RM0364
Reference manual

STM32F334xx advanced Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F334xx microcontroller memory and peripherals.

The STM32F334xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics refer to the STM32F334x4/x6/x8 datasheet.

For information on the Arm® Cortex®-M4 core with FPU, refer to the *STM32 Cortex®-M4 MCUs and MPUs programming manual* (PM0214).

Related documents

Available from STMicroelectronics web site *www.st.com*:

- STM32F334x4/x6/x8 datasheet
- *STM32 Cortex®-M4 MCUs and MPUs programming manual* (PM0214)

Contents

1 Documentation conventions 43

1.1 General information 43

1.2 List of abbreviations for registers 43

1.3 Glossary 44

1.4 Availability of peripherals 44

2 System and memory overview 45

2.1 System architecture 45

2.1.1 S0: I-bus 46

2.1.2 S1: D-bus 46

2.1.3 S2: S-bus 46

2.1.4 S3: DMA-bus 46

2.1.5 BusMatrix 46

2.2 Memory organization 47

2.2.1 Introduction 47

2.2.2 Memory map and register boundary addresses 48

2.2.3 Parity check 51

2.2.4 CCM SRAM write protection 51

2.3 Embedded SRAM 51

2.4 Flash memory overview 52

2.5 Boot configuration 52

2.5.1 Embedded boot loader 53

3 Embedded Flash memory 54

3.1 Flash main features 54

3.2 Flash memory functional description 54

3.2.1 Flash memory organization 54

3.2.2 Read operations 55

3.2.3 Flash program and erase operations 57

3.3 Memory protection 63

3.3.1 Read protection 63

3.3.2 Write protection 65

3.3.3 Option byte block write protection 65

3.4 Flash interrupts 65
3.5 Flash register description 66
3.5.1 Flash access control register (FLASH_ACR) 66
3.5.2 Flash key register (FLASH_KEYR) 66
3.5.3 Flash option key register (FLASH_OPTKEYR) 67
3.5.4 Flash status register (FLASH_SR) 67
3.5.5 Flash control register (FLASH_CR) 68
3.5.6 Flash address register (FLASH_AR) 69
3.5.7 Option byte register (FLASH_OBR) 70
3.5.8 Write protection register (FLASH_WRPR) 71
3.6 Flash register map 71

4 Option byte description 73

5 Cyclic redundancy check calculation unit (CRC) 76
5.1 Introduction 76
5.2 CRC main features 76
5.3 CRC functional description 77
5.3.1 CRC block diagram 77
5.3.2 CRC internal signals 77
5.3.3 CRC operation 77
5.4 CRC registers 79
5.4.1 CRC data register (CRC_DR) 79
5.4.2 CRC independent data register (CRC_IDR) 79
5.4.3 CRC control register (CRC_CR) 80
5.4.4 CRC initial value (CRC_INIT) 80
5.4.5 CRC polynomial (CRC_POL) 81
5.4.6 CRC register map 81

6 Power control (PWR) 82
6.1 Power supplies 82
6.1.1 Independent A/D and D/A converter supply and reference voltage 83
6.1.2 Battery Backup domain 83
6.1.3 Voltage regulator 84
6.2 Power supply supervisor 84
6.2.1 Power on reset (POR)/power down reset (PDR) 84

6.2.2 Programmable voltage detector (PVD) . . . 85
6.3 Low-power modes . . . 86
6.3.1 Slowing down system clocks . . . 87
6.3.2 Peripheral clock gating . . . 87
6.3.3 Sleep mode . . . 87
6.3.4 Stop mode . . . 88
6.3.5 Standby mode . . . 90
6.3.6 Auto-wakeup from low-power mode . . . 92
6.4 Power control registers . . . 93
6.4.1 Power control register (PWR_CR) . . . 93
6.4.2 Power control/status register (PWR_CSR) . . . 94
6.4.3 PWR register map . . . 95

7 Peripheral interconnect matrix . . . 96
7.1 Introduction . . . 96
7.2 Connection summary . . . 96
7.3 Interconnection details . . . 97
7.3.1 DMA interconnections . . . 97
7.3.2 From ADC to ADC . . . 97
7.3.3 From ADC to TIM . . . 97
7.3.4 From TIM and EXTI to ADC . . . 98
7.3.5 From OPAMP to ADC . . . 98
7.3.6 From TS to ADC . . . 98
7.3.7 From VBAT to ADC . . . 98
7.3.8 From VREFINT to ADC . . . 98
7.3.9 From COMP to TIM . . . 98
7.3.10 From TIM to COMP . . . 99
7.3.11 From DAC to COMP . . . 100
7.3.12 From VREFINT to COMP . . . 100
7.3.13 From TIM to OPAMP . . . 100
7.3.14 From TIM to TIM . . . 100
7.3.15 From system errors to TIM . . . 101
7.3.16 From HSE, HSI, LSE, LSI, MCO, RTC to TIM . . . 101
7.3.17 From TIM and EXTI to DAC . . . 102
7.3.18 From TIM to IRTIM . . . 102
7.3.19 From ADC to HRTIM1 . . . 102
7.3.20 From system faults to HRTIM1 . . . 102

7.3.21 From COMP to HRTIM1 . . . 102
7.3.22 From OPAMP to HRTIM1 . . . 103
7.3.23 From TIM to HRTIM1 . . . 103
7.3.24 From HRTIM1 to ADC . . . 103
7.3.25 From HRTIM1 to DAC . . . 103

8 Reset and clock control (RCC) . . . 104

8.1 Reset . . . 104
8.1.1 Power reset . . . 104
8.1.2 System reset . . . 104
8.1.3 RTC domain reset . . . 105
8.2 Clocks . . . 106
8.2.1 HSE clock . . . 108
8.2.2 HSI clock . . . 109
8.2.3 PLL . . . 110
8.2.4 LSE clock . . . 110
8.2.5 LSI clock . . . 111
8.2.6 System clock (SYSCLK) selection . . . 111
8.2.7 Clock security system (CSS) . . . 111
8.2.8 ADC clock . . . 112
8.2.9 RTC clock . . . 112
8.2.10 Timers (TIMx) clock . . . 112
8.2.11 High-resolution timer (HRTIM) clock . . . 113
8.2.12 Watchdog clock . . . 113
8.2.13 Clock-out capability . . . 113
8.2.14 Internal/external clock measurement with TIM16 . . . 114
8.3 Low-power modes . . . 115
8.4 RCC registers . . . 116
8.4.1 Clock control register (RCC_CR) . . . 116
8.4.2 Clock configuration register (RCC_CFGR) . . . 118
8.4.3 Clock interrupt register (RCC_CIR) . . . 120
8.4.4 APB2 peripheral reset register (RCC_APB2RSTR) . . . 123
8.4.5 APB1 peripheral reset register (RCC_APB1RSTR) . . . 124
8.4.6 AHB peripheral clock enable register (RCC_AHBENR) . . . 126
8.4.7 APB2 peripheral clock enable register (RCC_APB2ENR) . . . 127
8.4.8 APB1 peripheral clock enable register (RCC_APB1ENR) . . . 129
8.4.9 RTC domain control register (RCC_BDCR) . . . 131

8.4.10 Control/status register (RCC_CSR) . . . 132
8.4.11 AHB peripheral reset register (RCC_AHBRSTR) . . . 134
8.4.12 Clock configuration register 2 (RCC_CFGR2) . . . 135
8.4.13 Clock configuration register 3 (RCC_CFGR3) . . . 136
8.4.14 RCC register map . . . 137

9 General-purpose I/Os (GPIO) . . . 139

9.1 Introduction . . . 139
9.2 GPIO main features . . . 139
9.3 GPIO functional description . . . 139
9.3.1 General-purpose I/O (GPIO) . . . 142
9.3.2 I/O pin alternate function multiplexer and mapping . . . 142
9.3.3 I/O port control registers . . . 143
9.3.4 I/O port data registers . . . 143
9.3.5 I/O data bitwise handling . . . 143
9.3.6 GPIO locking mechanism . . . 144
9.3.7 I/O alternate function input/output . . . 144
9.3.8 External interrupt/wakeup lines . . . 144
9.3.9 Input configuration . . . 144
9.3.10 Output configuration . . . 145
9.3.11 Alternate function configuration . . . 146
9.3.12 Analog configuration . . . 147
9.3.13 Using the HSE or LSE oscillator pins as GPIOs . . . 148
9.3.14 Using the GPIO pins in the RTC supply domain . . . 148
9.4 GPIO registers . . . 148
9.4.1 GPIO port mode register (GPIOx_MODER) (x =A to D and F) . . . 148
9.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to D and F) . . . 149
9.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to D and F) . . . 149
9.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to D and F) . . . 150
9.4.5 GPIO port input data register (GPIOx_IDR) (x = A to D and F) . . . 150
9.4.6 GPIO port output data register (GPIOx_ODR) (x = A to D and F) . . . 151
9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to D and F) . . . 151

9.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to E and F) . . . 151
9.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to D and F) . . . 153
9.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to D and F) . . . 153
9.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to D and F) . . . 154
9.4.12 GPIO register map . . . 155

10 System configuration controller (SYSCFG) . . . 157
10.1 SYSCFG registers . . . 157
10.1.1 SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . 157
10.1.2 SYSCFG CCM SRAM protection register (SYSCFG_RCR) . . . 159
10.1.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . 160
10.1.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . 161
10.1.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . 163
10.1.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . 164
10.1.7 SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . 165
10.1.8 SYSCFG configuration register 3 (SYSCFG_CFGR3) . . . 167
10.1.9 SYSCFG register map . . . 168

11 Direct memory access controller (DMA) . . . 170
11.1 Introduction . . . 170
11.2 DMA main features . . . 170
11.3 DMA implementation . . . 171
11.3.1 DMA . . . 171
11.3.2 DMA request mapping . . . 171
11.4 DMA functional description . . . 173
11.4.1 DMA block diagram . . . 173
11.4.2 DMA transfers . . . 174
11.4.3 DMA arbitration . . . 175
11.4.4 DMA channels . . . 176
11.4.5 DMA data width, alignment and endianness . . . 180
11.4.6 DMA error management . . . 181
11.5 DMA interrupts . . . 182

11.6 DMA registers 182
11.6.1 DMA interrupt status register (DMA_ISR) 182
11.6.2 DMA interrupt flag clear register (DMA_IFCR) 185
11.6.3 DMA channel x configuration register (DMA_CCRx) 186
11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx) . 189
11.6.5 DMA channel x peripheral address register (DMA_CPARx) 189
11.6.6 DMA channel x memory address register (DMA_CMARx) 190
11.6.7 DMA register map 190

12 Interrupts and events 193
12.1 Nested vectored interrupt controller (NVIC) 193
12.1.1 NVIC main features 193
12.1.2 SysTick calibration value register 193
12.1.3 Interrupt and exception vectors 193
12.2 Extended interrupts and events controller (EXTI) 196
12.2.1 Main features 197
12.2.2 Block diagram 197
12.2.3 Wakeup event management 198
12.2.4 Asynchronous Internal Interrupts 198
12.2.5 Functional description 198
12.2.6 External and internal interrupt/event line mapping 200
12.3 EXTI registers 201
12.3.1 Interrupt mask register (EXTI_IMR1) 201
12.3.2 Event mask register (EXTI_EMR1) 202
12.3.3 Rising trigger selection register (EXTI_RTSR1) 202
12.3.4 Falling trigger selection register (EXTI_FTSR1) 203
12.3.5 Software interrupt event register (EXTI_SWIER1) 204
12.3.6 Pending register (EXTI_PR1) 205
12.3.7 Interrupt mask register (EXTI_IMR2) 206
12.3.8 Event mask register (EXTI_EMR2) 206
12.3.9 Rising trigger selection register (EXTI_RTSR2) 206
12.3.10 Falling trigger selection register (EXTI_FTSR2) 207
12.3.11 Software interrupt event register (EXTI_SWIER2) 207
12.3.12 Pending register (EXTI_PR2) 208
12.3.13 EXTI register map 208

13 Analog-to-digital converters (ADC) 211

13.1 Introduction 211
13.2 ADC main features 212
13.3 ADC functional description 214
13.3.1 ADC block diagram 214
13.3.2 Pins and internal signals 215
13.3.3 Clocks 215
13.3.4 ADC1/2 connectivity 218
13.3.5 Slave AHB interface 219
13.3.6 ADC voltage regulator (ADVREGEN) 219
13.3.7 Single-ended and differential input channels 219
13.3.8 Calibration (ADCAL, ADCALDIF, ADCx_CALFACT) 220
13.3.9 ADC on-off control (ADEN, ADDIS, ADRDY) 223
13.3.10 Constraints when writing the ADC control bits 224
13.3.11 Channel selection (SQRx, JSQRx) 224
13.3.12 Channel-wise programmable sampling time (SMPR1, SMPR2) 225
13.3.13 Single conversion mode (CONT=0) 226
13.3.14 Continuous conversion mode (CONT=1) 226
13.3.15 Starting conversions (ADSTART, JADSTART) 227
13.3.16 Timing 228
13.3.17 Stopping an ongoing conversion (ADSTP, JADSTP) 228
13.3.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) 230
13.3.19 Injected channel management 233
13.3.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 234
13.3.21 Queue of context for injected conversions 235
13.3.22 Programmable resolution (RES) - fast conversion mode 244
13.3.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . 244
13.3.24 End of conversion sequence (EOS, JEOS) 245
13.3.25 Timing diagrams example (single/continuous modes, hardware/software triggers) 245
13.3.26 Data management 246
13.3.27 Dynamic low-power features 252
13.3.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) 257
13.3.29 Dual ADC modes 261
13.3.30 Temperature sensor 274
13.3.31 VBAT supply monitoring 276
13.3.32 Monitoring the internal voltage reference 277

13.4 ADC interrupts 279
13.5 ADC registers (for each ADC) 280
13.5.1 ADC interrupt and status register (ADCx_ISR, x=1..2) 280
13.5.2 ADC interrupt enable register (ADCx_IER, x=1..2) 282
13.5.3 ADC control register (ADCx_CR, x=1..2) 284
13.5.4 ADC configuration register (ADCx_CFGR, x=1..2) 287
13.5.5 ADC sample time register 1 (ADCx_SMPR1, x=1..2) 291
13.5.6 ADC sample time register 2 (ADCx_SMPR2, x=1..2) 293
13.5.7 ADC watchdog threshold register 1 (ADCx_TR1, x=1..2) 293
13.5.8 ADC watchdog threshold register 2 (ADCx_TR2, x = 1..2) 294
13.5.9 ADC watchdog threshold register 3 (ADCx_TR3, x=1..2) 295
13.5.10 ADC regular sequence register 1 (ADCx_SQR1, x=1..2) 296
13.5.11 ADC regular sequence register 2 (ADCx_SQR2, x=1..2) 297
13.5.12 ADC regular sequence register 3 (ADCx_SQR3, x=1..2) 299
13.5.13 ADC regular sequence register 4 (ADCx_SQR4, x=1..2) 300
13.5.14 ADC regular Data Register (ADCx_DR, x=1..2) 301
13.5.15 ADC injected sequence register (ADCx_JSQR, x=1..2) 302
13.5.16 ADC offset register (ADCx_OFRy, x=1..2) (y=1..4) 304
13.5.17 ADC injected data register (ADCx_JDRy, x=1..2, y= 1..4) 305
13.5.18 ADC Analog Watchdog 2 Configuration Register (ADCx_AWD2CR, x=1..2) 305
13.5.19 ADC Analog Watchdog 3 Configuration Register (ADCx_AWD3CR, x=1..2) 306
13.5.20 ADC Differential Mode Selection Register (ADCx_DIFSEL, x=1..2) . . 306
13.5.21 ADC Calibration Factors (ADCx_CALFACT, x=1..2) 307
13.6 ADC common registers 308
13.6.1 ADC Common status register (ADCx_CSR, x=12) 308
13.6.2 ADC common control register (ADCx_CCR, x=12) 310
13.6.3 ADC common regular data register for dual mode (ADCx_CDR, x=12) 313
13.7 ADC register map 313

14 Digital-to-analog converter (DAC1 and DAC2) 317
14.1 Introduction 317
14.2 DAC1/2 main features 317
14.3 DAC output buffer enable/DAC output switch 319
14.4 DAC channel enable 320

14.5 Single mode functional description 320
14.5.1 DAC data format 320
14.5.2 DAC channel conversion 320
14.5.3 DAC output voltage 322
14.5.4 DAC trigger selection 322
14.6 Dual-mode functional description 323
14.6.1 DAC data format 323
14.6.2 DAC channel conversion in dual mode 324
14.6.3 Description of dual conversion modes 324
14.6.4 DAC output voltage 327
14.6.5 DAC trigger selection 328
14.7 Noise generation 328
14.8 Triangle-wave generation 329
14.9 DMA request 330
14.10 DAC registers 331
14.10.1 DAC control register (DAC_CR) 331
14.10.2 DAC software trigger register (DAC_SWTRIGR) 335
14.10.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 335
14.10.4 DAC channel1 12-bit left-aligned data holding register (DAC_DHR12L1) 336
14.10.5 DAC channel1 8-bit right-aligned data holding register (DAC_DHR8R1) 336
14.10.6 DAC channel2 12-bit right-aligned data holding register (DAC_DHR12R2) 336
14.10.7 DAC channel2 12-bit left-aligned data holding register (DAC_DHR12L2) 337
14.10.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) 337
14.10.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) 338
14.10.10 Dual DAC 12-bit left-aligned data holding register (DAC_DHR12LD) 338
14.10.11 Dual DAC 8-bit right-aligned data holding register (DAC_DHR8RD) 338
14.10.12 DAC channel1 data output register (DAC_DOR1) 339
14.10.13 DAC channel2 data output register (DAC_DOR2) 339
14.10.14 DAC status register (DAC_SR) 339
14.10.15 DAC register map 341

15 Comparator (COMP) 343
15.1 Introduction 343
15.2 COMP main features 343
15.3 COMP functional description 344
15.3.1 COMP block diagram 344
15.3.2 COMP pins and internal signals 344
15.3.3 COMP reset and clocks 345
15.3.4 Comparator LOCK mechanism 345
15.3.5 Comparator output blanking function 346
15.4 COMP interrupts 347
15.5 COMP registers 347
15.5.1 COMP2 control and status register (COMP2_CSR) 347
15.5.2 COMP4 control and status register (COMP4_CSR) 348
15.5.3 COMP6 control and status register (COMP6_CSR) 350
15.5.4 COMP register map 352

16 Operational amplifier (OPAMP) 353
16.1 OPAMP introduction 353
16.2 OPAMP main features 353
16.3 OPAMP functional description 353
16.3.1 General description 353
16.3.2 Clock 353
16.3.3 Operational amplifiers and comparators interconnections 354
16.3.4 Using the OPAMP outputs as ADC inputs 354
16.3.5 Calibration 354
16.3.6 Timer controlled Multiplexer mode 355
16.3.7 OPAMP modes 356
16.4 OPAMP registers 360
16.4.1 OPAMP2 control register (OPAMP2_CSR) 360
16.4.2 OPAMP register map 363

17 Touch sensing controller (TSC) 364
17.1 Introduction 364
17.2 TSC main features 364
17.3 TSC functional description 365
17.3.1 TSC block diagram 365

17.3.2 Surface charge transfer acquisition overview . . . 365
17.3.3 Reset and clocks . . . 367
17.3.4 Charge transfer acquisition sequence . . . 368
17.3.5 Spread spectrum feature . . . 369
17.3.6 Max count error . . . 369
17.3.7 Sampling capacitor I/O and channel I/O mode selection . . . 370
17.3.8 Acquisition mode . . . 371
17.3.9 I/O hysteresis and analog switch control . . . 371
17.4 TSC low-power modes . . . 372
17.5 TSC interrupts . . . 372
17.6 TSC registers . . . 373
17.6.1 TSC control register (TSC_CR) . . . 373
17.6.2 TSC interrupt enable register (TSC_IER) . . . 375
17.6.3 TSC interrupt clear register (TSC_ICR) . . . 376
17.6.4 TSC interrupt status register (TSC_ISR) . . . 377
17.6.5 TSC I/O hysteresis control register (TSC_IOHCR) . . . 377
17.6.6 TSC I/O analog switch control register (TSC_IOASCR) . . . 378
17.6.7 TSC I/O sampling control register (TSC_IOSCR) . . . 378
17.6.8 TSC I/O channel control register (TSC_IOCCR) . . . 379
17.6.9 TSC I/O group control status register (TSC_IOGCSR) . . . 379
17.6.10 TSC I/O group x counter register (TSC_IOGxCR) . . . 380
17.6.11 TSC register map . . . 381

18 Advanced-control timer (TIM1) . . . 383
18.1 TIM1 introduction . . . 383
18.2 TIM1 main features . . . 384
18.3 TIM1 functional description . . . 387
18.3.1 Time-base unit . . . 387
18.3.2 Counter modes . . . 389
18.3.3 Repetition counter . . . 400
18.3.4 External trigger input . . . 402
18.3.5 Clock selection . . . 403
18.3.6 Capture/compare channels . . . 407
18.3.7 Input capture mode . . . 410
18.3.8 PWM input mode . . . 411
18.3.9 Forced output mode . . . 411

18.3.10 Output compare mode . . . 412
18.3.11 PWM mode . . . 413
18.3.12 Asymmetric PWM mode . . . 416
18.3.13 Combined PWM mode . . . 417
18.3.14 Combined 3-phase PWM mode . . . 418
18.3.15 Complementary outputs and dead-time insertion . . . 419
18.3.16 Using the break function . . . 421
18.3.17 Clearing the OCxREF signal on an external event . . . 426
18.3.18 6-step PWM generation . . . 428
18.3.19 One-pulse mode . . . 429
18.3.20 Retriggerable one pulse mode . . . 430
18.3.21 Encoder interface mode . . . 431
18.3.22 UIF bit remapping . . . 433
18.3.23 Timer input XOR function . . . 434
18.3.24 Interfacing with Hall sensors . . . 434
18.3.25 Timer synchronization . . . 437
18.3.26 ADC synchronization . . . 441
18.3.27 DMA burst mode . . . 441
18.3.28 Debug mode . . . 442
18.4 TIM1 registers . . . 443
18.4.1 TIM1 control register 1 (TIM1_CR1) . . . 443
18.4.2 TIM1 control register 2 (TIM1_CR2) . . . 444
18.4.3 TIM1 slave mode control register (TIM1_SMCR) . . . 447
18.4.4 TIM1 DMA/interrupt enable register (TIM1_DIER) . . . 449
18.4.5 TIM1 status register (TIM1_SR) . . . 451
18.4.6 TIM1 event generation register (TIM1_EGR) . . . 453
18.4.7 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . 454
18.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . 455
18.4.9 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . 458
18.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . 459
18.4.11 TIM1 capture/compare enable register (TIM1_CCER) . . . 461
18.4.12 TIM1 counter (TIM1_CNT) . . . 464
18.4.13 TIM1 prescaler (TIM1_PSC) . . . 464

18.4.14 TIM1 auto-reload register (TIM1_ARR) 464
18.4.15 TIM1 repetition counter register (TIM1_RCR) 465
18.4.16 TIM1 capture/compare register 1 (TIM1_CCR1) 465
18.4.17 TIM1 capture/compare register 2 (TIM1_CCR2) 466
18.4.18 TIM1 capture/compare register 3 (TIM1_CCR3) 466
18.4.19 TIM1 capture/compare register 4 (TIM1_CCR4) 467
18.4.20 TIM1 break and dead-time register (TIM1_BDTR) 467
18.4.21 TIM1 DMA control register (TIM1_DCR) 470
18.4.22 TIM1 DMA address for full transfer (TIM1_DMAR) 471
18.4.23 TIM1 option registers (TIM1_OR) 472
18.4.24 TIM1 capture/compare mode register 3 (TIM1_CCMR3) 472
18.4.25 TIM1 capture/compare register 5 (TIM1_CCR5) 473
18.4.26 TIM1 capture/compare register 6 (TIM1_CCR6) 474
18.4.27 TIM1 register map 475

19 General-purpose timers (TIM2/TIM3) 478

19.1 TIM2/TIM3 introduction 478
19.2 TIM2/TIM3 main features 478
19.3 TIM2/TIM3 functional description 480
19.3.1 Time-base unit 480
19.3.2 Counter modes 482
19.3.3 Clock selection 492
19.3.4 Capture/Compare channels 496
19.3.5 Input capture mode 498
19.3.6 PWM input mode 499
19.3.7 Forced output mode 500
19.3.8 Output compare mode 501
19.3.9 PWM mode 502
19.3.10 Asymmetric PWM mode 505
19.3.11 Combined PWM mode 506
19.3.12 Clearing the OCxREF signal on an external event 507
19.3.13 One-pulse mode 509
19.3.14 Retriggerable one pulse mode 510
19.3.15 Encoder interface mode 511
19.3.16 UIF bit remapping 513

19.3.17 Timer input XOR function . . . 513
19.3.18 Timers and external trigger synchronization . . . 514
19.3.19 Timer synchronization . . . 517
19.3.20 DMA burst mode . . . 522
19.3.21 Debug mode . . . 523
19.4 TIM2/TIM3 registers . . . 524
19.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 3) . . . 524
19.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 3) . . . 525
19.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3) . . . 527
19.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3) . . . 530
19.4.5 TIMx status register (TIMx_SR)(x = 2 to 3) . . . 531
19.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 3) . . . 532
19.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3) . . . 533
19.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3) . . . 535
19.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 3) . . . 537
19.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 3) . . . 538
19.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3) . . . 539
19.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3) . . . 540
19.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3) . . . 541
19.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 3) . . . 541
19.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 3) . . . 542
19.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3) . . . 542
19.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3) . . . 543
19.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3) . . . 543
19.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3) . . . 544
19.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 3) . . . 545
19.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3) . . . 545
19.4.22 TIMx register map . . . 546

20 General-purpose timers (TIM15/TIM16/TIM17) . . . 548
20.1 TIM15/TIM16/TIM17 introduction . . . 548
20.2 TIM15 main features . . . 548
20.3 TIM16/TIM17 main features . . . 549

20.4 TIM15/TIM16/TIM17 functional description 552
20.4.1 Time-base unit 552
20.4.2 Counter modes 554
20.4.3 Repetition counter 558
20.4.4 Clock selection 559
20.4.5 Capture/compare channels 561
20.4.6 Input capture mode 564
20.4.7 PWM input mode (only for TIM15) 565
20.4.8 Forced output mode 566
20.4.9 Output compare mode 566
20.4.10 PWM mode 567
20.4.11 Combined PWM mode (TIM15 only) 568
20.4.12 Complementary outputs and dead-time insertion 570
20.4.13 Using the break function 572
20.4.14 One-pulse mode 575
20.4.15 Retriggerable one pulse mode (TIM15 only) 577
20.4.16 UIF bit remapping 577
20.4.17 Timer input XOR function (TIM15 only) 579
20.4.18 External trigger synchronization (TIM15 only) 580
20.4.19 Slave mode – combined reset + trigger mode (TIM15 only) 582
20.4.20 DMA burst mode 582
20.4.21 Timer synchronization (TIM15) 584
20.4.22 Using timer output as trigger for other timers (TIM16/TIM17) 584
20.4.23 Debug mode 584
20.5 TIM15 registers 585
20.5.1 TIM15 control register 1 (TIM15_CR1) 585
20.5.2 TIM15 control register 2 (TIM15_CR2) 586
20.5.3 TIM15 slave mode control register (TIM15_SMCR) 588
20.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER) 589
20.5.5 TIM15 status register (TIM15_SR) 590
20.5.6 TIM15 event generation register (TIM15_EGR) 592
20.5.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 593
20.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 594
20.5.9 TIM15 capture/compare enable register (TIM15_CCER) 597
20.5.10 TIM15 counter (TIM15_CNT) 600

20.5.11 TIM15 prescaler (TIM15_PSC) . . . 600
20.5.12 TIM15 auto-reload register (TIM15_ARR) . . . 600
20.5.13 TIM15 repetition counter register (TIM15_RCR) . . . 601
20.5.14 TIM15 capture/compare register 1 (TIM15_CCR1) . . . 601
20.5.15 TIM15 capture/compare register 2 (TIM15_CCR2) . . . 602
20.5.16 TIM15 break and dead-time register (TIM15_BDTR) . . . 602
20.5.17 TIM15 DMA control register (TIM15_DCR) . . . 604
20.5.18 TIM15 DMA address for full transfer (TIM15_DMAR) . . . 605
20.5.19 TIM15 register map . . . 605
20.6 TIM16/TIM17 registers . . . 608
20.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) . . . 608
20.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) . . . 609
20.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) . . . 610
20.6.4 TIMx status register (TIMx_SR)(x = 16 to 17) . . . 611
20.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) . . . 612
20.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17) . . . 613
20.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17) . . . 614
20.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . . . 616
20.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17) . . . 618
20.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) . . . 619
20.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) . . . 619
20.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) . . . 620
20.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) . . . 620
20.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) . . . 621
20.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) . . . 623
20.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) . . . 623
20.6.17 TIM16 option register (TIM16_OR) . . . 624
20.6.18 TIM16/TIM17 register map . . . 625

21 High-Resolution Timer (HRTIM) . . . 627
21.1 Introduction . . . 627
21.2 Main features . . . 628
21.3 Functional description . . . 629
21.3.1 General description . . . 629
21.3.2 HRTIM pins and internal signals . . . 630

21.3.3 Clocks . . . 632
21.3.4 Timer A..E timing units . . . 635
21.3.5 Master timer . . . 653
21.3.6 Set/reset events priorities and narrow pulses management . . . 654
21.3.7 External events global conditioning . . . 657
21.3.8 External event filtering in timing units . . . 661
21.3.9 Delayed Protection . . . 666
21.3.10 Register preload and update management . . . 672
21.3.11 Events propagation within or across multiple timers . . . 675
21.3.12 Output management . . . 679
21.3.13 Burst mode controller . . . 681
21.3.14 Chopper . . . 690
21.3.15 Fault protection . . . 691
21.3.16 Auxiliary outputs . . . 694
21.3.17 Synchronizing the HRTIM with other timers or HRTIM instances . . . 697
21.3.18 ADC triggers . . . 700
21.3.19 DAC triggers . . . 701
21.3.20 HRTIM Interrupts . . . 703
21.3.21 DMA . . . 705
21.3.22 HRTIM initialization . . . 709
21.3.23 Debug . . . 710
21.4 Application use cases . . . 711
21.4.1 Buck converter . . . 711
21.4.2 Buck converter with synchronous rectification . . . 712
21.4.3 Multiphase converters . . . 712
21.4.4 Transition mode Power Factor Correction . . . 714
21.5 HRTIM registers . . . 716
21.5.1 HRTIM Master Timer Control Register (HRTIM_MCR) . . . 716
21.5.2 HRTIM Master Timer Interrupt Status Register (HRTIM_MISR) . . . 719
21.5.3 HRTIM Master Timer Interrupt Clear Register (HRTIM_MICR) . . . 720
21.5.4 HRTIM Master Timer DMA / Interrupt Enable Register (HRTIM_MDIER) . . . 721
21.5.5 HRTIM Master Timer Counter Register (HRTIM_MCNTR) . . . 723
21.5.6 HRTIM Master Timer Period Register (HRTIM_MPER) . . . 723
21.5.7 HRTIM Master Timer Repetition Register (HRTIM_MREP) . . . 724
21.5.8 HRTIM Master Timer Compare 1 Register (HRTIM_MCMP1R) . . . 724
21.5.9 HRTIM Master Timer Compare 2 Register (HRTIM_MCMP2R) . . . 725

21.5.10 HRTIM Master Timer Compare 3 Register (HRTIM_MCMP3R) 725
21.5.11 HRTIM Master Timer Compare 4 Register (HRTIM_MCMP4R) 726
21.5.12 HRTIM Timerx Control Register (HRTIM_TIMxCR) 727
21.5.13 HRTIM Timerx Interrupt Status Register (HRTIM_TIMxISR) 731
21.5.14 HRTIM Timerx Interrupt Clear Register (HRTIM_TIMxICR) 733
21.5.15 HRTIM Timerx DMA / Interrupt Enable Register (HRTIM_TIMxDIER) 734
21.5.16 HRTIM Timerx Counter Register (HRTIM_CNTxR) 737
21.5.17 HRTIM Timerx Period Register (HRTIM_PERxR) 737
21.5.18 HRTIM Timerx Repetition Register (HRTIM_REPxR) 738
21.5.19 HRTIM Timerx Compare 1 Register (HRTIM_CMP1xR) 738
21.5.20 HRTIM Timerx Compare 1 Compound Register (HRTIM_CMP1CxR) 739
21.5.21 HRTIM Timerx Compare 2 Register (HRTIM_CMP2xR) 739
21.5.22 HRTIM Timerx Compare 3 Register (HRTIM_CMP3xR) 740
21.5.23 HRTIM Timerx Compare 4 Register (HRTIM_CMP4xR) 740
21.5.24 HRTIM Timerx Capture 1 Register (HRTIM_CPT1xR) 741
21.5.25 HRTIM Timerx Capture 2 Register (HRTIM_CPT2xR) 741
21.5.26 HRTIM Timerx Deadtime Register (HRTIM_DTxR) 742
21.5.27 HRTIM Timerx Output1 Set Register (HRTIM_SETx1R) 744
21.5.28 HRTIM Timerx Output1 Reset Register (HRTIM_RSTx1R) 746
21.5.29 HRTIM Timerx Output2 Set Register (HRTIM_SETx2R) 746
21.5.30 HRTIM Timerx Output2 Reset Register (HRTIM_RSTx2R) 747
21.5.31 HRTIM Timerx External Event Filtering Register 1 (HRTIM_EEFxR1) 748
21.5.32 HRTIM Timerx External Event Filtering Register 2 (HRTIM_EEFxR2) 750
21.5.33 HRTIM Timerx Reset Register (HRTIM_RSTxR) 751
21.5.34 HRTIM Timerx Chopper Register (HRTIM_CHPxR) 754
21.5.35 HRTIM Timerx Capture 1 Control Register (HRTIM_CPT1xCR) 756
21.5.36 HRTIM Timerx Capture 2 Control Register (HRTIM_CPT2xCR) 757
21.5.37 HRTIM Timerx Output Register (HRTIM_OUTxR) 760
21.5.38 HRTIM Timerx Fault Register (HRTIM_FLTxR) 763
21.5.39 HRTIM Control Register 1 (HRTIM_CR1) 764
21.5.40 HRTIM Control Register 2 (HRTIM_CR2) 766
21.5.41 HRTIM Interrupt Status Register (HRTIM_ISR) 767
21.5.42 HRTIM Interrupt Clear Register (HRTIM_ICR) 768
21.5.43 HRTIM Interrupt Enable Register (HRTIM_IER) 769

21.5.44 HRTIM Output Enable Register (HRTIM_OENR) 770
21.5.45 HRTIM Output Disable Register (HRTIM_ODISR) 771
21.5.46 HRTIM Output Disable Status Register (HRTIM_ODSR) 772
21.5.47 HRTIM Burst Mode Control Register (HRTIM_BMCR) 773
21.5.48 HRTIM Burst Mode Trigger Register (HRTIM_BMTRGR) 775
21.5.49 HRTIM Burst Mode Compare Register (HRTIM_BMCMPR) 777
21.5.50 HRTIM Burst Mode Period Register (HRTIM_BMPER) 777
21.5.51 HRTIM Timer External Event Control Register 1 (HRTIM_EECR1) . . 778
21.5.52 HRTIM Timer External Event Control Register 2 (HRTIM_EECR2) . . 780
21.5.53 HRTIM Timer External Event Control Register 3 (HRTIM_EECR3) . . 781
21.5.54 HRTIM ADC Trigger 1 Register (HRTIM_ADC1R) 782
21.5.55 HRTIM ADC Trigger 2 Register (HRTIM_ADC2R) 783
21.5.56 HRTIM ADC Trigger 3 Register (HRTIM_ADC3R) 784
21.5.57 HRTIM ADC Trigger 4 Register (HRTIM_ADC4R) 786
21.5.58 HRTIM DLL Control Register (HRTIM_DLLCR) 788
21.5.59 HRTIM Fault Input Register 1 (HRTIM_FLTINR1) 789
21.5.60 HRTIM Fault Input Register 2 (HRTIM_FLTINR2) 791
21.5.61 HRTIM Burst DMA Master timer update Register (HRTIM_BDMUPR) 793
21.5.62 HRTIM Burst DMA Timerx update Register (HRTIM_BDTxUPR) 794
21.5.63 HRTIM Burst DMA Data Register (HRTIM_BDMADR) 795
21.5.64 HRTIM register map 796

22 Infrared interface (IRTIM) 805

23 Basic timers (TIM6/TIM7) 806

23.1 TIM6/TIM7 introduction 806
23.2 TIM6/TIM7 main features 806
23.3 TIM6/TIM7 functional description 807
23.3.1 Time-base unit 807
23.3.2 Counting mode 809
23.3.3 UIF bit remapping 812
23.3.4 Clock source 812
23.3.5 Debug mode 813
23.4 TIM6/TIM7 registers 813
23.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) 813
23.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) 815

23.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) 815
23.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) 816
23.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) 816
23.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) 816
23.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) 817
23.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7) 817
23.4.9 TIMx register map 818

24 Independent watchdog (IWDG) 819

24.1 Introduction 819
24.2 IWDG main features 819
24.3 IWDG functional description 819
24.3.1 IWDG block diagram 819
24.3.2 Window option 820
24.3.3 Hardware watchdog 821
24.3.4 Register access protection 821
24.3.5 Debug mode 821
24.4 IWDG registers 822
24.4.1 IWDG key register (IWDG_KR) 822
24.4.2 IWDG prescaler register (IWDG_PR) 823
24.4.3 IWDG reload register (IWDG_RLR) 824
24.4.4 IWDG status register (IWDG_SR) 825
24.4.5 IWDG window register (IWDG_WINR) 826
24.4.6 IWDG register map 827

25 System window watchdog (WWDG) 828

25.1 Introduction 828
25.2 WWDG main features 828
25.3 WWDG functional description 828
25.3.1 WWDG block diagram 829
25.3.2 Enabling the watchdog 829
25.3.3 Controlling the down-counter 829
25.3.4 How to program the watchdog timeout 829
25.3.5 Debug mode 831
25.4 WWDG interrupts 831
25.5 WWDG registers 831

25.5.1 WWDG control register (WWDG_CR) 831
25.5.2 WWDG configuration register (WWDG_CFR) 832
25.5.3 WWDG status register (WWDG_SR) 832
25.5.4 WWDG register map 833

26 Real-time clock (RTC) 834
26.1 Introduction 834
26.2 RTC main features 835
26.3 RTC functional description 836
26.3.1 RTC block diagram 836
26.3.2 GPIOs controlled by the RTC 837
26.3.3 Clock and prescalers 839
26.3.4 Real-time clock and calendar 839
26.3.5 Programmable alarms 840
26.3.6 Periodic auto-wakeup 840
26.3.7 RTC initialization and configuration 841
26.3.8 Reading the calendar 842
26.3.9 Resetting the RTC 843
26.3.10 RTC synchronization 844
26.3.11 RTC reference clock detection 844
26.3.12 RTC smooth digital calibration 845
26.3.13 Time-stamp function 847
26.3.14 Tamper detection 848
26.3.15 Calibration clock output 849
26.3.16 Alarm output 850
26.4 RTC low-power modes 850
26.5 RTC interrupts 850
26.6 RTC registers 851
26.6.1 RTC time register (RTC_TR) 851
26.6.2 RTC date register (RTC_DR) 852
26.6.3 RTC control register (RTC_CR) 854
26.6.4 RTC initialization and status register (RTC_ISR) 857
26.6.5 RTC prescaler register (RTC_PRER) 860
26.6.6 RTC wakeup timer register (RTC_WUTR) 861
26.6.7 RTC alarm A register (RTC_ALRMAR) 862
26.6.8 RTC alarm B register (RTC_ALRMBR) 863

26.6.9 RTC write protection register (RTC_WPR) 864
26.6.10 RTC sub second register (RTC_SSR) 864
26.6.11 RTC shift control register (RTC_SHIFTR) 865
26.6.12 RTC timestamp time register (RTC_TSTR) 866
26.6.13 RTC timestamp date register (RTC_TSDR) 867
26.6.14 RTC time-stamp sub second register (RTC_TSSSR) 868
26.6.15 RTC calibration register (RTC_CALR) 869
26.6.16 RTC tamper and alternate function configuration register (RTC_TAFCR) 870
26.6.17 RTC alarm A sub second register (RTC_ALRMASSR) 873
26.6.18 RTC alarm B sub second register (RTC_ALRMBSSR) 874
26.6.19 RTC backup registers (RTC_BKPxR) 875
26.6.20 RTC register map 875

27 Inter-integrated circuit (I2C) interface 878

27.1 Introduction 878
27.2 I2C main features 878
27.3 I2C implementation 879
27.4 I2C functional description 879
27.4.1 I2C block diagram 880
27.4.2 I2C pins and internal signals 881
27.4.3 I2C clock requirements 881
27.4.4 Mode selection 881
27.4.5 I2C initialization 882
27.4.6 Software reset 887
27.4.7 Data transfer 888
27.4.8 I2C slave mode 890
27.4.9 I2C master mode 899
27.4.10 I2C_TIMINGR register configuration examples 911
27.4.11 SMBus specific features 912
27.4.12 SMBus initialization 915
27.4.13 SMBus: I2C_TIMEOUTR register configuration examples 917
27.4.14 SMBus slave mode 918
27.4.15 Wakeup from Stop mode on address match 926
27.4.16 Error conditions 926
27.4.17 DMA requests 928
27.4.18 Debug mode 929

27.5 I2C low-power modes . . . 929
27.6 I2C interrupts . . . 930
27.7 I2C registers . . . 931
27.7.1 I2C control register 1 (I2C_CR1) . . . 931
27.7.2 I2C control register 2 (I2C_CR2) . . . 934
27.7.3 I2C own address 1 register (I2C_OAR1) . . . 937
27.7.4 I2C own address 2 register (I2C_OAR2) . . . 938
27.7.5 I2C timing register (I2C_TIMINGR) . . . 939
27.7.6 I2C timeout register (I2C_TIMEOUTR) . . . 940
27.7.7 I2C interrupt and status register (I2C_ISR) . . . 941
27.7.8 I2C interrupt clear register (I2C_ICR) . . . 943
27.7.9 I2C PEC register (I2C_PECR) . . . 944
27.7.10 I2C receive data register (I2C_RXDR) . . . 945
27.7.11 I2C transmit data register (I2C_TXDR) . . . 945
27.7.12 I2C register map . . . 946

28 Universal synchronous/asynchronous receiver transmitter (USART/UART) . . . 948

28.1 Introduction . . . 948
28.2 USART main features . . . 948
28.3 USART extended features . . . 949
28.4 USART implementation . . . 950
28.5 USART functional description . . . 950
28.5.1 USART character description . . . 953
28.5.2 USART transmitter . . . 955
28.5.3 USART receiver . . . 957
28.5.4 USART baud rate generation . . . 964
28.5.5 Tolerance of the USART receiver to clock deviation . . . 966
28.5.6 USART auto baud rate detection . . . 967
28.5.7 Multiprocessor communication using USART . . . 968
28.5.8 Modbus communication using USART . . . 970
28.5.9 USART parity control . . . 971
28.5.10 USART LIN (local interconnection network) mode . . . 972
28.5.11 USART synchronous mode . . . 974
28.5.12 USART Single-wire Half-duplex communication . . . 977
28.5.13 USART Smartcard mode . . . 977

28.5.14 USART IrDA SIR ENDEC block 982
28.5.15 USART continuous communication in DMA mode 984
28.5.16 RS232 hardware flow control and RS485 driver enable using USART 986
28.5.17 Wakeup from Stop mode using USART 988
28.6 USART low-power modes 990
28.7 USART interrupts 990
28.8 USART registers 992
28.8.1 USART control register 1 (USART_CR1) 992
28.8.2 USART control register 2 (USART_CR2) 995
28.8.3 USART control register 3 (USART_CR3) 999
28.8.4 USART baud rate register (USART_BRR) 1003
28.8.5 USART guard time and prescaler register (USART_GTPR) 1003
28.8.6 USART receiver timeout register (USART_RTOR) 1004
28.8.7 USART request register (USART_RQR) 1005
28.8.8 USART interrupt and status register (USART_ISR) 1006
28.8.9 USART interrupt flag clear register (USART_ICR) 1011
28.8.10 USART receive data register (USART_RDR) 1012
28.8.11 USART transmit data register (USART_TDR) 1012
28.8.12 USART register map 1013

29 Serial peripheral interface (SPI) 1015
29.1 Introduction 1015
29.2 SPI main features 1015
29.3 SPI implementation 1015
29.4 SPI functional description 1016
29.4.1 General description 1016
29.4.2 Communications between one master and one slave 1017
29.4.3 Standard multi-slave communication 1019
29.4.4 Multi-master communication 1020
29.4.5 Slave select (NSS) pin management 1021
29.4.6 Communication formats 1022
29.4.7 Configuration of SPI 1024
29.4.8 Procedure for enabling SPI 1025
29.4.9 Data transmission and reception procedures 1025
29.4.10 SPI status flags 1035
29.4.11 SPI error flags 1036

29.4.12 NSS pulse mode 1037
29.4.13 TI mode 1037
29.4.14 CRC calculation 1038
29.5 SPI interrupts 1040
29.6 SPI registers 1041
29.6.1 SPI control register 1 (SPIx_CR1) 1041
29.6.2 SPI control register 2 (SPIx_CR2) 1043
29.6.3 SPI status register (SPIx_SR) 1045
29.6.4 SPI data register (SPIx_DR) 1046
29.6.5 SPI CRC polynomial register (SPIx_CRCPR) 1047
29.6.6 SPI Rx CRC register (SPIx_RXCRCR) 1047
29.6.7 SPI Tx CRC register (SPIx_TXCRCR) 1047
29.6.8 SPI register map 1049

30 Controller area network (bxCAN) 1050
30.1 Introduction 1050
30.2 bxCAN main features 1050
30.3 bxCAN general description 1050
30.3.1 CAN 2.0B active core 1051
30.3.2 Control, status and configuration registers 1051
30.3.3 Tx mailboxes 1051
30.3.4 Acceptance filters 1051
30.4 bxCAN operating modes 1052
30.4.1 Initialization mode 1052
30.4.2 Normal mode 1053
30.4.3 Sleep mode (low-power) 1053
30.5 Test mode 1054
30.5.1 Silent mode 1054
30.5.2 Loop back mode 1055
30.5.3 Loop back combined with silent mode 1055
30.6 Behavior in debug mode 1056
30.7 bxCAN functional description 1056
30.7.1 Transmission handling 1056
30.7.2 Time triggered communication mode 1058
30.7.3 Reception handling 1058
30.7.4 Identifier filtering 1059

30.7.5 Message storage 1063
30.7.6 Error management 1065
30.7.7 Bit timing 1065
30.8 bxCAN interrupts 1068
30.9 CAN registers 1069
30.9.1 Register access protection 1069
30.9.2 CAN control and status registers 1069
30.9.3 CAN mailbox registers 1079
30.9.4 CAN filter registers 1086
30.9.5 bxCAN register map 1090

31 Debug support (DBG) 1094
31.1 Overview 1094
31.2 Reference Arm documentation 1095
31.3 SWJ debug port (serial wire and JTAG) 1095
31.3.1 Mechanism to select the JTAG-DP or the SW-DP 1096
31.4 Pinout and debug port pins 1096
31.4.1 SWJ debug port pins 1097
31.4.2 Flexible SWJ-DP pin assignment 1097
31.4.3 Internal pull-up and pull-down on JTAG pins 1098
31.4.4 Using serial wire and releasing the unused debug pins as GPIOs . . 1099
31.5 STM32F334xx JTAG TAP connection 1099
31.6 ID codes and locking mechanism 1100
31.6.1 MCU device ID code 1100
31.6.2 Boundary scan TAP 1101
31.6.3 Cortex®-M4 TAP 1101
31.6.4 Cortex®-M4 JEDEC-106 ID code 1101
31.7 JTAG debug port 1101
31.8 SW debug port 1103
31.8.1 SW protocol introduction 1103
31.8.2 SW protocol sequence 1103
31.8.3 SW-DP state machine (reset, idle states, ID code) 1104
31.8.4 DP and AP read/write accesses 1105
31.8.5 SW-DP registers 1105
31.8.6 SW-AP registers 1106

31.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP 1106
31.10 Core debug 1107
31.11 Capability of the debugger host to connect under system reset 1108
31.12 FPB (Flash patch breakpoint) 1108
31.13 DWT (data watchpoint trigger) 1108
31.14 ITM (instrumentation trace macrocell) 1109
31.14.1 General description 1109
31.14.2 Time stamp packets, synchronization and overflow packets 1109
31.15 MCU debug component (DBGMCU) 1111
31.15.1 Debug support for low-power modes 1111
31.15.2 Debug support for timers, watchdog, bxCAN and I^2C 1111
31.15.3 Debug MCU configuration register 1111
31.15.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ) 1113
31.15.5 Debug MCU APB2 freeze register (DBGMCU_APB2_FZ) 1114
31.16 TPIU (trace port interface unit) 1115
31.16.1 Introduction 1115
31.17 DBG register map 1116

32 Device electronic signature 1117
32.1 Unique device ID register (96 bits) 1117
32.2 Flash memory size data register 1118

33 Revision history 1119

List of tables

Table 1. STM32F334xx peripheral register boundary addresses . . . 48
Table 2. CCM SRAM organization . . . 51
Table 3. Boot modes . . . 52
Table 4. Flash module organization . . . 54
Table 5. Flash memory read protection status . . . 63
Table 6. Access status versus protection level and execution modes . . . 64
Table 7. Flash interrupt request . . . 65
Table 8. Flash interface - register map and reset values . . . 71
Table 9. Option byte format . . . 73
Table 10. Option byte organization . . . 73
Table 11. Description of the option bytes . . . 74
Table 12. CRC internal input/output signals . . . 77
Table 13. CRC register map and reset values . . . 81
Table 14. Low-power mode summary . . . 86
Table 15. Sleep-now . . . 88
Table 16. Sleep-on-exit . . . 88
Table 17. Stop mode . . . 90
Table 18. Standby mode . . . 91
Table 19. PWR register map and reset values . . . 95
Table 20. STM32F334 peripherals interconnect matrix . . . 96
Table 21. Comparator outputs to timer inputs . . . 99
Table 22. Timer output selection as comparator blanking source . . . 99
Table 23. DAC output selection as comparator inverting input . . . 100
Table 24. Timer synchronization . . . 101
Table 25. Timer and EXTI signals triggering DAC conversions . . . 102
Table 26. RCC register map and reset values . . . 137
Table 27. Port bit configuration table . . . 141
Table 28. GPIO register map and reset values . . . 155
Table 29. SYSCFG register map and reset values . . . 168
Table 30. DMA implementation . . . 171
Table 31. DMA requests for each channel . . . 173
Table 32. Programmable data width and endian behavior (when PINC = MINC = 1) . . . 180
Table 33. DMA interrupt requests . . . 182
Table 34. DMA register map and reset values . . . 190
Table 35. STM32F334xx vector table . . . 193
Table 36. External interrupt/event controller register map and reset values . . . 208
Table 37. ADC internal signals . . . 215
Table 38. ADC pins . . . 215
Table 39. Configuring the trigger polarity for regular external triggers . . . 230
Table 40. ADC1 (master) & 2 (slave) - External triggers for regular channels . . . 231
Table 41. ADC1 & ADC2 - External trigger for injected channels . . . 232
Table 42. TSAR timings depending on resolution . . . 244
Table 43. Offset computation versus data resolution . . . 247
Table 44. Analog watchdog channel selection . . . 257
Table 45. Analog watchdog 1 comparison . . . 258
Table 46. Analog watchdog 2 and 3 comparison . . . 258
Table 47. ADC interrupts per each ADC . . . 279
Table 48. DELAY bits versus ADC resolution . . . 312

Table 49. ADC global register map . . . 313
Table 50. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2) . . . 314
Table 51. ADC register map and reset values (master and slave ADC common registers) offset =0x300, x=1) . . . 316
Table 52. DACx pins . . . 319
Table 53. External triggers (DAC1) . . . 322
Table 54. External triggers (DAC2) . . . 322
Table 55. DAC register map and reset values . . . 341
Table 56. STM32F334xx comparator input/outputs summary . . . 345
Table 57. COMP register map and reset values . . . 352
Table 58. Connections with dedicated I/O . . . 353
Table 59. OPAMP register map and reset values . . . 363
Table 60. Acquisition sequence summary . . . 367
Table 61. Spread spectrum deviation versus AHB clock frequency . . . 369
Table 62. I/O state depending on its mode and IODEF bit value . . . 370
Table 63. Effect of low-power modes on TSC . . . 372
Table 64. Interrupt control bits . . . 372
Table 65. TSC register map and reset values . . . 381
Table 66. Behavior of timer outputs versus BRK/BRK2 inputs . . . 425
Table 67. Counting direction versus encoder signals . . . 432
Table 68. TIM1 internal trigger connection . . . 449
Table 69. Output control bits for complementary OCx and OCxN channels with break feature . . . 463
Table 70. TIM1 register map and reset values . . . 475
Table 71. Counting direction versus encoder signals . . . 512
Table 72. TIMx internal trigger connection . . . 530
Table 73. Output control bit for standard OCx channels . . . 540
Table 74. TIM2/TIM3 register map and reset values . . . 546
Table 75. TIMx Internal trigger connection . . . 589
Table 76. Output control bits for complementary OCx and OCxN channels with break feature (TIM15) . . . 599
Table 77. TIM15 register map and reset values . . . 605
Table 78. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . 618
Table 79. TIM16/TIM17 register map and reset values . . . 625
Table 80. HRTIM Input/output summary . . . 630
Table 81. Timer resolution and min. PWM frequency for f_{HRTIM} = 144 MHz . . . 632
Table 82. Period and Compare registers min and max values . . . 635
Table 83. Timer operating modes . . . 636
Table 84. Events mapping across Timer A to E . . . 642
Table 85. Deadtime resolution and max absolute values . . . 651
Table 86. External events mapping and associated features . . . 658
Table 87. Output set/reset latency and jitter vs external event operating mode . . . 659
Table 88. Filtering signals mapping per time . . . 662
Table 89. Windowing signals mapping per timer (EEFLTR[3:0] = 1111) . . . 664
Table 90. HRTIM preloadable control registers and associated update sources . . . 673
Table 91. Update enable inputs and sources . . . 674
Table 92. Master timer update event propagation . . . 676
Table 93. TIMx update event propagation . . . 676
Table 94. Reset events able to generate an update . . . 677
Table 95. Update event propagation for a timer reset . . . 678
Table 96. Output state programming, x= A..E, y = 1 or 2 . . . 679

Table 97. Timer output programming for burst mode . . . 682
Table 98. Burst mode clock sources from general purpose timer. . . . 684
Table 99. Fault inputs . . . 692
Table 100. Sampling rate and filter length vs FLTFxF[3:0] and clock setting . . . 693
Table 101. Effect of sync event vs timer operating modes . . . 698
Table 102. HRTIM interrupt summary . . . 704
Table 103. HRTIM DMA request summary. . . . 705
Table 104. RTIM global register map . . . 796
Table 105. HRTIM Register map and reset values: Master timer. . . . 796
Table 106. HRTIM Register map and reset values: TIMx (x= A..E) . . . 798
Table 107. HRTIM Register map and reset values: Common functions. . . . 802
Table 108. TIMx register map and reset values . . . 818
Table 109. IWDG register map and reset values . . . 827
Table 110. WWDG register map and reset values . . . 833
Table 111. RTC pin PC13 configuration . . . 838
Table 112. LSE pin PC14 configuration . . . 838
Table 113. LSE pin PC15 configuration . . . 838
Table 114. Effect of low-power modes on RTC . . . 850
Table 115. Interrupt control bits . . . 851
Table 116. RTC register map and reset values . . . 875
Table 117. STM32F334xx I2C implementation . . . 879
Table 118. I2C input/output pins. . . . 881
Table 119. I2C internal input/output signals . . . 881
Table 120. Comparison of analog vs. digital filters . . . 883
Table 121. I2C-SMBus specification data setup and hold times . . . 886
Table 122. I2C configuration. . . . 890
Table 123. I2C-SMBus specification clock timings . . . 901
Table 124. Examples of timing settings for fI2CCLK = 8 MHz . . . 911
Table 125. Examples of timings settings for fI2CCLK = 16 MHz . . . 911
Table 126. Examples of timings settings for fI2CCLK = 48 MHz . . . 912
Table 127. SMBus timeout specifications . . . 914
Table 128. SMBus with PEC configuration . . . 916
Table 129. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms) . . . 917
Table 130. Examples of TIMEOUTB settings for various I2CCLK frequencies . . . 917
Table 131. Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs) . . . 918
Table 132. Effect of low-power modes on the I2C . . . 929
Table 133. I2C Interrupt requests . . . 930
Table 134. I2C register map and reset values . . . 946
Table 135. STM32F334xx USART features . . . 950
Table 136. Noise detection from sampled data . . . 962
Table 137. Error calculation for programmed baud rates at f_{CK} = 72MHz in both cases of oversampling by 16 or by 8. . . . 965
Table 138. Tolerance of the USART receiver when BRR [3:0] = 0000. . . . 967
Table 139. Tolerance of the USART receiver when BRR [3:0] is different from 0000 . . . 967
Table 140. Frame formats . . . 971
Table 141. Effect of low-power modes on the USART . . . 990
Table 142. USART interrupt requests. . . . 990
Table 143. USART register map and reset values . . . 1013
Table 144. STM32F334xx SPI implementation . . . 1016
Table 145. SPI interrupt requests . . . 1040

Table 146. SPI register map and reset values . . . 1049
Table 147. Transmit mailbox mapping . . . 1064
Table 148. Receive mailbox mapping . . . 1064
Table 149. bxCAN register map and reset values . . . 1090
Table 150. SWJ debug port pins . . . 1097
Table 151. Flexible SWJ-DP pin assignment . . . 1097
Table 152. JTAG debug port data registers . . . 1101
Table 153. 32-bit debug port registers addressed through the shifted value A[3:2] . . . 1102
Table 154. Packet request (8-bits) . . . 1103
Table 155. ACK response (3 bits) . . . 1104
Table 156. DATA transfer (33 bits) . . . 1104
Table 157. SW-DP registers . . . 1105
Table 158. Cortex®-M4 AHB-AP registers . . . 1107
Table 159. Core debug registers . . . 1107
Table 160. Main ITM registers . . . 1110
Table 161. Flexible TRACE pin assignment . . . 1115
Table 162. DBG register map and reset values . . . 1116
Table 163. Document revision history . . . 1119

List of figures

Figure 1. System architecture . . . 45
Figure 2. Programming procedure . . . 58
Figure 3. Flash memory Page Erase procedure . . . 60
Figure 4. Flash memory Mass Erase procedure . . . 61
Figure 5. CRC calculation unit block diagram . . . 77
Figure 6. Power supply overview . . . 82
Figure 7. Power on reset/power down reset waveform . . . 85
Figure 8. PVD thresholds . . . 86
Figure 9. Simplified diagram of the reset circuit . . . 105
Figure 10. STM32F334xx clock tree . . . 107
Figure 11. HSE/ LSE clock sources . . . 108
Figure 12. Frequency measurement with TIM16 in capture mode . . . 114
Figure 13. Basic structure of an I/O port bit . . . 140
Figure 14. Basic structure of a 5-Volt tolerant I/O port bit . . . 140
Figure 15. Input floating/pull up/pull down configurations . . . 145
Figure 16. Output configuration . . . 146
Figure 17. Alternate function configuration . . . 147
Figure 18. High impedance-analog configuration . . . 147
Figure 19. DMA request mapping . . . 172
Figure 20. DMA block diagram . . . 174
Figure 21. External interrupt/event block diagram . . . 197
Figure 22. External interrupt/event GPIO mapping . . . 200
Figure 23. ADC block diagram . . . 214
Figure 24. ADC clock scheme . . . 216
Figure 25. ADC1 and ADC2 connectivity . . . 218
Figure 26. ADC calibration . . . 221
Figure 27. Updating the ADC calibration factor . . . 222
Figure 28. Mixing single-ended and differential channels . . . 222
Figure 29. Enabling / Disabling the ADC . . . 223
Figure 30. Analog to digital conversion time . . . 228
Figure 31. Stopping ongoing regular conversions . . . 229
Figure 32. Stopping ongoing regular and injected conversions . . . 230
Figure 33. Triggers are shared between ADC master & ADC slave . . . 231
Figure 34. Injected conversion latency . . . 234
Figure 35. Example of JSQR queue of context (sequence change) . . . 237
Figure 36. Example of JSQR queue of context (trigger change) . . . 237
Figure 37. Example of JSQR queue of context with overflow before conversion . . . 238
Figure 38. Example of JSQR queue of context with overflow during conversion . . . 238
Figure 39. Example of JSQR queue of context with empty queue (case JQM=0) . . . 239
Figure 40. Example of JSQR queue of context with empty queue (case JQM=1) . . . 240
Figure 41. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion. . . . 240
Figure 42. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs. . . . 241
Figure 43. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion . . . 241
Figure 44. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1) . . . 242

Figure 45. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0). 242
Figure 46. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1). 243
Figure 47. Example of JSQR queue of context when changing SW and HW triggers. 243
Figure 48. Single conversions of a sequence, software trigger 245
Figure 49. Continuous conversion of a sequence, software trigger. 245
Figure 50. Single conversions of a sequence, hardware trigger 246
Figure 51. Continuous conversions of a sequence, hardware trigger 246
Figure 52. Right alignment (offset disabled, unsigned value) 248
Figure 53. Right alignment (offset enabled, signed value). 248
Figure 54. Left alignment (offset disabled, unsigned value) 249
Figure 55. Left alignment (offset enabled, signed value). 249
Figure 56. Example of overrun (OVR) 250
Figure 57. AUTODLY=1, regular conversion in continuous mode, software trigger 253
Figure 58. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0) 254
Figure 59. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1) 255
Figure 60. AUTODLY=1, regular continuous conversions interrupted by injected conversions 256
Figure 61. AUTODLY=1 in auto- injected mode (JAUTO=1). 256
Figure 62. Analog watchdog's guarded area 257
Figure 63. ADCy_AWDx_OUT signal generation (on all regular channels). 259
Figure 64. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW) 260
Figure 65. ADCy_AWDx_OUT signal generation (on a single regular channel) 260
Figure 66. ADCy_AWDx_OUT signal generation (on all injected channels) 260
Figure 67. Dual ADC block diagram(1). 262
Figure 68. Injected simultaneous mode on 4 channels: dual ADC mode 263
Figure 69. Regular simultaneous mode on 16 channels: dual ADC mode 265
Figure 70. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode. 267
Figure 71. Interleaved mode on 1 channel in single conversion mode: dual ADC mode. 267
Figure 72. Interleaved conversion with injection 268
Figure 73. Alternate trigger: injected group of each ADC 269
Figure 74. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode. 270
Figure 75. Alternate + regular simultaneous 271
Figure 76. Case of trigger occurring during injected conversion 271
Figure 77. DMA Requests in regular simultaneous mode when MDMA=0b00 272
Figure 78. DMA requests in regular simultaneous mode when MDMA=0b10 273
Figure 79. DMA requests in interleaved mode when MDMA=0b10 273
Figure 80. Temperature sensor channel block diagram 275
Figure 81. VBAT channel block diagram 277
Figure 82. V_{REFINT} channel block diagram 277
Figure 83. DAC1 block diagram 318
Figure 84. DAC2 block diagram. 319
Figure 85. Data registers in single DAC channel mode 320
Figure 86. Timing diagram for conversion with trigger disabled TEN = 0 321
Figure 87. Data registers in dual DAC channel mode 323
Figure 88. DAC LFSR register calculation algorithm 328
Figure 89. DAC conversion (SW trigger enabled) with LFSR wave generation. 328
Figure 90. DAC triangle wave generation 329
Figure 91. DAC conversion (SW trigger enabled) with triangle wave generation 329
Figure 92. Comparator 2 block diagram 344
Figure 93. Comparator output blanking 346
Figure 94. STM32F334xx comparator and operational amplifier

connections . . . 354
Figure 95. Timer controlled Multiplexer mode . . . 356
Figure 96. Standalone mode: external gain setting mode . . . 357
Figure 97. Follower configuration . . . 358
Figure 98. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used . . . 359
Figure 99. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering . . . 359
Figure 100. TSC block diagram . . . 365
Figure 101. Surface charge transfer analog I/O group structure . . . 366
Figure 102. Sampling capacitor voltage variation . . . 367
Figure 103. Charge transfer acquisition sequence . . . 368
Figure 104. Spread spectrum variation principle . . . 369
Figure 105. Advanced-control timer block diagram . . . 385
Figure 106. Counter timing diagram with prescaler division change from 1 to 2 . . . 388
Figure 107. Counter timing diagram with prescaler division change from 1 to 4 . . . 388
Figure 108. Counter timing diagram, internal clock divided by 1 . . . 390
Figure 109. Counter timing diagram, internal clock divided by 2 . . . 390
Figure 110. Counter timing diagram, internal clock divided by 4 . . . 391
Figure 111. Counter timing diagram, internal clock divided by N . . . 391
Figure 112. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 392
Figure 113. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 392
Figure 114. Counter timing diagram, internal clock divided by 1 . . . 394
Figure 115. Counter timing diagram, internal clock divided by 2 . . . 394
Figure 116. Counter timing diagram, internal clock divided by 4 . . . 395
Figure 117. Counter timing diagram, internal clock divided by N . . . 395
Figure 118. Counter timing diagram, update event when repetition counter is not used . . . 396
Figure 119. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 397
Figure 120. Counter timing diagram, internal clock divided by 2 . . . 398
Figure 121. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 398
Figure 122. Counter timing diagram, internal clock divided by N . . . 399
Figure 123. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . 399
Figure 124. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 400
Figure 125. Update rate examples depending on mode and TIMx_RCR register settings . . . 401
Figure 126. External trigger input block . . . 402
Figure 127. Control circuit in normal mode, internal clock divided by 1 . . . 403
Figure 128. TI2 external clock connection example . . . 404
Figure 129. Control circuit in external clock mode 1 . . . 405
Figure 130. External trigger input block . . . 405
Figure 131. Control circuit in external clock mode 2 . . . 406
Figure 132. Capture/compare channel (example: channel 1 input stage) . . . 407
Figure 133. Capture/compare channel 1 main circuit . . . 408
Figure 134. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . 408
Figure 135. Output stage of capture/compare channel (channel 4) . . . 409
Figure 136. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . 409
Figure 137. PWM input mode timing . . . 411
Figure 138. Output compare mode, toggle on OC1 . . . 413
Figure 139. Edge-aligned PWM waveforms (ARR=8) . . . 414
Figure 140. Center-aligned PWM waveforms (ARR=8) . . . 415
Figure 141. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 417
Figure 142. Combined PWM mode on channel 1 and 3 . . . 418
Figure 143. 3-phase combined PWM signals with multiple trigger pulses per period . . . 419
Figure 144. Complementary output with dead-time insertion . . . 420

Figure 145. Dead-time waveforms with delay greater than the negative pulse 420
Figure 146. Dead-time waveforms with delay greater than the positive pulse. 421
Figure 147. Various output behavior in response to a break event on BKIN (OSSI = 1) 424
Figure 148. PWM output state following BKIN and BKIN2 pins assertion (OSSI=1) 425
Figure 149. PWM output state following BKIN assertion (OSSI=0) 426
Figure 150. Clearing TIMx OCxREF 427
Figure 151. 6-step generation, COM example (OSSR=1) 428
Figure 152. Example of one pulse mode. 429
Figure 153. Retriggerable one pulse mode 431
Figure 154. Example of counter operation in encoder interface mode. 432
Figure 155. Example of encoder interface mode with TI1FP1 polarity inverted. 433
Figure 156. Measuring time interval between edges on 3 signals 434
Figure 157. Example of Hall sensor interface 436
Figure 158. Control circuit in reset mode 437
Figure 159. Control circuit in Gated mode 438
Figure 160. Control circuit in trigger mode 439
Figure 161. Control circuit in external clock mode 2 + trigger mode 440
Figure 162. General-purpose timer block diagram 479
Figure 163. Counter timing diagram with prescaler division change from 1 to 2 481
Figure 164. Counter timing diagram with prescaler division change from 1 to 4 481
Figure 165. Counter timing diagram, internal clock divided by 1 482
Figure 166. Counter timing diagram, internal clock divided by 2 483
Figure 167. Counter timing diagram, internal clock divided by 4 483
Figure 168. Counter timing diagram, internal clock divided by N. 484
Figure 169. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded). 484
Figure 170. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded). 485
Figure 171. Counter timing diagram, internal clock divided by 1 486
Figure 172. Counter timing diagram, internal clock divided by 2 486
Figure 173. Counter timing diagram, internal clock divided by 4 487
Figure 174. Counter timing diagram, internal clock divided by N. 487
Figure 175. Counter timing diagram, Update event when repetition counter is not used 488
Figure 176. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 489
Figure 177. Counter timing diagram, internal clock divided by 2 490
Figure 178. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 490
Figure 179. Counter timing diagram, internal clock divided by N. 491
Figure 180. Counter timing diagram, Update event with ARPE=1 (counter underflow). 491
Figure 181. Counter timing diagram, Update event with ARPE=1 (counter overflow) 492
Figure 182. Control circuit in normal mode, internal clock divided by 1 493
Figure 183. TI2 external clock connection example. 493
Figure 184. Control circuit in external clock mode 1 494
Figure 185. External trigger input block 495
Figure 186. Control circuit in external clock mode 2 496
Figure 187. Capture/Compare channel (example: channel 1 input stage) 497
Figure 188. Capture/Compare channel 1 main circuit 497
Figure 189. Output stage of Capture/Compare channel (channel 1) 498
Figure 190. PWM input mode timing 500
Figure 191. Output compare mode, toggle on OC1. 502
Figure 192. Edge-aligned PWM waveforms (ARR=8) 503
Figure 193. Center-aligned PWM waveforms (ARR=8). 504
Figure 194. Generation of 2 phase-shifted PWM signals with 50% duty cycle 505
Figure 195. Combined PWM mode on channels 1 and 3 507

Figure 196. Clearing TIMx OCxREF . . . 508
Figure 197. Example of one-pulse mode. . . . 509
Figure 198. Retriggerable one-pulse mode. . . . 511
Figure 199. Example of counter operation in encoder interface mode . . . 512
Figure 200. Example of encoder interface mode with TI1FP1 polarity inverted . . . 513
Figure 201. Control circuit in reset mode . . . 514
Figure 202. Control circuit in gated mode . . . 515
Figure 203. Control circuit in trigger mode . . . 516
Figure 204. Control circuit in external clock mode 2 + trigger mode . . . 517
Figure 205. Master/Slave timer example . . . 518
Figure 206. Master/slave connection example with 1 channel only timers . . . 518
Figure 207. Gating TIM2 with OC1REF of TIM3 . . . 519
Figure 208. Gating TIM2 with Enable of TIM3 . . . 520
Figure 209. Triggering TIM2 with update of TIM3 . . . 521
Figure 210. Triggering TIM2 with Enable of TIM3 . . . 521
Figure 211. Triggering TIM3 and TIM2 with TIM3 TI1 input. . . . 522
Figure 212. TIM15 block diagram . . . 550
Figure 213. TIM16/TIM17 block diagram . . . 551
Figure 214. Counter timing diagram with prescaler division change from 1 to 2 . . . 553
Figure 215. Counter timing diagram with prescaler division change from 1 to 4 . . . 553
Figure 216. Counter timing diagram, internal clock divided by 1 . . . 555
Figure 217. Counter timing diagram, internal clock divided by 2 . . . 555
Figure 218. Counter timing diagram, internal clock divided by 4 . . . 556
Figure 219. Counter timing diagram, internal clock divided by N. . . . 556
Figure 220. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded). . . . 557
Figure 221. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). . . . 557
Figure 222. Update rate examples depending on mode and TIMx_RCR register settings . . . 559
Figure 223. Control circuit in normal mode, internal clock divided by 1 . . . 560
Figure 224. TI2 external clock connection example. . . . 560
Figure 225. Control circuit in external clock mode 1 . . . 561
Figure 226. Capture/compare channel (example: channel 1 input stage) . . . 562
Figure 227. Capture/compare channel 1 main circuit . . . 562
Figure 228. Output stage of capture/compare channel (channel 1). . . . 563
Figure 229. Output stage of capture/compare channel (channel 2 for TIM15) . . . 563
Figure 230. PWM input mode timing . . . 565
Figure 231. Output compare mode, toggle on OC1 . . . 567
Figure 232. Edge-aligned PWM waveforms (ARR=8) . . . 568
Figure 233. Combined PWM mode on channel 1 and 2 . . . 569
Figure 234. Complementary output with dead-time insertion. . . . 570
Figure 235. Dead-time waveforms with delay greater than the negative pulse. . . . 571
Figure 236. Dead-time waveforms with delay greater than the positive pulse. . . . 571
Figure 237. Output behavior in response to a break . . . 574
Figure 238. Example of one pulse mode . . . 576
Figure 239. Retriggerable one pulse mode . . . 577
Figure 240. Measuring time interval between edges on 2 signals . . . 579
Figure 241. Control circuit in reset mode . . . 580
Figure 242. Control circuit in gated mode . . . 581
Figure 243. Control circuit in trigger mode . . . 582
Figure 244. High-resolution timer block diagram . . . 630
Figure 245. Counter and capture register format vs clock prescaling factor . . . 633

Figure 246. Timer A..E overview . . . 635
Figure 247. Continuous timer operation . . . 637
Figure 248. Single-shot timer operation . . . 637
Figure 249. Timer reset resynchronization (prescaling ratio above 32) . . . 639
Figure 250. Repetition rate vs HRTIM_REPxR content in continuous mode . . . 640
Figure 251. Repetition counter behavior in single-shot mode . . . 640
Figure 252. Compare events action on outputs: set on compare 1, reset on compare 2 . . . 642
Figure 253. Timing unit capture circuitry . . . 644
Figure 254. Auto-delayed overview (Compare 2 only) . . . 645
Figure 255. Auto-delayed compare . . . 646
Figure 256. Push-pull mode block diagram . . . 648
Figure 257. Push-pull mode example . . . 649
Figure 258. Complementary outputs with deadtime insertion . . . 649
Figure 259. Deadtime insertion vs deadtime sign (1 indicates negative deadtime) . . . 650
Figure 260. Complementary outputs for low pulse width (SDTRx = SDTFx = 0) . . . 651
Figure 261. Complementary outputs for low pulse width (SDTRx = SDTFx = 1) . . . 651
Figure 262. Complementary outputs for low pulse width (SDTRx = 0, SDTFx = 1) . . . 652
Figure 263. Complementary outputs for low pulse width (SDTRx = 1, SDTFx=0) . . . 652
Figure 264. Master timer overview . . . 653
Figure 265. Short distance set/reset management for narrow pulse generation . . . 655
Figure 266. External event conditioning overview (1 channel represented) . . . 657
Figure 267. Latency to external events falling edge (counter reset and output set) . . . 660
Figure 268. Latency to external events (output reset on external event) . . . 661
Figure 269. Event blanking mode . . . 661
Figure 270. Event postpone mode . . . 662
Figure 271. External trigger blanking with edge-sensitive trigger . . . 663
Figure 272. External trigger blanking, level sensitive triggering . . . 663
Figure 273. Event windowing mode . . . 664
Figure 274. External trigger windowing with edge-sensitive trigger . . . 665
Figure 275. External trigger windowing, level sensitive triggering . . . 665
Figure 276. Delayed Idle mode entry . . . 667
Figure 277. Burst mode and delayed protection priorities (DIDL = 0) . . . 668
Figure 278. Burst mode and delayed protection priorities (DIDL = 1) . . . 669
Figure 279. Balanced Idle protection example . . . 670
Figure 280. Output management overview . . . 680
Figure 281. HRTIM output states and transitions . . . 680
Figure 282. Burst mode operation example . . . 682
Figure 283. Burst mode trigger on external event . . . 684
Figure 284. Delayed burst mode entry with deadtime enabled and IDLESx = 1 . . . 686
Figure 285. Delayed Burst mode entry during deadtime . . . 687
Figure 286. Burst mode exit when the deadtime generator is enabled . . . 688
Figure 287. Burst mode emulation example . . . 690
Figure 288. Carrier frequency signal insertion . . . 690
Figure 289. HRTIM outputs with Chopper mode enabled . . . 691
Figure 290. Fault protection circuitry (FAULT1 fully represented, FAULT2..5 partially) . . . 692
Figure 291. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING}$ = fHRTIM, N = 4) . . . 693
Figure 292. Auxiliary outputs . . . 695
Figure 293. Auxiliary and main outputs during burst mode (DIDLx = 0) . . . 696
Figure 294. Deadtime distortion on auxiliary output when exiting burst mode . . . 696
Figure 295. Counter behavior in synchronized start mode . . . 700
Figure 296. ADC trigger selection overview . . . 701
Figure 297. Combining several updates on a single DACtrigOutx output . . . 702

Figure 298. DMA burst overview . . . 706
Figure 299. Burst DMA operation flowchart . . . 707
Figure 300. Registers update following DMA burst transfer . . . 708
Figure 301. Buck converter topology . . . 711
Figure 302. Dual Buck converter management . . . 711
Figure 303. Synchronous rectification depending on output current . . . 712
Figure 304. Buck with synchronous rectification . . . 712
Figure 305. 3-phase interleaved buck converter . . . 713
Figure 306. 3-phase interleaved buck converter control . . . 714
Figure 307. Transition mode PFC . . . 714
Figure 308. Transition mode PFC waveforms . . . 715
Figure 309. IRTIM internal hardware connections with TIM16 and TIM17 . . . 805
Figure 310. Basic timer block diagram . . . 806
Figure 311. Counter timing diagram with prescaler division change from 1 to 2 . . . 808
Figure 312. Counter timing diagram with prescaler division change from 1 to 4 . . . 808
Figure 313. Counter timing diagram, internal clock divided by 1 . . . 809
Figure 314. Counter timing diagram, internal clock divided by 2 . . . 810
Figure 315. Counter timing diagram, internal clock divided by 4 . . . 810
Figure 316. Counter timing diagram, internal clock divided by N . . . 811
Figure 317. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 811
Figure 318. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 812
Figure 319. Control circuit in normal mode, internal clock divided by 1 . . . 813
Figure 320. Independent watchdog block diagram . . . 819
Figure 321. Watchdog block diagram . . . 829
Figure 322. Window watchdog timing diagram . . . 830
Figure 323. RTC block diagram . . . 836
Figure 324. I2C block diagram . . . 880
Figure 325. I2C bus protocol . . . 882
Figure 326. Setup and hold timings . . . 884
Figure 327. I2C initialization flowchart . . . 887
Figure 328. Data reception . . . 888
Figure 329. Data transmission . . . 889
Figure 330. Slave initialization flowchart . . . 892
Figure 331. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0 . . . 894
Figure 332. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1 . . . 895
Figure 333. Transfer bus diagrams for I2C slave transmitter . . . 896
Figure 334. Transfer sequence flowchart for slave receiver with NOSTRETCH=0 . . . 897
Figure 335. Transfer sequence flowchart for slave receiver with NOSTRETCH=1 . . . 898
Figure 336. Transfer bus diagrams for I2C slave receiver . . . 898
Figure 337. Master clock generation . . . 900
Figure 338. Master initialization flowchart . . . 902
Figure 339. 10-bit address read access with HEAD10R=0 . . . 902
Figure 340. 10-bit address read access with HEAD10R=1 . . . 903
Figure 341. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes . . . 904
Figure 342. Transfer sequence flowchart for I2C master transmitter for N>255 bytes . . . 905
Figure 343. Transfer bus diagrams for I2C master transmitter . . . 906
Figure 344. Transfer sequence flowchart for I2C master receiver for N≤255 bytes . . . 908
Figure 345. Transfer sequence flowchart for I2C master receiver for N >255 bytes . . . 909

Figure 346. Transfer bus diagrams for I2C master receiver . . . 910
Figure 347. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 915
Figure 348. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC . . . 919
Figure 349. Transfer bus diagrams for SMBus slave transmitter (SBC=1) . . . 919
Figure 350. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC . . . 921
Figure 351. Bus transfer diagrams for SMBus slave receiver (SBC=1) . . . 922
Figure 352. Bus transfer diagrams for SMBus master transmitter . . . 923
Figure 353. Bus transfer diagrams for SMBus master receiver . . . 925
Figure 354. USART block diagram . . . 952
Figure 355. Word length programming . . . 954
Figure 356. Configurable stop bits . . . 956
Figure 357. TC/TXE behavior when transmitting . . . 957
Figure 358. Start bit detection when oversampling by 16 or 8 . . . 958
Figure 359. Data sampling when oversampling by 16 . . . 962
Figure 360. Data sampling when oversampling by 8 . . . 962
Figure 361. Mute mode using Idle line detection . . . 969
Figure 362. Mute mode using address mark detection . . . 970
Figure 363. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 973
Figure 364. Break detection in LIN mode vs. Framing error detection . . . 974
Figure 365. USART example of synchronous transmission . . . 975
Figure 366. USART data clock timing diagram (M bits = 00) . . . 975
Figure 367. USART data clock timing diagram (M bits = 01) . . . 976
Figure 368. RX data setup/hold time . . . 976
Figure 369. ISO 7816-3 asynchronous protocol . . . 978
Figure 370. Parity error detection using the 1.5 stop bits . . . 979
Figure 371. IrDA SIR ENDEC- block diagram . . . 983
Figure 372. IrDA data modulation (3/16) -Normal Mode . . . 984
Figure 373. Transmission using DMA . . . 985
Figure 374. Reception using DMA . . . 986
Figure 375. Hardware flow control between 2 USARTs . . . 986
Figure 376. RS232 RTS flow control . . . 987
Figure 377. RS232 CTS flow control . . . 988
Figure 378. USART interrupt mapping diagram . . . 991
Figure 379. SPI block diagram . . . 1016
Figure 380. Full-duplex single master/ single slave application . . . 1017
Figure 381. Half-duplex single master/ single slave application . . . 1018
Figure 382. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . 1019
Figure 383. Master and three independent slaves . . . 1020
Figure 384. Multi-master application . . . 1021
Figure 385. Hardware/software slave select management . . . 1022
Figure 386. Data clock timing diagram . . . 1023
Figure 387. Data alignment when data length is not equal to 8-bit or 16-bit . . . 1024
Figure 388. Packing data in FIFO for transmission and reception . . . 1028
Figure 389. Master full-duplex communication . . . 1031
Figure 390. Slave full-duplex communication . . . 1032
Figure 391. Master full-duplex communication with CRC . . . 1033
Figure 392. Master full-duplex communication in packed mode . . . 1034
Figure 393. NSSP pulse generation in Motorola SPI master mode . . . 1037
Figure 394. TI mode transfer . . . 1038
Figure 395. CAN network topology . . . 1051
Figure 396. Single-CAN block diagram . . . 1052

Figure 397. bxCAN operating modes . . . 1054
Figure 398. bxCAN in silent mode . . . 1055
Figure 399. bxCAN in loop back mode . . . 1055
Figure 400. bxCAN in combined mode . . . 1056
Figure 401. Transmit mailbox states . . . 1057
Figure 402. Receive FIFO states . . . 1058
Figure 403. Filter bank scale configuration - register organization . . . 1061
Figure 404. Example of filter numbering . . . 1062
Figure 405. Filtering mechanism - example . . . 1063
Figure 406. CAN error state diagram . . . 1064
Figure 407. Bit timing . . . 1066
Figure 408. CAN frames . . . 1067
Figure 409. Event flags and interrupt generation . . . 1068
Figure 410. CAN mailbox registers . . . 1079
Figure 411. Block diagram of STM32 MCU and Cortex®-M4-level debug support . . . 1094
Figure 412. SWJ debug port . . . 1096
Figure 413. JTAG TAP connections . . . 1100

1 Documentation conventions

1.1 General information

The STM32F334xx devices have an Arm®(a) Cortex®-M4 core.

arm

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

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

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

b. This is an exhaustive list of all abbreviations applicable to STM microcontrollers, some of them may not be used in the current document.

1.3 Glossary

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

- The Arm® Cortex®-M4 core integrates one debug port: **SWD debug port (SWD-DP)** provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol. Refer to the Cortex®-M4 technical reference manual.
- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **OBL**: option byte loader.
- **AHB**: advanced high-performance bus.

1.4 Availability of peripherals

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

2 System and memory overview

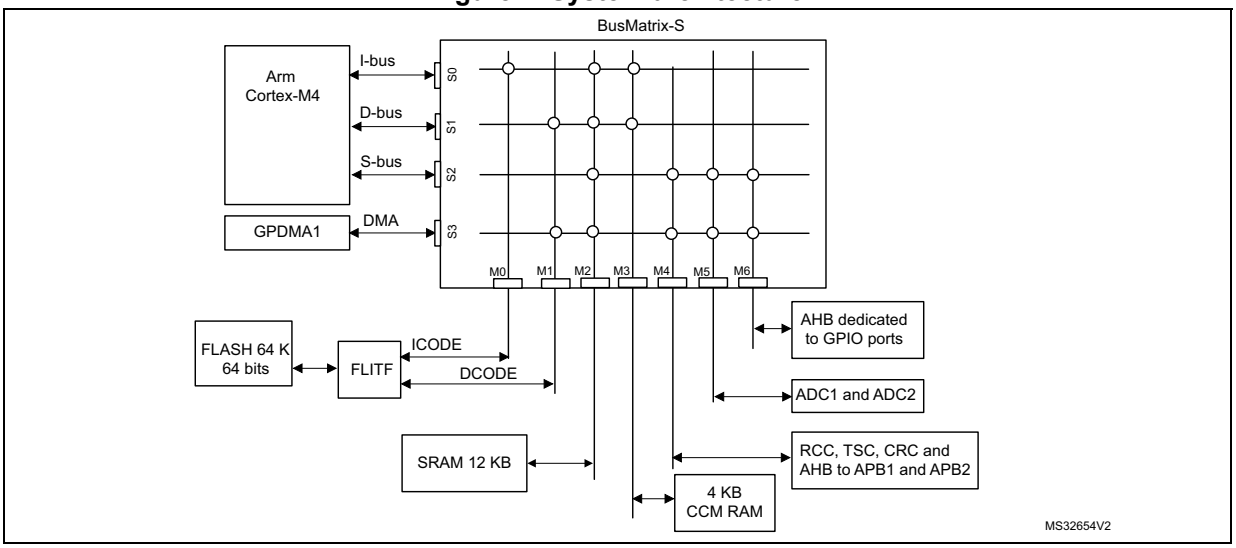
2.1 System architecture

The STM32F334xx main system consists of:

- Four masters:
  - Cortex®-M4 core I-bus
  - Cortex®-M4 core D-bus
  - Cortex®-M4 core S-bus
  - DMA1 (general-purpose DMA)
- Seven slaves:
  - Internal Flash memory on the DCode
  - Internal Flash memory on ICode
  - Up to Internal 12-Kbyte SRAM
  - Internal 4-Kbyte CCM SRAM
  - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
  - AHB dedicated to GPIO ports
  - ADCs 1 and 2

These are interconnected using a multilayer AHB bus architecture as shown in *Figure 1*:

Figure 1. System architecture



2.1.1 S0: I-bus

This bus connects the Instruction bus of the Cortex®-M4 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory, the SRAM up to 16 Kbytes and the CCM SRAM (4 Kbytes).

2.1.2 S1: D-bus

This bus connects the DCode bus (literal load and debug access) of the Cortex®-M4 core to the BusMatrix. The targets of this bus are the internal Flash memory, the SRAM (16 Kbytes) and the CCM SRAM (4 Kbytes).

2.1.3 S2: S-bus

This bus connects the system bus of the Cortex®-M4 core to the BusMatrix. This bus is used to access data located in the peripheral or SRAM area. The targets of this bus are the SRAM (16 Kbytes), the AHB to APB1/APB2 bridges, the AHB IO port and the 2 ADCs.

2.1.4 S3: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of different Masters to Flash, SRAM (16 Kbytes) and peripherals.

2.1.5 BusMatrix

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

AHB/APB bridges

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

Refer to *Section 2.2.2: Memory map and register boundary addresses on page 48* for the address mapping of the peripherals connected to this bridge.

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

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

2.2 Memory organization

2.2.1 Introduction

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

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

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

2.2.2 Memory map and register boundary addresses

Refer to the device datasheet for a comprehensive diagram of the memory map.

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

Table 1. STM32F334xx peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5000 0000 - 0x5000 03FF	1 K	ADC1 - ADC2	*Section 13.7 on page 313*
-	0x4800 1800 - 0x4FFF FFFF	~132 M	Reserved	-
AHB2	0x4800 1400 - 0x4800 17FF	1 K	GPIOF	*Section 9.4.12 on page 155*
	0x4800 1000 - 0x4800 13FF	1 K	Reserved	
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
-	0x4002 4400 - 0x47FF FFFF	~128 M	Reserved	
AHB1	0x4002 4000 - 0x4002 43FF	1 K	TSC	*Section 17.6.11 on page 381*
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 K	CRC	*Section 5.4.6 on page 81*
	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 K	Flash interface	*Section 3.6 on page 71*
	0x4002 1400 - 0x4002 1FFF	3 K	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 K	RCC	*Section 8.4.14 on page 137*
	0x4002 0400 - 0x4002 0FFF	3 K	Reserved	-
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	*Section 11.6.7 on page 190*
-	0x4001 8000 - 0x4001 FFFF	32 K	Reserved	-

Table 1. STM32F334xx peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 7400 - 0x4001 77FF	1 K	HRTIM1	*Section 21.5.64 on page 796*
	0x4001 4C00 - 0x4001 73FF	12 K	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*Section 20.6.18 on page 625*
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x4001 4000 - 0x4001 43FF	1 K	TIM15	*Section 20.5.19 on page 605*
	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*Section 25.7.12 on page 708*
	0x4001 3400 - 0x4001 37FF	1 K	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	*Section 29.6.8 on page 1049*
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	*Section 18.4.27 on page 475*
	0x4001 0800 - 0x4001 2BFF	9 K	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 K	EXTI	*Section 12.3.13 on page 208*
	0x4001 0000 - 0x4001 03FF	1 K	SYSCFG + COMP + OPAMP	*Section 10.1.9 on page 168*, *Section 15.5.4 on page 352*, *Section 16.4.2 on page 363*
-	0x4000 9C00 - 0x4000 FFFF	25 K	Reserved	-

Table 1. STM32F334xx peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 9800 - 0x4000 9BFF	1 K	DAC2	*Section 14.10.15 on page 341*
	0x4000 7800 - 0x4000 97FF	8 K	Reserved	-
	0x4000 7400 - 0x4000 77FF	1 K	DAC1	*Section 14.10.15 on page 341*
	0x4000 7000 - 0x4000 73FF	1 K	PWR	*Section 6.4.3 on page 95*
	0x4000 6800 - 0x4000 6FFF	2 K	Reserved	-
	0x4000 6400 - 0x4000 67FF	1 K	bxCAN	*Section 30.9.5 on page 1090*
	0x4000 5800 - 0x4000 63FF	3 K	Reserved	-
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	*Section 27.7.12 on page 946*
	0x4000 4C00 - 0x4000 53FF	2 K	Reserved	-
	0x4000 4800 - 0x4000 4BFF	1 K	USART3	*Section 25.7.12 on page 708*
	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x4000 3400 - 0x4000 43FF	4 K	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*Section 24.4.6 on page 827*
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	*Section 25.5.4 on page 833*
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	*Section 26.6.20 on page 875*
	0x4000 1800 - 0x4000 27FF	4 K	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 K	TIM7	*Section 23.4.9 on page 818*
	0x4000 1000 - 0x4000 13FF	1 K	TIM6	*Section 23.4.9 on page 818*
	0x4000 0800 - 0x4000 0FFF	2 K	Reserved	-
	0x4000 0400 - 0x4000 07FF	1 K	TIM3	*Section 19.4.22 on page 546*
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	
-	0x2000 A000 - 3FFF FFFF	~512 M	Reserved	-
-	0x2000 0000 - 0x2000 2FFF	12 K	SRAM	-
-	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	-
-	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	-
-	0x1000 2000 - 0x1FFF D7FF	~256 M	Reserved	-
-	0x1000 0000 - 0x1000 0FFF	4 K	CCM SRAM	-
-	0x0804 0000 - 0x0FFF FFFF	~128 M	Reserved	-
-	0x0800 0000 - 0x0800 FFFF	64 K	Main Flash memory	-
-	0x0004 0000 - 0x07FF FFFF	~128 M	Reserved	-
-	0x0000 000 - 0x0000 FFFF	64 K	Main Flash memory, system memory or SRAM depending on BOOT configuration	-

2.2.3 Parity check

The parity check is implemented on all of the SRAM and CCM SRAM. The SRAM parity check is disabled by default. It is enabled by the user, when needed, using an option bit.

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

The parity bits are computed on data and address and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated if the SRAM parity check is enabled. The same error can also be linked to the Break input of TIMER 1, 8, 15, 16 and 17, by setting the SRAM_PARITY_LOCK control bit in the SYSCFG configuration register 2 (SYSCFG_CFGR2). In case of parity error, the SRAM Parity Error flag (SRAM_PEF) is set in SYSCFG_CFGR2. For more details, refer to SYSCFG_CFGR2.

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

2.2.4 CCM SRAM write protection

The CCM SRAM is write protected with a page granularity of 1 Kbyte.

Table 2. CCM SRAM organization

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF

The write protection can be enabled in the CCM SRAM protection register (SYSCFG_RCR) in the SYSCFG block. This is a register with write 1 once mechanism, which means by writing 1 on a bit, it sets up the write protection for that page of SRAM and it can be removed/cleared by a system reset only. For more details, refer to the SYSCFG section.

2.3 Embedded SRAM

STM32F334xx devices feature up to 16 Kbytes of static SRAM. It can be accessed as bytes, halfwords (16 bits) or full words (32 bits):

- Up to 12 Kbytes of SRAM that can be addressed at maximum system clock frequency
- without wait states and can be accessed by both CPU and DMA;
- 4 Kbytes of CCM SRAM. It is used to execute critical routines or to access data. It can be accessed by the CPU only. No DMA accesses are allowed. This memory can be addressed at maximum system clock frequency without wait state.

2.4 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of two parts:
  - Option bytes for hardware and memory protection user configuration
  - System memory which contains the proprietary boot loader code. Refer to *Section 3: Embedded Flash memory* for more details.

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

2.5 Boot configuration

In the STM32F334xx, three different boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the user option byte, as shown in the following table:

Table 3. Boot modes

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

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

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

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

2.5.1 Embedded boot loader

The embedded boot loader is located in the system memory, programmed by ST during production. It is used to reprogram the Flash memory through USART1(PA9/PA10), USART2(PA2/PA3) or I2C1(PB6/PB7).

3 Embedded Flash memory

3.1 Flash main features

Up to 64 Kbytes of Flash memory

- Memory organization:
  - Main memory block:
    8 Kbits × 64 bits
  - Information block:
    1280 × 64 bits

Flash memory interface (FLITF) features:

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

3.2 Flash memory functional description

3.2.1 Flash memory organization

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

The memory organization is based on a main memory block containing 32 pages of 2 Kbytes and an information block as shown in *Table 4*.

Table 4. Flash module organization(1)

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	0x0800 F800-0x0800 FFFF	2 K	Page 31

Table 4. Flash module organization(1) (continued)

Flash area	Flash memory addresses	Size (bytes)	Name
Information block	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory
	0x1FFF F800 - 0x1FFF F80F	16	Option bytes
Flash memory interface registers	0x4002 2000 - 0x4002 2003	4	FLASH_ACR
	0x4002 2004 - 0x4002 2007	4	FLASH_KEYR
	0x4002 2008 - 0x4002 200B	4	FLASH_OPTKEYR
	0x4002 200C - 0x4002 200F	4	FLASH_SR
	0x4002 2010 - 0x4002 2013	4	FLASH_CR
	0x4002 2014 - 0x4002 2017	4	FLASH_AR
	0x4002 2018 - 0x4002 201B	4	Reserved
	0x4002 201C - 0x4002 201F	4	FLASH_OBR
	0x4002 2020 - 0x4002 2023	4	FLASH_WRPR

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

The information block is divided into two parts:

- System memory is used to boot the device in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the boot loader which is used to reprogram the Flash memory through one of the following interfaces: USART1, USART2 or I2C1. It is programmed by ST when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606 available from *www.st.com*.
- Option bytes

3.2.2 Read operations

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

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

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

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

Instruction fetch

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

Prefetch buffer

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

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

Prefetch controller

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

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

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

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

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

Access latency

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

DCode interface

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

Flash Access controller

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

DCode interface requests have priority over other requests.

3.2.3 Flash program and erase operations

The STM32F334xx embedded Flash memory can be programmed using in-circuit programming or in-application programming.

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

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

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

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

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

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

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

Unlocking the Flash memory

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

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

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

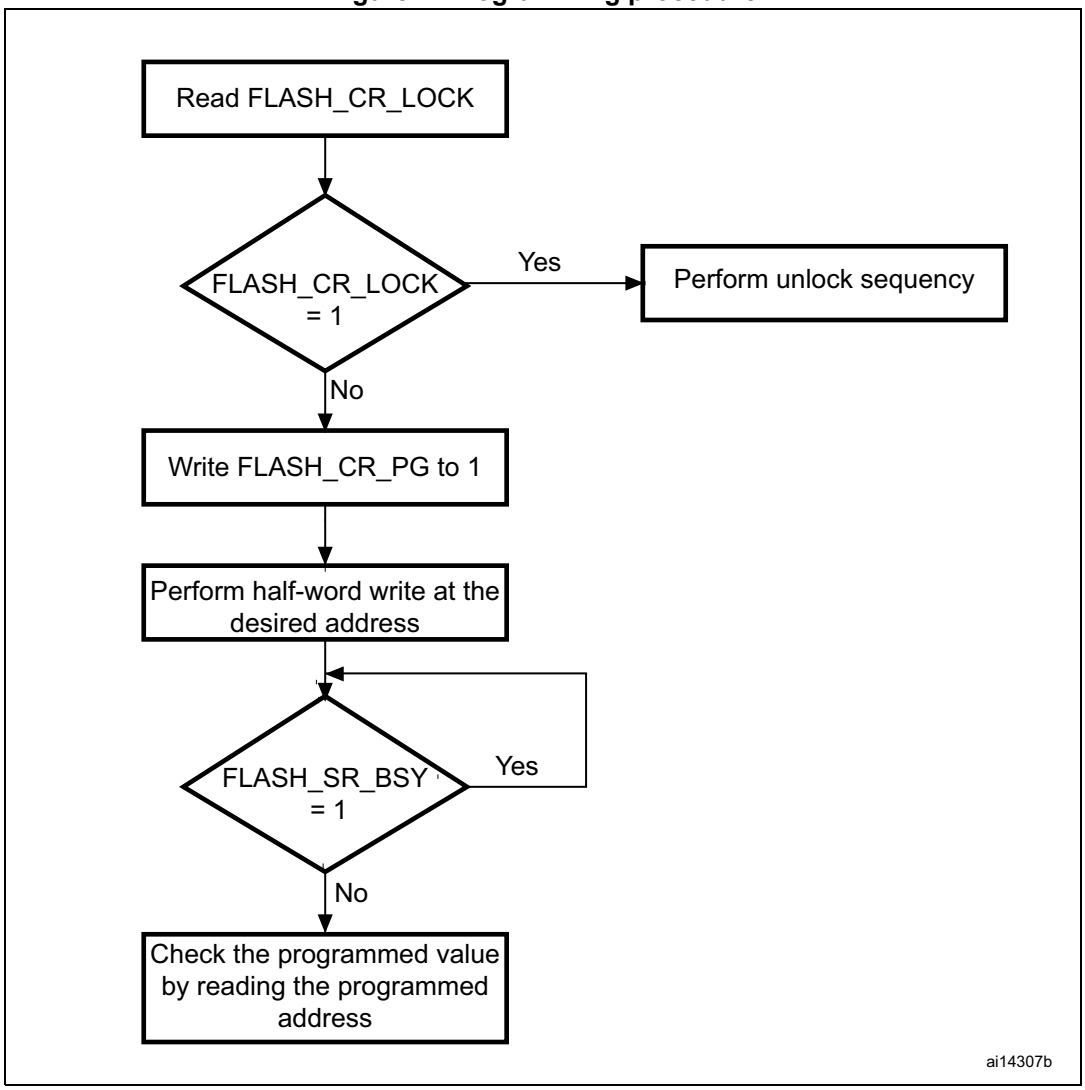
In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated. This is done after the first write cycle if KEY1 does not match, or during the second write cycle if KEY1 has been correctly written but KEY2 does not match.

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

Main Flash memory programming

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

Figure 2. Programming procedure



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

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

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

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

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

Flash memory erase

The Flash memory can be erased page by page or completely (mass erase).

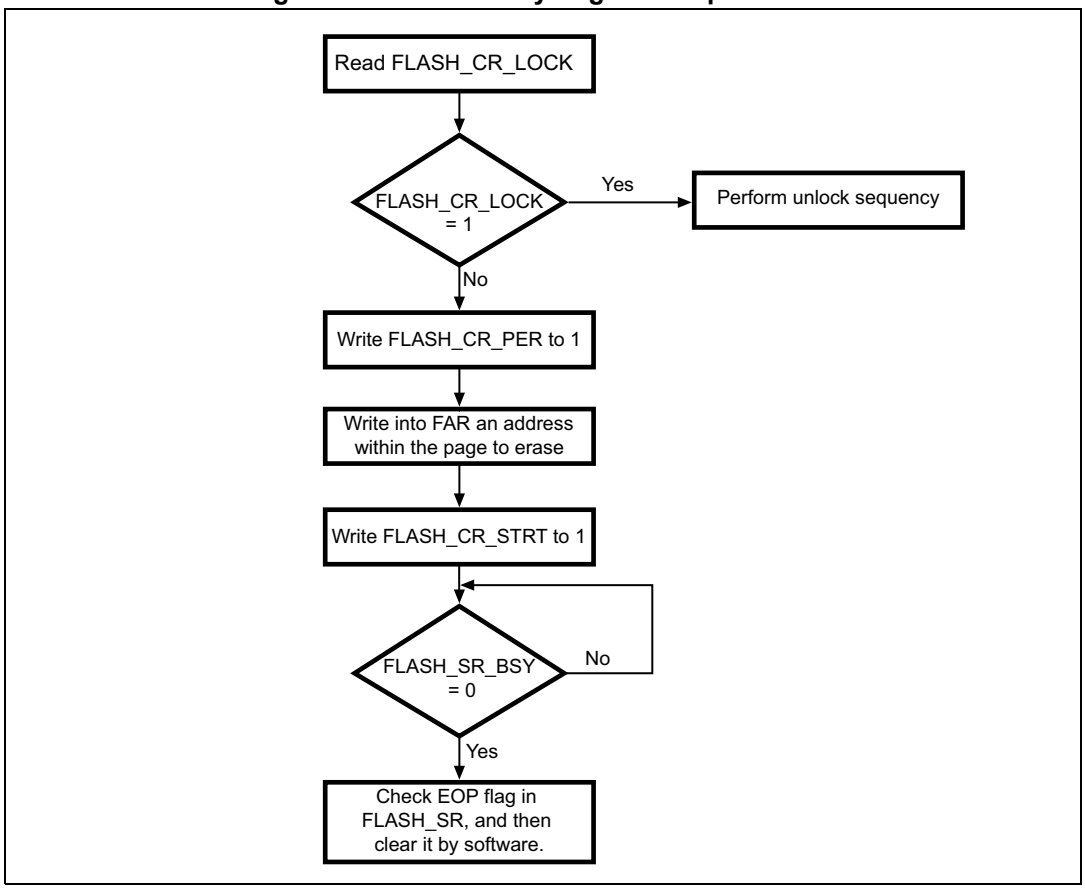
Page erase

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

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

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

Figure 3. Flash memory Page Erase procedure



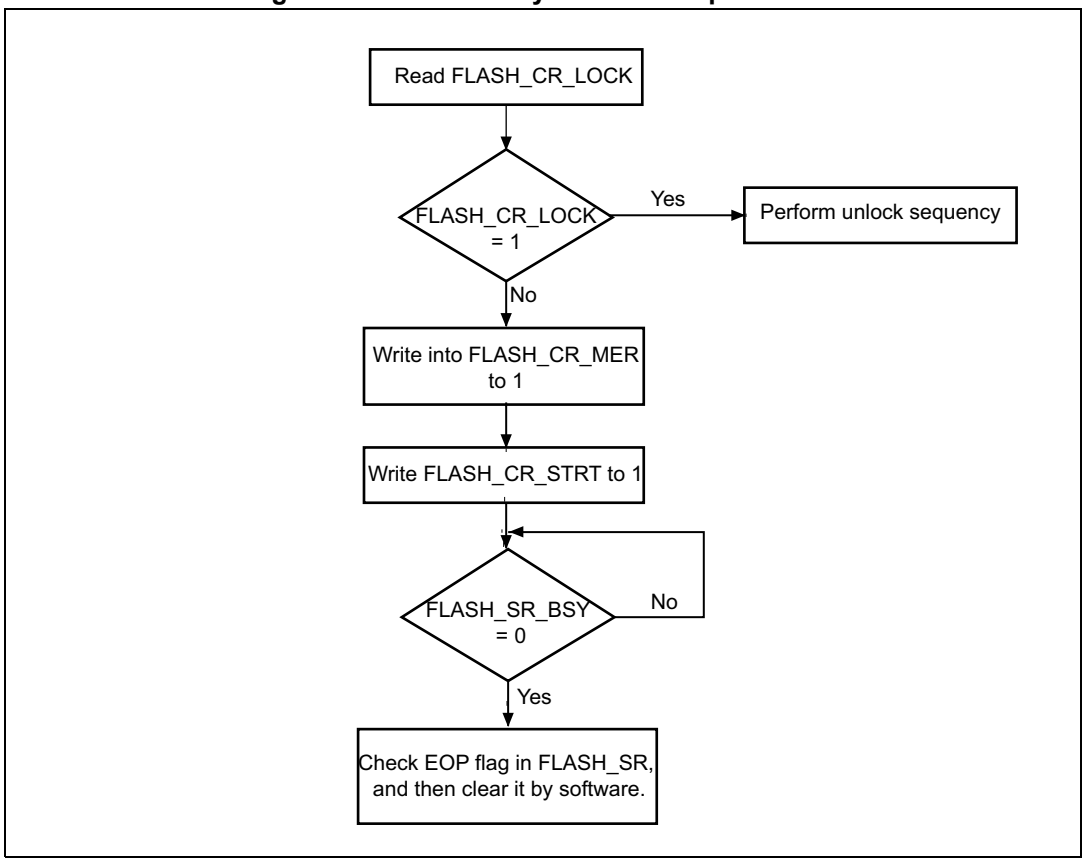
Mass erase

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

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

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

Figure 4. Flash memory Mass Erase procedure



Option byte programming

The option bytes are programmed differently from normal user addresses. The number of option bytes is limited to 6 (2 for write protection, 1 for readout protection, 1 for hardware configuration, and 2 for data storage). After unlocking the FPEC, the user has to authorize the programming of the option bytes by writing the same set of KEYS (KEY1 and KEY2) to the FLASH_OPTKEYR register (refer to *Unlocking the Flash memory* for key values). Then, the OPTWRE bit in the FLASH_CR register is set by hardware and the user has to set the OPTPG bit in the FLASH_CR register and perform a half-word write operation at the desired Flash address.

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

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

The sequence is as follows:

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

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

Erase procedure

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

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

3.3 Memory protection

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

3.3.1 Read protection

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

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

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

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in *Table 5*.

Table 5. Flash memory read protection status

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

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

Level 0: no protection

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

Level 1: Read protection

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

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

erase of main memory Flash is performed and the backup registers (RTC_BKPxR in the RTC) are reset.

Level 2: No debug

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

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

Note: The debug feature is also disabled under reset.

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

Table 6. Access status versus protection level and execution modes

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

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

Changing read protection level

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

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

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

3.3.2 Write protection

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

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

Write unprotection

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

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

3.3.3 Option byte block write protection

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

3.4 Flash interrupts

Table 7. Flash interrupt request

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

3.5 Flash register description

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

3.5.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0030

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

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

Bit 5 **PRFTBS**: Prefetch buffer status
This bit provides the status of the prefetch buffer.
0: Prefetch buffer is disabled
1: Prefetch buffer is enabled

Bit 4 **PRFTBE**: Prefetch buffer enable
0: Prefetch is disabled
1: Prefetch is enabled

Bit 3 **HLFCYA:** Flash half cycle access enable
0: Half cycle is disabled
1: Half cycle is enabled

Bits 2:0 **LATENCY[2:0]**: Latency
These bits represent the ratio of the HCLK period to the Flash access time.
000: Zero wait state, if 0 < HCLK ≤ 24 MHz
001: One wait state, if 24 MHz < HCLK ≤ 48 MHz
010: Two wait sates, if 48 < HCLK ≤ 72 MHz

3.5.2 Flash key register (FLASH_KEYR)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

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

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

These bits represent the keys to unlock the Flash.

3.5.3 Flash option key register (FLASH_OPTKEYR)

Address offset: 0x08

Reset value: 0xXXXX XXXX

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

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

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

These bits represent the keys to unlock the OPTWRE.

3.5.4 Flash status register (FLASH_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOP	WRPRT ERR	Res.	PG ERR	Res.	BSY
										rw	rw		rw		r

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

Bit 5 **EOP**: End of operation

Set by hardware when a Flash operation (programming / erase) is completed.
Reset by writing a 1

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

Bit 4 **WRPRTERR**: Write protection error

Set by hardware when programming a write-protected address of the Flash memory.
Reset by writing 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PGERR**: Programming error

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

Reset by writing 1.

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

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BSY**: Busy

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

3.5.5 Flash control register (FLASH_CR)

Address offset: 0x10

Reset value: 0x0000 0080

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

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

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

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

0: Inactive

1: Active

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

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

0: Interrupt generation disabled

1: Interrupt generation enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ERRIE**: Error interrupt enable

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

0: Interrupt generation disabled

1: Interrupt generation enabled

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

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

This bit can be reset by software

Bit 8 Reserved, must be kept at reset value.

Bit 7 **LOCK**: Lock

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

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

Bit 6 **STRT**: Start

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

Bit 5 **OPTER**: Option byte erase

Option byte erase chosen.

Bit 4 **OPTPG**: Option byte programming

Option byte programming chosen.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **MER**: Mass erase

Erase of all user pages chosen.

Bit 1 **PER**: Page erase

Page Erase chosen.

Bit 0 **PG**: Programming

Flash programming chosen.

3.5.6 Flash address register (FLASH_AR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

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

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

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

3.5.7 Option byte register (FLASH_OBR)

Address offset 0x1C

Reset value: 0xXXXX XX0X

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

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

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

Bits 31:24 Data1

Bits 23:16 Data0

Bits 15:8 **OBR:** User Option Byte
- Bit 15: Reserved, must be kept at reset value.
- Bit 14: SRAM_PE.
- Bit 13: VDDA_MONITOR
- Bit 12: nBOOT1
- Bit 11: Reserved, must be kept at reset value.
- Bit 10: nRST_STDBY
- Bit 9: nRST_STOP
- Bit 8: WDG_SW

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

Bits 2:1 **RDPRT[1:0]**: Read protection Level status
- 00: Read protection Level 0 is enabled (ST production set up)
- 01: Read protection Level 1 is enabled
- 10: Reserved
- 11: Read protection Level 2 is enabled

Note: These bits are read-only.

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

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

Note: This bit is read-only.

3.5.8 Write protection register (FLASH_WRPR)

Address offset: 0x20

Reset value: 0xFFFF FFFF

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

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

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

3.6 Flash register map

Table 8. Flash interface - register map and reset values

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

Table 8. Flash interface - register map and reset values (continued)

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

Refer to *Section 2.2 on page 47* for the register boundary addresses.

4 Option byte description

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

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

Table 9. Option byte format

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

The organization of these bytes inside the information block is as shown in *Table 10*.

The option bytes can be read from the memory locations listed in *Table 10* or from the Option byte register (FLASH_OBR).

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

Table 10. Option byte organization

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0

Table 11. Description of the option bytes

Flash memory address	Option bytes
0x1FFF F800	Bits [31:24]: **nUSER** Bits [23:16]: **USER:** User option byte (stored in FLASH_OBR[15:8]) This byte is used to configure the following features: - Select the watchdog event: Hardware or software - Reset event when entering Stop mode - Reset event when entering Standby mode Bit 23: Reserved Bit 22: **SRAM_PE** The SRAM hardware parity check is disabled by default. This bit allows the user to enable the SRAM hardware parity check. 0: Parity check enabled. 1: Parity check disabled. Bit 21: **VDDA_MONITOR** This bit selects the analog monitoring on the VDDA power source: 0: VDDA power supply supervisor disabled. 1: VDDA power supply supervisor enabled. Bit 20: **nBOOT1** Together with the BOOT0 pin, this bit selects Boot mode from the main Flash memory, SRAM or System memory. Refer to *Section 2.5 on page 52*. Bit 19: Reserved, must be kept at reset. Bit 18: **nRST_STDBY** 0: Reset generated when entering Standby mode. 1: No reset generated. Bit 17: **nRST_STOP** 0: Reset generated when entering Stop mode 1: No reset generated Bit 16: **WDG_SW** 0: Hardware watchdog 1: Software watchdog Bits [15:8]: **nRDP** Bits [7:0]: **RDP:** Read protection option byte The value of this byte defines the Flash memory protection level 0xAA: Level 0 0xXX (except 0xAA and 0xCC): Level 1 0xCC: Level 2 The protection levels are stored in the Flash_OBR Flash option bytes register (RDPRT bits).

Table 11. Description of the option bytes (continued)

Flash memory address	Option bytes
0x1FFF F804	**Datax**: Two bytes for user data storage. These addresses can be programmed using the option byte programming procedure. Bits [31:24]: **nData1** Bits [23:16]: **Data1** (stored in FLASH_OBR[31:24]) Bits [15:8]: **nData0** Bits [7:0]: **Data0** (stored in FLASH_OBR[23:16])
0x1FFF F808	**WRPx**: Flash memory write protection option bytes Bits [31:24]: **nWRP1** Bits [23:16]: **WRP1** (stored in FLASH_WRPR[15:8]) Bits [15:8]: **nWRP0** Bits [7:0]: **WRP0** (stored in FLASH_WRPR[7:0]) 0: Write protection active 1: Write protection not active Refer to *Section 3.3.2: Write protection* for more details. In total, 2 user option bytes are used to protect the whole main Flash memory. WRP0: Write-protects pages 0 to 15 WRP1: Write-protects pages 16 to 31 *Note: Even if WRP2 and WRP3 are not available, they must be kept at reset value.*

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

5 Cyclic redundancy check calculation unit (CRC)

5.1 Introduction

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

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

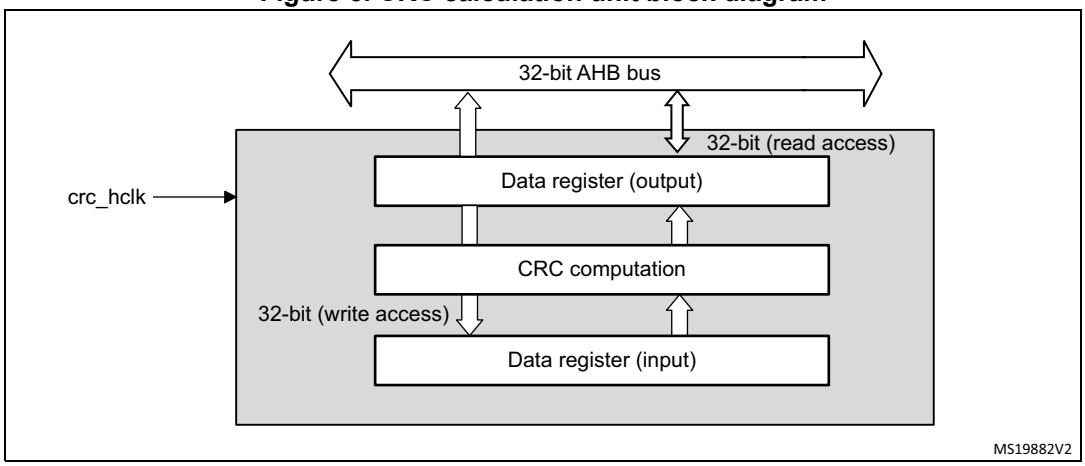
5.2 CRC main features

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

5.3 CRC functional description

5.3.1 CRC block diagram

Figure 5. CRC calculation unit block diagram



5.3.2 CRC internal signals

Table 12. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

5.3.3 CRC operation

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

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

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

The duration of the computation depends on data width:

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

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

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

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

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

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

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

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

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

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

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

Polynomial programmability

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

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

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

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

5.4 CRC registers

5.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

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

This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

5.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

These bits can be used as a temporary storage location for four bytes.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

5.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 7 **REV_OUT**: Reverse output data
This bit controls the reversal of the bit order of the output data.
0: Bit order not affected
1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data
These bits control the reversal of the bit order of the input data
00: Bit order not affected
01: Bit reversal done by byte
10: Bit reversal done by half-word
11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size
These bits control the size of the polynomial.
00: 32 bit polynomial
01: 16 bit polynomial
10: 8 bit polynomial
11: 7 bit polynomial

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

Bit 0 **RESET**: RESET bit
This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

5.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT[31:0]**: *Programmable initial CRC value*
This register is used to write the CRC initial value.

5.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

5.4.6 CRC register map

Table 13. CRC register map and reset values

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

Refer to *Section 2.2 on page 47* for the register boundary addresses.

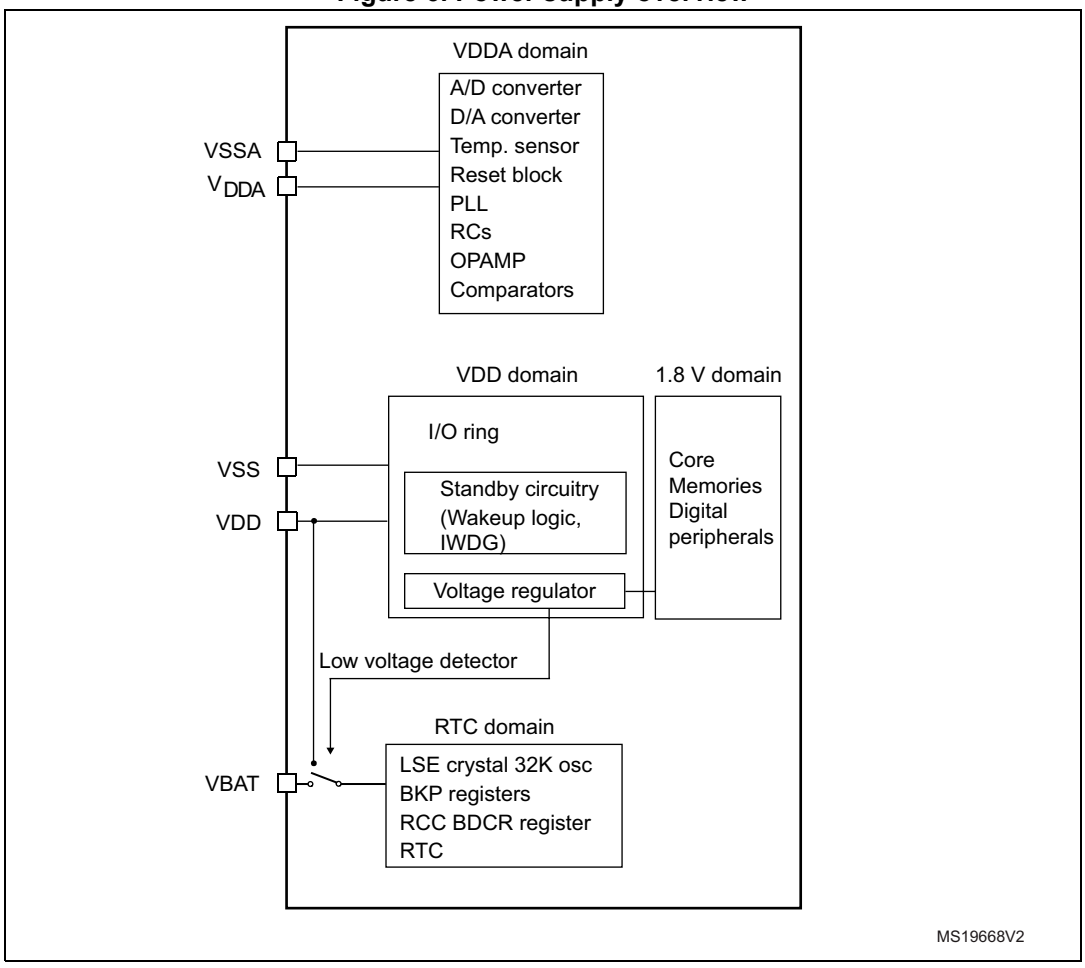
6 Power control (PWR)

6.1 Power supplies

The STM32F334xx devices require a 2.0 V - 3.6 V operating supply voltage (V_{DD}) and a 2.0 V - 3.6 V analog supply voltage (V_{DDA}). The embedded regulator is used to supply the internal 1.8 V digital power.

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

Figure 6. Power supply overview



The following supply voltages are available:

- V_{DD} and V_{SS}: external power supply for I/Os and core.
  These supply voltages are provided externally through V_{DD} and V_{SS} pins. V_{DD} = 2.0 to 3.6 V .
  V_{DD} must always be kept lower than or equal to V_{DDA}.
- VDD18 = 1.65 to 1.95 V (VDD18 domain): power supply for digital core, SRAM and Flash memory.
  VDD18 is internally generated through an internal voltage regulator .

- V_{DDA}, V_{SSA}= 2.0 to 3.6 V : external power supply for ADC, DAC, comparators, operational amplifiers, temperature sensor, PLL, HSI 8 MHz oscillator, LSI 40 kHz oscillator, and reset block.
  V_{DDA} must be in the 2.4 to 3.6 V range when the OPAMP and DAC are used.
  It is forbidden to have $V_{DDA} < V_{DD}$ - 0.4 V. An external Schottky diode must be placed between V_{DD} and V_{DDA} to guarantee that this condition is met.
- V_{BAT}= 1.65 to 3.6 V: Backup power supply for RTC, LSE oscillator, PC13 to PC15 and backup registers when V_{DD} is not present. When V_{DD} supply is present, the internal power switch switches the backup power to V_{DD}. If V_{BAT} is not used, it must be connected to V_{DD}.

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

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

The ADC and DAC voltage supply input is available on a separate VDDA pin. An isolated supply ground connection is provided on the VSSA pin.

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

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

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

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

6.1.2 Battery Backup domain

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

The V_{BAT} pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The switch to the V_{BAT} supply is controlled by the power-down reset (PDR) embedded in the reset block.

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

connect an external low-drop diode between this power supply and the V_{BAT} pin.

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

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

- PC13, PC14 and PC15 can be used as GPIO pins
- PC13, PC14 and PC15 can be configured by RTC or LSE (refer to *Section 26.3: RTC functional description on page 836*)

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

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

- PC13, PC14 and PC15 can be controlled only by RTC or LSE (refer to *Section 26.3: RTC functional description on page 836*)

6.1.3 Voltage regulator

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

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

6.2 Power supply supervisor

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

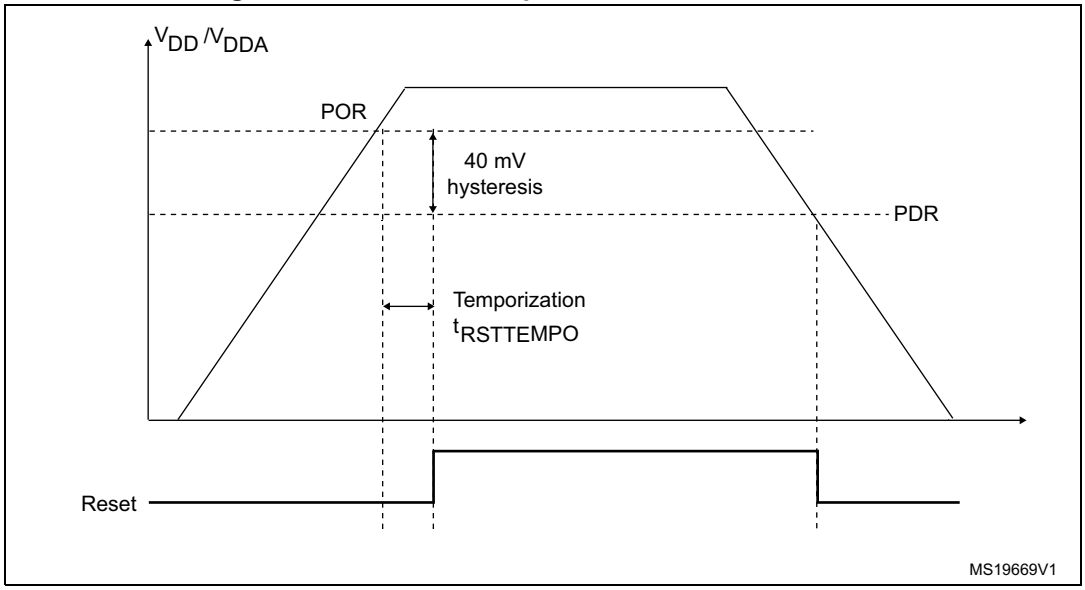
The device has an integrated power-on reset (POR) and power-down reset (PDR) circuits which are always active and ensure proper operation above a threshold of 2 V.

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

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

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

Figure 7. Power on reset/power down reset waveform



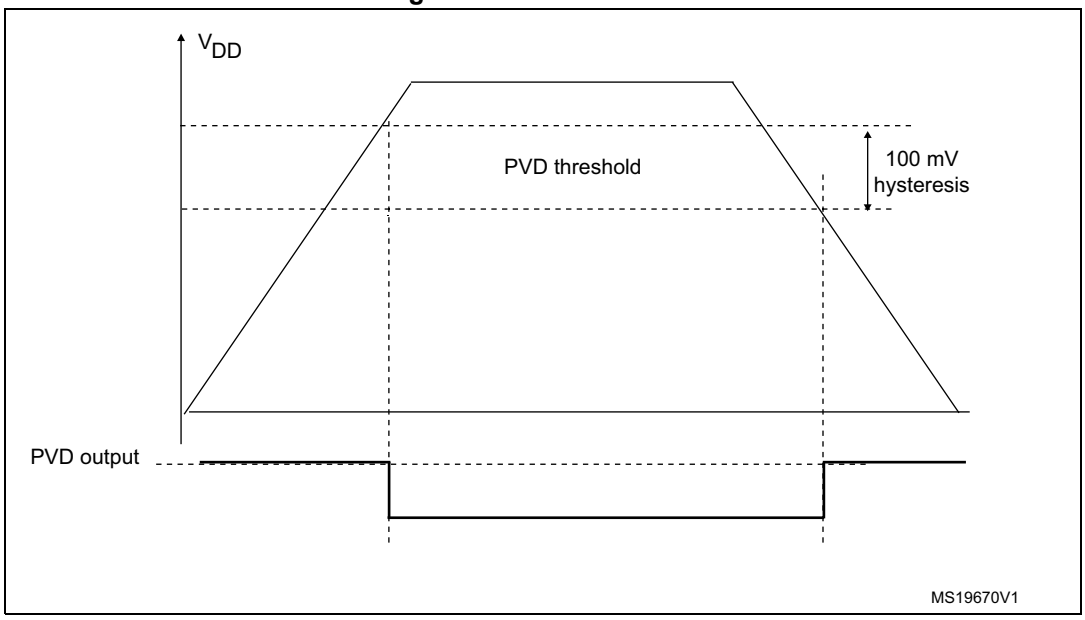
6.2.2 Programmable voltage detector (PVD)

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

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the *Power control/status register (PWR_CSR)*, to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI Line16 should be configured according to PVD output behavior i.e. if the EXTI line 16 is configured to rising edge sensitivity, the interrupt is generated when V_{DD} drops below the PVD threshold. As an example the service routine could perform emergency shutdown tasks.

Figure 8. PVD thresholds



6.3 Low-power modes

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

The device features three low-power modes:

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

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

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

Table 14. Low-power mode summary

Mode name	Entry	wakeup	Effect on 1.8V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on - exit)	WFI	Any interrupt	CPU clock OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			

Table 14. Low-power mode summary

Mode name	Entry	wakeup	Effect on 1.8V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers) Specific communication peripherals on reception events (USART, I2C)	All 1.8V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on *Power control register (PWR_CR)*)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

6.3.1 Slowing down system clocks

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

For more details refer to *Section 8.4.2: Clock configuration register (RCC_CFGR)*.

6.3.2 Peripheral clock gating

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

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

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

6.3.3 Sleep mode

Entering Sleep mode

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

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

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

Refer to *Table 15* and *Table 16* for details on how to enter Sleep mode.

Exiting Sleep mode

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

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

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

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

Refer to *Table 15* and *Table 16* for more details on how to exit Sleep mode.

Table 15. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex-M4 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to *Table 35: STM32F334xx vector table* If WFE was used for entry Wakeup event: Refer to *Section 12.2.3: Wakeup event management*
Wakeup latency	None

Table 16. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex-M4 System Control register.
Mode exit	Interrupt: refer to *Table 35: STM32F334xx vector table*.
Wakeup latency	None

6.3.4 Stop mode

The Stop mode is based on the Cortex-M4 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode in the STM32F334xx devices. In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering Stop mode

Refer to *Table 17* for details on how to enter the Stop mode.

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

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

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See *Section 24.3: IWDG functional description* in *Section 24: Independent watchdog (IWDG)*.
- real-time clock (RTC): this is configured by the RTCEN bit in the *RTC domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the *Control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the *RTC domain control register (RCC_BDCR)*.

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

Exiting Stop mode

Refer to *Table 17* 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 17. Stop mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP bit in Arm® Cortex®-M4 System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR **Note:** To enter Stop mode, all EXTI Line pending bits (in *Pending register (EXTI_PR1)*), all peripherals interrupt pending bits and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues. If the application needs to disable the external oscillator (external clock) before entering Stop mode, the system clock source must be first switched to HSI and then clear the HSEON bit. Otherwise, if before entering Stop mode the HSEON bit is kept at 1, the security system (CSS) feature must be enabled to detect any external oscillator (external clock) failure and avoid a malfunction when entering Stop mode.
Mode exit	If WFI was used for entry: – Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). – Some specific communication peripherals (USART, I2C) interrupts, when programmed in wakeup mode (the peripheral must be programmed in wakeup mode and the corresponding interrupt vector must be enabled in the NVIC). Refer to *Table 35: STM32F334xx vector table*. If WFE was used for entry: Any EXTI Line configured in event mode. Refer to *Section 12.2.3: Wakeup event management*
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

6.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex-M4 deepsleep mode, with the voltage regulator disabled. The 1.8 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the RTC domain and Standby circuitry (see *Figure 6*).

Entering Standby mode

Refer to *Table 18* 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 24.3: IWDG functional description in *Section 24: Independent watchdog (IWDG)*.

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

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin or the rising edge of an RTC alarm occurs (see *Figure 323: RTC block diagram*). All registers are reset after wakeup from Standby except for *Power control/status register (PWR_CSR)*.

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

Refer to *Table 18* for more details on how to exit Standby mode.

Table 18. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex-M4 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR)
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

I/O states in Standby mode

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

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

Debug mode

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

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

6.3.6 Auto-wakeup from low-power mode

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

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

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

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

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

6.4 Power control registers

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

6.4.1 Power control register (PWR_CR)

Address offset: 0x00

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

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

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

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

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

0: Access to RTC and Backup registers disabled
1: Access to RTC and Backup registers enabled

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

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

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

000: 2.2V
001: 2.3V
010: 2.4V
011: 2.5V
100: 2.6V
101: 2.7V
110: 2.8V
111: 2.9V

Notes:

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

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

This bit is set and cleared by software.

0: PVD disabled
1: PVD enabled

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

This bit is always read as 0.

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

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

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

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

6.4.2 Power control/status register (PWR_CSR)

Address offset: 0x04

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

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

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

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

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

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

Bit 8 **EWUP1:** Enable WKUP1 pin

This bit is set and cleared by software.

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

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

Note: This bit is reset by a system Reset.

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

Bit 2 **PVDO:** PVD output

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

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

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

Notes:

1. The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.
2. Once the PVD is enabled and configured in the PWR_CR register, PVDO can be used to generate an interrupt through the External Interrupt controller.
3. Once the PVD_LOCK is enabled (for CLASS B protection) PVDO cannot be disabled anymore.

Bit 1 **SBF:** Standby flag

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

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF:** Wakeup flag

This bit is set by hardware and cleared by a system reset or by setting the CWUF bit in the *Power control register (PWR_CR)*

0: No wakeup event occurred

1: A wakeup event was received from the WKUP pin or from the RTC alarm

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

6.4.3 PWR register map

Table 19. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWUP3	EWUP2	EWUP1	Res.	Res.	Res.	Res.	Res.	PVDO	SBF	WUF
	Reset value																						0	0	0						0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

7 Peripheral interconnect matrix

7.1 Introduction

Several STM32F3 peripherals have internal interconnections. Knowing these interconnections allows the following benefits:

- Autonomous communication between peripherals,
- Efficient synchronization between peripherals,
- Discard the software latency and minimize GPIOs configuration,
- Optimum number of available pins even with small packages,
- Avoid the use of connectors and design an optimized PCB with less dissipated energy.

7.2 Connection summary

The following table presents the matrix for the peripheral interconnect.

Table 20. STM32F334 peripherals interconnect matrix(1)

Source / Destination	DMA1	ADC1	ADC2	COMP2	COMP4	COMP6	OPAMP	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3	DAC1	DAC2	IRTIM	HRTIM1
ADC1	x	-	x	-	-	-	-	x	-	-	-	-	-	-	-	-	x
ADC2	x	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	x
COMP2	-	-	-	-	-	-	-	x	-	-	-	x	x	-	-	-	x
COMP4	-	-	-	-	-	-	-	-	x	-	-	-	x	-	-	-	x
COMP6	-	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	x
OPAMP2	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	x
TIM1	x	x	x	x	-	-	x	-	-	-	-	x	x	-	-	-	x
SPI1	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USART1	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM15	x	x	x	-	x	x	-	x	-	-	-	-	x	x	x	-	x
TIM16	x	-	-	-	-	-	-	-	x	-	-	-	-	-	-	x	x
TIM17	x	-	-	-	-	-	-	x	x	-	-	-	-	-	-	x	x
TIM2	x	x	x	x	-	x	-	x	x	-	-	-	x	x	x	-	x
TIM3	x	x	x	x	x	-	-	x	x	-	-	x	-	x	x	-	x
TIM6	x	x	x	-	-	-	-	-	-	-	-	-	-	x	x	-	x
TIM7	x	-	-	-	-	-	-	-	-	-	-	-	-	x	x	-	x
USART2	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USART3	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I2C1	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1	x	-	-	x	x	x	-	-	-	-	-	-	-	-	-	-	-

Table 20. STM32F334 peripherals interconnect matrix(1) (continued)

Source / Destination	DMA1	ADC1	ADC2	COMP2	COMP4	COMP6	OPAMP	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3	DAC1	DAC2	IRTIM	HRTIM1
DAC2	x	-	-	x	x	x	-	-	-	-	-	-	-	-	-	-	-
TS	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VBAT	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Vrefint	-	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-
CSS	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-	x
PVD	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-	x
SRAM Parity error	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-	x
CPU Hardfault	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-	x
HSE	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
HSI	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
LSE	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
LSI	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
MCO	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
RTC	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
HRTIM1	x	x	x	-	-	-	-	-	-	-	-	-	-	x	x	-	-

1. The cells with gray shading indicate that there is no interconnection.

7.3 Interconnection details

7.3.1 DMA interconnections

Hardware DMA requests are managed by peripherals. The DMA channels dedicated to each peripheral are summarized in *Section 11.3.2: DMA request mapping*.

7.3.2 From ADC to ADC

ADC1 can be used as a "master" to trigger ADC2 "slave" start of conversion.

In dual ADC mode, the converted data of the master and slave ADCs can be read in parallel.

A description of dual ADC mode is provided in *Section 13.3.29: Dual ADC modes*.

7.3.3 From ADC to TIM

ADCx (x=1, 2) can provide trigger event through watchdog signals to advanced-control timer TIM1.

A description of the ADC analog watchdog settings is provided in *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*.

The output (from ADC) is on signals ADCx_AWDy_OUT (x = 1, 2 and y = 1..3 as there are 3 analog watchdogs per ADC) and the input (to timer) on signal TIM1_ETR (external trigger).

TIM1_ETR is connected to ADCx_AWDy_OUT through bits in TIM1_OR registers; refer to *Section 18.4.23: TIM1 option registers (TIM1_OR)*.

7.3.4 From TIM and EXTI to ADC

General-purpose timers (TIM2/TIM3), basic timers (TIM6/TIM7), advanced-control timer (TIM1), general-purpose timer (TIM15/TIM16/TIM17) and EXTI can be used to generate an ADC triggering event.

The output (from timer) is on signal TIMx_TRGO, TIMx_TRGO2 or TIMx_CCx event.

The input (to ADC) is on signal EXT[15:0], JEXT[15:0].

The connection between timers and ADCs or also EXTI & ADCs is provided in:

- *Table 40: ADC1 (master) & 2 (slave) - External triggers for regular channels*
- *Table 41: ADC1 & ADC2 - External trigger for injected channels*

7.3.5 From OPAMP to ADC

There are two interconnection types:

1. Connect OPAMP output reference voltage to an internal ADC channel. This connection can be used for OPAMP calibration. For more details, please refer to the *Section 16.3.5: Calibration*.

   ADC2_IN17 is the channel connected internally to the reference voltage for OPAMP2.
2. OPAMP2 can be connected to ADC2_IN3. Refer to *Section 16.3.4: Using the OPAMP outputs as ADC inputs*.

7.3.6 From TS to ADC

Internal temperature sensor (VTS) is connected internally to ADC1_IN16. Refer to *Section 13.3.30: Temperature sensor*.

7.3.7 From VBAT to ADC

VBAT/2 output voltage can be converted using ADC1_IN17. This interconnection is explained in *Section 13.3.31: VBAT supply monitoring*.

7.3.8 From VREFINT to ADC

VREFINT is internally connected to channel 18 of the two ADCs. This allows the monitoring of its value as described in *Section 13.3.32: Monitoring the internal voltage reference*.

7.3.9 From COMP to TIM

The comparators outputs can be redirected internally to different timer inputs:

- break input 1/2 for fast PWM shutdowns,
- OCREF_CLR input,
- Input capture.

To select which timer input must be connected to the comparator output, the bits field COMPxOUTSEL in the COMPx_CSR register are used.

The following table gives an overview of all possible comparator outputs redirection to the timer inputs.

Table 21. Comparator outputs to timer inputs

COMP output selection					
-	**TIM1**	**TIM2**	**TIM3**	**TIM15**	**TIM16**
COMP2	TIM1_BRK_ACTH TIM1_BRK2 TIM1_OCrefClear TIM1_IC1	TIM2_IC4 TIM2_OCrefClear	TIM3_IC1 TIM3_OCrefClear	-	-
COMP4	TIM1_BRK TIM1_BRK2	-	TIM3_IC3 TIM3_OCrefClear	TIM15_OCrefClearTIM15_IC2	-
COMP6	TIM1_BRK_ACTH TIM1_BRK2	TIM2_IC2 TIM2_OCrefClear	-	-	TIM16_OCrefClear TIM16_IC1

Note: When the comparator output is configured to be connected internally to timers break input, the following must be considered:

1/ COMP2/6 can be used to control TIM1_BRK_ACTH (this break is always active high with no digital filter) and to control also TIM1_BRK2 input.

2/ COMP4 can be used to control TIM1_BRK and TIM1_BRK2 input (same as the other comparators).

7.3.10 From TIM to COMP

The timers output can be selected as comparators outputs blanking signals using the “COMPx_BLANKING” bits in “COMPx_CSR” register. More details on the blanking function can be found in *Section 15.3.5: Comparator output blanking function*.

Table 22. Timer output selection as comparator blanking source

COMP blanking source			
-	**COMP2**	**COMP4**	**COMP6**
TIM1	TIM1 OC5	-	-
TIM15	-	TIM15 OC1	TIM15 OC2
TIM2	TIM2 OC3	-	TIM2 OC4
TIM3	TIM3 OC3	TIM3 OC4	-

7.3.11 From DAC to COMP

The comparators inverting input may be a DAC channel output (DAC1_CH1, DAC1_CH2 or DAC2_CH1).

The selection is made based on “COMPxINMSEL” bits value in “COMPx_CSR” register.

The following table summarizes these interconnections.

Table 23. DAC output selection as comparator inverting input

COMP inverting inputs			
-	**COMP2**	**COMP4**	**COMP6**
DAC1_CH1	X	X	X
DAC1_CH2	X	X	X
DAC2_CH1	X	X	X

7.3.12 From VREFINT to COMP

Besides to the DAC channel output, Vrefint (x1, x3/4, x1/2, x1/4) can be selected as comparator inverting input using “COMPxINMSEL” bits in “COMPx_CSR” register.

7.3.13 From TIM to OPAMP

The switch between OPAMP inverting and non-inverting inputs can be done automatically. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP input multiplexers. More details on this feature are available in *Section 16.3.6: Timer controlled Multiplexer mode*.

7.3.14 From TIM to TIM

Some STM32F3 timers are linked together internally for timer synchronization or chaining.

When one timer is configured in Master Mode, it can reset, start, stop or clock the counter of another timer configured in Slave Mode.

A description of the feature with the various synchronization modes is available in:

- *Section 18.3.25: Timer synchronization* for *the advanced-control timer TIM1*
- *Section 18.3.25: Timer synchronization* for *the general-purpose timers (TIM2/TIM3)*

The slave mode selection is made using “SMS” bits, as described in:

- *Section 18.4.3: TIM1 slave mode control register (TIM1_SMCR),*
- *Section 19.4.3: TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3)* for the *general-purpose timers (TIM2/TIM3),*
- *Section 20.5.3: TIM15 slave mode control register (TIM15_SMCR).*

The possible master/slave connections are summarized in the following table providing the internal trigger connection:

Table 24. Timer synchronization

-		SLAVE			
		TIM1	**TIM2**	**TIM3**	**TIM15**
MASTER	TIM1	-	TIM2_ITR0	TIM3_ITR0	-
	TIM2	TIM1_ITR1	-	TIM3_ITR1	TIM15_ITR0
	TIM3	TIM1_ITR2	TIM2_ITR2	-	TIM15_ITR1
	TIM15	TIM1_ITR0		TIM3_ITR2	-
	TIM16	-		-	TIM15_ITR2
	TIM17	TIM1_ITR3		-	TIM15_ITR3

7.3.15 From system errors to TIM

In addition to comparators outputs, other sources can be used as trigger for the internal break events of some timers (TIM1/TIM15/TIM16/TIM17). For example:

- the clock failure event generated by CSS, refer to *Section 8.2.6: System clock (SYSCLK) selection* for more details,
- the PVD output, refer to *Section 6.2.2: Programmable voltage detector (PVD)* for more details,
- the SRAM parity error signal, refer to *Section 2.2.3: Parity check* for more details,
- the Cortex-M4 LOCKUP (Hardfault) output.

The sources mentioned above can be connected internally to TIMx_BRK_ACTH input, x = 1,15,16,17.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

More details on the break feature are provided in:

- *Section 18.3.16: Using the break function* for the advanced-control timers (TIM1)
- *Section 20.4.13: Using the break function* for the general-purpose timers (TIM15/TIM16/TIM17)

7.3.16 From HSE, HSI, LSE, LSI, MCO, RTC to TIM

TIM16 can be used for the measurement of internal/external clock sources. TIM16 channel1 input capture is connected to HSE/32, GPIO, RTC clock and MCO to output clocks among (HSE, HSI, LSE, LSI, SYSCLK, PLLCLK, PLLCLK/2).

The selection is performed through the TI1_RMP [1:0] bits in the TIM16_OR register.

This allows calibrating the HSI/LSI clocks.

More details are provided in *Section 8.2.14: Internal/external clock measurement with TIM16*.

7.3.17 From TIM and EXTI to DAC

A timer counter may be used as a trigger for DAC conversions.

The TRGO event is the internal signal that will trigger conversion.

The following table provides a summary of DACs interconnections with timers:

This is described in *Section 14.5.4: DAC trigger selection*.

Table 25. Timer and EXTI signals triggering DAC conversions

-	DAC1	DAC2
TIM2	X	X
TIM3	X	X
TIM6	X	X
TIM7	X	X
TIM15	X	X
EXTI line9	X	X

7.3.18 From TIM to IRTIM

General-purpose timer (TIM16/TIM17) output channels TIMx_OC1 are used to generate the waveform of infrared signal output. The functionality is described in *Section 22: Infrared interface (IRTIM)*.

7.3.19 From ADC to HRTIM1

ADCx (x=1, 2) provides the trigger event through watchdog signals to the high resolution timer HRTIM1.

The exact mapping between HRTIM1 external events and ADC watchdog signals is provided in *Table 86: External events mapping and associated features*.

7.3.20 From system faults to HRTIM1

The HRTIM1 system fault input (SYSFLT) gathers MCU internal fault events coming from:

- the clock failure event generated by the clock security system (CSS),
- the PVD output,
- the SRAM parity error signal,
- the Cortex-M4 LOCKUP (Hardfault) output.

Refer to *Section 21.3.15: Fault protection* for more details on the HRTIM1 fault protection feature.

7.3.21 From COMP to HRTIM1

The comparator output can be redirected internally to HRTIM1 inputs.

Table 56: STM32F334xx comparator input/outputs summary provides the exact mapping between comparators outputs and HRTIM internal signals. It is also explained in *Table 86: External events mapping and associated features*.

The comparator outputs are connected directly to HRTIM1 in order to speed-up the propagation delay.

7.3.22 From OPAMP to HRTIM1

The OPAMP2_VOUT can be used as a HRTIM1 internal event source connected to HRTIM1_EEV4 or HRTIM1_EEV9 as shown in *Table 86: External events mapping and associated features*

7.3.23 From TIM to HRTIM1

The connections between timers and HRTIM1 are listed in *Table 86: External events mapping and associated features*.

7.3.24 From HRTIM1 to ADC

The HRTIM1 can be used to generate an ADC trigger event on signal HRTIM1_ADCTRG1/2/3/4.

More details on ADC triggering using HRTIM1 signals are provided in *Section 21.3.18: ADC triggers*.

7.3.25 From HRTIM1 to DAC

The HRTIM1 DACTRGx events can be selected as internal signals to trigger DAC conversion depending on the value of TSELx[2:0] control bits in DAC_CR register.

More details on ADC triggering using HRTIM1 signals are provided in *Section 21.3.18: ADC triggers*.

8 Reset and clock control (RCC)

8.1 Reset

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

8.1.1 Power reset

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

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

A power reset sets all registers to their reset values except the RTC domain (see *Figure 6*).

8.1.2 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the RTC domain (see *Figure 6*).

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

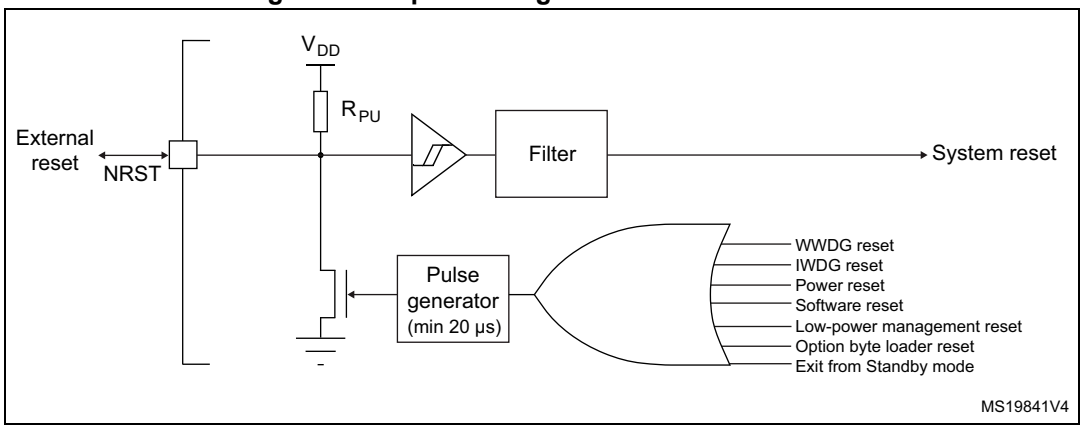
1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software reset (SW reset) (see *Software reset*)
5. Low-power management reset (see *Low-power management reset*)
6. Option byte loader reset (see *Option byte loader reset*)
7. A power reset

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR (see *Section 8.4.10: Control/status register (RCC_CSR)*).

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

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 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 9. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex®-M4 application interrupt and reset control register must be set to force a software reset on the device. Refer to the *STM32 Cortex®-M4 MCUs and MPUs programming manual* (PM0214) for more details.

Low-power management reset

There are two ways to generate a low-power management reset:

1. Reset generated when entering Standby mode:

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

2. Reset when entering Stop mode:

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

For further information on the User Option Bytes, refer to *Section 2: Option bytes*.

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit (bit 13) is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

8.1.3 RTC domain reset

The RTC domain has two specific resets that affect only the RTC domain (*Figure 6*).

An RTC domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC *RTC domain control register (RCC_BDCR)*. It is generated when one of the following events occurs.

1. Software reset, triggered by setting the BDRST bit in the *RTC domain control register (RCC_BDCR)*.
2. V_{DD} power-up if V_{BAT} has been disconnected when it was low.

The backup registers are also reset when one of the following events occurs:

1. RTC tamper detection event.
2. Change of the read out protection from level 1 to level 0.

8.2 Clocks

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

- HSI 8 MHZ RC oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following additional clock sources:

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

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

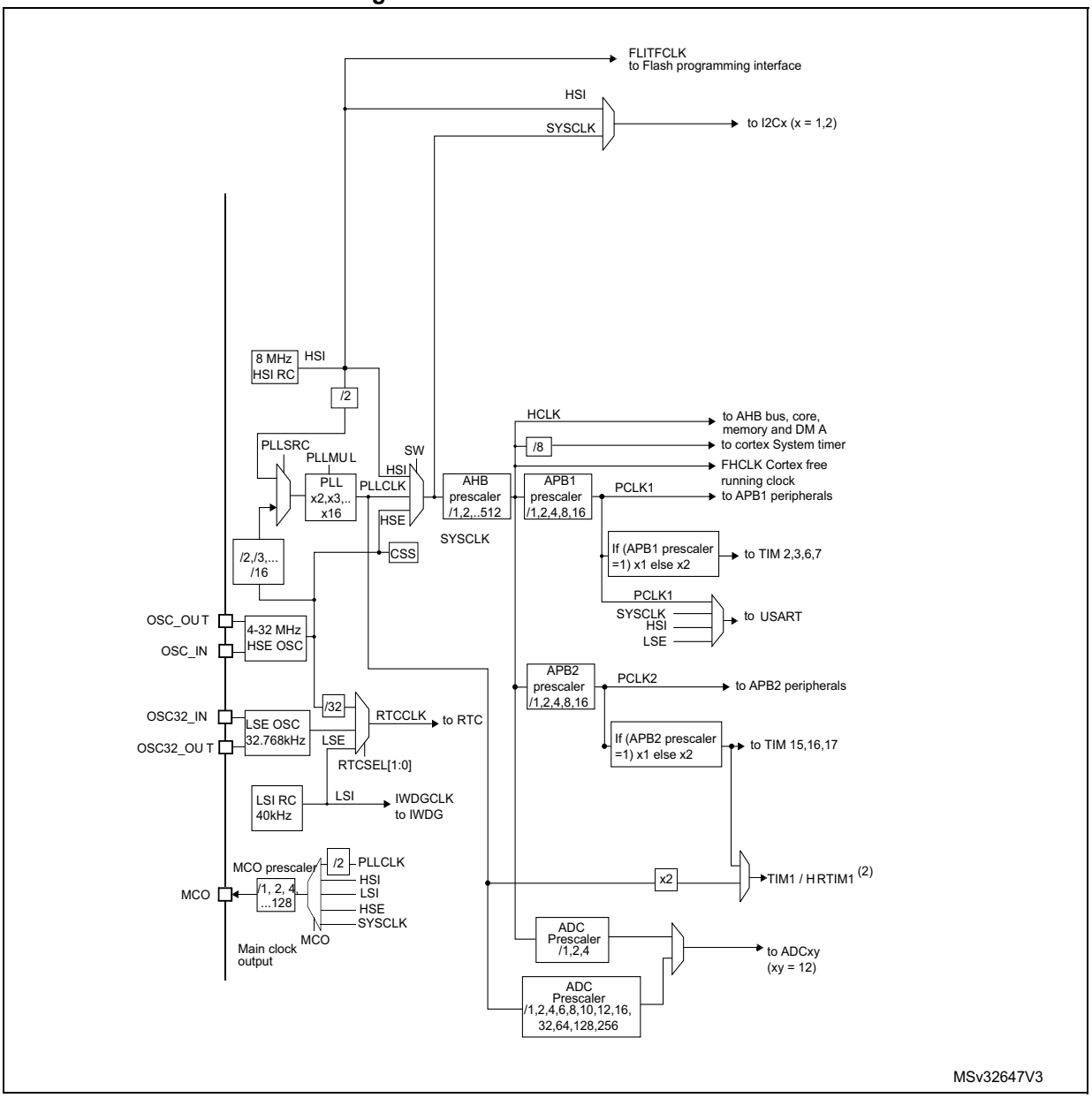
Several prescalers can be used to configure the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB and APB2 domains is 72 MHz. The maximum allowed frequency of the APB1 domain is 36 MHz.

All the peripheral clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except:

- The Flash memory programming interface clock (FLITFCLK) which is always the HSI clock.
- The option byte loader clock which is always the HSI clock
- The ADCs clock which is derived from the PLL output. It can reach 72 MHz and can then be divided by 1,2,4,6,8,10,12,16,32,64,128 or 256.
- The U(S)ARTs clock which is derived (selected by software) from one of the four following sources:
  - system clock
  - HSI clock
  - LSE clock
  - APB1 or APB2 clock (PCLK1 or PCLK2 depending on which APB is mapped the USART)
- The I2C1/2 clock which is derived (selected by software) from one of the two following sources:
  - system clock
  - HSI clock
- The RTC clock which is derived from the LSE, LSI or from the HSE clock divided by 32.
- The IWDG clock which is always the LSI clock.

The RCC feeds the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex clock (HCLK), configurable in the SysTick Control and Status Register.

Figure 10. STM32F334xx clock tree



1. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet.
2. TIM1 can be clocked from the PLLCLKx2 running up to 144 MHz when the system clock source is the PLL. Refer to *Section 8.2.10: Timers (TIMx) clock*.
   HRTIM1 can be clocked from the PLLCLKx2 with 2 possible configurations:
   - HSE is the PLL clock source and PLLCLK is set to 72 MHz (HRTIM frequency is 144 MHz)
   - HSI is the PLL clock source and PLLCLK is set to 64 MHz (HRTIM frequency is 128 MHz)
   Refer to *Section 8.2.11: High-resolution timer (HRTIM) clock*.
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is ‘1’, the AHB prescaler must be equal to ‘1’.

FCLK acts as Cortex®-M4 free-running clock. For more details refer to the *STM32 Cortex®-M4 MCUs and MPUs programming manual* (PM0214).

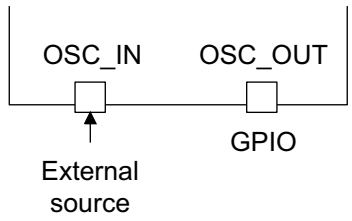
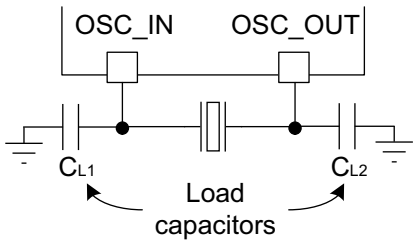
8.2.1 HSE clock

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

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

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

Figure 11. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	 MSv31915V1
Crystal/Ceramic resonators	 MSv31916V1

External crystal/ceramic resonator (HSE crystal)

The 4 to 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Figure 11*. Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the *Clock control register (RCC_CR)* indicates if the HSE oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.

The HSE Crystal can be switched on and off using the HSEON bit in the *Clock control register (RCC_CR)*.

Caution: To switch ON the HSE oscillator, 512 HSE clock pulses need to be seen by an internal stabilization counter after the HSEON bit is set. Even in the case that no crystal or resonator is connected to the device, excessive external noise on the OSC_IN pin may still lead the oscillator to start. Once the oscillator is started, it needs another 6 HSE clock pulses to complete a switching OFF sequence. If for any reason the oscillations are no more present on the OSC_IN pin, the oscillator cannot be switched OFF, locking the OSC pins from any other use and introducing unwanted power consumption. To avoid such situation, it is strongly recommended to always enable the Clock Security System (CSS) which is able to switch OFF the oscillator even in this case.

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 32 MHz. Select this mode by setting the HSEBYP and HSEON bits in the *Clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~40-60% duty cycle depending on the frequency (refer to the datasheet) has to drive the OSC_IN pin while the OSC_OUT pin can be used a GPIO. See *Figure 11*.

8.2.2 HSI clock

The HSI clock signal is generated from an internal 8 MHz RC Oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at T_A=25°C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *Clock control register (RCC_CR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. The user can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the *Clock control register (RCC_CR)*.

For more details on how to measure the HSI frequency variation, refer to *Section 8.2.14: Internal/external clock measurement with TIM16*.

The HSIRDY flag in the *Clock control register (RCC_CR)* indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC can be switched on and off using the HSION bit in the *Clock control register (RCC_CR)*.

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 8.2.7: Clock security system (CSS) on page 111*.

8.2.3 PLL

The internal PLL can be used to multiply the HSI or HSE output clock frequency. Refer to *Figure 10* and *Clock control register (RCC_CR)*.

The PLL configuration (selection of the input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.

An interrupt can be generated when the PLL is ready, if enabled in the *Clock interrupt register (RCC_CIR)*.

The PLL output frequency must be set in the range 16-72 MHz.

8.2.4 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit in *RTC domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RTC domain control register (RCC_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other.

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

Caution: To switch ON the LSE oscillator, 4096 LSE clock pulses need to be seen by an internal stabilization counter after the LSEON bit is set. Even in the case that no crystal or resonator is connected to the device, excessive external noise on the OSC32_IN pin may still lead the oscillator to start. Once the oscillator is started, it needs another 6 LSE clock pulses to complete a switching OFF sequence. If for any reason the oscillations are no more present on the OSC_IN pin, the oscillator cannot be switched OFF, locking the OSC32 pins from any other use and introducing unwanted power consumption. The only way to recover such situation is to perform the RTC domain reset by software.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. Select this mode by setting the LSEBYP and LSEON bits in the *RTC domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO. See *Figure 11*.

8.2.5 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is around 40 kHz (between 30 kHz and 50 kHz). For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the *Control/status register (RCC_CSR)*.

The LSIRDY flag in the *Control/status register (RCC_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.

8.2.6 System clock (SYSCLK) selection

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

- HSI oscillator
- HSE oscillator
- PLL

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *Clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

8.2.7 Clock security system (CSS)

Clock Security System can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (HRTIM1_SYSFLT, TIM1 and TIM15/16/17) and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 NMI (non-maskable interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the Clock interrupt register (RCC_CIR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the HSI oscillator and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

8.2.8 ADC clock

The ADC clock is derived from the PLL output. It can reach 72 MHz and can be divided by the following prescalers values: 1, 2, 4, 6, 8,10,12,16, 32, 64, 128 or 256. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADCx_CCR.

If the programmed factor is ‘1’, the AHB prescaler must be set to ‘1’.

8.2.9 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *RTC domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the RTC domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC.

The LSE clock is in the RTC domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock:
  - The RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
  - The RTC remains clocked and functional under system reset.
- If LSI is selected as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock divided by 32 is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.8 V domain).

8.2.10 Timers (TIMx) clock

APB clock source

The timers clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain.

PLL clock source

A clock issued from the PLL (PLLCLKx2) can be selected for TIM1.This configuration allows to feed TIM1 with a frequency up to 144 MHz when the system clock source is the PLL.

8.2.11 High-resolution timer (HRTIM) clock

APB clock source

If the high resolution is not required, the HRTIM1SW bit in the RCC_CFGR3 register can be kept cleared. In this case, CKPSC[2:0] in the HRTIM1_MCR register must be greater or equal to 5 (prescaling ratio greater or equal to 32).

HRTIM1 input clock frequency is automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain.

PLL clock source

If HRTIM1 high resolution is required, the clock issued from the PLL must be selected by setting the HRTIM1SW bit in the RCC_CFGR3 register. In this case, any value of CK_PSC[2:0] in the HRTIM1_MCR register can be used. In this configuration, AHB and APB2 prescalers (HPRE and PPRE2 bits in the RCC_CFGR register) must be set in order to keep a ratio of 1 or 2 between the system clock SYSCLK and the APB2 clock PCLK2.

Two configurations are supported:

1. HSE is the PLL clock source and PLLCLK is set to 72MHz (HRTIM frequency is 144 MHz)
2. HSI is the PLL clock source and PLLCLK is set to 64MHz (HRTIM frequency is 128 MHz)

Note: The HRTIM operating temperature range is limited when the HRTIM frequency is 128MHz. Refer to the datasheet for operating conditions.

8.2.12 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.

8.2.13 Clock-out capability

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. The configuration registers of the corresponding GPIO port must be programmed in alternate function mode. One of 5 clock signals can be selected as the MCO clock.

- LSI
- LSE
- SYSCLK
- HSI
- HSE
- PLL clock not divided or divided by 2 (using the PLLNODIV bit in RCC_CFGR register)

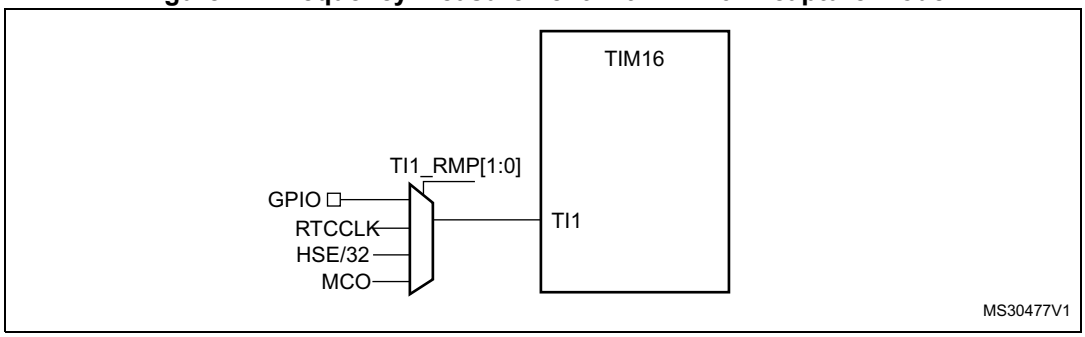
The selection is controlled by the MCO[2:0] bits in the *Clock configuration register (RCC_CFGR)*.

The additional bit PLLNODIV in this register controls the divider bypass for a PLL clock input to MCO. The MCO frequency can be reduced by a configurable divider, controlled by the MCOPRE[2:0] bits of the Clock configuration register (RCC_CFGR).

8.2.14 Internal/external clock measurement with TIM16

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM16 channel 1 input capture. As represented on *Figure 12*.

Figure 12. Frequency measurement with TIM16 in capture mode



The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP [1:0] bits in the TIM16_OR register. The possibilities available are the following ones.

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM16 Channel1 is connected to the RTCCLK.
- TIM16 Channel1 is connected to the HSE/32 Clock.
- TIM16 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCO[2:0] bits of the Clock configuration register (RCC_CFGR).

Calibration of the HSI

The primary purpose of connecting the LSE, through the MCO multiplexer, to the channel 1 input capture is to be able to precisely measure the HSI system clocks (for this, the HSI should be used as the system clock source). The number of HSI clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm's), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

The HSI oscillator has dedicated user-accessible calibration bits for this purpose.

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

If LSE is not available, HSE/32 is the better option in order to reach the most precise calibration possible.

Calibration of the LSI

The calibration of the LSI follows the same pattern that for the HSI, but changing the reference clock. It is necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

8.3 Low-power modes

APB peripheral clocks and DMA clock can be disabled by software.

Sleep mode stops the CPU clock. The memory interface clocks (Flash and RAM interfaces) can be stopped by software during sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

Stop mode stops all the clocks in the V18 domain and disables the PLL, the HSI and the HSE oscillators.

All U(S)ARTs and I2Cs have the capability to enable the HSI oscillator even when the MCU is in Stop mode (if HSI is selected as the clock source for that peripheral).

All U(S)ARTs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON) but they do not have the capability to turn on the LSE oscillator.

Standby mode stops all the clocks in the V18 domain and disables the PLL and the HSI and HSE oscillators.

The CPU’s deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register.

When waking up from deepsleep after an interrupt (Stop mode) or reset (Standby mode), the HSI oscillator is selected as system clock.

If a Flash programming operation is on going, deepsleep mode entry is delayed until the Flash interface access is finished. If an access to the APB domain is ongoing, deepsleep mode entry is delayed until the APB access is finished.

8.4 RCC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

8.4.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83

(where X is undefined)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLLON	Res.	Res.	Res.	Res.	CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** PLL clock ready flag

Set by hardware to indicate that the PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON:** PLL enable

Set and cleared by software to enable PLL.

Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock.

0: PLL OFF

1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSSON:** Clock security system enable

Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected.

0: Clock detector OFF

1: Clock detector ON (Clock detector ON if the HSE oscillator is ready, OFF if not).

Bit 18 **HSEBYP:** HSE crystal oscillator bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

Bit 17 **HSERDY:** HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. This bit needs 6 cycles of the HSE oscillator clock to fall down after HSEON reset.

0: HSE oscillator not ready
1: HSE oscillator ready

Bit 16 **HSEON:** HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF
1: HSE oscillator ON

Bits 15:8 **HSICAL[7:0]:** HSI clock calibration

These bits are initialized automatically at startup.

Bits 7:3 **HSITRIM[4:0]:** HSI clock trimming

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

The default value is 16, which, when added to the HSICAL value, should trim the HSI to 8 MHz ± 1%. The trimming step ($F_{hsitrim}$) is around 40 kHz between two consecutive HSICAL steps.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSIRDY:** HSI clock ready flag

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

0: HSI oscillator not ready
1: HSI oscillator ready

Bit 0 **HSION:** HSI clock enable

Set and cleared by software.

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

0: HSI oscillator OFF
1: HSI oscillator ON

8.4.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLNO DIV	MCOPRE[2:0]			Res.	MCO[2:0]			Res.	Res.	PLLMUL[3:0]				PLL XTPRE	PLL SRC
rw	rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit 31 **PLLNODIV:** Do not divide PLL to MCO
This bit is set and cleared by software. It switch-off divider-by-2 for PLL connection to MCO
0: PLL is divided by 2 before MCO
1: PLL is not divided before MCO

Bits 30:28 **MCOPRE[2:0]:** Microcontroller Clock Output Prescaler
There bits are set and cleared by software. It is highly recommended to change this prescaler before MCO output is enabled
000: MCO is divided by 1
001: MCO is divided by 2
010: MCO is divided by 4
.....
111: MCO is divided by 128

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **MCO[2:0]:** Microcontroller clock output
Set and cleared by software.
000: MCO output disabled, no clock on MCO
001: Reserved
010: LSI clock selected.
011: LSE clock selected.
100: System clock (SYSCLK) selected
101: HSI clock selected
110: HSE clock selected
111: PLL clock selected (divided by 1 or 2 depending on PLLNODIV bit).
Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **PLLMUL[3:0]:** PLL multiplication factor

These bits are written by software to define the PLL multiplication factor. These bits can be written only when PLL is disabled.

Caution: The PLL output frequency must not exceed 72 MHz.

0000: PLL input clock x 2
0001: PLL input clock x 3
0010: PLL input clock x 4
0011: PLL input clock x 5
0100: PLL input clock x 6
0101: PLL input clock x 7
0110: PLL input clock x 8
0111: PLL input clock x 9
1000: PLL input clock x 10
1001: PLL input clock x 11
1010: PLL input clock x 12
1011: PLL input clock x 13
1100: PLL input clock x 14
1101: PLL input clock x 15
1110: PLL input clock x 16
1111: PLL input clock x 16

Bit 17 **PLLXTPRE:** HSE divider for PLL input clock

This bits is set and cleared by software to select the HSE division factor for the PLL. It can be written only when the PLL is disabled.

Note: This bit is the same as the LSB of PREDIV in Clock configuration register 2 (RCC_CFGR2) (for compatibility with other STM32 products)

0000: HSE input to PLL not divided
0001: HSE input to PLL divided by 2

Bit 16 **PLLSRC:** PLL entry clock source

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

0: HSI/2 selected as PLL input clock
1: HSE/PREDIV selected as PLL input clock (refer to *Section 8.4.12: Clock configuration register 2 (RCC_CFGR2) on page 135*

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]:** APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB2 clock (PCLK).

0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 10:8 **PPRE1[2:0]:** APB Low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB1 clock (PCLK).

0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 7:4 **HPRE[3:0]:** HLCK prescaler

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

0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

Note: The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock. Refer to section Read operations on page 55 for more details.

Bits 3:2 **SWS[1:0]:** System clock switch status

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

00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

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

Set and cleared by software to select SYSCLK source.

Cleared by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).

00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

8.4.3 Clock interrupt register (RCC_CIR)

Address offset: 0x08

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Res.	Res.	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

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

Bit 23 **CSSC:** Clock security system interrupt clear
This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **PLLRDYC:** PLL ready interrupt clear
This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: Clear PLLRDYF flag

Bit 19 **HSERDYC:** HSE ready interrupt clear
This bit is set by software to clear the HSERDYF flag.
0: No effect
1: Clear HSERDYF flag

Bit 18 **HSIRDYC:** HSI ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: Clear HSIRDYF flag

Bit 17 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear
This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLLRDYIE:** PLL ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL lock.
0: PLL lock interrupt disabled
1: PLL lock interrupt enabled

Bit 11 **HSERDYIE:** HSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled

Bit 10 **HSIRDYIE:** HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled

Bit 9 **LSERDYIE:** LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 8 **LSIRDYIE:** LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

Bit 7 **CSSF:** Clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **PLLRDYF:** PLL ready interrupt flag
Set by hardware when the PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit 3 **HSERDYF:** HSE ready interrupt flag
Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 2 **HSIRDYF:** HSI ready interrupt flag
Set by hardware when the HSI clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to *Clock control register (RCC_CR)*). When HSION is not set but the HSI oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.
Cleared by software setting the HSIRDYC bit.
0: No clock ready interrupt caused by the HSI oscillator
1: Clock ready interrupt caused by the HSI oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag
Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.
Cleared by software setting the LSERDYC bit.
0: No clock ready interrupt caused by the LSE oscillator
1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSIRDYF:** LSI ready interrupt flag
Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.
Cleared by software setting the LSIRDYC bit.
0: No clock ready interrupt caused by the LSI oscillator
1: Clock ready interrupt caused by the LSI oscillator

8.4.4 APB2 peripheral reset register (RCC_APB2RSTR)

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
Res.	Res.	HRTIM 1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17 RST	TIM16 RST	TIM15 RST
		rw											rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFG RST
	rw		rw	rw											rw

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

Bit 29 **HRTIM1RST:** High Resolution Timer1 reset
Set and cleared by software.
0: No effect
1: Reset HRTIM1 timer

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST:** TIM17 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM17 timer

Bit 17 **TIM16RST:** TIM16 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM16 timer

Bit 16 **TIM15RST:** TIM15 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM15 timer

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST:** USART1 reset
Set and cleared by software.
0: No effect
1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI1 reset
Set and cleared by software.
0: No effect
1: Reset SPI1

Bit 11 **TIM1RST:** TIM1 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM1 timer

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST:** SYSCFG, Comparators and operational amplifiers reset
Set and cleared by software.
0: No effect
1: Reset SYSCFG, COMP, and OPAMP

8.4.5 APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DAC1 RST	PWR RST	Res.	DAC2R ST	CAN RST	Res.	Res.	Res.	I2C1 RST	Res.	Res.	USART3 RST	USART2 RST	Res.
		rw	rw		rw	rw				rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res.	Res.	WWDG RST	Res.	Res.	Res.	Res.	Res.	TIM7 RST	TIM6 RST	Res.	Res	TIM3 RST	TIM2 RST
				rw						rw	rw			rw	rw

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

Bit 29 **DAC1RST:** DAC1 interface reset
Set and cleared by software.
0: No effect
1: Reset DAC1 interface

Bit 28 **PWRRST:** Power interface reset
Set and cleared by software.
0: No effect
1: Reset power interface

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DAC2RST:** DAC2 interface reset
Set and cleared by software.
0: No effect
1: Reset DAC2 interface

Bit 25 **CANRST:** CAN reset
Set and reset by software.
0: does not reset the CAN
1: resets the CAN

Bits 24:22 Reserved, must be kept at reset value

Bit 21 **I2C1RST:** I2C1 reset
Set and cleared by software.
0: No effect
1: Reset I2C1

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **USART3RST:** USART3 reset
Set and cleared by software.
0: No effect
1: Reset USART3

Bit 17 **USART2RST:** USART2 reset
Set and cleared by software.
0: No effect
1: Reset USART2

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **WWDGRST:** Window watchdog reset
Set and cleared by software.
0: No effect
1: Reset window watchdog

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7RST:** TIM7 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM7

Bit 4 **TIM6RST:** TIM6 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM6

Bits 3:21 Reserved, must be kept at reset value.

Bit 1 **TIM3RST:** TIM3 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM3

Bit 0 **TIM2RST:** TIM2 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM2

8.4.6 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x14

Reset value: 0x0000 0014

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

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	TSCEN	Res.	IOPF EN	Res..	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.
			rw				rw		rw		rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC EN	Res.	FLITF EN	Res.	SRAM EN	Res.	DMA1 EN
									rw		rw		rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADC12EN:** ADC1 and ADC2 enable
Set and reset by software.
0: ADC1 and ADC2 clock disabled
1: ADC1 and ADC2 clock enabled

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

Bit 24 **TSCEN:** Touch sensing controller clock enable
Set and cleared by software.
0: TSC clock disabled
1: TSC clock enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 **IOPFEN:** I/O port F clock enable
Set and cleared by software.
0: I/O port F clock disabled
1: I/O port F clock enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **IOPDEN:** I/O port D clock enable
Set and cleared by software.
0: I/O port D clock disabled
1: I/O port D clock enabled

Bit 19 **IOPCEN:** I/O port C clock enable
Set and cleared by software.
0: I/O port C clock disabled
1: I/O port C clock enabled

Bit 18 **IOPBEN:** I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1: I/O port B clock enabled

Bit 17 **IOPAEN:** I/O port A clock enable
Set and cleared by software.
0: I/O port A clock disabled
1: I/O port A clock enabled

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **CRCEN:** CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **FLITFEN:** FLITF clock enable
Set and cleared by software to disable/enable FLITF clock during Sleep mode.
0: FLITF clock disabled during Sleep mode
1: FLITF clock enabled during Sleep mode

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAMEN:** SRAM interface clock enable
Set and cleared by software to disable/enable SRAM interface clock during Sleep mode.
0: SRAM interface clock disabled during Sleep mode.
1: SRAM interface clock enabled during Sleep mode

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DMA1EN:** DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

8.4.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HRTIMER1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
		rw											rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	rw		rw	rw											rw

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

Bit 29 **HRTIMER1EN:** High Resolution Timer 1 clock enable
Set and cleared by software.
0: HRTIM1 clock disabled
1: HRTIM1 clock enabled

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN:** TIM17 timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled
1: TIM17 timer clock enabled

Bit 17 **TIM16EN:** TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled
1: TIM16 timer clock enabled

Bit 16 **TIM15EN:** TIM15 timer clock enable
Set and cleared by software.
0: TIM15 timer clock disabled
1: TIM15 timer clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN:** USART1 clock enable
Set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

Bit 13 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 **TIM1EN:** TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1 timer clock enabled

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN:** SYSCFG clock enable
Set and cleared by software.
0: SYSCFG clock disabled
1: SYSCFG clock enabled

8.4.8 APB1 peripheral clock enable register (RCC_APB1ENR)

Address: 0x1C

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait state, except if the access occurs while an access to a peripheral on APB1 domain is on going. In this case, wait states are inserted until this access to APB1 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DAC1EN	PWREN	Res.	DAC2EN	CANEN	Res	Res.	Res.	I2C1EN	Res.	Res.	USART3EN	USART2EN	Res.
		rw	rw		rw	rw				rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3ENRes.	TIM2EN
				rw						rw	rw			rw	rw

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

Bit 29 **DAC1EN:** DAC1 interface clock enable
Set and cleared by software.
0: DAC1 interface clock disabled
1: DAC1 interface clock enabled

Bit 28 **PWREN:** Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DAC2EN:** DAC2 interface clock enable
Set and cleared by software.
0: DAC2 interface clock disabled
1: DAC2 interface clock enabled

Bit 25 **CANEN:** CAN clock enable
Set and reset by software.
0: CAN clock disabled
1: CAN clock enabled

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN:** I2C1 clock enable
Set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled

Bits 20:19 Reserved, must be kept at reset value.

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

Bits 16:12 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 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN:** TIM7 timer clock enable
Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN:** TIM6 timer clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bits 3:21 Reserved, must be kept at reset value.

Bit 1 **TIM3EN:** TIM3 timer clock enable
Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN:** TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

8.4.9 RTC domain control register (RCC_BDCR)

Address offset: 0x20

Reset value: 0x0000 0018 (reset by RTC 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.

Note: The LSEON, LSEBYP, RTCSEL and RTCEN bits of the RTC domain control register (RCC_BDCR) are in the RTC domain. As a result, after Reset, these bits are write-protected and the DBP bit in the Power control register (PWR_CR) has to be set before these can be modified. These bits are only reset after a RTC domain Reset (see Section 8.1.3: RTC domain reset). Any internal or external reset does not have any effect on these bits.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	Res.	Res.	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
rw						rw	rw				rw	rw	rw	r	rw

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

Bit 16 **BDRST:** RTC domain software reset
Set and cleared by software.
0: Reset not activated
1: Resets the entire RTC domain

Bit 15 **RTCEN:** RTC clock enable
Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection
Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the RTC domain is reset. The BDRST bit can be used to reset them.
00: No clock
01: LSE oscillator clock used as RTC clock
10: LSI oscillator clock used as RTC clock
11: HSE oscillator clock divided by 32 used as RTC clock

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:3 **LSEDRV[1:0]:** LSE oscillator drive capability
Set and reset by software to modulate the LSE oscillator's drive capability. A reset of the RTC domain restores the default value.
00: 'Xtal mode' lower driving capability
01: 'Xtal mode' medium high driving capability
10: 'Xtal mode' medium low driving capability
11: 'Xtal mode' higher driving capability (reset value)
Note: The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP:** LSE oscillator bypass
Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the external 32 kHz oscillator is disabled.
0: LSE oscillator not bypassed
1: LSE oscillator bypassed

Bit 1 **LSERDY:** LSE oscillator ready
Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.
0: LSE oscillator not ready
1: LSE oscillator ready

Bit 0 **LSEON:** LSE oscillator enable
Set and cleared by software.
0: LSE oscillator OFF
1: LSE oscillator ON

8.4.10 Control/status register (RCC_CSR)

Address: 0x24

Reset value: 0x0C00 0000 (reset by system Reset, except reset flags by power Reset only)

Access: 0 ≤ 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 STF	IW WDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	OB LRSTF	RMVF	V18PW RRSTF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI RDY	LSION
														r	rw

Bit 31 **LPWRSTF:** 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 *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 window 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 flag
Set by hardware when a POR/PDR occurs. Cleared by writing to the RMVF bit.
0: No POR/PDR occurred
1: POR/PDR occurred

Bit 26 **PINRSTF:** PIN reset flag
Set by hardware when a reset from the NRST pin occurs. Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF:** Option byte loader reset flag
Set by hardware when a reset from the OBL occurs. Cleared by writing to the RMVF bit.
0: No reset from OBL occurred
1: Reset from OBL occurred

Bit 24 **RMVF:** Remove reset flag
Set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

Bit 23 **V18PWRRSTF**: Reset flag of the 1.8 V domain.
Set by hardware when a POR/PDR of the 1.8 V domain occurred. Cleared by writing to the RMVF bit.
0: No POR/PDR reset of the 1.8 V domain occurred
1: POR/PDR reset of the 1.8 V domain occurred
Note: On the STM32F3x8 products, this flag is reserved.

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

Bit 1 **LSIRDY:** LSI oscillator ready
Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 0 **LSION:** LSI oscillator enable
Set and cleared by software.
0: LSI oscillator OFF
1: LSI oscillator ON

8.4.11 AHB peripheral reset register (RCC_AHBRSTR)

Address: 0x28

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC12 RST	Res.	Res.	Res.	TSC RST	Res.	IOPF RST	Res.	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.
			rw				rw		rw		rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADC12RST:** ADC1 and ADC2 reset
Set and reset by software.
0: does not reset the ADC1 and ADC2
1: resets the ADC1 and ADC2

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

Bit 24 **TSCRST:** Touch sensing controller reset
Set and cleared by software.
0: No effect
1: Reset TSC

Bit 23 Reserved, must be kept at reset value.

Bit 22 **IOPFRST:** I/O port F reset
Set and cleared by software.
0: No effect
1: Reset I/O port F

Bit 21 Reserved, must be kept at reset value.

Bit 20 **IOPDRST:** I/O port D reset
Set and cleared by software.
0: No effect
1: Reset I/O port D

Bit 19 **IOPCRST:** I/O port C reset
Set and cleared by software.
0: No effect
1: Reset I/O port C

Bit 18 **IOPBRST:** I/O port B reset
Set and cleared by software.
0: No effect
1: Reset I/O port B

Bit 17 **IOPARST:** I/O port A reset
Set and cleared by software.
0: No effect
1: Reset I/O port A

Bits 16:0 Reserved, must be kept at reset value.

8.4.12 Clock configuration register 2 (RCC_CFGR2)

Address: 0x2C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12PRES[4:0]					PREDIV[3:0]			
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:4 **ADC12PRES:** ADC12 prescaler
Set and reset by software to control PLL clock to ADC12 division factor.
0xxxx: ADC12 clock disabled, ADC12 can use AHB clock
10000: PLL clock divided by 1
10001: PLL clock divided by 2
10010: PLL clock divided by 4
10011: PLL clock divided by 6
10100: PLL clock divided by 8
10101: PLL clock divided by 10
10110: PLL clock divided by 12
10111: PLL clock divided by 16
11000: PLL clock divided by 32
11001: PLL clock divided by 64
11010: PLL clock divided by 128
11011: PLL clock divided by 256
others: PLL clock divided by 256

Bits 3:0 **PREDIV**: PREDIV division factor

These bits are set and cleared by software to select PREDIV division factor. They can be written only when the PLL is disabled.

Note: Bit 0 is the same bit as bit17 in Clock configuration register (RCC_CFGR), so modifying bit17 Clock configuration register (RCC_CFGR) also modifies bit 0 in Clock configuration register 2 (RCC_CFGR2) (for compatibility with other STM32 products)

0000: HSE input to PLL not divided
0001: HSE input to PLL divided by 2
0010: HSE input to PLL divided by 3
0011: HSE input to PLL divided by 4
0100: HSE input to PLL divided by 5
0101: HSE input to PLL divided by 6
0110: HSE input to PLL divided by 7
0111: HSE input to PLL divided by 8
1000: HSE input to PLL divided by 9
1001: HSE input to PLL divided by 10
1010: HSE input to PLL divided by 11
1011: HSE input to PLL divided by 12
1100: HSE input to PLL divided by 13
1101: HSE input to PLL divided by 14
1110: HSE input to PLL divided by 15
1111: HSE input to PLL divided by 16

8.4.13 Clock configuration register 3 (RCC_CFGR3)

Address: 0x30

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	HRTIM 1SW	Res.	Res.	Res.	TIM1 SW	Res.	Res.	Res.	I2C1 SW	Res.	Res.	USART1SW[1:0]	
			rw				rw				rw			rw	rw

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

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **HRTIM1SW:** High Resolution Timer1 clock source selection

Set and reset by software to select HRTIM1 clock source.

The bit is writable only when the following conditions occur: clock system = PLL, and AHBxAPB2 maximum prescaler factor is 2 and respect the clock system.

The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)

0: APB high speed clock (doubled frequency when prescaled) (default)
1: PLL vco output (running up to 144 MHz)

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

Bit 8 **TIM1SW:** Timer1 clock source selection

Set and reset by software to select TIM1 clock source.

The bit is writable only when the following conditions occur: clock system = PLL, and AHB and APB2 subsystem clock not divided respect the clock system.

The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)

0: PCLK2 clock (doubled frequency when prescaled) (default)

1: PLL vco output (running up to 144 MHz)

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **I2C1SW:** I2C1 clock source selection

This bit is set and cleared by software to select the I2C1 clock source.

0: HSI clock selected as I2C1 clock source (default)

1: SYSCLK clock selected as I2C1 clock

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

Bits 1:0 **USART1SW[1:0]:** USART1 clock source selection

This bit is set and cleared by software to select the USART1 clock source.

00: PCLK selected as USART1 clock source (default)

01: System clock (SYSCLK) selected as USART1 clock

10: LSE clock selected as USART1 clock

11: HSI clock selected as USART1 clock

8.4.14 RCC register map

Table 26. RCC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RCC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERD	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
	Reset value							0	0					0	0	0	0	x	x	x	x	x	x	x	x	1	0	0	0	0		1	1
0x04	**RCC_CFGR**	PLLNODIV	MCOPRE[2:0]			Res.	MCO [2:0]			Res.	Res.	PLLMUL [3:0]				PLLXTPRE	PLLSRC	Res.	Res.	PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [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
0x08	**RCC_CIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Res.	Res.	Res.	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res.	Res.	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0			0	0	0	0	0				0	0	0	0	0	0			0	0	0	0	0
0x0C	**RCC_ APB2RSTR**	Res.	Res.	HRTIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM16RST	TIM15RST	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGRST
	Reset value			0												0	0		0		0	0											0

Table 26. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	**RCC_APB1RSTR**	Res.	Res.	DAC1RST	PWRRST	Res.	DAC2RST	CANRST	Res.	Res.	Res.	I2C1RST	Res.	Res.	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	WWDGRST	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	Res.	Res.	TIM3RST	TIM2RST
	Reset value			0	0		0	0				0			0	0		0	0			0						0	0			0	0
0x14	**RCC_AHBENR**	Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	TSCEN	Res.	IOPFEN	Res.	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN		DMA1EN
	Reset value				0				0		0		0	0	0	0												0		1		1		0
0x18	**RCC_APB2ENR**	Res.	Res.	HRTIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	Reset value			0											0	0	0		0		0	0												0
0x1C	**RCC_APB1ENR**	Res.	Res.	DAC1EN	PWREN	Res.	DAC2EN	CANEN	Res.	Res.	Res.	I2C1EN	Res.	Res.	USART3EN	USART2EN	Res.	Res.	Res.	Res.	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
	Reset value			0	0		0	0				0			0	0						0						0	0			0	0
0x20	**RCC_BDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	RTC SEL [1:0]		Res.	Res.	Res.	LSE DRV [1:0]		LSEBYP	LSERDY	LSEON
	Reset value						0	0									0	0						0	0				1	1	0	0	0
0x24	**RCC_CSR**	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	0	0	0	0																							0	0
0x28	**RCC_AHBRSTR**	Res.	Res.	Res.	ADC12RST	Res.	Res.	Res.	Res.	Res	IOPFRST	Res.	IOPDRST	IOPCRST	IOPBRST	IOPARST	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.
	Reset value				0						0		0	0	0	0																	
0x2C	**RCC_CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12PRES [4:0]					PREDIV[3:0]			
	Reset value																								0					0			
0x30	**RCC_CFGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRTIM1EN	Res.	Res.	Res.	TIM1SW	Res.	Res.	Res.	I2C1SW	Res.	Res.	USART1SW[1:0]	
	Reset value																				0				0				0			0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

9 General-purpose I/Os (GPIO)

9.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

9.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the port A, B, C, D and F I/O configuration.
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

9.3 GPIO functional description

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

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

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

Figure 13 and *Figure 14* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 27* gives the possible port bit configurations.

Figure 13. Basic structure of an I/O port bit

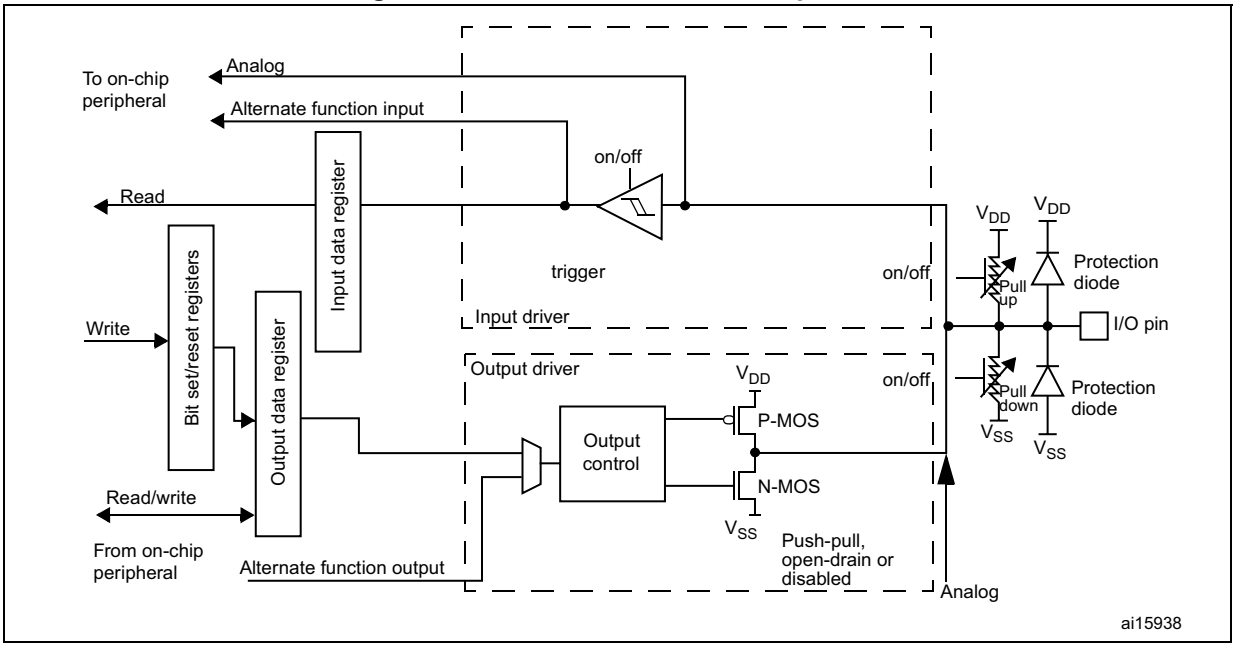
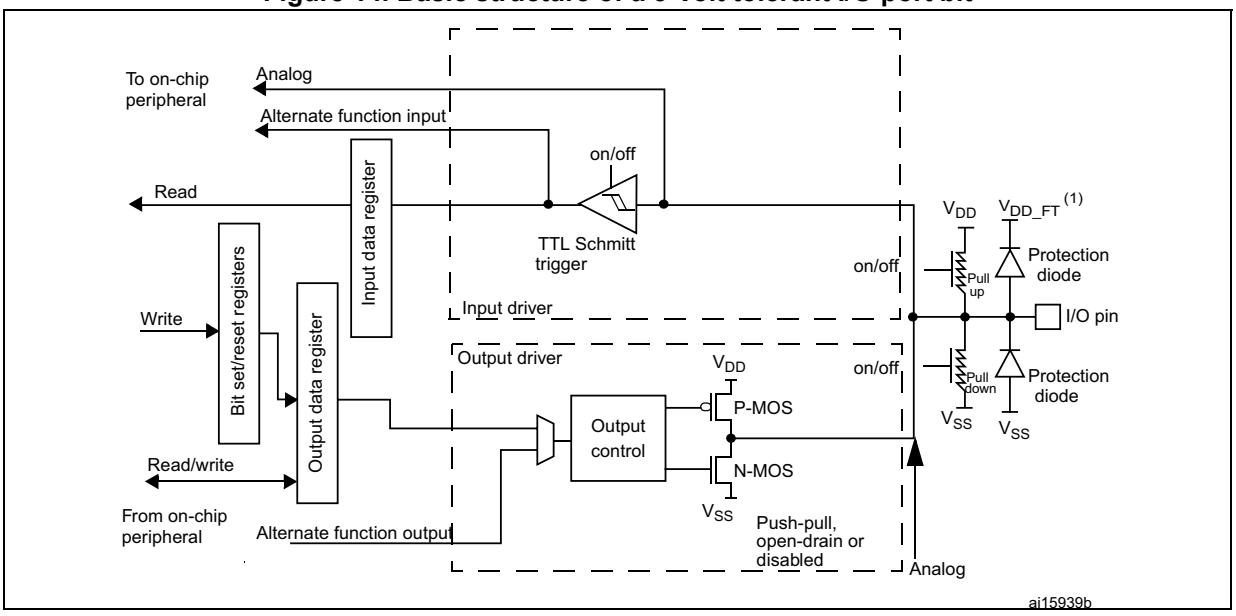


Figure 14. Basic structure of a 5-Volt tolerant I/O port bit



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

Table 27. Port bit configuration table(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [1:0]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

9.3.1 General-purpose I/O (GPIO)

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

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

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

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

9.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

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

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.

  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC, DAC, OPAMP and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, OPAMP, and COMP registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

9.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

9.3.4 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 9.4.5: GPIO port input data register (GPIOx_IDR) (x = A to D and F)* and *Section 9.4.6: GPIO port output data register (GPIOx_ODR) (x = A to D and F)* for the register descriptions.

9.3.5 I/O data bitwise handling

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

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

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

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

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

9.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to *Section 9.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to E and F)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 9.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to E and F)*.

9.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin refer to the device datasheet.

9.3.8 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: Extended interrupts and events controller (EXTI)* and to *Section 12.2.3: Wakeup event management.*

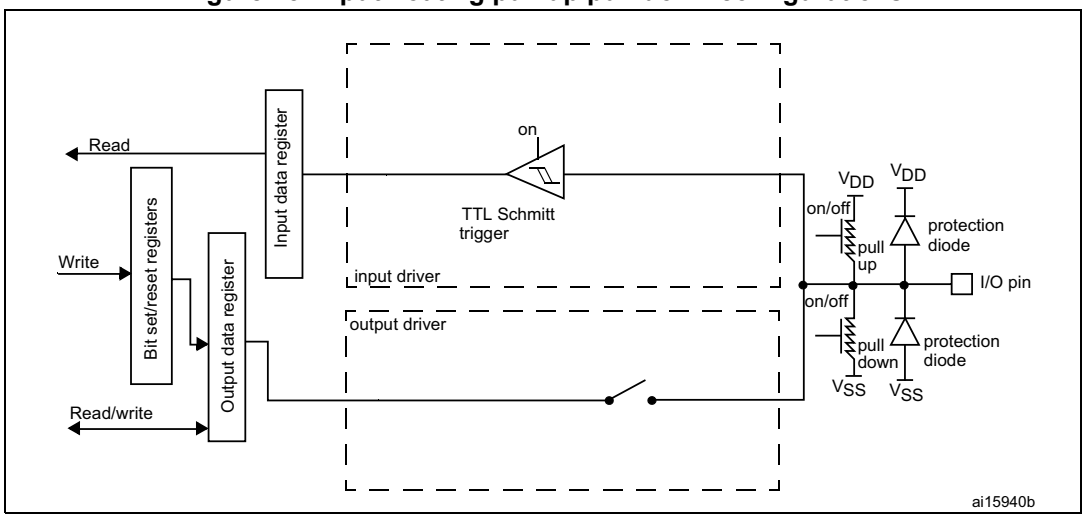
9.3.9 Input configuration

When the I/O port is programmed as input:

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

Figure 15 shows the input configuration of the I/O port bit.

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



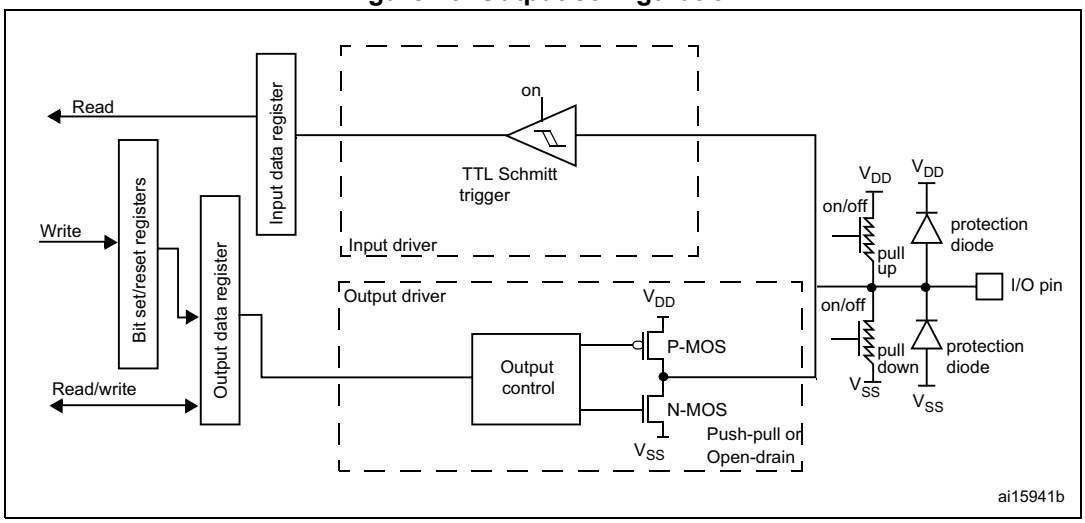
9.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

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

Figure 16. Output configuration



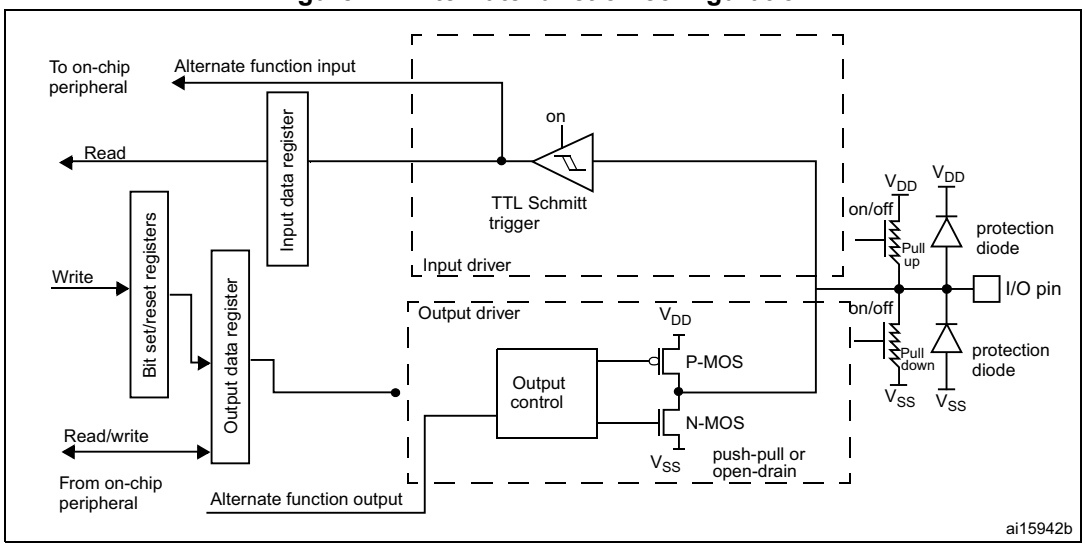
9.3.11 Alternate function configuration

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

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

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

Figure 17. Alternate function configuration



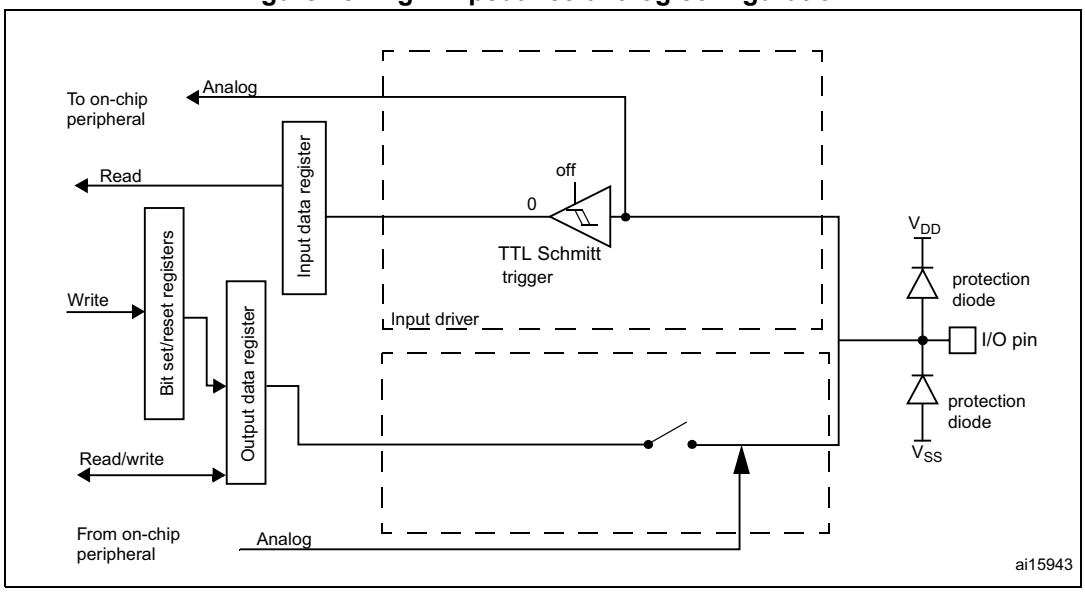
9.3.12 Analog configuration

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

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

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

Figure 18. High impedance-analog configuration



9.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

9.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 26.3: RTC functional description*.

9.4 GPIO registers

For a summary of register bits, register address offsets and reset values, refer to *Table 28*.

The peripheral registers can be written in word, half word or byte mode.

9.4.1 GPIO port mode register (GPIOx_MODER) (x =A to D and F)

Address offset:0x00

Reset value: 0xA800 0000 for port A

Reset value: 0x0000 0280 for port B

Reset value: 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Note: bits 10 and 11 of GPIOF_MODER are reserved and must be kept at reset state.

9.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to D and F)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **OT[15:0]:** Port x configuration I/O pin y (y = 15 to 0)

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

0: Output push-pull (reset state)

1: Output open-drain

9.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to D and F)

Address offset: 0x08

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 31:0 **OSPEEDR[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

x0: Low speed

01: Medium speed

11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed..

9.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to D and F)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0C00 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
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 31:0 **PUPDR[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

9.4.5 GPIO port input data register (GPIOx_IDR) (x = A to D and F)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 **IDR[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

9.4.6 GPIO port output data register (GPIOx_ODR) (x = A to D and F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ODR[15:0]:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..F).

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to D and F)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

9.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to E and F)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the

LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

RD LCKR

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0.

0: Port configuration not locked

1: Port configuration locked

9.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to D and F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRy[3:0]:** Alternate function selection for port x pin y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFRy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8 (Ports A and B only)
1001: AF9 (Ports A and B only)
1010: AF10 (Ports A and B only)
1011: AF11 (Ports A and B only)
1100: AF12 (Ports A and B only)
1101: AF13 (Ports A and B only)
1110: AF14 (Ports A and B only)
1111: AF15 (Ports A and B only)

9.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to D and F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
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
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRy[3:0]:** Alternate function selection for port x pin y (y = 8..15)
These bits are written by software to configure alternate function I/Os

AFRy selection:

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8 (Ports A and B only)
1001: AF9 (Ports A and B only)
1010: AF10 (Ports A and B only)
1011: AF11 (Ports A and B only)
1100: AF12 (Ports A and B only)
1101: AF13 (Ports A and B only)
1110: AF14 (Ports A and B only)
1111: AF15 (Ports A and B only)

9.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to D and F)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 15:0 **BR[15:0]:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Reset the corresponding ODx bit

9.4.12 GPIO register map

The following table gives the GPIO register map and reset values.

Table 28. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_MODER**	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, D and F**)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_OTYPER** (where x = **A..D and F**)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOA_OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOB_OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x08	**GPIOx_OSPEEDR** (where x = **C, D and F**)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = **A..D and F**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = **A..D and F**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = **A..D and F**)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = **A..D and F**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = **A..D and F**)	AFRLAFR7[3:0]				AFRLAFR6[3:0]				AFRLAFR5[3:0]				AFRLAFR4[3:0]				AFRLAFR3[3:0]				AFRLAFR2[3:0]				AFRLAFR1[3:0]				AFRLAFR0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = **A..D and F**)	AFRHAFR15[3:0]				AFRHAFR14[3:0]				AFRHAFR13[3:0]				AFRHAFR12[3:0]				AFRHAFR11[3:0]				AFRHAFR10[3:0]				AFRHAFR9[3:0]				AFRHAFR8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**GPIOx_BRR** (where x = **A..D and F**))	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

10 System configuration controller (SYSCFG)

The STM32F334xx devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Enabling/disabling I^2C Fm+ on some I/O ports
- Remapping some DMA trigger sources from TIM16, TIM17, TIM6, SPI1, I2C1, DAC1_CH1,TIM7 and to different DMA channels
- Remapping the memory located at the beginning of the code area
- Managing the external interrupt line connection to the GPIOs
- Remapping TIM1 ITR3 source
- Remapping DAC1 and DAC2 triggers
- Managing robustness feature
- Configuring encoder mode
- CCM SRAM pages protection

10.1 SYSCFG registers

10.1.1 SYSCFG configuration register 1 (SYSCFG_CFGR1)

This register is used for specific configurations on memory remap.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pin and the option bit setting.

After reset these bits take the value selected by the BOOT pin (BOOT0) and by the option bit (BOOT1).

Address offset: 0x00

Reset value: 0x7C00 000X (X is the memory mode selected by the BOOT0 pin and BOOT1 option bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPU_IE[5..0]						Res	Res	ENCODER_MODE		Res	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
rw	rw	rw	rw	rw	rw			rw			rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC2_CH1_DMA_RMP	TIM7_DAC2_DMA_RMP	TIM6_DAC1_DMA_RMP	TIM17_DMA_RMP	TIM16_DMA_RMP	Res	Res	Res	DAC_TRIG_RMP	TIM1_ITR3_RMP	Res	Res	Res	Res	MEM_MODE	
rw	rw	rw	rw	rw				rw	rw					rw	rw

Bits 31:26 **FPU_IE[5..0]**: Floating Point Unit interrupts enable bits
FPU_IE[5]: Inexact interrupt enable
FPU_IE[4]: Input normal interrupt enable
FPU_IE[3]: Overflow interrupt enable
FPU_IE[2]: underflow interrupt enable
FPU_IE[1]: Divide-by-zero interrupt enable
FPU_IE[0]: Invalid operation interrupt enable

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:22 **ENCODER_MODE:** Encoder mode
This bit is set and cleared by software.
00: No redirection.
01: TIM2 IC1 and TIM2 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively.
10: TIM3 IC1 and TIM3 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively .
11: Reserved.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **I2C1_FMP:** I2C1 Fm+ driving capability activation
This bit is set and cleared by software. It enables the Fm+ on I2C1 pins selected through AF selection bits.
0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits
1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits.

Bits 19:16 **I2C_PBx_FMP**: Fm+ driving capability activation on the pad
These bits are set and cleared by software. Each bit enables I^2C Fm+ mode for PB6, PB7, PB8, and PB9 I/Os.
0: PBx pin operates in standard mode (Sm), x = 6..9
1: I^2C Fm+ mode enabled on PBx pin, and the Speed control is bypassed.

Bit 15 **DAC2_CH1_DMA_RMP:**DAC2 channel1 DMA remap
This bit is set and cleared by software. It controls the remapping of DAC2 channel1 DMA request.
0: No remap
1: Remap (DAC2_CH1 DMA requests mapped on DMA1 channel 5)

Note: In STM32F334xx, this bit must be set.

Bit 14 **TIM7_DAC1_CH2_DMA_RMP:** TIM7 and DAC channel2 DMA remap
This bit is set and cleared by software. It controls the remapping of TIM7(UP) and DAC channel2 DMA request.
0: No remap
1: Remap (TIM7_UP and DAC_CH2 DMA requests mapped on DMA1 channel 4)

Note: In STM32F334xx, this bit must be set as there is no DMA2 in these products.

Bit 13 **TIM6_DAC1_CH1_DMA_RMP:** TIM6 and DAC channel1 DMA remap
This bit is set and cleared by software. It controls the remapping of TIM6 (UP) and DAC channel1 DMA request.
0: No remap (TIM6_UP and DAC_CH1 DMA requests mapped on DMA2 channel 3)
1: Remap (TIM6_UP and DAC_CH1 DMA requests mapped on DMA1 channel 3)

Note: In STM32F334xx, this bit must be set as there is no DMA2 in these products.

Bit 12 **TIM17_DMA_RMP:** TIM17 DMA request remapping bit
This bit is set and cleared by software. It controls the remapping of TIM17 DMA request.
0: No remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA1 channel 1)
1: Remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA1 channel 7)

Bit 11 **TIM16_DMA_RMP:** TIM16 DMA request remapping bit
This bit is set and cleared by software. It controls the remapping of TIM16 DMA request.
0: No remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA1 channel 3)
1: Remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA1 channel 6)

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **DAC1_TRIG_RMP:** DAC trigger remap (when TSEL = 001) This bit is set and cleared by software. It controls the mapping of the DAC trigger source.
0: No remap
1: Remap (DAC trigger is TIM3_TRGO)

Bit 6 **TIM1_ITR3_RMP:** Timer 1 ITR3 selection
This bit is set and cleared by software. It controls the mapping of TIM1 ITR3.
0: No remap
1: Remap (TIM1_ITR3 = TIM17_OC)

Bits 5:2 Reserved, must be kept at reset value.

Bits 1:0 **MEM_MODE:** Memory mapping selection bits
This bit is set and cleared by software. It controls the memory internal mapping at address 0x0000 0000. After reset these bits take on the memory mapping selected by BOOT0 pin and BOOT1 option bit.
x0: Main Flash memory mapped at 0x0000 0000
01: System Flash memory mapped at 0x0000 0000
11: Embedded SRAM (on the D-Code bus) mapped at 0x0000 0000

10.1.2 SYSCFG CCM SRAM protection register (SYSCFG_RCR)

The CCM SRAM has a size of 4 Kbytes, organized in 4 pages (1 Kbyte each). .

Each page can be write protected.

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	PAGE 3_WP	PAGE 2_WP	PAGE 1_WP	PAGE 0_WP
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PAGEx_WP** (x= 0 to 3): CCM SRAM page write protection bit)
These bits are set by software. They can be cleared only by system reset.
0: Write protection of pagex is disabled.
1: Write protection of pagex is enabled.

10.1.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI3[3:0]**: EXTI 3 configuration bits
These bits are written by software to select the source input for the EXTI3 external interrupt.
x000: PA[3] pin
x001: PB[3] pin
x010: PC[3] pin
x011: PD[3] pin
x100: PE[3] pin
other configurations: reserved

Bits 11:8 **EXTI2[3:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

x000: PA[2] pin
x001: PB[2] pin
x010: PC[2] pin
x011: PD[2] pin
x100: PE[2] pin
x101: PF[2] pin
other configurations: reserved

Bits 7:4 **EXTI1[3:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

x000: PA[1] pin
x001: PB[1] pin
x010: PC[1] pin
x011: PD[1] pin
x100: PE[1] pin
x101: PF[1] pin
other configurations: reserved

Bits 3:0 **EXTI0[3:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

Note: x000: PA[0] pin
x001: PB[0] pin
x010: PC[0] pin
x011: PD[0] pin
x100: PE[0] pin
x101: PF[0] pin
other configurations: reserved

10.1.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI7[3:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

x000: PA[7] pin
x001: PB[7] pin
x010: PC[7] pin
x011: PD[7] pin
x100: PE[7] pin
Other configurations: reserved

Bits 11:8 **EXTI6[3:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.

x000: PA[6] pin
x001: PB[6] pin
x010: PC[6] pin
x011: PD[6] pin
x100: PE[6] pin
x101: PF[6] pin
Other configurations: reserved

Bits 7:4 **EXTI5[3:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.

x000: PA[5] pin
x001: PB[5] pin
x010: PC[5] pin
x011: PD[5] pin
x100: PE[5] pin
x101: PF[5] pin
Other configurations: reserved

Bits 3:0 **EXTI4[3:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.

x000: PA[4] pin
x001: PB[4] pin
x010: PC[4] pin
x011: PD[4] pin
x100: PE[4] pin
x101: PF[4] pin
Other configurations: reserved

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.1.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI11[3:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

x000: PA[11] pin
x001: PB[11] pin
x010: PC[11] pin
x011: PD[11] pin
x100: PE[11] pin
other configurations: reserved

Bits 11:8 **EXTI10[3:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

x000: PA[10] pin
x001: PB[10] pin
x010: PC[10] pin
x011:PD[10] pin
x100:PE[10] pin
x101:PF[10] pin
other configurations: reserved

Bits 7:4 **EXTI9[3:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

x000: PA[9] pin
x001: PB[9] pin
x010: PC[9] pin
x011: PD[9] pin
x100: PE[9] pin
x101: PF[9] pin
other configurations: reserved

Bits 3:0 **EXTI8[3:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

x000: PA[8] pin
x001: PB[8] pin
x010: PC[8] pin
x011: PD[8] pin
x100: PE[8] pin
other configurations: reserved

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.1.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI15[3:0]**: EXTI15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

x000: PA[15] pin
x001: PB[15] pin
x010: PC[15] pin
x011: PD[15] pin
x100: PE[15] pin
Other configurations: reserved

Bits 11:8 **EXTI14[3:0]**: EXTI14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

x000: PA[14] pin
x001: PB[14] pin
x010: PC[14] pin
x011: PD[14] pin
x100: PE[14] pin
Other configurations: reserved

Bits 7:4 **EXTI13[3:0]**: EXTI13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

x000: PA[13] pin
x001: PB[13] pin
x010: PC[13] pin
x011: PD[13] pin
x100: PE[13] pin
Other configurations: reserved

Bits 3:0 **EXTI12[3:0]**: EXTI12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

x000: PA[12] pin
x001: PB[12] pin
x010: PC[12] pin
x011: PD[12] pin
x100: PE[12] pin
Other configurations: reserved

Note: Some of the I/O pins mentioned in the above register may not be available on small packages.

10.1.7 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x18

System reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	SRAM_PEF	Res	Res	Res	BYP_ADDR_PAR	Res	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK
							rc_w1				rw		rw	rw	rw

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

Bit 8 **SRAM_PEF**: SRAM parity error flag

This bit is set by hardware when an SRAM parity error is detected. It is cleared by software by writing ‘1’.

0: No SRAM parity error detected

1: SRAM parity error detected

Bits 7:5 Reserved, must be kept at reset value

Bit 4 **BYP_ADDR_PAR**: Bypass address bit 29 in parity calculation

This bit is set by software and cleared by a system reset. It is used to prevent an unwanted parity error when the user writes a code in the RAM at address 0x2XXXXXXX (address in the address range 0x20000000-0x20002000) and then executes the code from RAM at boot (RAM is remapped at address 0x00). In this case, a read operation is performed from the range 0x00000000-0x00002000 resulting in a parity error (the parity on the address is different).

0: The ramload operation is performed taking into consideration bit 29 of the address when the parity is calculated.

1: The ramload operation is performed without taking into consideration bit 29 of the address when the parity is calculated.

Bit 3 Reserved, must be kept at reset value

Bit 2 **PVD_LOCK**: PVD lock enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the PVD connection to TIM1/15/16/17 Break input and HRTIM1 SYSFLT, as well as the PVDE and PLS[2:0] in the PWR_CR register.

0: PVD interrupt disconnected from TIM1/15/16/17 and HRTIM1 SYSFLT Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/15/16/17 and HRTIM1 SYSFLT Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SRAM_PARITY_LOCK**: SRAM parity lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the SRAM parity error signal connection to TIM1/15/16/17 Break inputs and HRTIM1 SYSFLT.

0: SRAM parity error signal disconnected from TIM1/15/16/17 and HRTIM1 SYSFLT Break inputs

1: SRAM parity error signal connected to TIM1/15/16/17 and HRTIM1 SYSFLT Break inputs

Bit 0 **LOCKUP_LOCK:** Cortex®-M4 LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared by a system reset. It can be use to enable and lock the connection of Cortex®-M4 LOCKUP (Hardfault) output to TIM1/15/16/17 Break input.

0: Cortex®-M4 LOCKUP output disconnected from TIM1/15/16/17 Break inputs and HRTIM1 SYSFLT.

1: Cortex®-M4 LOCKUP output connected to TIM1/15/16/17 and HRTIM1 SYSFLT Break inputs

10.1.8 SYSCFG configuration register 3 (SYSCFG_CFGR3)

Address offset: 0x50

System reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DAC1_TRIG5_RMP	DAC1_TRIG3_RMP
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	ADC2_DMA_RMP		I2C1_TX_DMA_RMP		I2C1_RX_DMA_RMP		SPI1_TX_DMA_RMP		SPI1_RX_DMA_RMP	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **DAC1_TRIG5_RMP:** DAC1_CH1 / DAC1_CH2 Trigger remap
Set and cleared by software. This bit controls the mapping of DAC1 trigger
0: No remap
1: Remap (DAC trigger is HRTIM1_DAC1_TRIG2)

Bit 16 **DAC1_TRIG3_RMP** DAC1_CH1 / DAC1_CH2 Trigger remap
Set and cleared by software. This bit controls the mapping of DAC1 trigger
0: Remap (DAC trigger is TIM15_TRGO)
1: Remap (DAC trigger is HRTIM1_DAC1_TRIG1)

Bits 15:10 Reserved, must be kept at reset value

Bit 9 **ADC2_DMA_RMP[1]**: ADC2 DMA controller remapping bit
0: Reserved
1: ADC2 mapped on DMA1

Bit 8 **ADC2_DMA_RMP[0]**: ADC2 DMA channel remapping bit
0: ADC2 mapped on DMA1 channel 2
1: ADC2 mapped on DMA1 channel 4

Bits 7:6 **I2C1_TX_DMA_RMP:** I2C1_TX DMA remapping bit
This bit is set and cleared by software. It defines on which DMA1 channel I2C1_TX is mapped.
00: I2C1_TX mapped on DMA1 CH6
01: I2C1_TX mapped on DMA1 CH2
10: I2C1_TX mapped on DMA1 CH4
11: I2C1_TX mapped on DMA1 CH6

Bits 5:4 **I2C1_RX_DMA_RMP:** I2C1_RX DMA remapping bit
This bit is set and cleared by software. It defines on which DMA1 channel I2C1_RX is mapped.
00: I2C1_RX mapped on DMA1 CH7
01: I2C1_RX mapped on DMA1 CH3
10: I2C1_RX mapped on DMA1 CH5
11: I2C1_RX mapped on DMA1 CH7

Bits 3:2 **SPI1_TX_DMA_RMP:** SPI1_TX DMA remapping bit
This bit is set and cleared by software. It defines on which DMA1 channel SPI1_TX is mapped.
00: SPI1_TX mapped on DMA1 CH3
01: SPI1_TX mapped on DMA1 CH5
10: SPI1_TX mapped on DMA1 CH7
11: SPI1_TX mapped on DMA1 CH3

Bits 1:0 **SPI1_RX_DMA_RMP:** SPI1_RX DMA remapping bit
This bit is set and cleared by software. It defines on which DMA1 channel SPI1_RXis mapped.
00: SPI1_RX mapped on DMA1 CH2
01: SPI1_RX mapped on DMA1 CH4
10: SPI1_RX mapped on DMA1 CH6
11: SPI1_RX mapped on DMA1 CH2

10.1.9 SYSCFG register map

Table 29. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SYSCFG_CFGR1**	FPU_IE[5..0]						Res	Res	ENCODER_MODE [1:0]		Res	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	DAC2_CH1_DMA_RMP	TIM7_DAC2_DMA_RMP	TIM6_DAC1_DMA_RMP	TIM17_DMA_RMP	TIM16_DMA_RMP	Res	Res	Res	DAC_TRIG_RMP	TIM1_ITR3_RMP	Res	Res	Res	Res	MEM_MODE	
	Reset value	1	1	1	1	1	0			0	0		0	0	0	0	0	0	0	0	0	0				0	0					X	X
0x04	**SYSCFG_RCR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PAGE[3:0]_WP			
	Reset value																													0	0	0	0
0x08	**SYSCFG_EXTICR1**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**SYSCFG_EXTICR2**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SYSCFG_EXTICR3**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**SYSCFG_EXTICR4**	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29. SYSCFG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	**SYSCFG_CFGR2**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SRAM_PEF	Res	Res	Res	BYP_ADDR_PAR	Res	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK
	Reset value																								0				0		0	0	0
. . .	. . .	.																															
0x50	**SYSCFG_CFGR3**	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DAC1_TRIG5_RMP	DAC1_TRIG3_RMP	Res.	Res	Res	Res	Res	Res	ADC2_DMA_RMP		I2C1_TX_DMA_RMP		I2C1_RX_DMA_RMP		SPI1_TX_DMA_RMP		SPI1_RX_DMA_RMP	
	Reset value															0	0							1	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

11 Direct memory access controller (DMA)

11.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There is one instance of DMA with 7 channels.

Each channel is dedicated to managing memory access requests from one or more peripherals. The DMA includes an arbiter for handling the priority between DMA requests.

11.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as Flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to 2^{16} - 1
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

11.3 DMA implementation

11.3.1 DMA

DMA is implemented with the hardware configuration parameters shown in the table below.

Table 30. DMA implementation

Feature	DMA
Number of channels	7

11.3.2 DMA request mapping

DMA controller

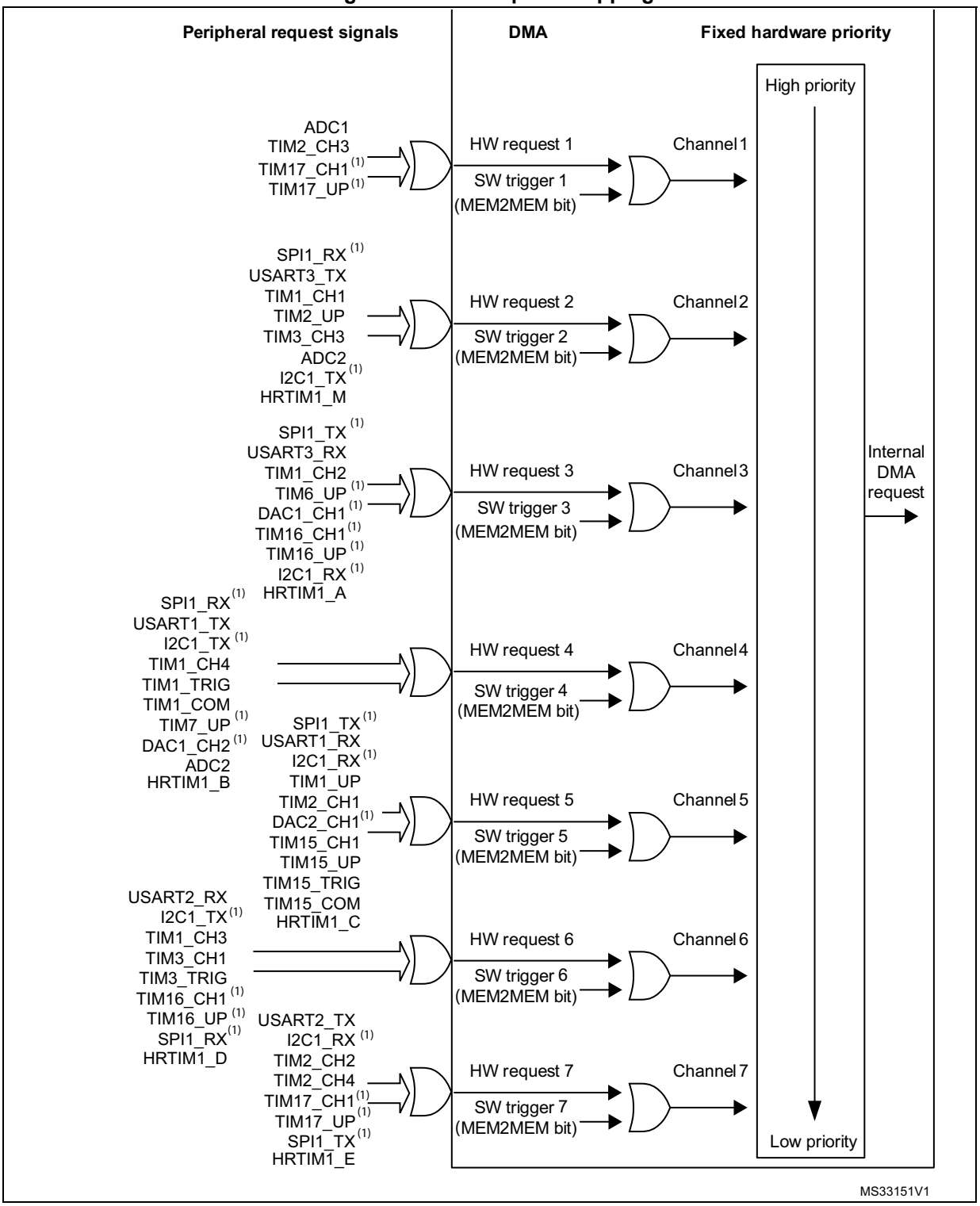
The hardware requests from the peripherals (TIMx(x=1...3, 6, 7, 15..17)HRTIM1,ADC1, ADC2, SPI1, I2C1, DAC1_Channel[1], DAC2_Channel[1] and USARTx (x=1..3)) are simply logically ORed before entering the DMA. This means that on one channel, only one request must be enabled at a time (see *Figure 19*).

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Caution: A same peripheral request can be assigned to two different channels only if the application ensures that these channels are not requested to be served at the same time. In other words, if two different channels receive a same asserted peripheral request at the same time, an unpredictable DMA hardware behavior occurs.

Table 31 lists the DMA requests for each channel.

Figure 19. DMA request mapping



1. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in *SYSCFG configuration register 1 (SYSCFG_CFGR1)* and *SYSCFG configuration register 3 (SYSCFG_CFGR3)*.

Table 31. DMA requests for each channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
ADC	ADC1	ADC2	-	ADC2(1)	-	-	-
SPI	-	SPI1_RX	SP1_TX	SPI1_RX(1)	SPI1_TX(1)	SPI1_RX(1)	SPI1_TX(1)
USART	-	USART3 _TX	USART3 _RX	USART1_TX	USART1 _RX	USART2 _RX	USART2_TX
I2C	-	I2C1_TX(1)	I2C1_RX(1)	I2C1_TX(1)	I2C1_RX(1)	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM6/DAC	-	-	TIM6_UP DAC1_CH1 (1)	-	-	-	-
TIM7/DAC	-	-	-	TIM7_UP DAC1_CH2 (1)	-	-	-
DAC	-	-	-	-	DAC2_CH1(1)	-	-
TIM15	-	-	-	-	TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM	-	-
TIM16	-	-	TIM16_CH1 TIM16_UP	-	-	TIM16_CH1 TIM16_UP(1)	-
TIM17	TIM17_CH1 TIM17_UP	-	-	-	-	-	TIM17_CH1 TIM17_UP(1)
HRTIM1	-	HRTIM1_M	HRTIM1_A	HRTIM1_B	HRTIM1_C	HRTIM1_D	HRTIM1_E

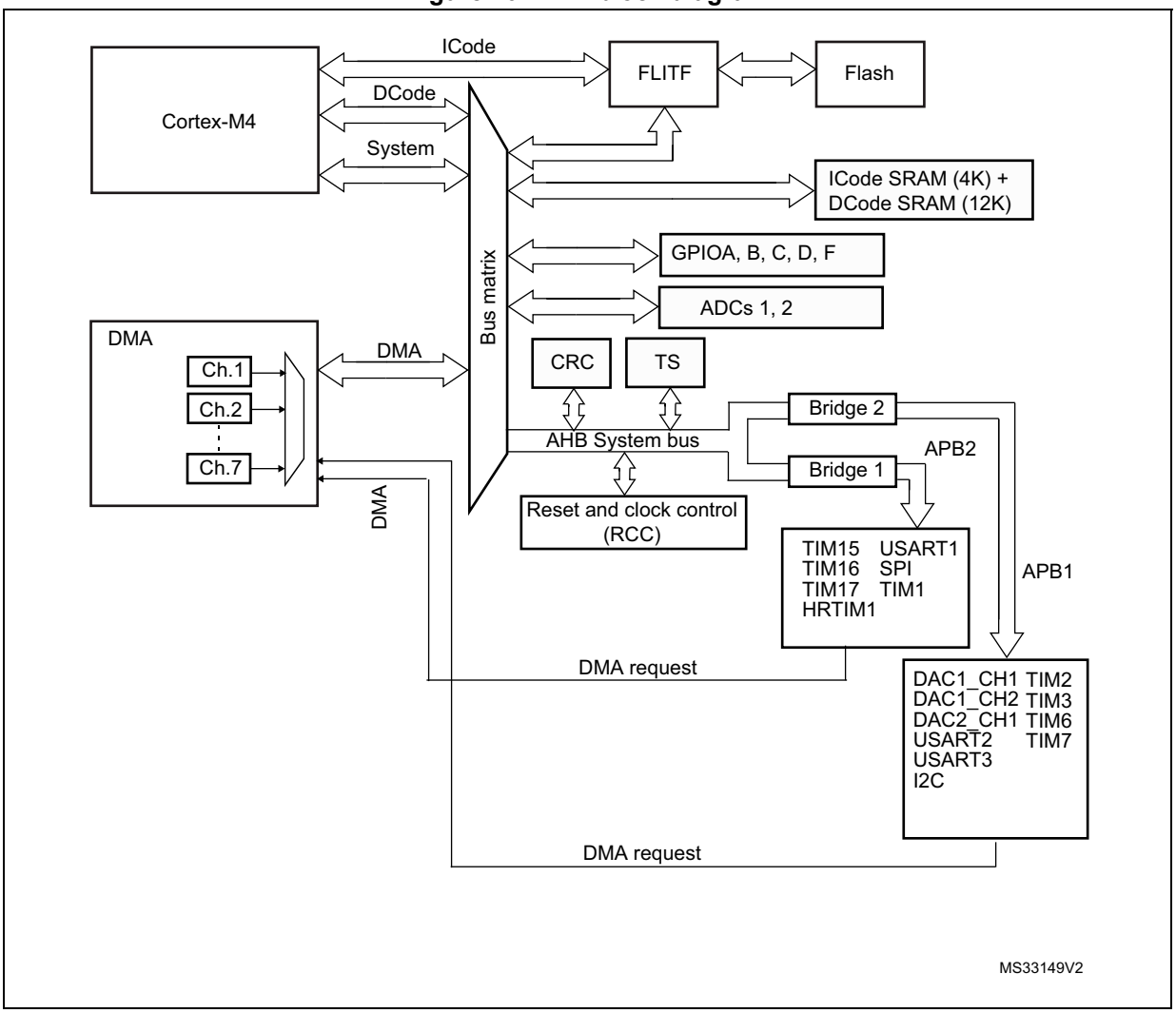
1. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in *SYSCFG configuration register 1 (SYSCFG_CFGR1)* and *SYSCFG configuration register 3 (SYSCFG_CFGR3)*

11.4 DMA functional description

11.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

Figure 20. DMA block diagram



The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates an interrupt per channel to the interrupt controller.

11.4.2 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.
  - a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA_CPARx or DMA_CMARx register.
- post-decrementing of the programmed DMA_CNDTRx register
  This register contains the remaining number of data items to transfer (number of AHB ‘read followed by write’ transfers).

This sequence is repeated until DMA_CNDTRx is null.

Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.

11.4.3 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB ‘read followed by write’ transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

11.4.4 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA_CCRx register.

Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA_CPARx or DMA_CMARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA_CNDTRx register.

Note: If the channel x is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In **circular mode**, after the last data transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx and DMA_CMARx registers.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the DMA_CPARx register.
   The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the DMA_CMARx register.
   The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA_CNDTRx register.
   After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA_CCRx register:
   – the channel priority
   – the data transfer direction
   – the circular mode
   – the peripheral and memory incremented mode
   – the peripheral and memory data size
   – the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

Note: The two last steps of the channel configuration procedure may be merged into a single access to the DMA_CCRx register, to configure and enable the channel.

Channel state and disabling a channel

A channel x in active state is an enabled channel (read DMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

  This corresponds to the two following actions:
  – An active channel is disabled by software (writing DMA_CCRx.EN = 0 whereas DMA_CCRx.EN = 1).
  – The software enables the channel again (DMA_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA_CNDTRx, DMA_CPARx and DMA_CMARx).

  This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.
- Stop and abort a channel

  If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA_CNDTRx

register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

  This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

  This is supported by the hardware if the following conditions are met:

  – The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.

  – The software must operate separated write accesses to the same DMA_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA_CCRx if a configuration change is needed. There are read-only DMA_CCRx register fields when DMA_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the DMA_ISR register is set.

Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA_CCRx register.

Note: The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA_CCRx register. When the circular mode is activated, the amount of data to transfer is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the DMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.

Memory-to-memory mode

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA_CNDTRx register reaches zero.

Note: The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA_CCRx register.

Peripheral-to-peripheral mode

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel

  This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the DMA channel

  The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA_CCRx register.

Programming transfer direction, assigning source/destination

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the DMA_PARx register, the PSIZE[1:0] field and PINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the DMA_MARx register, the MSIZE[1:0] field and MINC bit of the DMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

11.4.5 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in the table below.

Table 32. Programmable data width and endian behavior (when PINC = MINC = 1)

Source port width (MSIZE if DIR = 1, else PSIZE)	Destination port width (PSIZE if DIR = 1, else MSIZE)	Number of data items to transfer (NDT)	Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx)
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4 4: read BFBEBDBC[31:0] @0xC then write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8 4: read BFBEBDBC[31:0] @0xC then write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

11.4.6 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA_CCRx register.

The TEIFx bit of the DMA_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA_CCRx register is set.

The EN bit of the DMA_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.

11.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 33. DMA interrupt requests

Interrupt request	Interrupt event	Event flag	Interrupt enable bit
Channel x interrupt	Half transfer on channel x	HTIFx	HTIEx
	Transfer complete on channel x	TCIFx	TCIEx
	Transfer error on channel x	TEIFx	TEIEx
	Half transfer or transfer complete or transfer error on channel x	GIFx	-

11.6 DMA registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The DMA registers have to be accessed by words (32-bit).

11.6.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA_IFCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 27 **TEIF7**: transfer error (TE) flag for channel 7
0: no TE event
1: a TE event occurred

Bit 26 **HTIF7**: half transfer (HT) flag for channel 7
0: no HT event
1: a HT event occurred

Bit 25 **TCIF7**: transfer complete (TC) flag for channel 7
0: no TC event
1: a TC event occurred

Bit 24 **GIF7**: global interrupt flag for channel 7
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 23 **TEIF6**: transfer error (TE) flag for channel 6
0: no TE event
1: a TE event occurred

Bit 22 **HTIF6**: half transfer (HT) flag for channel 6
0: no HT event
1: a HT event occurred

Bit 21 **TCIF6**: transfer complete (TC) flag for channel 6
0: no TC event
1: a TC event occurred

Bit 20 **GIF6**: global interrupt flag for channel 6
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 19 **TEIF5**: transfer error (TE) flag for channel 5
0: no TE event
1: a TE event occurred

Bit 18 **HTIF5**: half transfer (HT) flag for channel 5
0: no HT event
1: a HT event occurred

Bit 17 **TCIF5**: transfer complete (TC) flag for channel 5
0: no TC event
1: a TC event occurred

Bit 16 **GIF5**: global interrupt flag for channel 5
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 15 **TEIF4**: transfer error (TE) flag for channel 4
0: no TE event
1: a TE event occurred

Bit 14 **HTIF4**: half transfer (HT) flag for channel 4
0: no HT event
1: a HT event occurred

Bit 13 **TCIF4**: transfer complete (TC) flag for channel 4
0: no TC event
1: a TC event occurred

Bit 12 **GIF4**: global interrupt flag for channel 4
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 11 **TEIF3**: transfer error (TE) flag for channel 3
0: no TE event
1: a TE event occurred

Bit 10 **HTIF3**: half transfer (HT) flag for channel 3
0: no HT event
1: a HT event occurred

Bit 9 **TCIF3**: transfer complete (TC) flag for channel 3
0: no TC event
1: a TC event occurred

Bit 8 **GIF3**: global interrupt flag for channel 3
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 7 **TEIF2**: transfer error (TE) flag for channel 2
0: no TE event
1: a TE event occurred

Bit 6 **HTIF2**: half transfer (HT) flag for channel 2
0: no HT event
1: a HT event occurred

Bit 5 **TCIF2**: transfer complete (TC) flag for channel 2
0: no TC event
1: a TC event occurred

Bit 4 **GIF2**: global interrupt flag for channel 2
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 3 **TEIF1**: transfer error (TE) flag for channel 1
0: no TE event
1: a TE event occurred

Bit 2 **HTIF1**: half transfer (HT) flag for channel 1
0: no HT event
1: a HT event occurred

Bit 1 **TCIF1**: transfer complete (TC) flag for channel 1
0: no TC event
1: a TC event occurred

Bit 0 **GIF1**: global interrupt flag for channel 1
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

11.6.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this DMA_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 27 **CTEIF7**: transfer error flag clear for channel 7

Bit 26 **CHTIF7**: half transfer flag clear for channel 7

Bit 25 **CTCIF7**: transfer complete flag clear for channel 7

Bit 24 **CGIF7**: global interrupt flag clear for channel 7

Bit 23 **CTEIF6**: transfer error flag clear for channel 6

Bit 22 **CHTIF6**: half transfer flag clear for channel 6

Bit 21 **CTCIF6**: transfer complete flag clear for channel 6

Bit 20 **CGIF6**: global interrupt flag clear for channel 6

Bit 19 **CTEIF5**: transfer error flag clear for channel 5

Bit 18 **CHTIF5**: half transfer flag clear for channel 5

Bit 17 **CTCIF5**: transfer complete flag clear for channel 5

Bit 16 **CGIF5**: global interrupt flag clear for channel 5

Bit 15 **CTEIF4**: transfer error flag clear for channel 4

Bit 14 **CHTIF4**: half transfer flag clear for channel 4

Bit 13 **CTCIF4**: transfer complete flag clear for channel 4

Bit 12 **CGIF4**: global interrupt flag clear for channel 4

Bit 11 **CTEIF3**: transfer error flag clear for channel 3

Bit 10 **CHTIF3**: half transfer flag clear for channel 3

Bit 9 **CTCIF3**: transfer complete flag clear for channel 3

Bit 8 **CGIF3**: global interrupt flag clear for channel 3

Bit 7 **CTEIF2**: transfer error flag clear for channel 2

Bit 6 **CHTIF2**: half transfer flag clear for channel 2

Bit 5 **CTCIF2**: transfer complete flag clear for channel 2

Bit 4 **CGIF2**: global interrupt flag clear for channel 2

Bit 3 **CTEIF1**: transfer error flag clear for channel 1

Bit 2 **CHTIF1**: half transfer flag clear for channel 1

Bit 1 **CTCIF1**: transfer complete flag clear for channel 1

Bit 0 **CGIF1**: global interrupt flag clear for channel 1

11.6.3 DMA channel x configuration register (DMA_CCRx)

Address offset: 0x08 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM**: memory-to-memory mode

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 13:12 **PL[1:0]**: priority level

00: low

01: medium

10: high

11: very high

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 11:10 **MSIZE[1:0]**: memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

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

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

n memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 5 **CIRC**: circular mode

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 4 **DIR**: data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

1: read from memory

- Destination attributes are defined by PSIZE and PINC, plus the DMA_CPARx register. This is still valid in a memory-to-memory mode.
- Source attributes are defined by MSIZE and MINC, plus the DMA_CMARx register. This is still valid in a peripheral-to-peripheral mode.

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 3 **TEIE**: transfer error interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 2 **HTIE**: half transfer interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 1 **TCIE**: transfer complete interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 0 **EN**: channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

0: disabled

1: enabled

Note: this bit is set and cleared by software.

11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx)

Address offset: 0x0C + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

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

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

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

Bits 15:0 **NDT[15:0]**: number of data to transfer (0 to 2^{16} - 1)

This field is updated by hardware when the channel is enabled:

- It is decremented after each single DMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

11.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 31:0 **PA[31:0]**: peripheral address

It contains the base address of the peripheral data register from/to which the data will be read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.

Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

11.6.6 DMA channel x memory address register (DMA_CMARx)

Address offset: 0x14 + 0x14 * (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **MA[31:0]**: peripheral address

It contains the base address of the memory from/to which the data will be read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

11.6.7 DMA register map

The table below gives the DMA register map and reset values.

Table 34. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DMA_IFCR	Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved	Reserved.																															
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved.																															
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved	Reserved.																															
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved.																															

Table 34. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved	Reserved.																															
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved	Reserved.																															
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

12 Interrupts and events

12.1 Nested vectored interrupt controller (NVIC)

12.1.1 NVIC main features

- 73 maskable interrupt channels (not including the sixteen Cortex-M4 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the PM0214 programming manual for Cortex-M4 products.

12.1.2 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (max $f_{HCLK}/8$).

12.1.3 Interrupt and exception vectors

Table 35 is the vector table for STM32F334xx devices.

Table 35. STM32F334xx vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-	3	settable	SVCall	System service call via SWI instruction	0x0000 002C

Table 35. STM32F334xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line 16 detection interrupt	0x0000 0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp interrupts through the EXTI line 19	0x0000 0048
3	10	settable	RTC_WKUP	RTC wakeup timer interrupts through the EXTI line 20	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2_TS	EXTI Line2 and Touch sensing interrupts	0x0000 0060
9	16	settable	EXTI3	EXTI Line3	0x0000 0064
10	17	settable	EXTI4	EXTI Line4	0x0000 0068
11	18	settable	DMA1_Channel1	DMA1 channel 1 interrupt	0x0000 006C
12	19	settable	DMA1_Channel2	DMA1 channel 2 interrupt	0x0000 0070
13	20	settable	DMA1_Channel3	DMA1 channel 3 interrupt	0x0000 0074
14	21	settable	DMA1_Channel4	DMA1 channel 4 interrupt	0x0000 0078
15	22	settable	DMA1_Channel5	DMA1 channel 5 interrupt	0x0000 007C
16	23	settable	DMA1_Channel6	DMA1 channel 6 interrupt	0x0000 0080
17	24	settable	DMA1_Channel7	DMA1 channel 7 interrupt	0x0000 0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
19	26	settable	CAN_TX	CAN_TX interrupts	0x0000 008C
20	27	settable	CAN_RX0	CAN_RX0 interrupts	0x0000 0090
21	28	settable	CAN_RX1	CAN_RX1 interrupt	0x0000 0094
22	29	settable	CAN_SCE	CAN_SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK/TIM15	TIM1 Break/TIM15 global interrupts	0x0000 00A0
25	32	settable	TIM1_UP/TIM16	TIM1 Update/TIM16 global interrupts	0x0000 00A4
26	33	settable	TIM1_TRG_COM /TIM17	TIM1 trigger and commutation/TIM17 interrupts	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0

Table 35. STM32F334xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	-	Reserved		0x0000 00B8
31	38	settable	I2C1_EV	I2C1 event interrupt & EXTI Line23 interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40	-	Reserved		0x0000 00C4
34	41	-	Reserved		0x0000 00C8
35	42	-	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	-	Reserved		0x0000 00D0
37	44	settable	USART1	USART1 global interrupt & EXTI Line 25	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt & EXTI Line 26	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt & EXTI Line 28	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC alarm interrupt	0x0000 00E4
42	49	-	Reserved		0x0000 00E8
43	50	-	Reserved		0x0000 00EC
44	51	-	Reserved		0x0000 00F0
45	52	-	Reserved		0x0000 00F4
46	53	-	Reserved		0x0000 00F8
47	54	-	Reserved		0x0000 00FC
48	55	-	Reserved		0x0000 0100
49	56	-	Reserved		0x0000 0104
50	57	-	Reserved		0x0000 0108
51	58	-	Reserved		0x0000 010C
52	59	-	Reserved		0x0000 0110
53	60	-	Reserved		0x0000 0114
54	61	settable	TIM6_DAC1	TIM6 global and DAC1 underrun interrupts	0x0000 0118
55	62	settable	TIM7_DAC2	TIM7 global and DAC2 underrun interrupt	0x0000 011C
56	63	-	Reserved		0x0000 0120
57	64	-	Reserved		0x0000 0124
58	65	-	Reserved		0x0000 0128
59	66	-	Reserved		0x0000 012C
60	67	-	Reserved		0x0000 0130
61	68	-	Reserved		0x0000 0134

Table 35. STM32F334xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
62	69	-	Reserved		0x0000 0138
63	70	-	Reserved		0x0000 013C
64	71	settable	COMP2	COMP2 interrupt combined with EXTI Lines 22 interrupt.	0x0000 0140
65	72	settable	COMP4_6	COMP4 & COMP6 interrupts combined with EXTI Lines 30 and 32 interrupts respectively.	0x0000 0144
66	73	-	Reserved		0x0000 0148
67	74	-	HRTIM_Master_IRQn	HRTIM master timer interrupt	0x0000 014C
68	75	-	HRTIM_TIMA_IRQn	HRTIM timer A interrupt	0x0000 0150
69	76	-	HRTIM_TIMB_IRQn	HRTIM timer B interrupt	0x0000 0154
70	77	-	HRTIM_TIMC_IRQn	HRTIM timer C interrupt	0x0000 0158
71	78	-	HRTIM_TIMD_IRQn	HRTIM timer D interrupt	0x0000 015C
72	79	-	HRTIM_TIME_IRQn	HRTIM timer E interrupt	0x0000 0160
73	80	-	HRTIM_TIM_FLT_IRQn	HRTIM fault interrupt	0x0000 0164
74	81	-	Reserved		0x0000 0168
75	82	-	Reserved		0x0000 016C
76	83	-	Reserved		0x0000 0170
77	84	-	Reserved		0x0000 0174
78	85	-	Reserved		0x0000 0178
79	86	-	Reserved		0x0000 017C
80	87	-	Reserved		0x0000 0180
81	88	settable	FPU	Floating point interrupt	0x0000 0184

12.2 Extended interrupts and events controller (EXTI)

The extended interrupts and events controller (EXTI) manages the external and internal asynchronous events/interrupts and generates the event request to the CPU/Interrupt Controller and a wake-up request to the Power Manager.

The EXTI allows the management of up to 36 external/internal event line (28 external event lines and 8 internal event lines).

The active edge of each external interrupt line can be chosen independently, whilst for internal interrupt the active edge is always the rising one. An interrupt could be left pending: in case of an external one, a status register is instantiated and indicates the source of the interrupt; an event is always a simple pulse and it's used for triggering the core wake-up. For internal interrupts, the pending status is assured by the generating peripheral, so no need

for a specific flag. Each input line can be masked independently for interrupt or event generation, in addition the internal lines are sampled only in STOP mode. This controller allows also to emulate the (only) external events by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

12.2.1 Main features

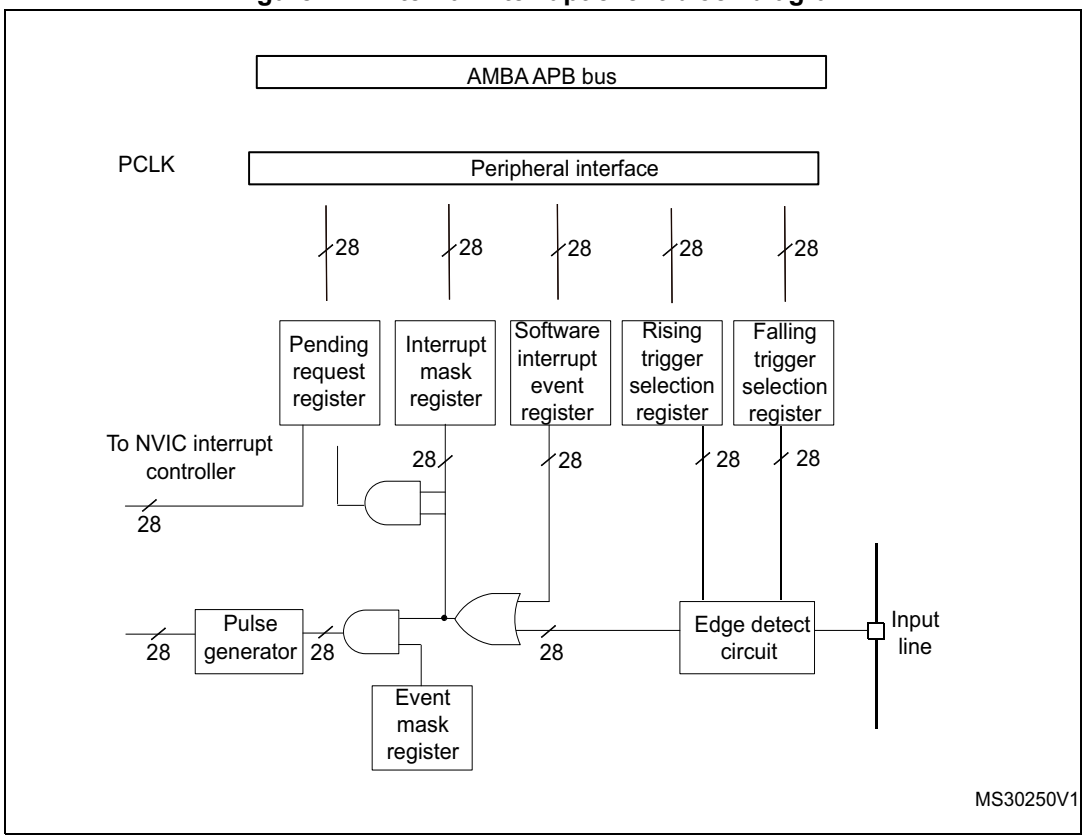
The EXTI main features are the following:

- support generation of up to 36 event/interrupt requests
- Independent configuration of each line as an external or an internal event requests
- Independent mask on each event/interrupt line
- Automatic disable of internal lines when system is not in STOP mode
- Independent trigger for external event/interrupt line
- Dedicated status bit for external interrupt line
- Emulation for all the external event requests.

12.2.2 Block diagram

The extended interrupt/event block diagram is shown in the following figure.

Figure 21. External interrupt/event block diagram



12.2.3 Wakeup event management

STM32F334xx devices are able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

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

12.2.4 Asynchronous Internal Interrupts

Some communication peripherals (UART, I2C) are able to generate events when the system is in run mode and also when the system is in stop mode allowing to wake up the system from stop mode.

To accomplish this, the peripheral is asked to generate both a synchronized (to the system clock, e.g. APB clock) and an asynchronous version of the event.

12.2.5 Functional description

For the external interrupt lines, to generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a 1 in the pending register.

For the internal interrupt lines, the active edge is always the rising edge, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

For the external lines, an interrupt/event request can also be generated by software by writing a 1 in the software interrupt/event register.

Note: The interrupts or events associated to the internal lines can be triggered only when the system is in STOP mode. If the system is still running, no interrupt/event is generated.

Hardware interrupt selection

To configure a line as interrupt source, use the following procedure:

- Configure the corresponding mask bit in the EXTI_IMR register.
- Configure the Trigger Selection bits of the Interrupt line (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI line can be correctly acknowledged.

Hardware event selection

To configure a line as event source, use the following procedure:

- Configure the corresponding mask bit in the EXTI_EMR register.
- Configure the Trigger Selection bits of the Event line (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

Any of the external lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

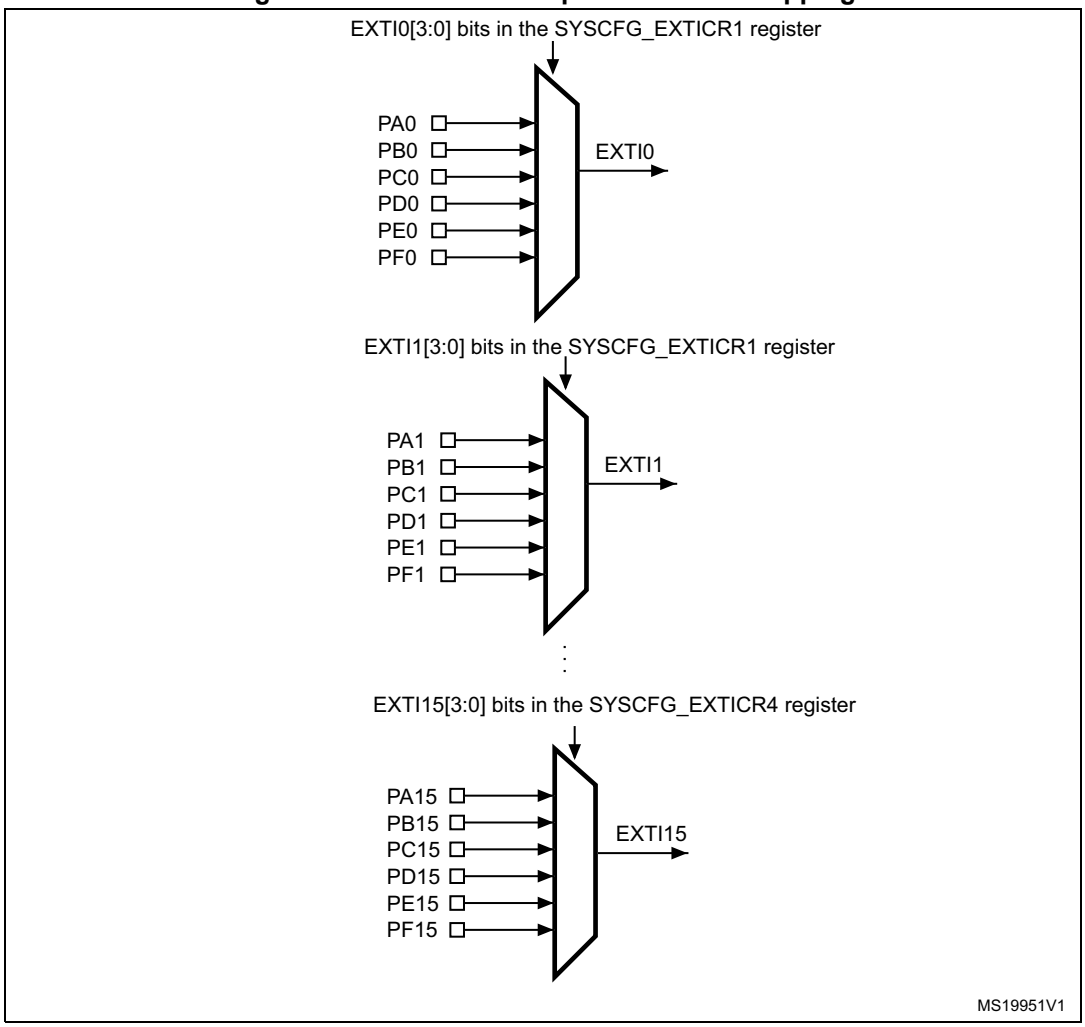
- Configure the corresponding mask bit (EXTI_IMR, EXTI_EMR)
- Set the required bit of the software interrupt register (EXTI_SWIER)

12.2.6 External and internal interrupt/event line mapping

36 interrupt/event lines are available: 8 lines are internal (including the reserved ones); the remaining 28 lines are external.

The GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Figure 22. External interrupt/event GPIO mapping



The remaining lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is reserved
- EXTI line 19 is connected to RTC tamper and Timestamps
- EXTI line 20 is connected to RTC wakeup timer
- EXTI line 21 is reserved
- EXTI line 22 is connected to Comparator 2 output
- EXTI line 23 is connected to I2C1 wakeup
- EXTI line 24 is reserved
- EXTI line 25 is connected to USART1 wakeup
- EXTI line 26 is reserved
- EXTI line 27 is reserved
- EXTI line 28 is reserved
- EXTI line 29 is reserved
- EXTI line 30 is connected to Comparator 4 output
- EXTI line 31 is reserved
- EXTI line 32 is connected to Comparator 6 output

Note: EXTI lines 23 and 25 are internal.

12.3 EXTI registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

12.3.1 Interrupt mask register (EXTI_IMR1)

Address offset: 0x00

Reset value: 0xBFA4 0000 (See note below)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MR30	Res.	Res.	Res.	Res.	MR25	Res.	MR23	MR22	Res.	MR20	MR19	Res.	MR17	MR16
	rw					rw		rw	rw		rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MRx:** Interrupt Mask on external/internal line x (x = 30)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

Bits 29:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Note: The reset value for the internal lines (23 and 25) and reserved lines is set to '1'.

12.3.2 Event mask register (EXTI_EMR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MR30	Res.	Res.	Res.	Res.	MR25	Res.	MR23	MR22	Res.	MR20	MR19	Res.	MR17	MR16
	rw					rw		rw	rw		rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

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

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

12.3.3 Rising trigger selection register (EXTI_RTSR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TR30	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR22	Res.	TR20	TR19	Res.	TR17	TR16
	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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TRx:** Rising trigger event configuration bit of line x (x = 31 to 29)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **TRx:** Rising trigger event configuration bit of line x (x = 22)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **TRx:** Rising trigger event configuration bit of line x (x = 20 to 19)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **TRx:** Rising trigger event configuration bit of line x (x = 17 to 0)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a rising edge on an external interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

12.3.4 Falling trigger selection register (EXTI_FTSR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TR30	Res	Res.	Res.	Res.	Res.	Res.	Res.	TR22	Res.	TR20	TR19	Res.	TR17	TR16
	rw								rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TRx:** Falling trigger event configuration bit of line x (x = 30)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **TRx:** Falling trigger event configuration bit of line x (x = 22)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **TRx:** Falling trigger event configuration bit of line x (x = 20 to 19)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **TRx:** Falling trigger event configuration bit of line x (x = 17 to 0)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a falling edge on an external interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

12.3.5 Software interrupt event register (EXTI_SWIER1)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SWIERx:** Software interrupt on line x (x = 30)
If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.
This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a ‘1’ into the bit).

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **SWIERx:** Software interrupt on line x (x = 22)
If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.
This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a ‘1’ into the bit).

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **SWIERx:** Software interrupt on line x (x = 20 to 19)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a ‘1’ into the bit).

Bit 18 Reserved, must be kept at reset value.

12.3.6 Pending register (EXTI_PR1)

Address offset: 0x14

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PR30	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR22	Res.	PR20	PR19	Res.	PR17	PR16
	rc_w1								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PRx:** Pending bit on line x (x = 30)

0: No trigger request occurred
1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ to the bit.

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **PRx:** Pending bit on line x (x = 22)

0: No trigger request occurred
1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ to the bit.

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **PRx:** Pending bit on line x (x = 20 to 19)

0: No trigger request occurred
1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ to the bit.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **PRx:** Pending bit on line x (x = 17 to 0)

0: No trigger request occurred
1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ to the bit.

12.3.7 Interrupt mask register (EXTI_IMR2)

Address offset: 0x20

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR32
															rw

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

Bit 0 **MRx:** Interrupt mask on EXTI line 32

0: Event request from Line 32 is masked
1: Event request from Line 32 is not masked

12.3.8 Event mask register (EXTI_EMR2)

Address offset: 0x24

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR32
															rw

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

Bit 0 **MR32:** Event mask on EXTI line 32

0: Event request from Line 32 is masked
1: Event request from Line 32 is not masked

12.3.9 Rising trigger selection register (EXTI_RTSR2)

Address offset: 0x28

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR32
															rw

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

Bit 0 **TRx:** Rising trigger event configuration bit of line x (x = 32)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a rising edge on an external interrupt line occurs during a write operation to the EXTI_RTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

12.3.10 Falling trigger selection register (EXTI_FTSR2)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR32
															rw

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

Bit 0 **TR32:** Falling trigger event configuration bit of line 32
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a falling edge on an external interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.r

12.3.11 Software interrupt event register (EXTI_SWIER2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 32
															rw

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

Bit 0 **SWIER32:** Software interrupt on line 32

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR register (by writing a ‘1’ into the bit).

12.3.12 Pending register (EXTI_PR2)

Address offset: 0x34

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR32
															rc_w1

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

Bit 0 **PRx:** Pending bit on line x (x = 32)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ into the bit.

12.3.13 EXTI register map

Table 36. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_IMR1**	Res.	MR[31:0]	Res.	Res.	Res.	Res.	MR[31:0]	Res.	MR[31:0]	MR[31:0]	Res.	MR[31:0]	MR[31:0]	Res.	MR[31:0]																	
	Reset value	0	0	0	1	1	1	1		1	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_EMR1**	Res.	MR[31:0]	Res.	Res.	Res.	Res.	MR[31:0]	Res.	MR[31:0]	MR[31:0]	Res.	MR[31:0]	MR[31:0]	Res.											MR[31:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_RTSR1**	TR[31:29]			Res.	Res.	Res.	Res	Res.	Res.	TR[22:0]																						
	Reset value	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36. External interrupt/event controller register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**EXTI_RTSR1**	Res.	TR[31:29]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[22:0]	Res.	TR[22:0]	TR[22:0]	Res.	TR[22:0]																	
	Reset value		0								0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR1**	TR[31:29]			Res.	Res.	Res.	Res.	Res.	Res.	TR[22:0]																						
	Reset value	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR1**	Res.	TR[31:29]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[22:0]	Res.	TR[22:0]	TR[22:0]	Res.	TR[22:0]																	
	Reset value		0								0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER1**	SWIER [31:29]			Res.	Res.	Res.	Res.	Res.	Res.	SWIER[22:0]																						
	Reset value	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER1**	Res.	SWIER[31:29]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER[31:29]	Res.	SWIER[31:29]	SWIER[31:29]	Res.	SWIER[22:0]																	
	Reset value		0								0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR1**	PR [31:29]			Res.	Res.	Res.	Res.	Res.	Res.	PR[22:0]																						
	Reset value	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR1**	Res.	PR[31:29]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[31:29]	Res.	PR[31:29]	PR[31:29]	Res.	PR[22:0]																	
	Reset value		0								0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**EXTI_IMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR32
	Reset value																																0
0x24	**EXTI_EMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR32
	Reset value																																0
0x28	**EXTI_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR32
	Reset value																																0
0x2C	**EXTI_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR32
	Reset value																																0

Table 36. External interrupt/event controller register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**EXTI_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER32
	Reset value																																0
0x34	**EXTI_PR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR32
	Reset value																																0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

13 Analog-to-digital converters (ADC)

13.1 Introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 18 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

13.2 ADC main features

- High-performance features
  - 2x ADC, each can operate in dual mode.
  - ADC1 is connected to 11 external channels + 3 internal channels
  - ADC2 is connected to 14 external channels + 3 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time:
    Fast channels: 0.19 µs for 12-bit resolution (5.1 Ms/s)
    Slow channels: 0.21 µs for 12-bit resolution (4.8 Ms/s)
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution: 0.16 µs for 10-bit resolution
  - Can manage Single-ended or differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by GP-DMA for regular channel conversions
  - 4 dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance (0.19 µs conversion time for fast channels can be kept whatever the AHB bus clock frequency)
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- External analog input channels for each of the 2 ADCs:
  - Up to 5 fast channels from dedicated GPIO pads
  - Up to 11 slow channels from dedicated GPIO pads
- In addition, there are four internal dedicated channels:
  - One from internal temperature sensor (V_{TS}), connected to ADC1
  - One from $V_{BAT}/2$, connected to ADC1
  - One from the internal reference voltage (V_{REFINT}), connected to the two ADCs
  - One from OPAMP2 reference voltage output (VREFOPAMP2), connected to ADC2
- Start-of-conversion can be initiated:
  - by software for both regular and injected conversions
  - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions

- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Dual ADC mode
- Interrupt generation at the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC supply requirements: 2.0 V to 3.6 V
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

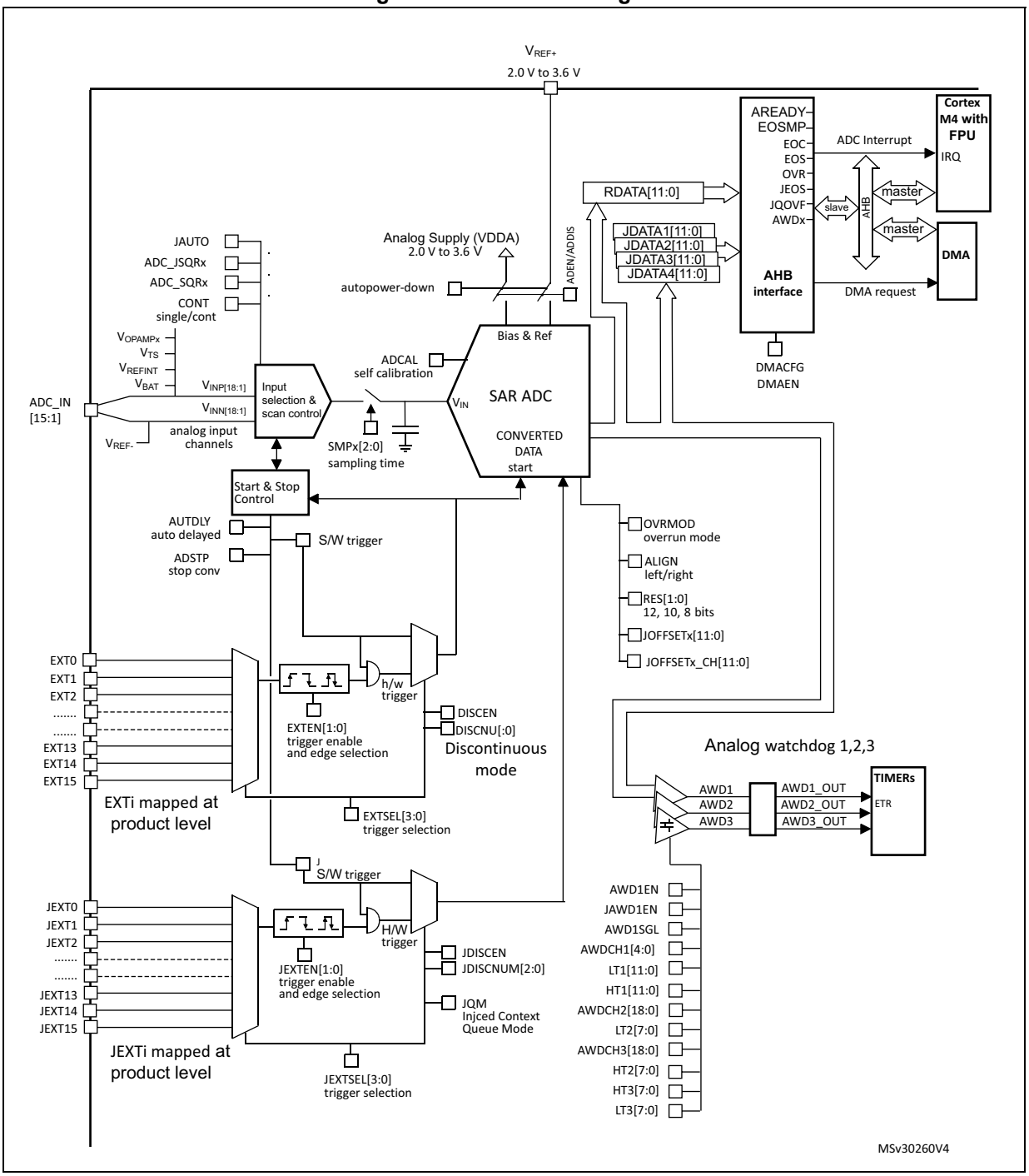
Figure 23 shows the block diagram of one ADC.

13.3 ADC functional description

13.3.1 ADC block diagram

Figure 23 shows the ADC block diagram and *Table 38* gives the ADC pin description.

Figure 23. ADC block diagram



13.3.2 Pins and internal signals

Table 37. ADC internal signals

Internal signal name	Signal type	Description
EXT[15:0]	Inputs	Up to 16 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
JEXT[15:0]	Inputs	Up to 16 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
ADC1_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
$V_{REFOPAMP2}$	Input	Reference voltage output from internal operational amplifier 2
V_{TS}	Input	Output voltage from internal temperature sensor
V_{REFINT}	Input	Output voltage from internal reference voltage
V_{BAT}	Input supply	External battery voltage supply

Table 38. ADC pins

Name	Signal type	Comments
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC, 2.0 V ≤ V_{REF+} ≤ V_{DDA}
VDDA	Input, analog supply	Analog power supply equal V_{DDA}: 2.0V ≤ V_{DDA} ≤ 3.6 V
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
VSSA	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
V_{INP}[18:1]	Positive input analog channels for each ADC	Connected either to external channels: ADC_IN*i* or internal channels.
V_{INN}[18:1]	Negative input analog channels for each ADC	Connected to V_{REF-} or external channels: ADC_IN*i-1*
ADCx_IN15:1	External analog input signals	Up to 16 analog input channels (x = ADC number = 1 or 2): – 5 fast channels – 10 slow channels

13.3.3 Clocks

Dual clock domain architecture

The dual clock-domain architecture means that each ADC clock is independent from the AHB bus clock.

The input clock of the two ADCs (master and slave) can be selected between two different clock sources (see *Figure 24: ADC clock scheme*):

a) The ADC clock can be a specific clock source, named “ADCxy_CK (xy=12 or 34) which is independent and asynchronous with the AHB clock”.

It can be configured in the RCC to deliver up to 72 MHz (PLL output). Refer to RCC Section for more information on generating ADC12_CK.

To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be reset.

b) The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

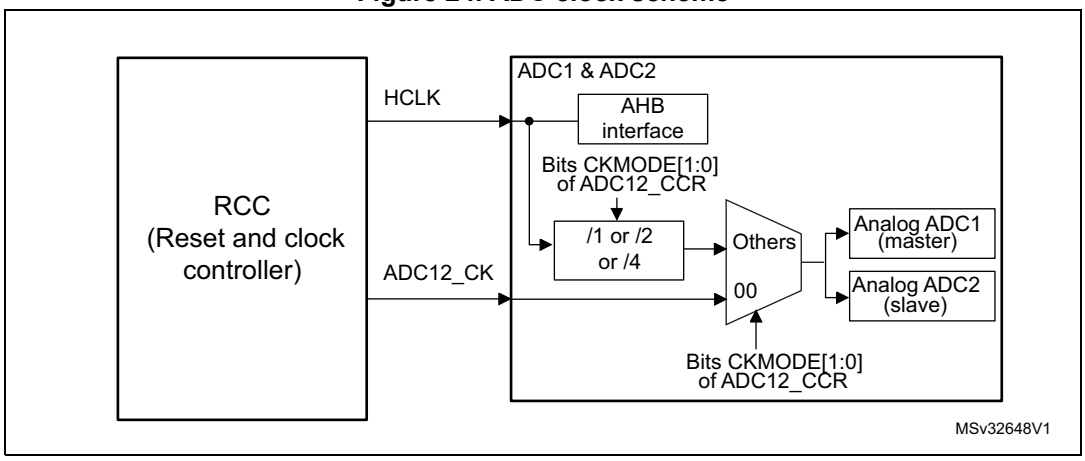
To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from “00”.

Note: Software can use option b) by writing CKMODE[1:0]=01 only if the AHB prescaler of the RCC is set to 1 (the duty cycle of the AHB clock must be 50% in this configuration).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits ADCxPRES[4:0] in register RCC_CFGR2 (Refer to *Section 8: Reset and clock control (RCC)*).

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

Figure 24. ADC clock scheme



1. Refer to the RCC section to see how HCLK and ADC12_CK can be generated.

Clock ratio constraint between ADC clock and AHB clock

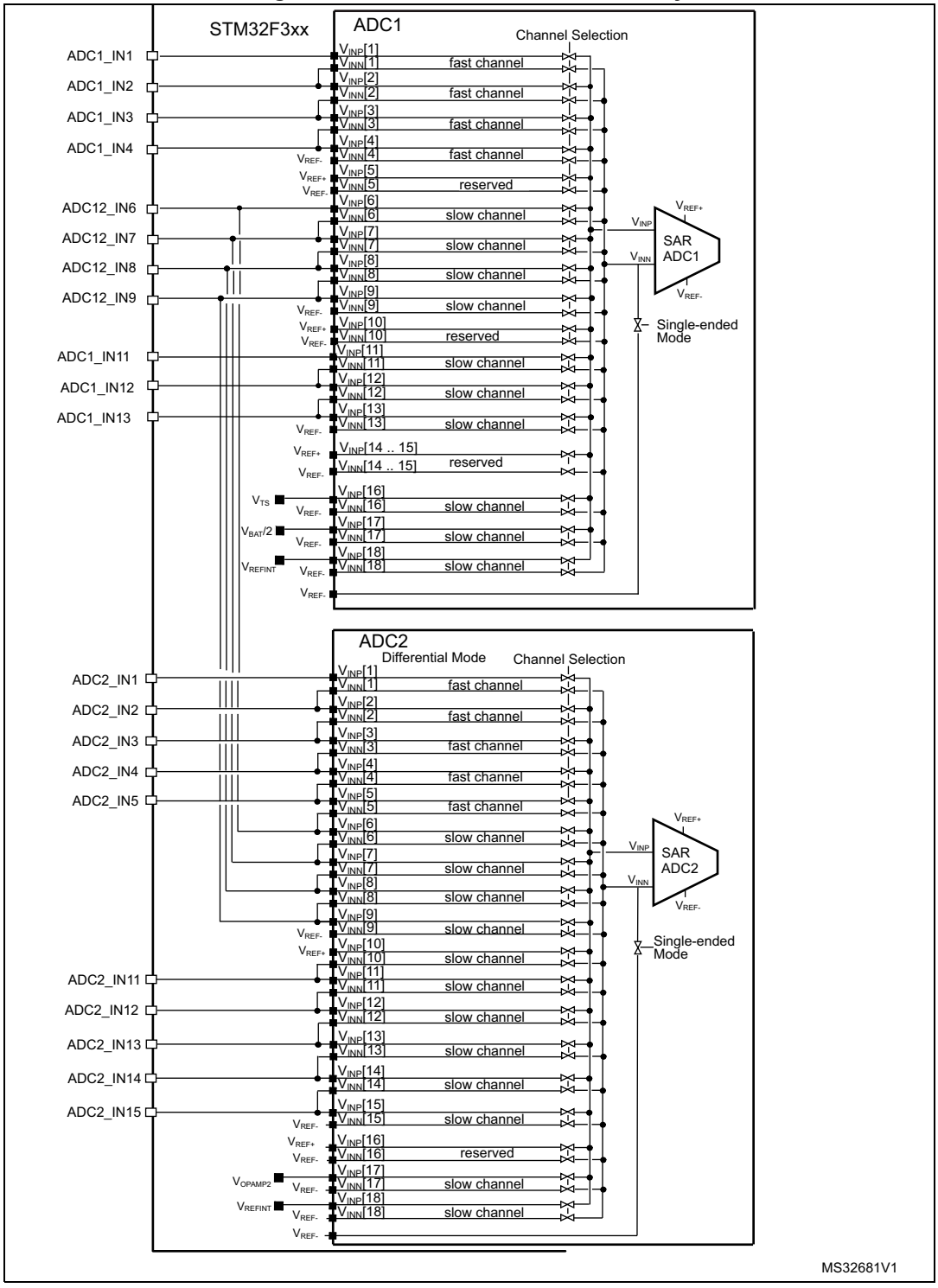
There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- F_{HCLK} >= F_{ADC} / 4 if the resolution of all channels are 12-bit or 10-bit
- F_{HCLK} >= F_{ADC} / 3 if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- F_{HCLK} >= F_{ADC} / 2 if there are some channels with resolutions equal to 6-bit

13.3.4 ADC1/2 connectivity

ADC1 and ADC2 are tightly coupled and share some external channels as described in *Figure 25*.

Figure 25. ADC1 and ADC2 connectivity



13.3.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

13.3.6 ADC voltage regulator (ADVREGEN)

The sequence below is required to start ADC operations:

1. Enable the ADC internal voltage regulator (refer to the ADC voltage regulator enable sequence).
2. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This temporization must be implemented by software. $T_{ADCVREG\_STUP}$ is equal to 10 µs in the worst case process/temperature/power supply.

After ADC operations are complete, the ADC is disabled (ADEN=0).

It is possible to save power by disabling the ADC voltage regulator (refer to the ADC voltage regulator disable sequence).

Note: When the internal voltage regulator is disabled, the internal analog calibration is kept.

ADVREG enable sequence

To enable the ADC voltage regulator, perform the sequence below:

1. Change ADVREGEN[1:0] bits from ‘10’ (disabled state, reset state) into ‘00’.
2. Change ADVREGEN[1:0] bits from ‘00’ into ‘01’ (enabled state).

ADVREG disable sequence

To disable the ADC voltage regulator, perform the sequence below:

1. Change ADVREGEN[1:0] bits from ‘01’ (enabled state) into ‘00’.
2. Change ADVREGEN[1:0] bits from ‘00’ into ‘10’ (disabled state)

13.3.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[15:1] in the ADCx_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0). Note that DIFSEL[18:16] are fixed to single ended channels (internal channels only) and are always read as 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage ADC_IN*i* (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage ADC_IN*i* (positive input) and ADC_IN*i+1* (negative input).

For a complete description of how the input channels are connected for each ADC, refer to *Figure 25: ADC1 and ADC2 connectivity on page 218*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage is connected to ADC_IN*i+1*. As a consequence, channel “*i+1*” is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1 and ADC2: this can make the channel on the other ADC unusable. Only exception is interleave mode for ADC master and the slave.

Example: Configuring ADC1_IN5 in differential input mode will make ADC12_IN6 not usable: in that case, the channels 6 of both ADC1 and ADC2 must never be converted.

Note: Channels 16, 17 and 18 of ADC1 and channels 17 and 18 of ADC2 are connected to internal analog channels and are internally fixed to single-ended inputs configuration (corresponding bits DIFSEL[i] is always zero). Channel 15 of ADC1 is also an internal channel and the user must configure the corresponding bit DIFSEL[15] to zero.

13.3.8 Calibration (ADCAL, ADCALDIF, ADCx_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADCx_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is kept if the ADC is disabled (ADEN=0). When the ADC operating conditions change (V_{REF+} changes are the main contributor to ADC offset variations, V_{DDA} and temperature change to a lesser extent), it is recommended to re-run a calibration cycle.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADCx_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

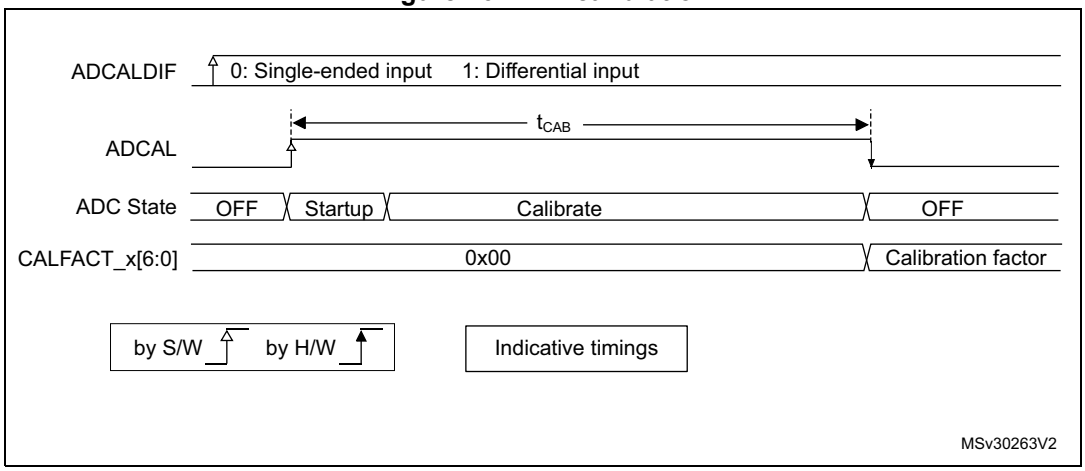
The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor

will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software procedure to calibrate the ADC

1. Ensure ADVREGEN[1:0]=01 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (Single-ended input) or ADCALDIF=1 (Differential input).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The calibration factor can be read from ADCx_CALFACT register.

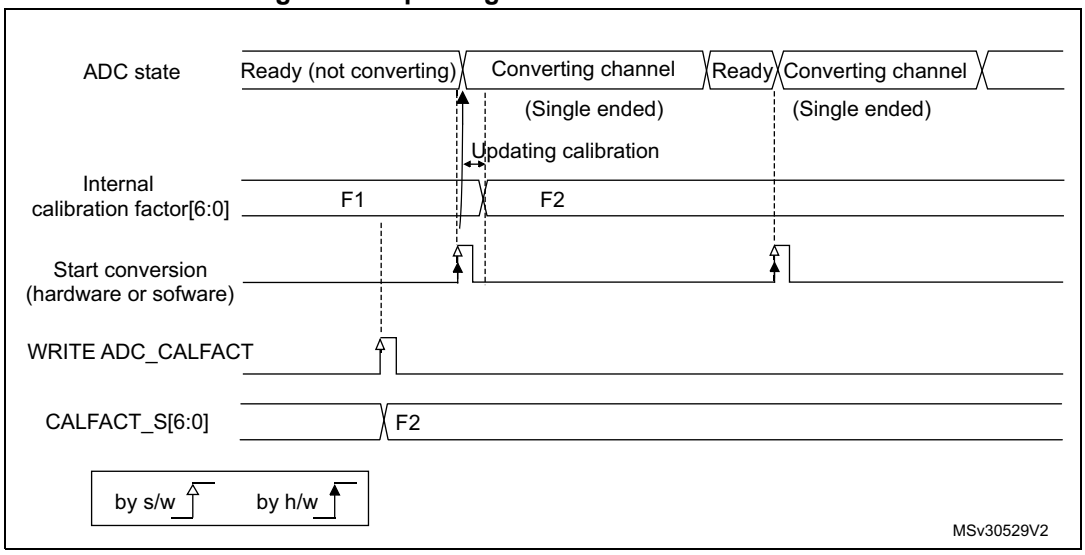
Figure 26. ADC calibration



Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 27. Updating the ADC calibration factor

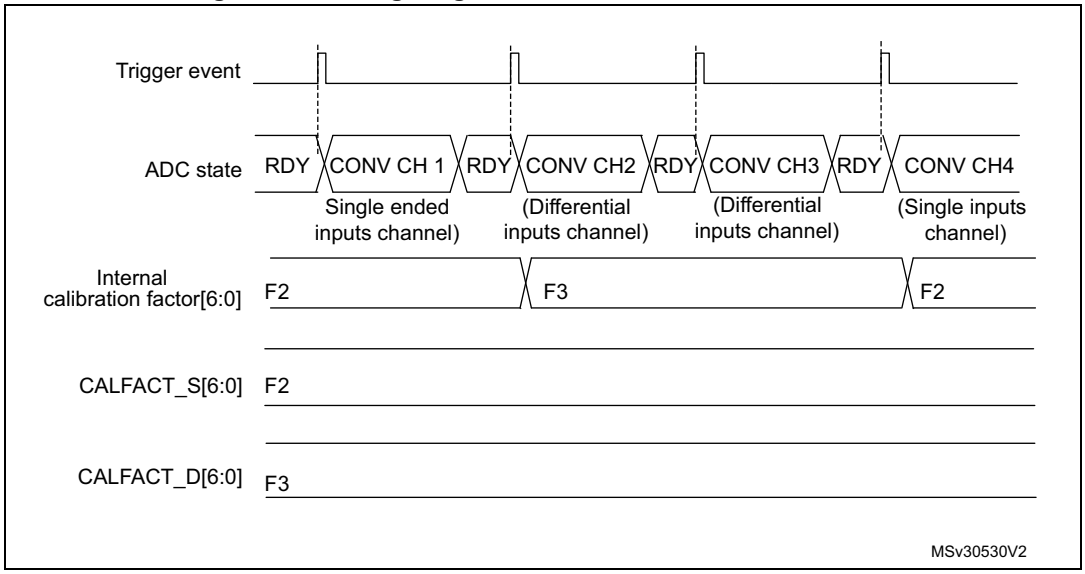


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in Differential input modes (with ADCALDIF=1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 28. Mixing single-ended and differential channels



13.3.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 13.3.6: ADC voltage regulator (ADVREGEN)*).

Once ADVREGEN[1:0] = 01, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 29*. Two control bits enable or disable the ADC:

- ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS=1 disables the ADC and disable the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to *Section 13.3.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable the ADC

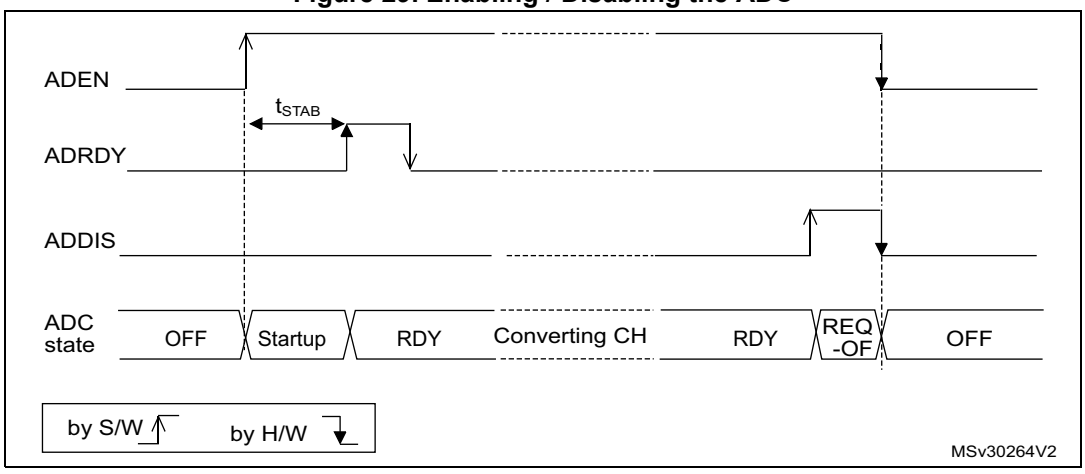
1. Set ADEN=1.
2. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).

Note: ADEN bit cannot be set during ADCAL=1 and 4 ADC clock cycle after the ADCAL bit is cleared by hardware(end of the calibration).

Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

Figure 29. Enabling / Disabling the ADC



13.3.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the control bits DIFSEL in the ADCx_DIFSEL register and the control bits ADCAL and ADEN in the ADCx_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADCx_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADCx_CFGR, ADCx_SMPRx, ADCx_TRx, ADCx_SQRx, ADCx_JDRy, ADCx_OFRy, ADCx_OFCHR and ADCx_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the control bits ADSTP or JADSTP of the ADCx_CR register only if the ADC is enabled and eventually converting and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADCx_JSQR at any time, when the ADC is enabled (ADEN=1).

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADCx_CR register).

13.3.11 Channel selection (SQRx, JSQRx)

There are up to 18 multiplexed channels per ADC:

- 5 fast analog inputs coming from GPIO pads (ADC_IN1..5)
- Up to 10 slow analog inputs coming from GPIO pads (ADC_IN5..15). Depending on the products, not all of them are available on GPIO pads.
- ADC1 is connected to 4 internal analog inputs:
  - ADC1_IN16 = V_{TS} = temperature sensor
  - ADC1_IN17 = $V_{BAT}/2$ = V_{BAT} channel
  - ADC1_IN18 = V_{REFINT} = internal reference voltage (also connected to ADC2_IN18).
- ADC2_IN17 = $V_{REFOPAMP2}$ = reference voltage for the operational amplifier 2

Warning: The user must ensure that only one of the two ADCs is converting V_{REFINT} at the same time (it is forbidden to have several ADCs converting V_{REFINT} at the same time).

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, TSEN or VBATEN in the ADCx_CCR *registers.*

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADCx_SQR registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADCx_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADCx_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADCx_JSQR register.

ADCx_SQR registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to *Section 13.3.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

It is possible to modify the ADCx_JSQR registers on-the-fly while injected conversions are occurring. Refer to *Section 13.3.21: Queue of context for injected conversions*

13.3.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADCx_SMPR1 and ADCx_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 4.5 ADC clock cycles
- SMP = 011: 7.5 ADC clock cycles
- SMP = 100: 19.5 ADC clock cycles
- SMP = 101: 61.5 ADC clock cycles
- SMP = 110: 181.5 ADC clock cycles
- SMP = 111: 601.5 ADC clock cycles

The total conversion time is calculated as follows:

Tconv = Sampling time + 12.5 ADC clock cycles

Example:

With F_{ADC_CLK} = 72 MHz and a sampling time of 1.5 ADC clock cycles:

Tconv = (1.5 + 12.5) ADC clock cycles = 14 ADC clock cycles = 0.194 µs (for fast channels)

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC charateristics section of the datasheets.

13.3.13 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADCx_CR register (for a regular channel)
- Setting the JADSTART bit in the ADCx_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADCx_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

13.3.14 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADCx_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).

13.3.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN /= 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN /= 0x0

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT=0, EXTSEL=0x0)
  - at any end of regular conversion sequence (EOS assertion) or at any end of sub-group processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
  - after execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTSEL /=0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- in single mode with software injected trigger (JEXTSEL=0x0)
  - at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN = 1
- in all cases (JEXTSEL=x)
  - after execution of the JADSTP procedure asserted by the software.

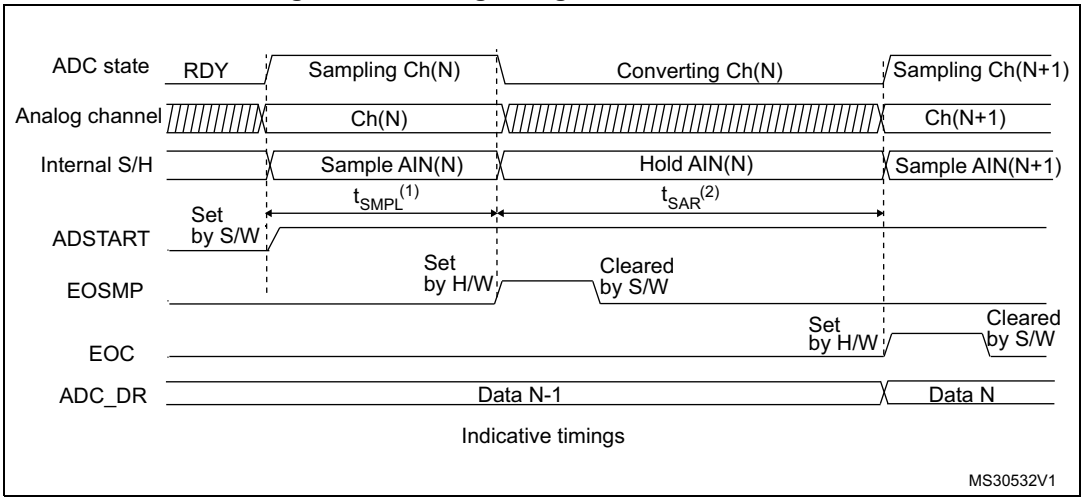
13.3.16 Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{ADC} = T_{SMPL} + T_{SAR} = [\ 1.5\ _{|min} + 12.5\ _{|12bit}\ ] \times T_{ADC\_CLK}$$

$$T_{ADC} = T_{SMPL} + T_{SAR} = 20.83\ ns\ _{|min} + 173.6\ ns\ _{|12bit} = 194.4\ ns\ (for\ F_{ADC\_CLK} = 72\ MHz)$$

Figure 30. Analog to digital conversion time



1. T_{SMPL} depends on SMP[2:0]
2. T_{SAR} depends on RES[2:0]

13.3.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADCx_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADCx_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would re-start a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must wait until ADSTART = 0 (or JADSTART = 0) before starting a new conversion.

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 31. Stopping ongoing regular conversions

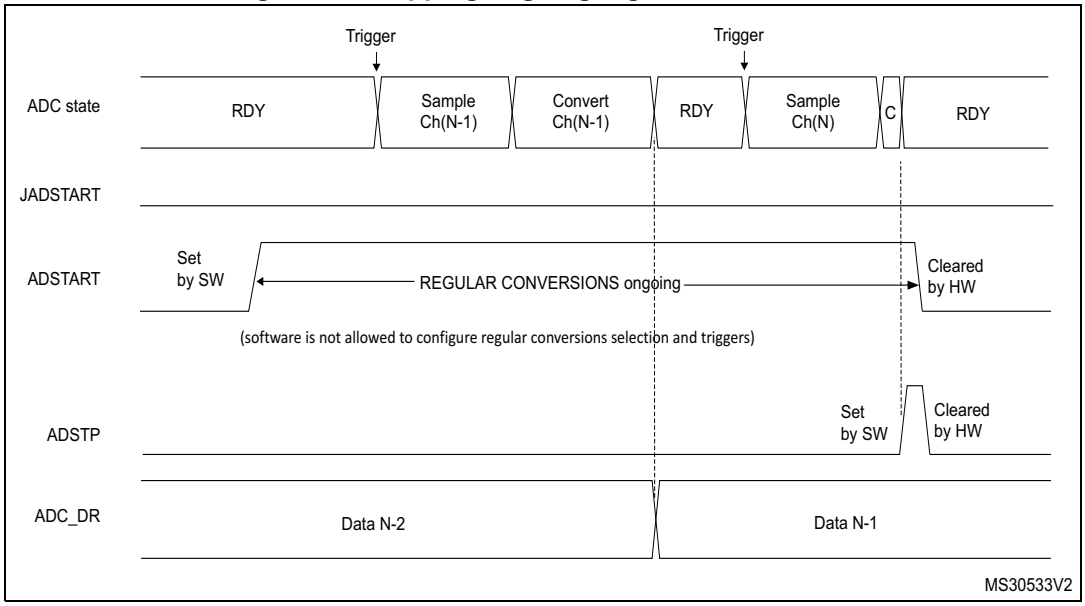
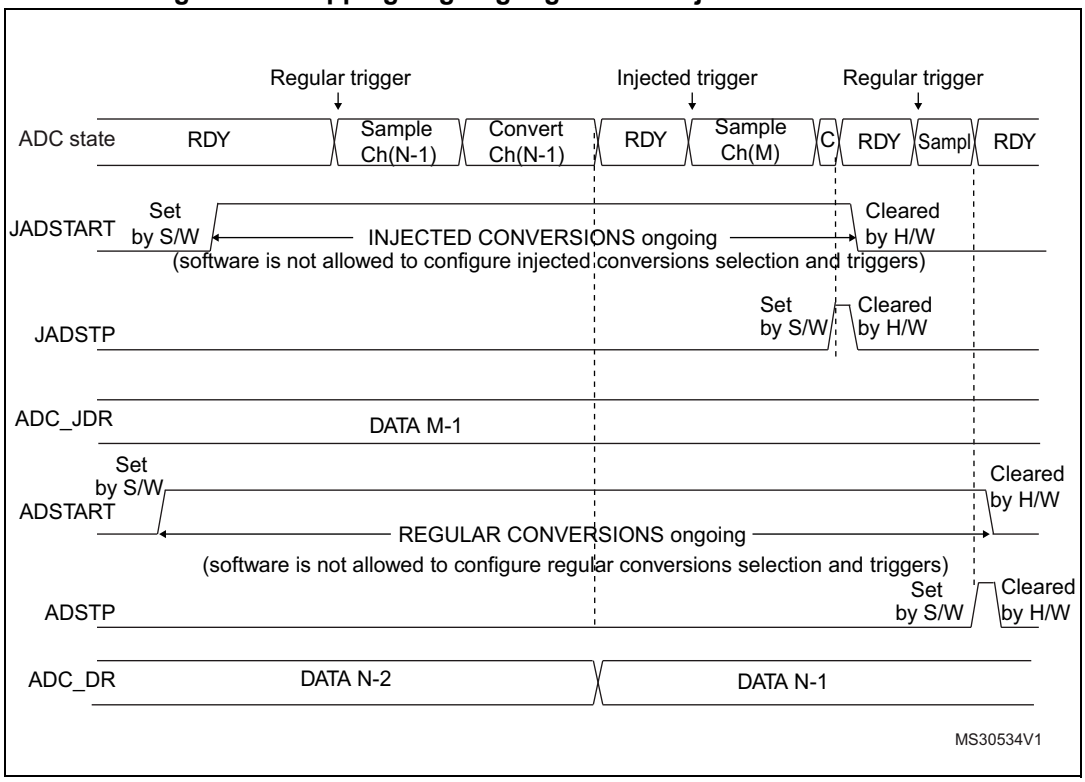


Figure 32. Stopping ongoing regular and injected conversions



13.3.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 39 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 39. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]/ JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly. Refer to Section 13.3.21: Queue of context for injected conversions.

The EXTSEL[3:0] and JEXTSEL[3:0] control bits select which out of 16 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly.
The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 13.3.21: Queue of context for injected conversions on page 235

Each ADC master shares the same input triggers with its ADC slave as described in *Figure 33*.

Figure 33. Triggers are shared between ADC master & ADC slave

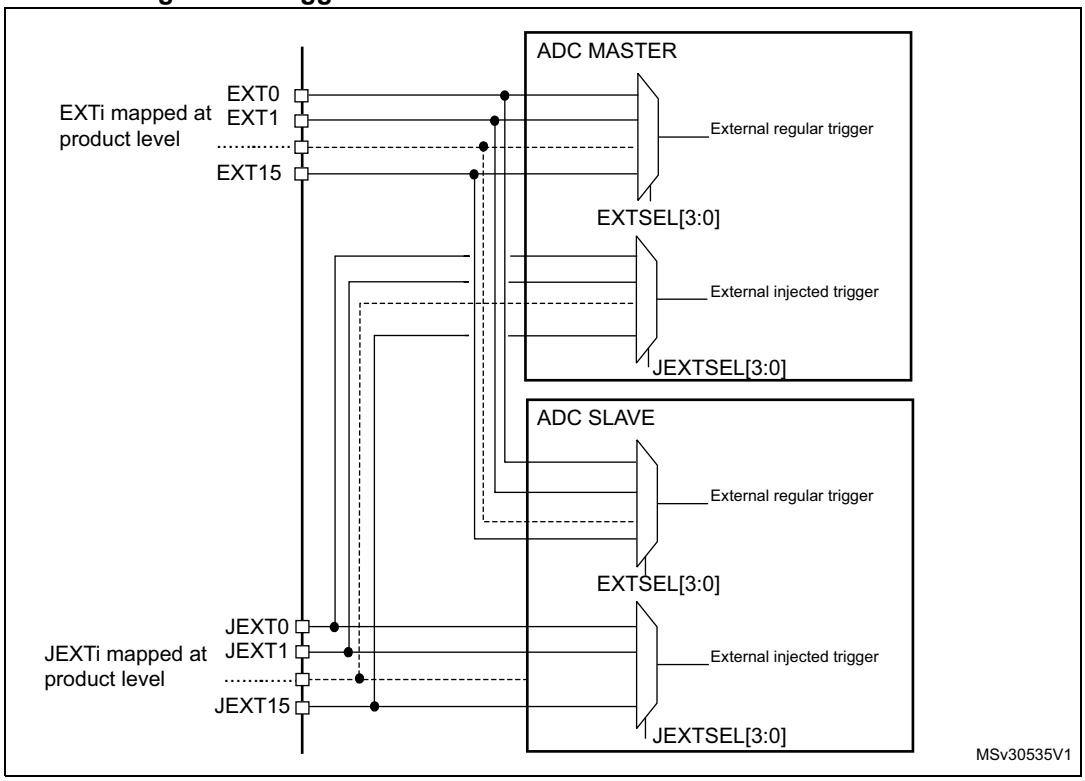


Table 40 to *Table 41* give all the possible external triggers of the two ADCs for regular and injected conversion.

Table 40. ADC1 (master) & 2 (slave) - External triggers for regular channels

Name	Source	Type	EXTSEL[3:0]
EXT0	TIM1_CC1 event	Internal signal from on chip timers	0000
EXT1	TIM1_CC2 event	Internal signal from on chip timers	0001
EXT2	TIM1_CC3 event	Internal signal from on chip timers	0010
EXT3	TIM2_CC2 event	Internal signal from on chip timers	0011

Table 40. ADC1 (master) & 2 (slave) - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[3:0]
EXT4	TIM3_TRGO event	Internal signal from on chip timers	0100
EXT5	Reserved	-	0101
EXT6	EXTI line 11	External pin	0110
EXT7	HRTIM_ADCTRG1 event	Internal signal from on chip timers	0111
EXT8	HRTIM_ADCTRG3 event	Internal signal from on chip timers	1000
EXT9	TIM1_TRGO event	Internal signal from on chip timers	1001
EXT10	TIM1_TRGO2 event	Internal signal from on chip timers	1010
EXT11	TIM2_TRGO event	Internal signal from on chip timers	1011
EXT12	Reserved	-	1100
EXT13	TIM6_TRGO event	Internal signal from on chip timers	1101
EXT14	TIM15_TRGO event	Internal signal from on chip timers	1110
EXT15	TIM3_CC4 event	Internal signal from on chip timers	1111

Table 41. ADC1 & ADC2 - External trigger for injected channels

Name	Source	Type	JEXTSEL[3..0]
JEXT0	TIM1_TRGO event	Internal signal from on chip timers	0000
JEXT1	TIM1_CC4 event	Internal signal from on chip timers	0001
JEXT2	TIM2_TRGO event	Internal signal from on chip timers	0010
JEXT3	TIM2_CC1 event	Internal signal from on chip timers	0011
JEXT4	TIM3_CC4 event	Internal signal from on chip timers	0100
JEXT5	Reserved	-	0101
JEXT6	EXTI line 15	External pin	0110
JEXT7	Reserved	-	0111
JEXT8	TIM1_TRGO2 event	Internal signal from on chip timers	1000
JEXT9	HRTIM_ADCTRG2 event	Internal signal from on chip timers	1001
JEXT10	HRTIM_ADCTRG4 event	Internal signal from on chip timers	1010
JEXT11	TIM3_CC3 event	Internal signal from on chip timers	1011
JEXT12	TIM3_TRGO event	Internal signal from on chip timers	1100
JEXT13	TIM3_CC1 event	Internal signal from on chip timers	1101
JEXT14	TIM6_TRGO event	Internal signal from on chip timers	1110
JEXT15	TIM15_TRGO event	Internal signal from on chip timers	1111

13.3.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADCx_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADCx_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADCx_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 34* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 28 ADC clock cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum interval between triggers must be 29 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADCx_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADCx_SQR and ADCx_JSQR registers.

In this mode, the ADSTART bit in the ADCx_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

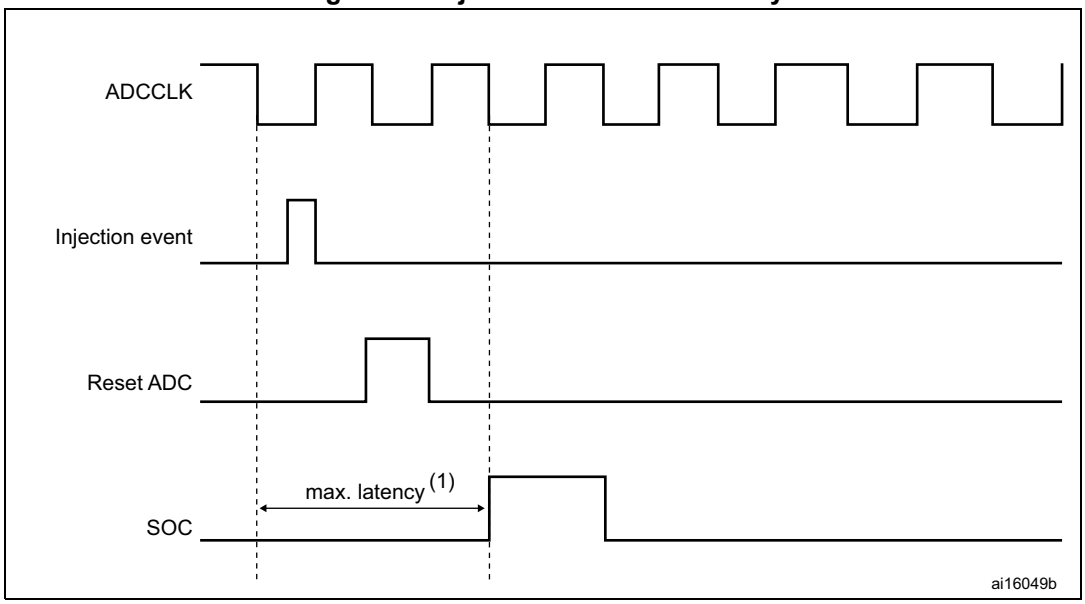
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer Complete event.

Figure 34. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F334xx datasheets.

13.3.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADCx_CFGR register.

It is used to convert a short sequence (sub-group) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADCx_SQR registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADCx_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADCx_SQR registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADCx_SQR1 register.

Example:

- DISCEN=1, n=3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - all the next trigger events will relaunch the complete sequence.

Note: When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADCx_CFGR register. It converts the sequence selected in the ADCx_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADCx_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADCx_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

13.3.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL[3:0] in ADCx_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADCx_JSQR register)

All the parameters of the context are defined into a single register ADCx_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADCx_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and both injected software and hardware triggers are disabled. Therefore, any further hardware or software injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

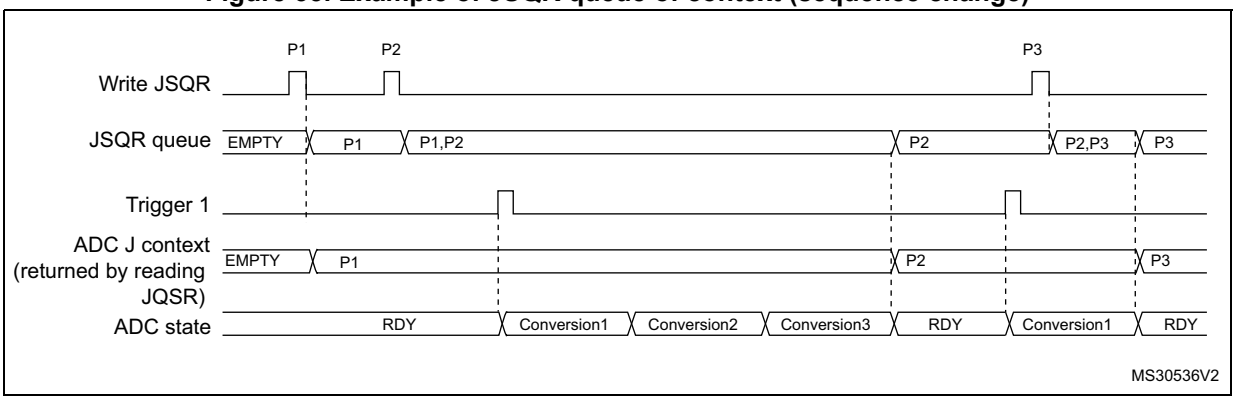
Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Behavior when changing the trigger or sequence context

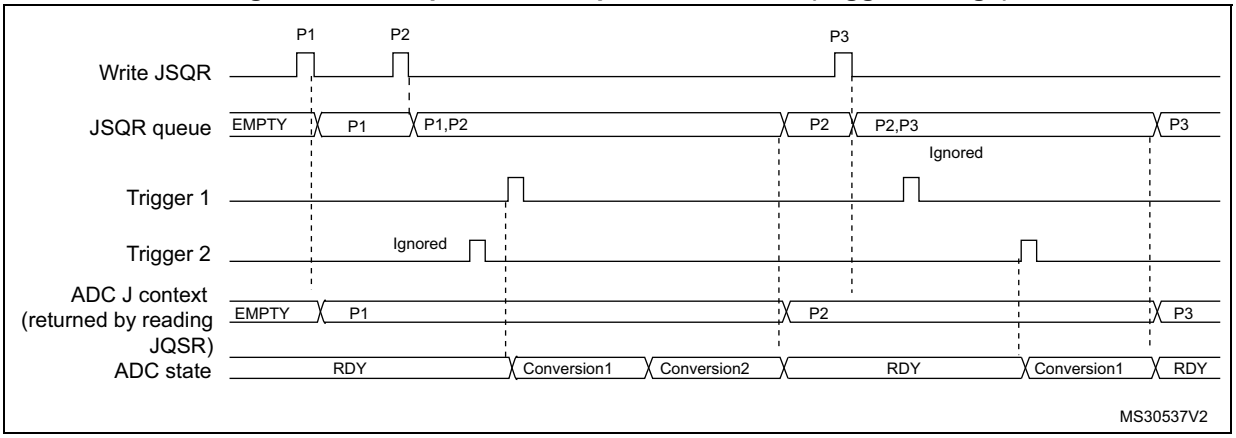
The *Figure 35* and *Figure 36* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 35. Example of JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 36. Example of JSQR queue of context (trigger change)

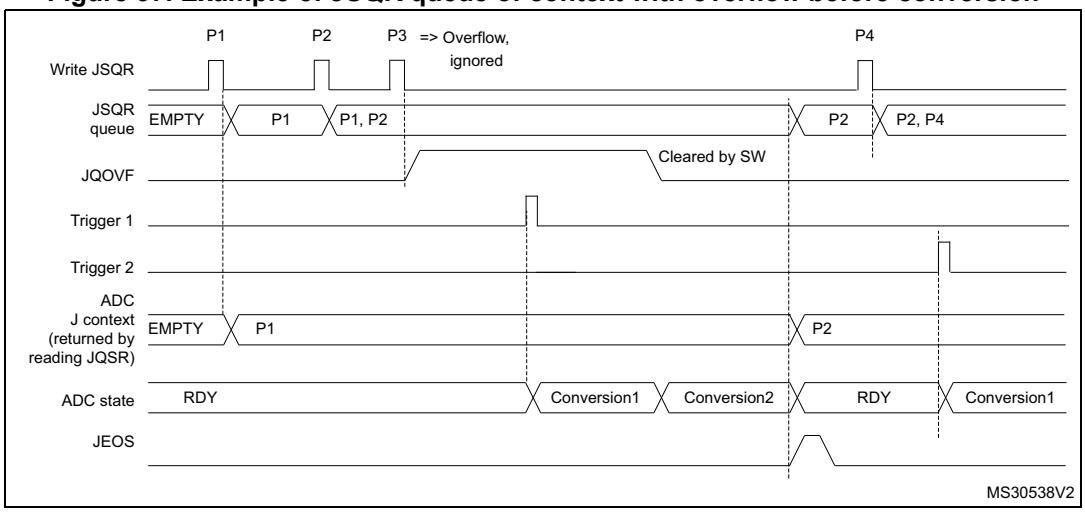


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

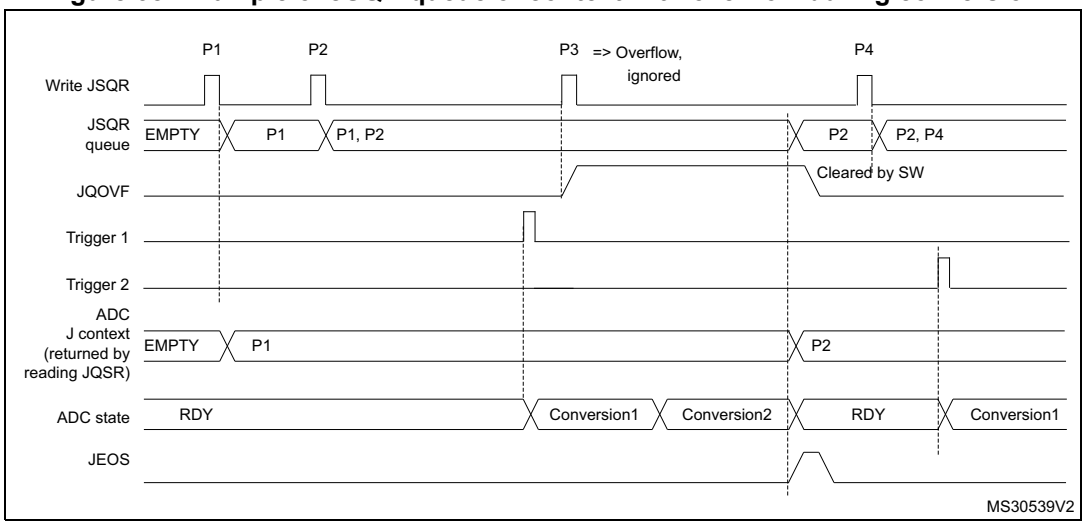
The *Figure 37* and *Figure 38* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 37. Example of JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 38. Example of JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

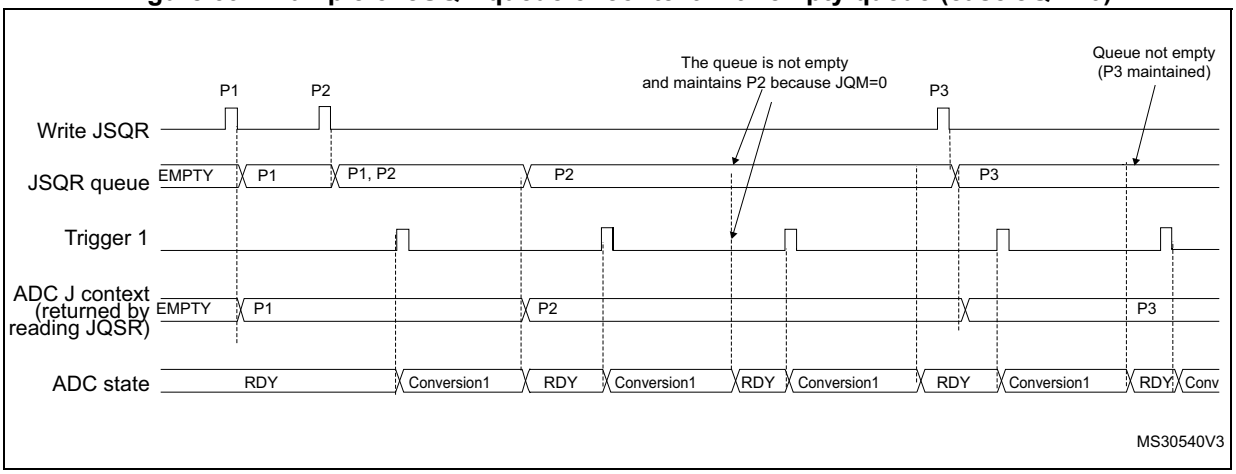
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 39 and *Figure 40* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

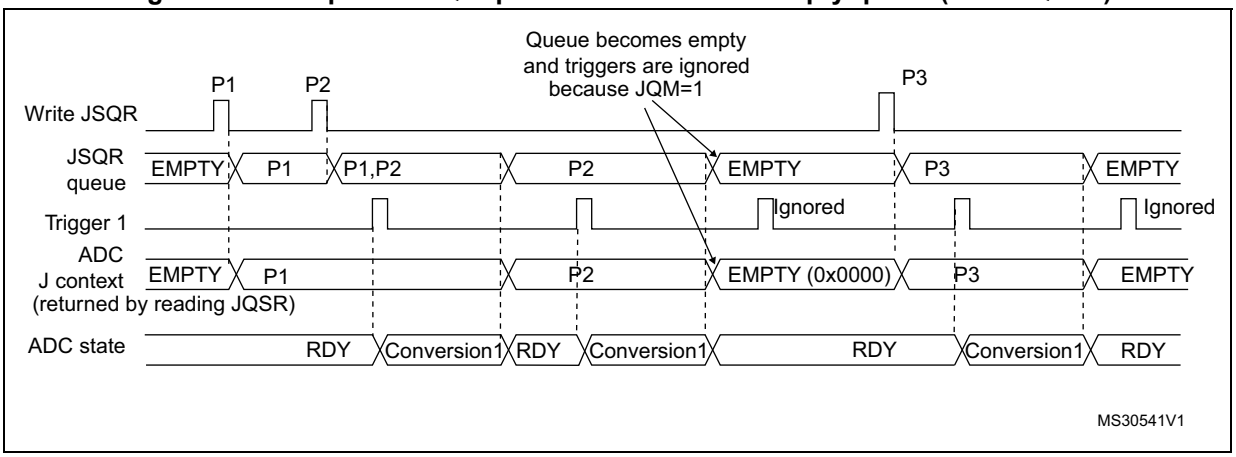
Figure 39. Example of JSQR queue of context with empty queue (case JQM=0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 40. Example of JSQR queue of context with empty queue (case JQM=1)

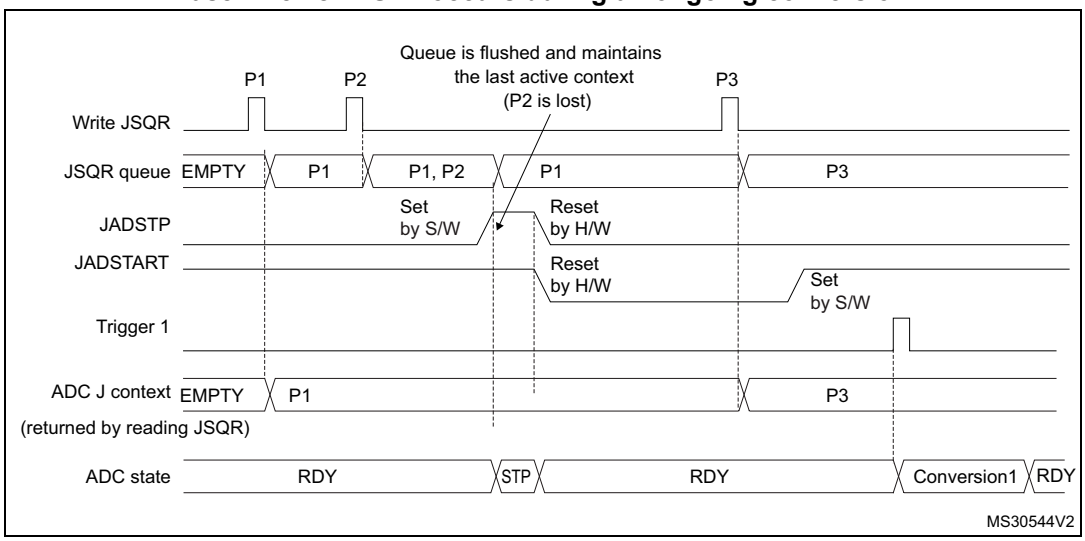


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

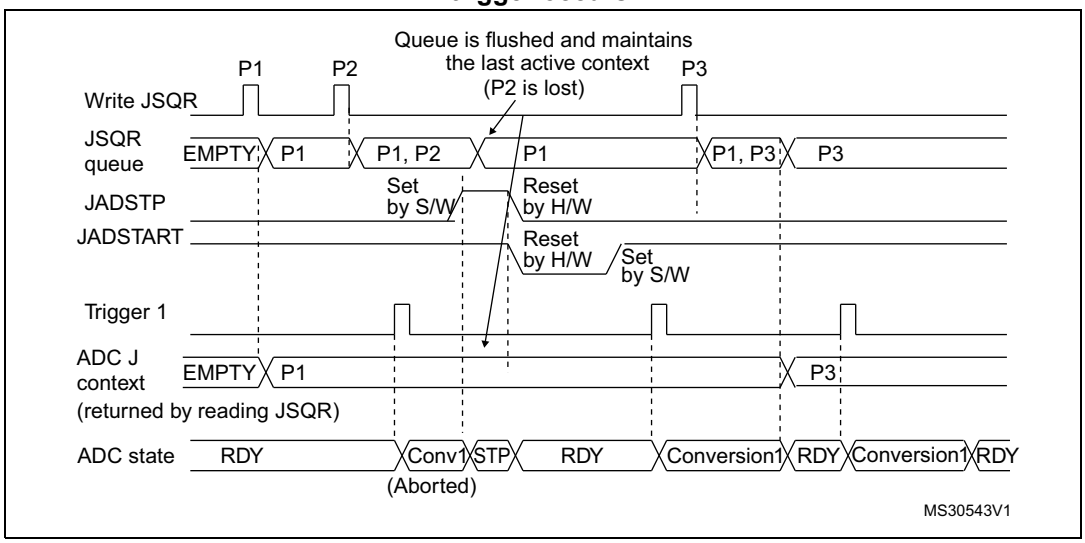
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 41. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



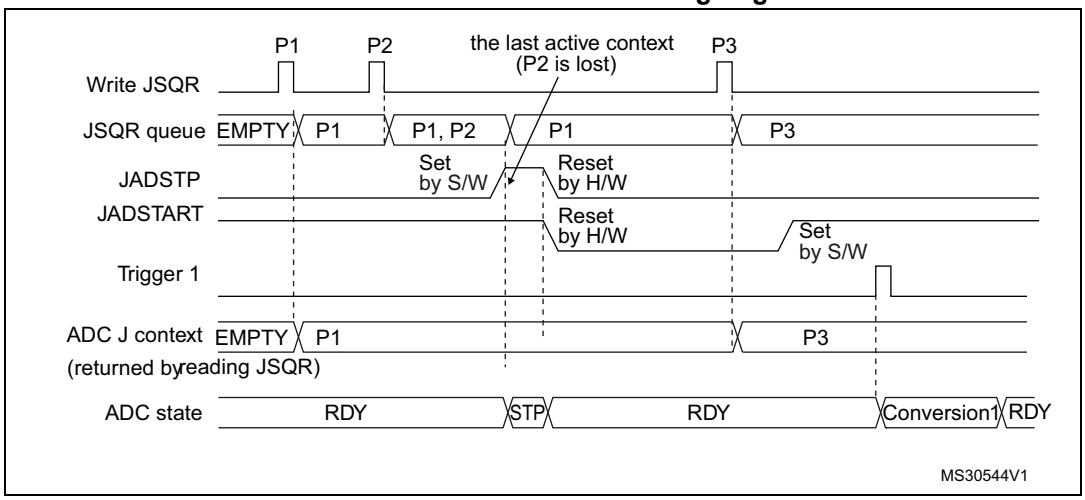
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 42. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.



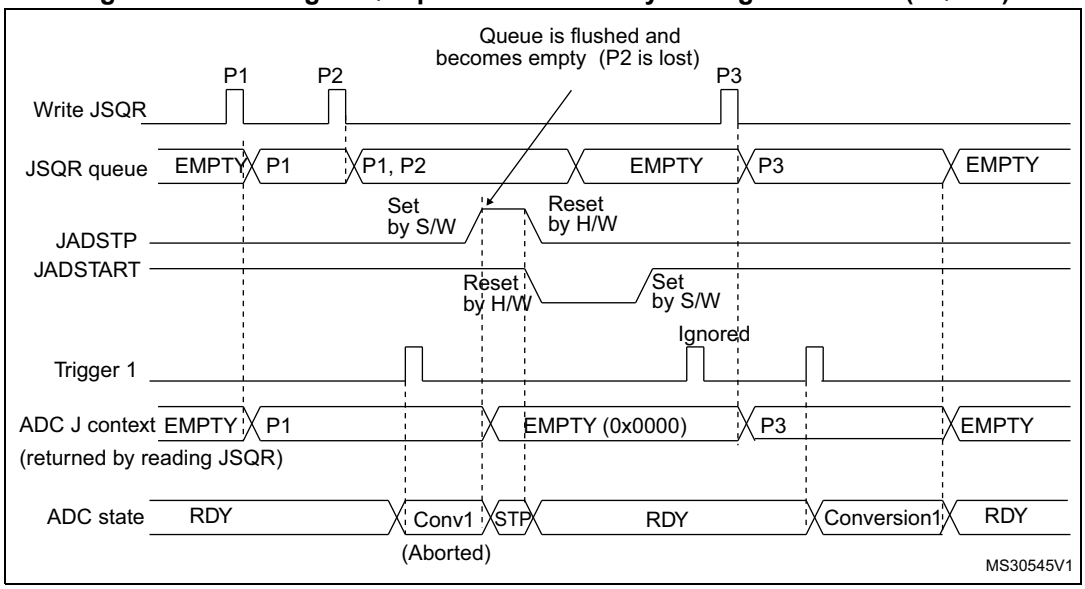
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 43. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion



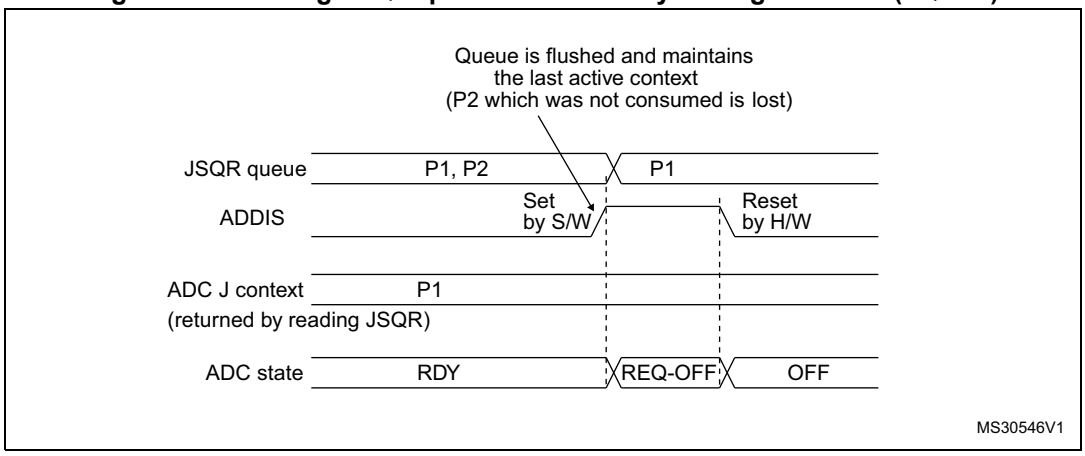
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 44. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)



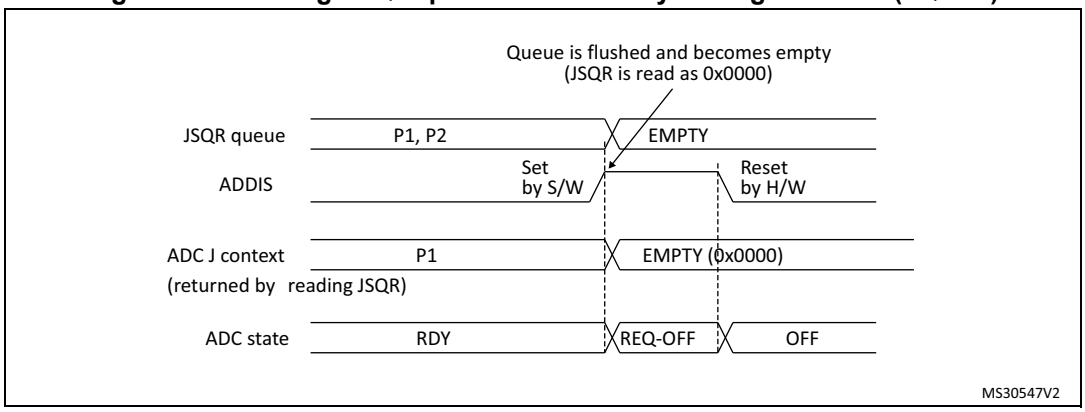
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 45. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 46. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1)



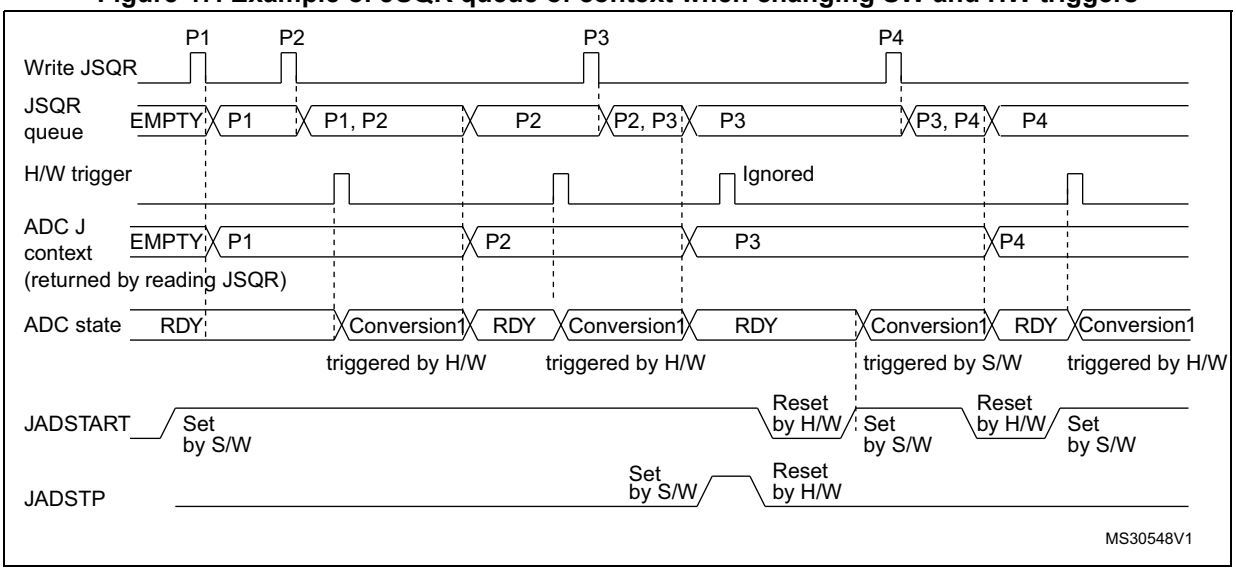
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Changing context from hardware to software (or software to hardware) injected trigger

When changing the context from hardware trigger to software injected trigger, it is necessary to stop the injected conversions by setting JADSTP=1 after the last hardware triggered conversions. This is necessary to re-enable the software trigger (a rising edge on JADSTART is necessary to start a software injected conversion). Refer to *Figure 47*.

When changing the context from software trigger to hardware injected trigger, after the last software trigger, it is necessary to set JADSTART=1 to enable the hardware triggers. Refer to *Figure 47*.

Figure 47. Example of JSQR queue of context when changing SW and HW triggers



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger (JEXTEN /=0x0)
   P2: sequence of 1 conversion, hardware trigger (JEXTEN /= 0x0)
   P3: sequence of 1 conversion, software trigger (JEXTEN = 0x0)
   P4: sequence of 1 conversion, hardware trigger (JEXTEN /= 0x0)

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

13.3.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 52*, *Figure 53*, *Figure 54* and *Figure 55* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 42*.

Table 42. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC}=72 MHz	T_{ADC} (ADC clock cycles) (with Sampling Time= 1.5 ADC clock cycles)	T_{ADC} (ns) at F_{ADC}=72 MHz
12	12.5 ADC clock cycles	173.6 ns	14 ADC clock cycles	194.4 ns
10	10.5 ADC clock cycles	145.8 ns	12 ADC clock cycles	166.7 ns
8	8.5 ADC clock cycles	118.0 ns	10 ADC clock cycles	138.9 ns
6	6.5 ADC clock cycles	90.3 ns	8 ADC clock cycles	111.1 ns

13.3.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADCx_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADCx_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADCx_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADCx_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

13.3.24 End of conversion sequence (EOS, JEOS)

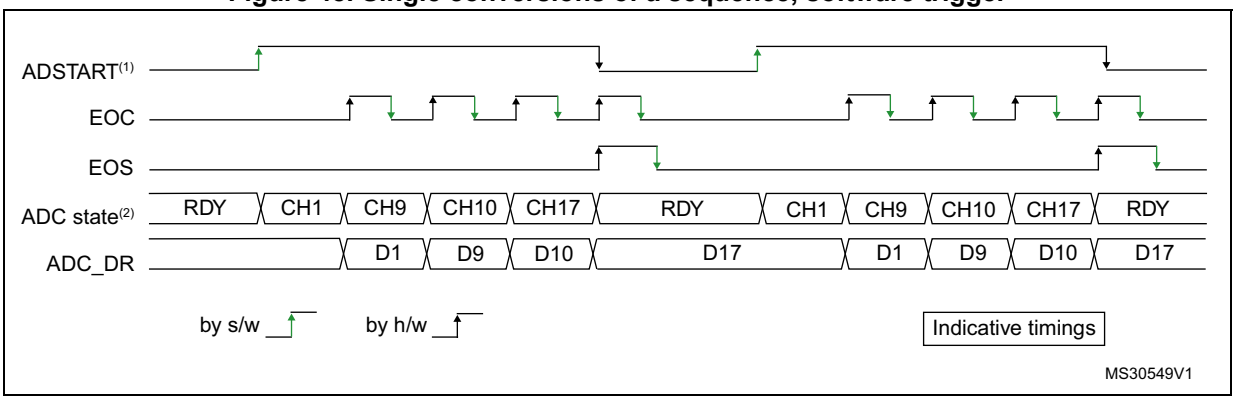
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADCx_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

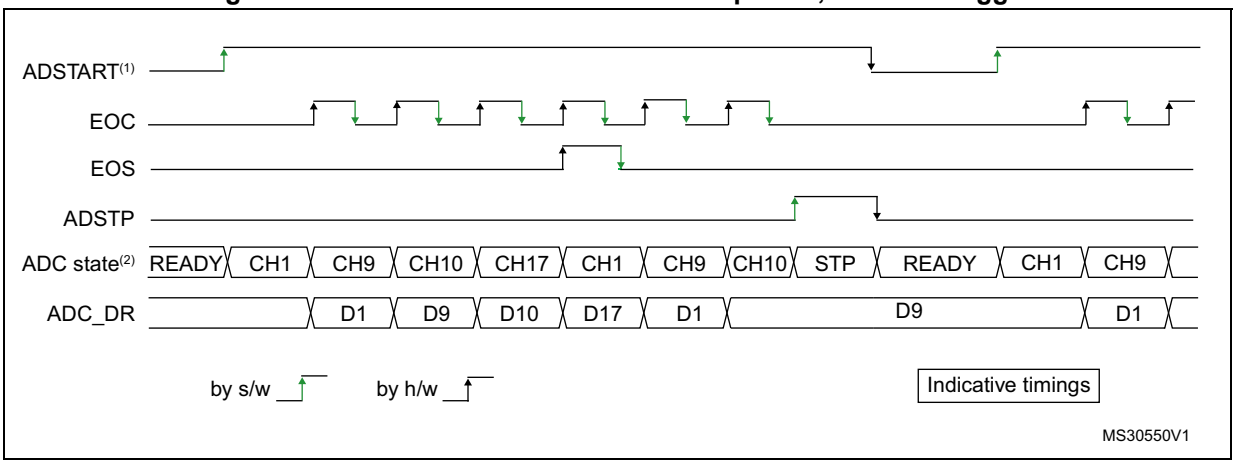
13.3.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 48. Single conversions of a sequence, software trigger



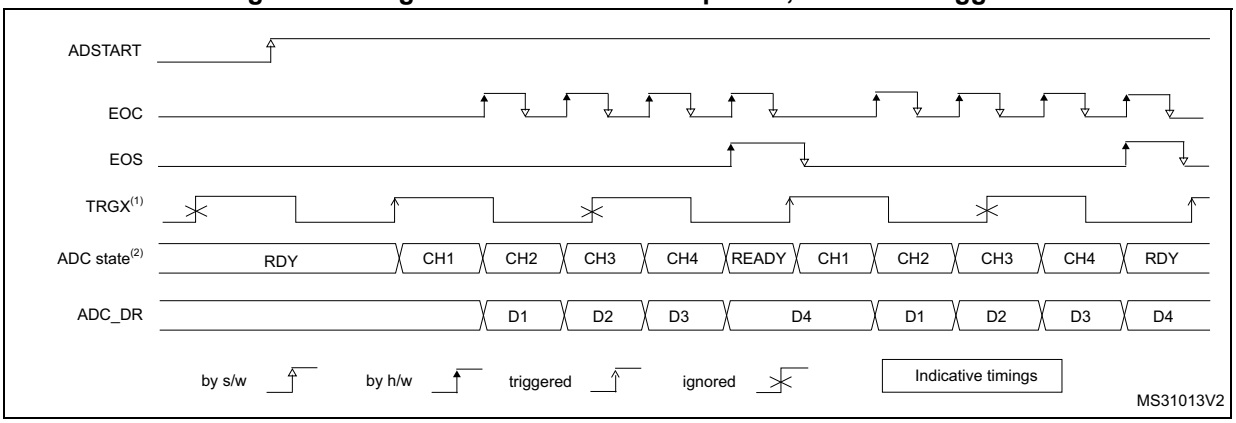
1. EXTEN=0x0, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 49. Continuous conversion of a sequence, software trigger



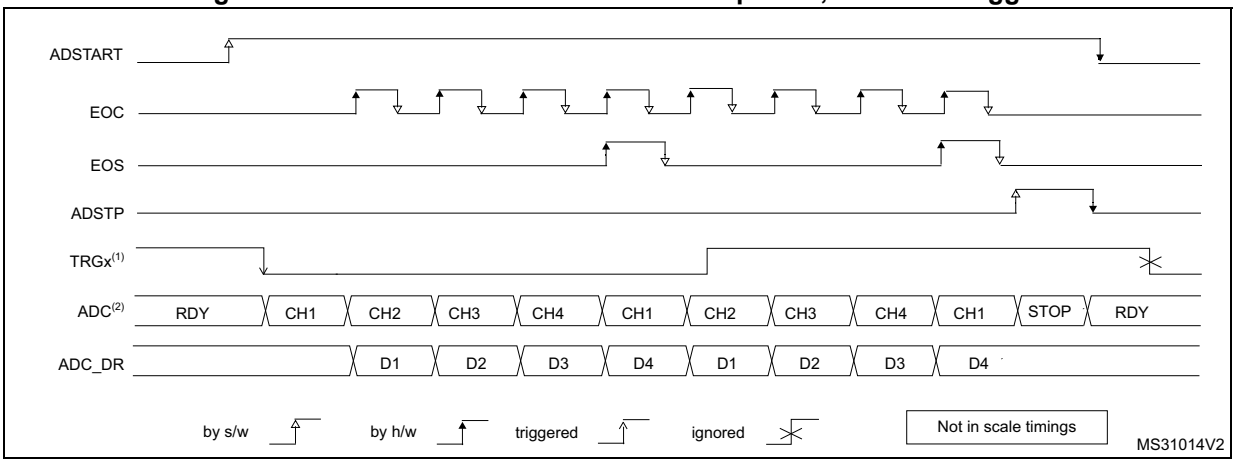
1. EXTEN=0x0, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 50. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 51. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

13.3.26 Data management

Data register, data alignment and offset (ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADCx_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADCx_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADCx_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 52*, *Figure 53*, *Figure 54* and *Figure 55*.

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 54* and *Figure 55*.

Offset

An offset y (y=1,2,3,4) can be applied to a channel by setting the bit OFFSETy_EN=1 into ADCx_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSETy_CH[4:0] of ADCx_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Table 45 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 43. Offset computation versus data resolution

Resolution (bits RES[1:0])	Substraction between raw converted data and offset:		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	signed 10-bit data	The user must configure OFFSET[1:0] to “00”
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	signed 8-bit data	The user must configure OFFSET[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	signed 6-bit data	The user must configure OFFSET[5:0] to “000000”

When reading data from ADCx_DR (regular channel) or from ADCx_JDRy (injected channel, y=1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSETy_EN=1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 52, *Figure 53*, *Figure 54* and *Figure 55* show alignments for signed and unsigned data.

Figure 52. Right alignment (offset disabled, unsigned value)

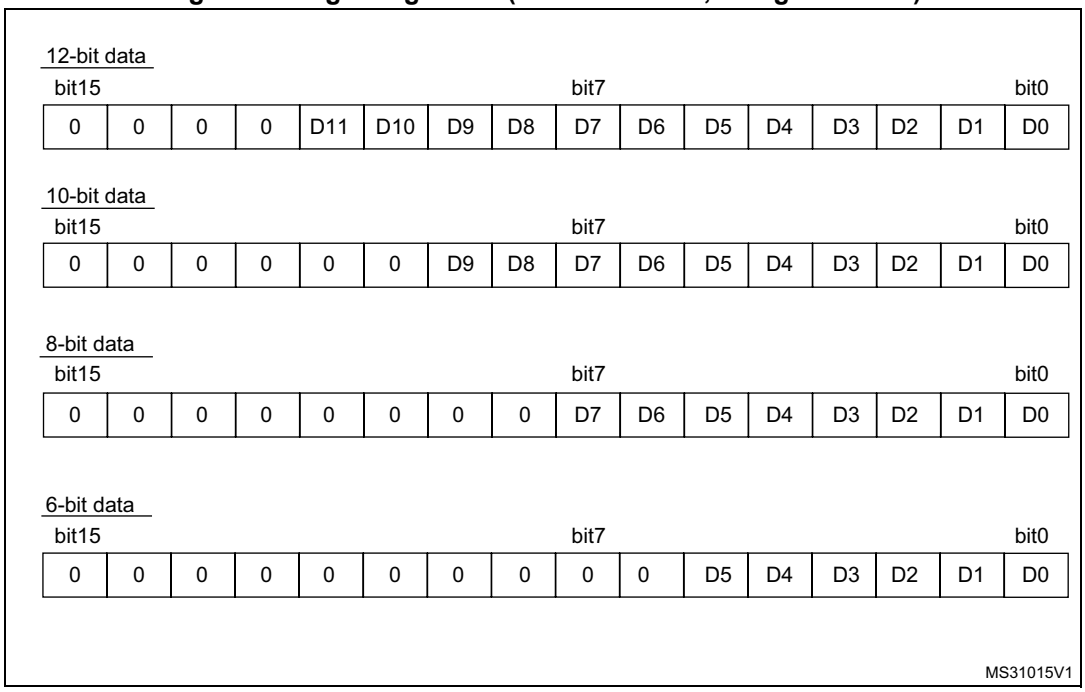


Figure 53. Right alignment (offset enabled, signed value)

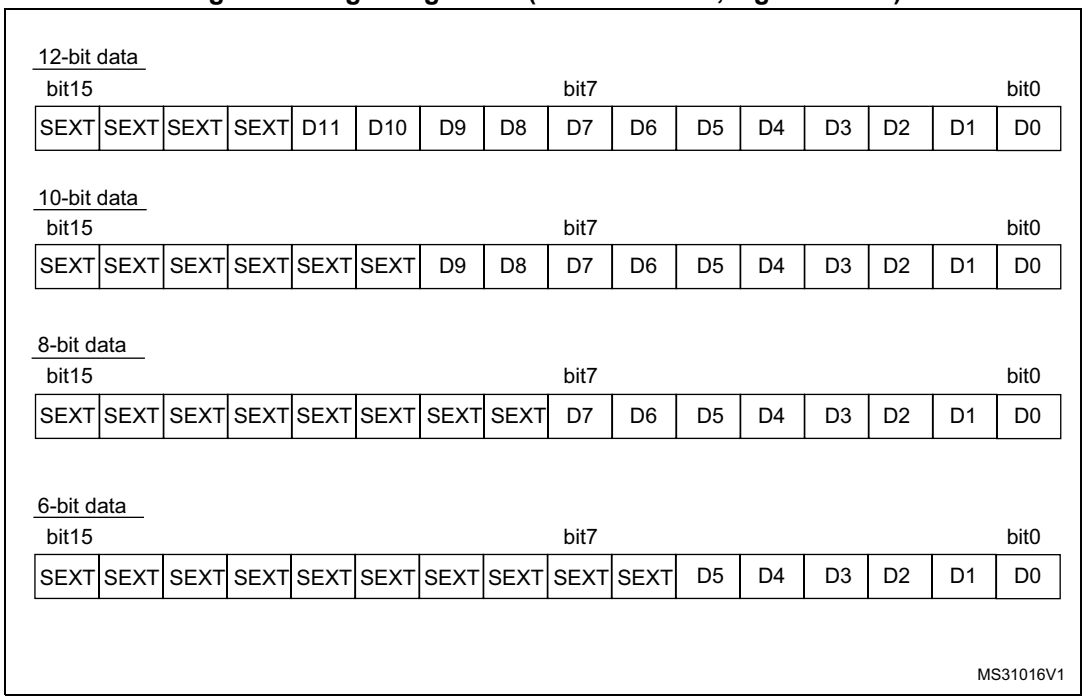


Figure 54. Left alignment (offset disabled, unsigned value)

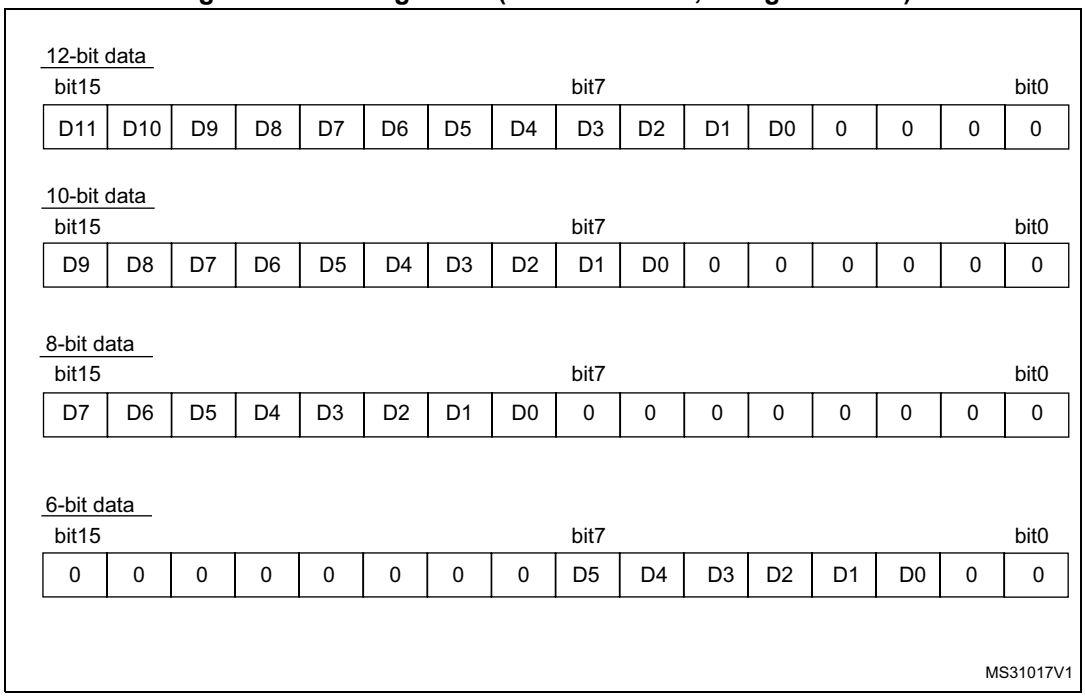
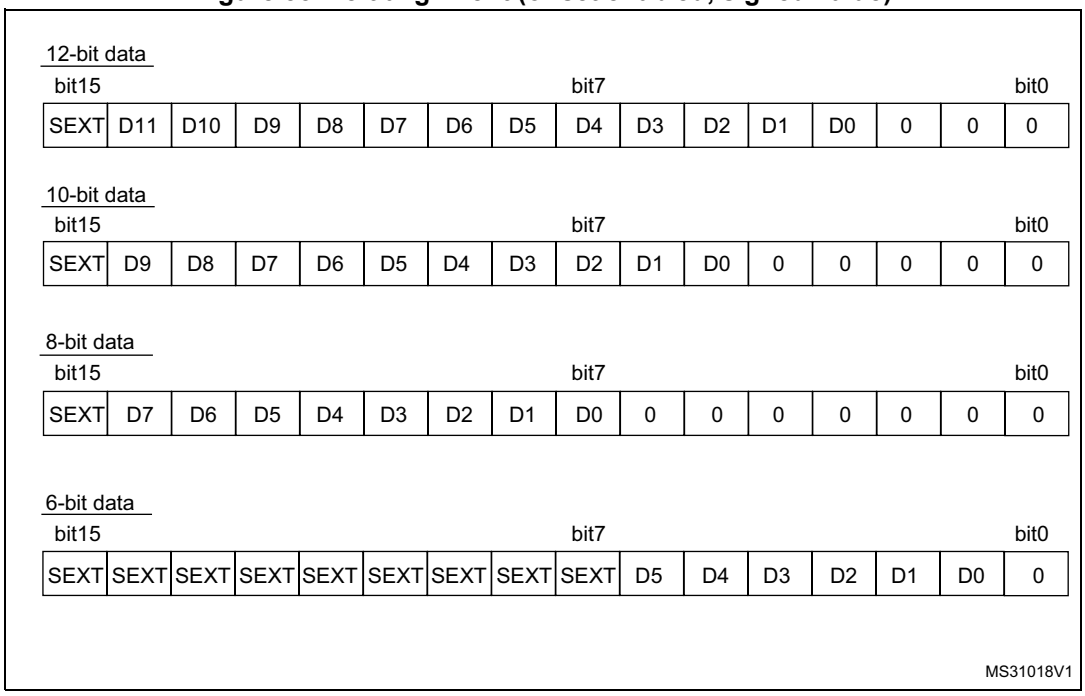


Figure 55. Left alignment (offset enabled, signed value)



ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies of a buffer overrun event, when the regular converted data was not read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

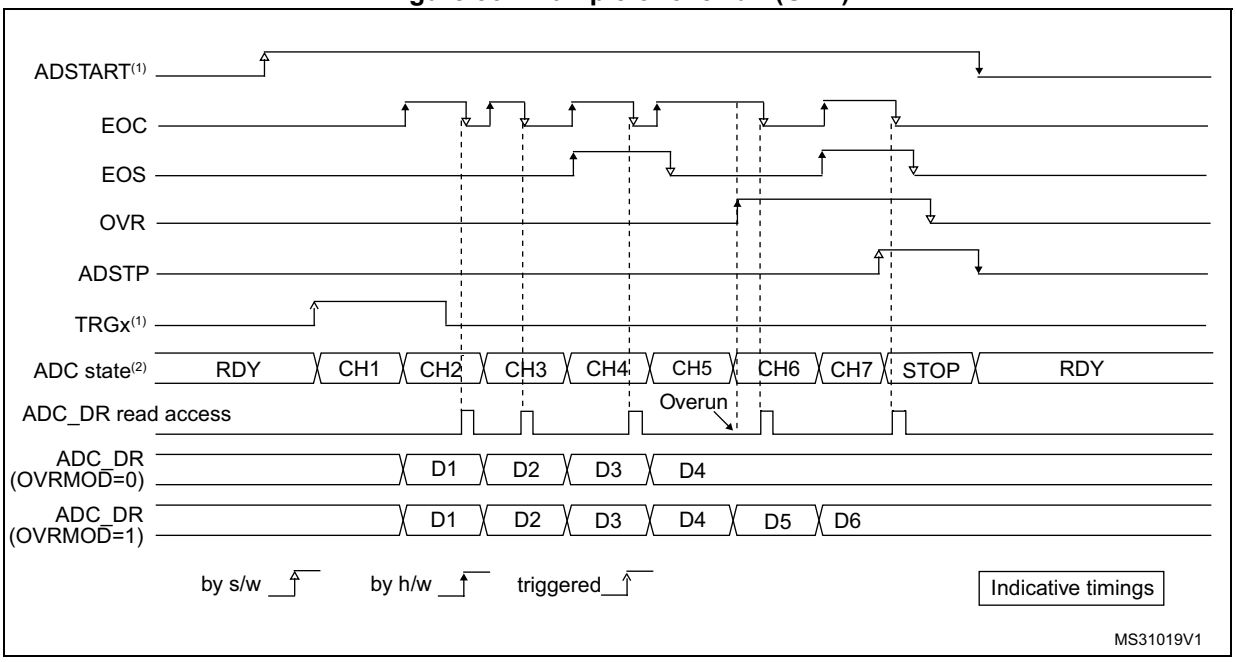
When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD=1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions will operate normally and the ADCx_DR register will always contain the latest converted data.

Figure 56. Example of overrun (OVR)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADCx_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADCx_DR register will always contain the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADCx_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADCx_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADCx_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADCx_CFGR register in single ADC mode, or with bit DMACFG of the ADCx_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG=0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG=1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG=0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when DMA_EOT interrupt occurs - refer to DMA paragraph) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMACFG=1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

13.3.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADCx_DR register has been read or if the EOC bit has been cleared (see *Figure 57*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 58*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to re-start a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 58*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 60*).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 61*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

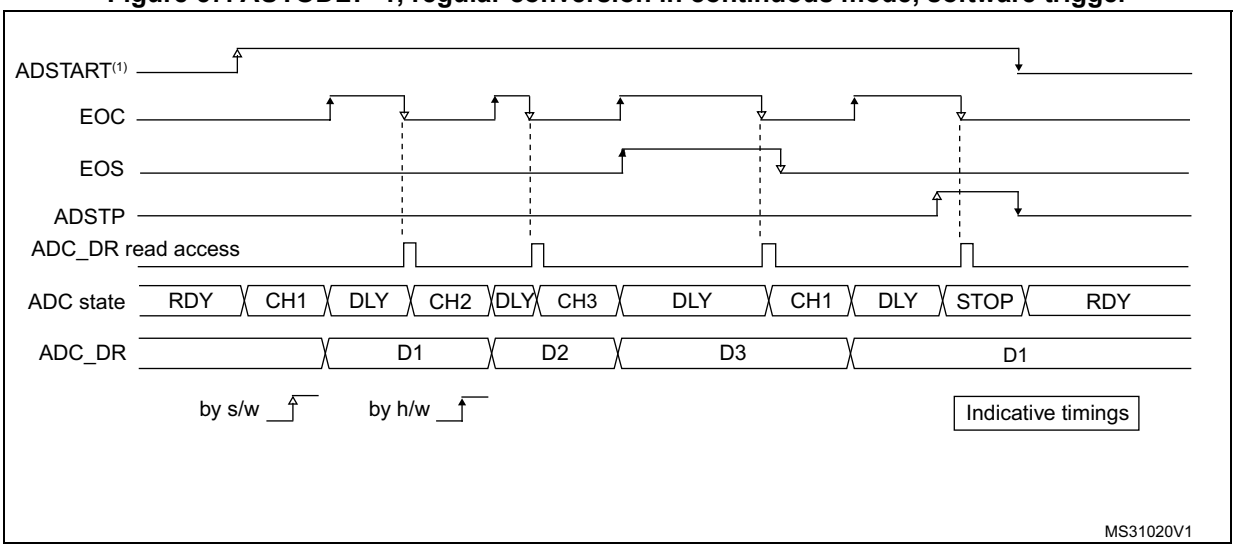
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

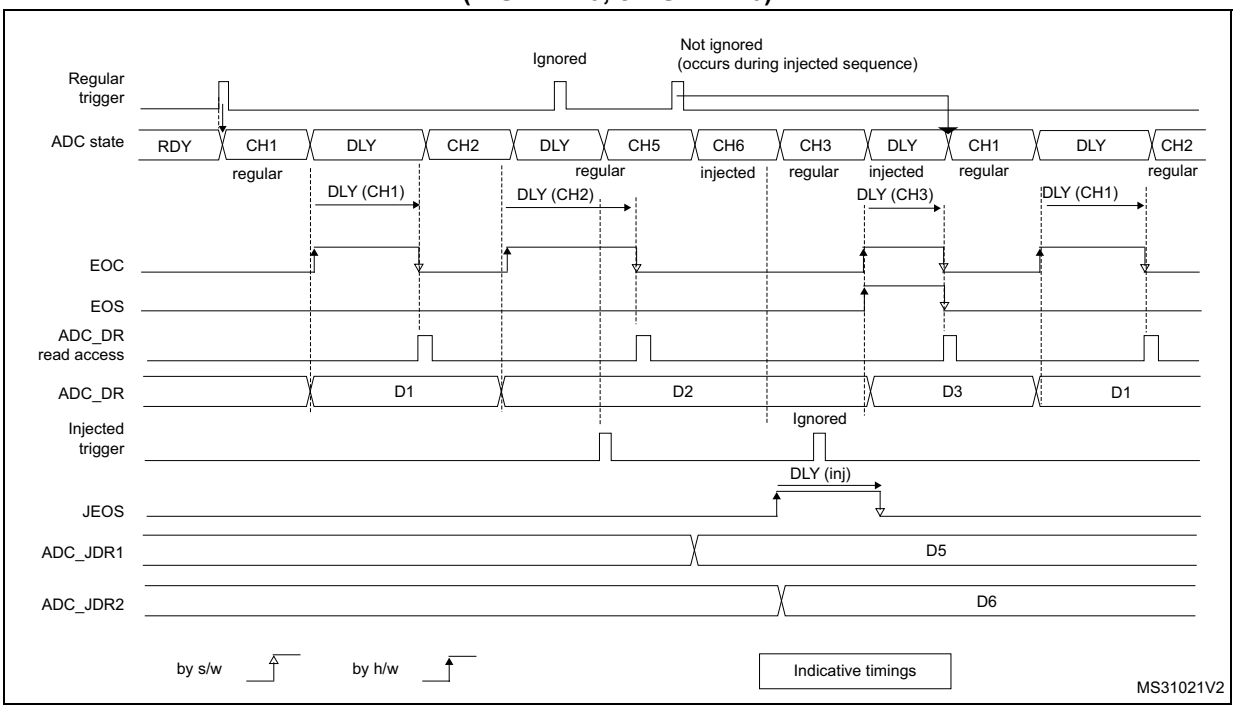
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 57. AUTODLY=1, regular conversion in continuous mode, software trigger



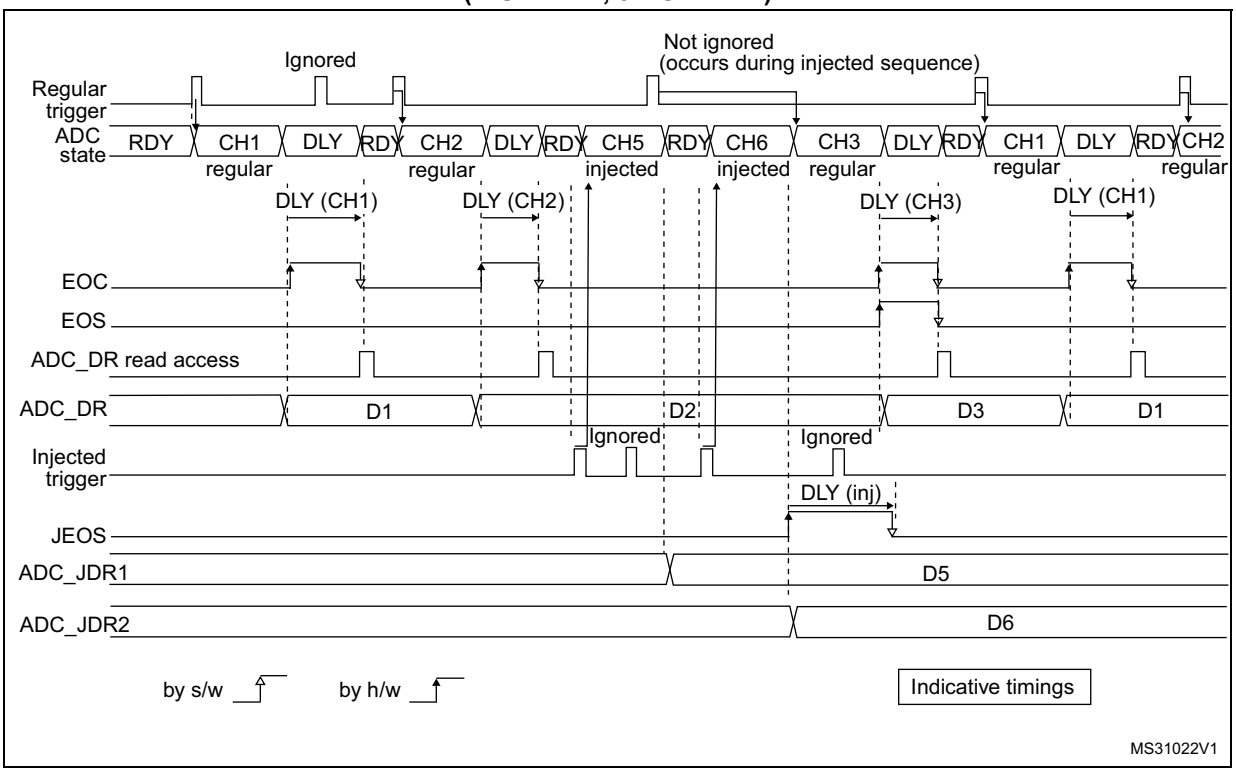
1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 58. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)



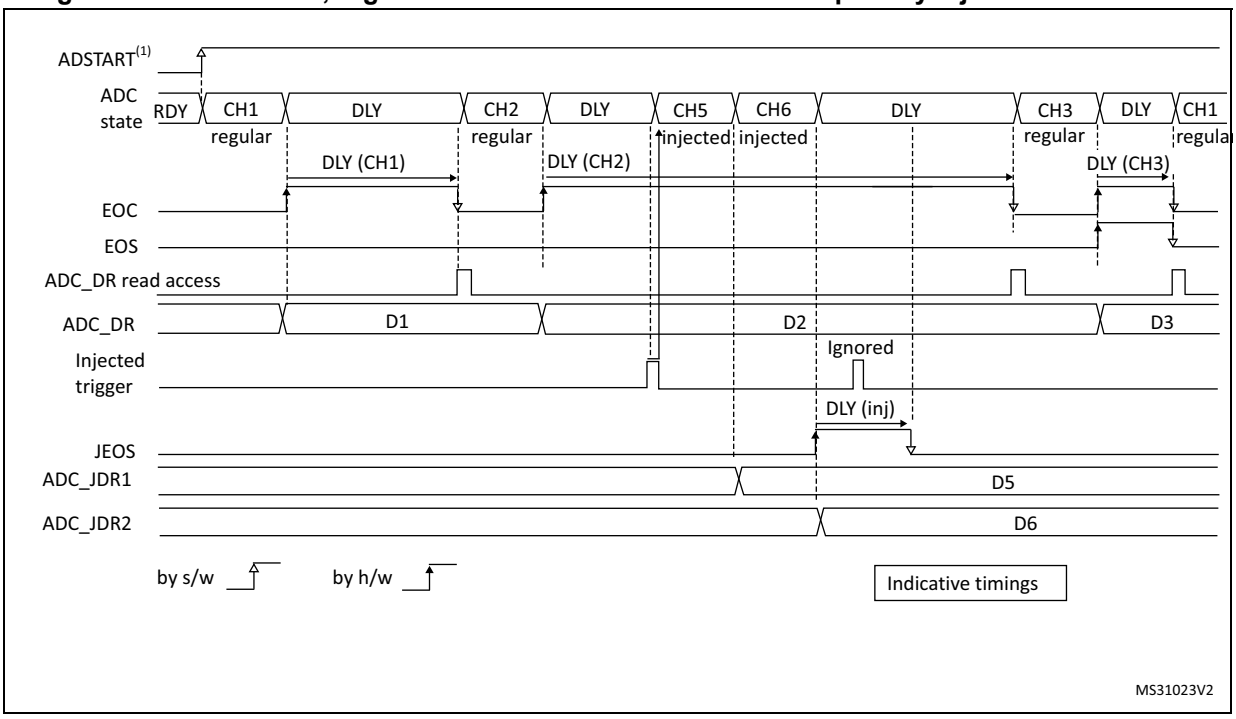
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 59. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



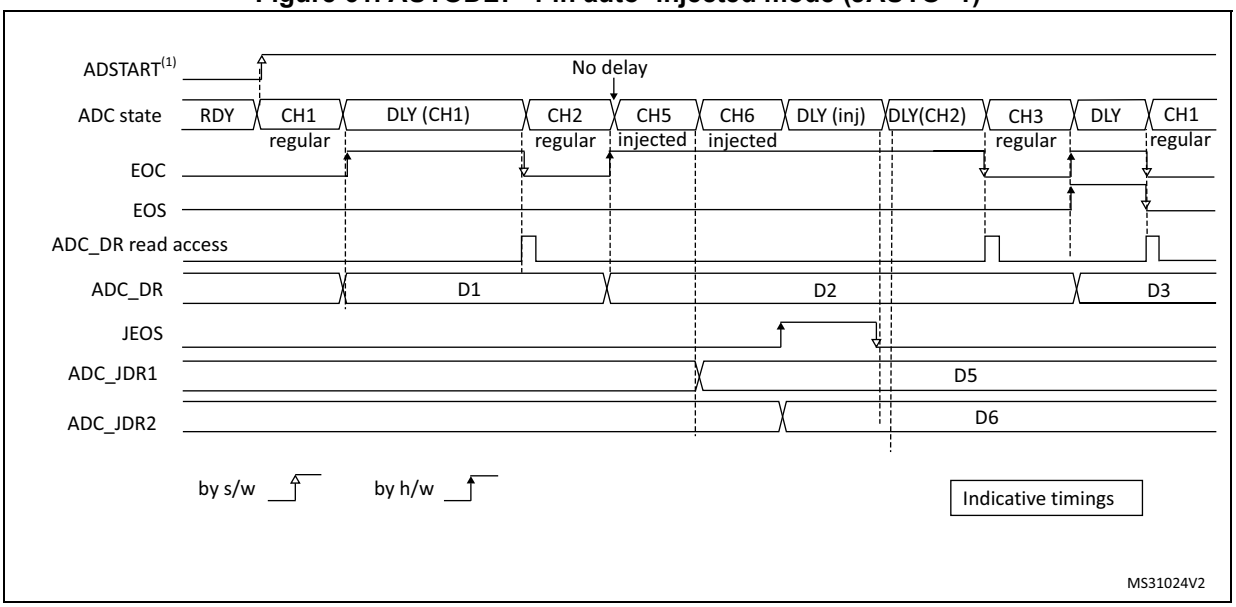
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 60. AUTODLY=1, regular continuous conversions interrupted by injected conversions



1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 61. AUTODLY=1 in auto- injected mode (JAUTO=1)

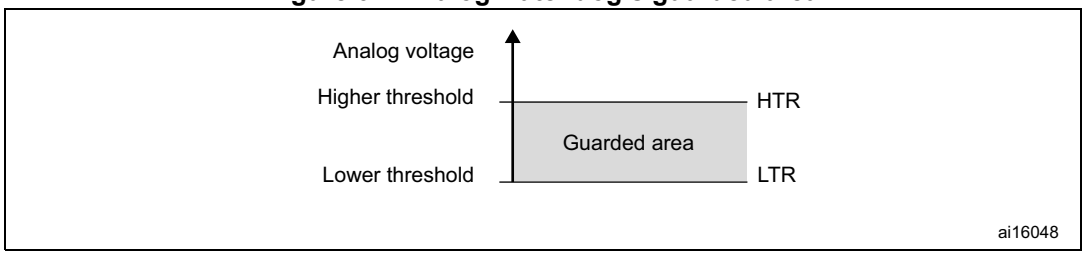


1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

13.3.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 62. Analog watchdog's guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADCx_IER register (x=1,2,3).

AWDx (x=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADCx_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels(1) remain within a configured voltage range (window).

Table 44 shows how the ADCx_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 44. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADCx_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 45 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 45. Analog watchdog 1 comparison

Resolution (bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[18:1] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:1] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 46* describes how the comparison is performed for all the possible resolutions.

Table 46. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS=1). Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 63. ADCy_AWDx_OUT signal generation (on all regular channels)

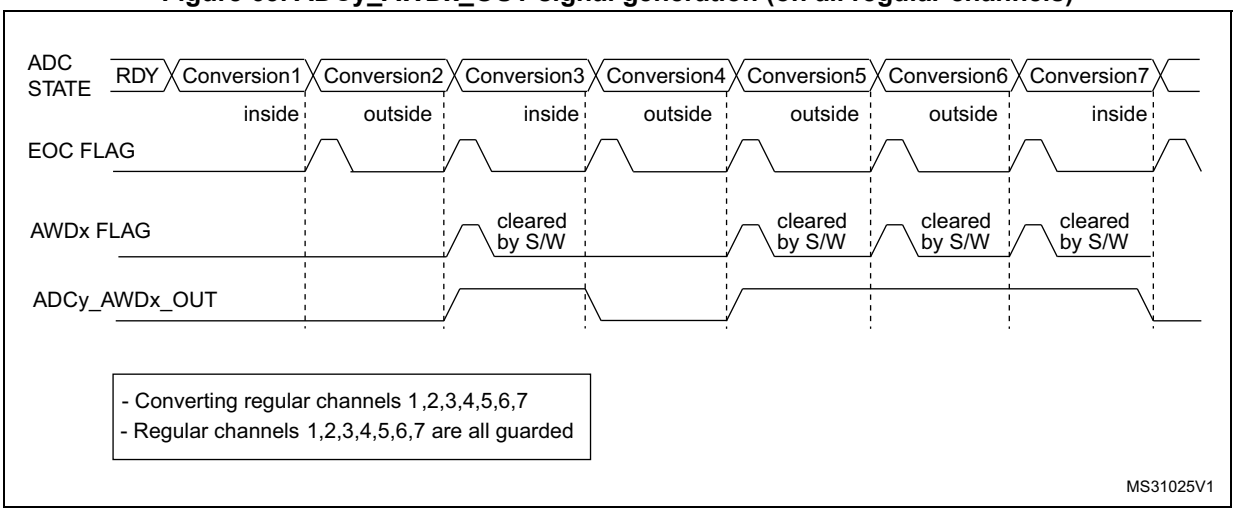


Figure 64. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)

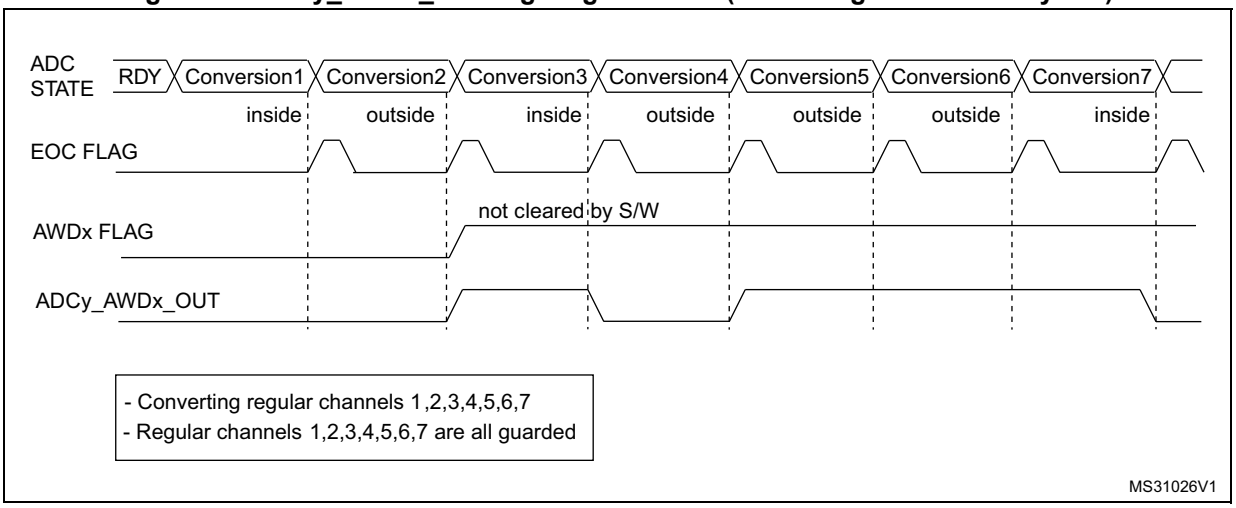


Figure 65. ADCy_AWDx_OUT signal generation (on a single regular channel)

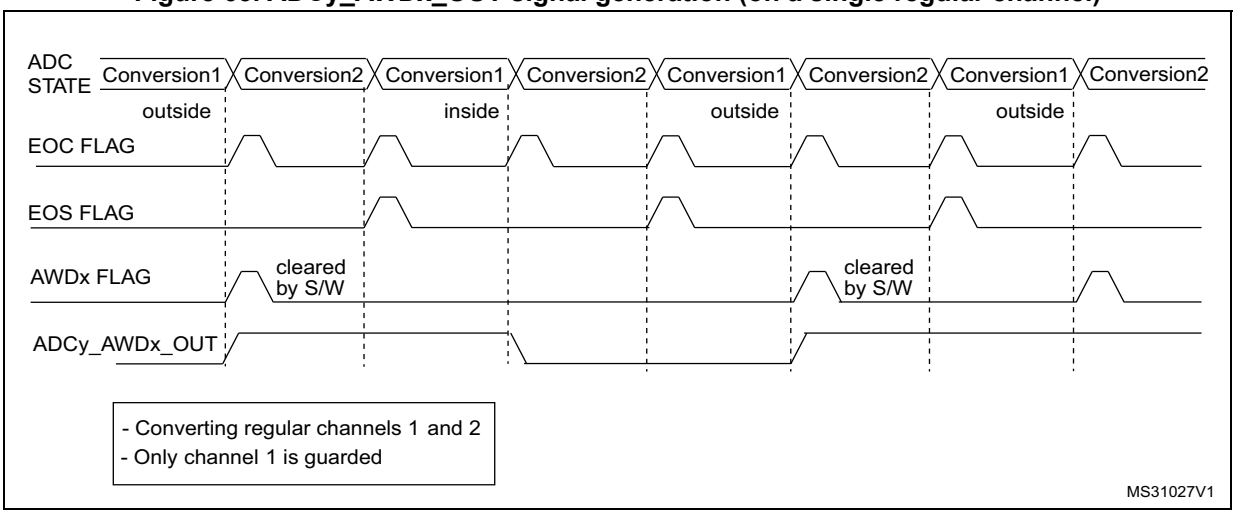
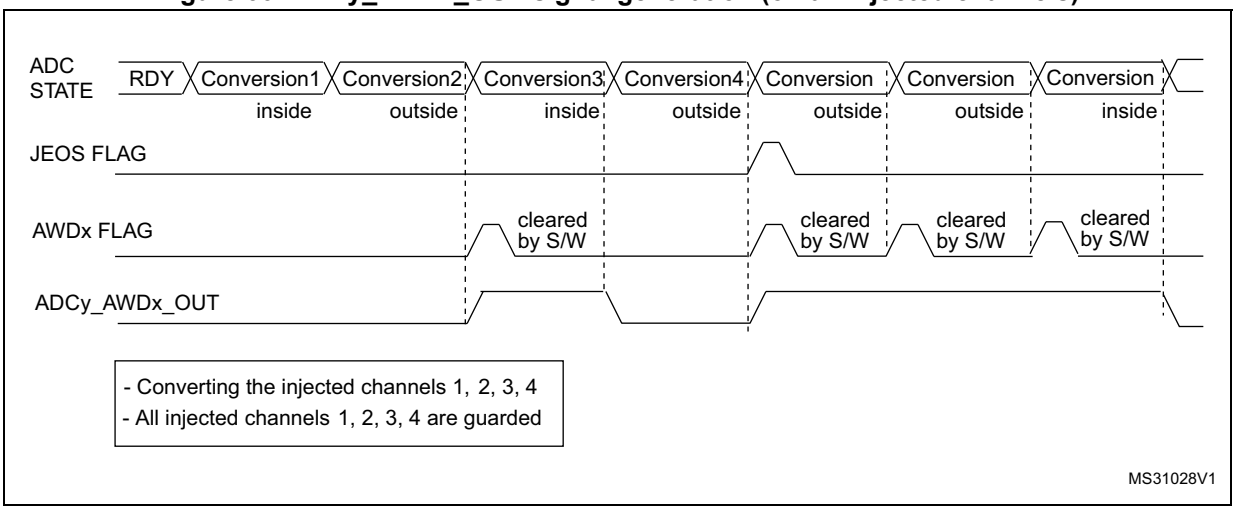


Figure 66. ADCy_AWDx_OUT signal generation (on all injected channels)



13.3.29 Dual ADC modes

In devices with two ADCs or more, dual ADC modes can be used (see *Figure 67*):

- ADC1 and ADC2 can be used together in dual mode (ADC1 is master)

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode

In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADCx_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

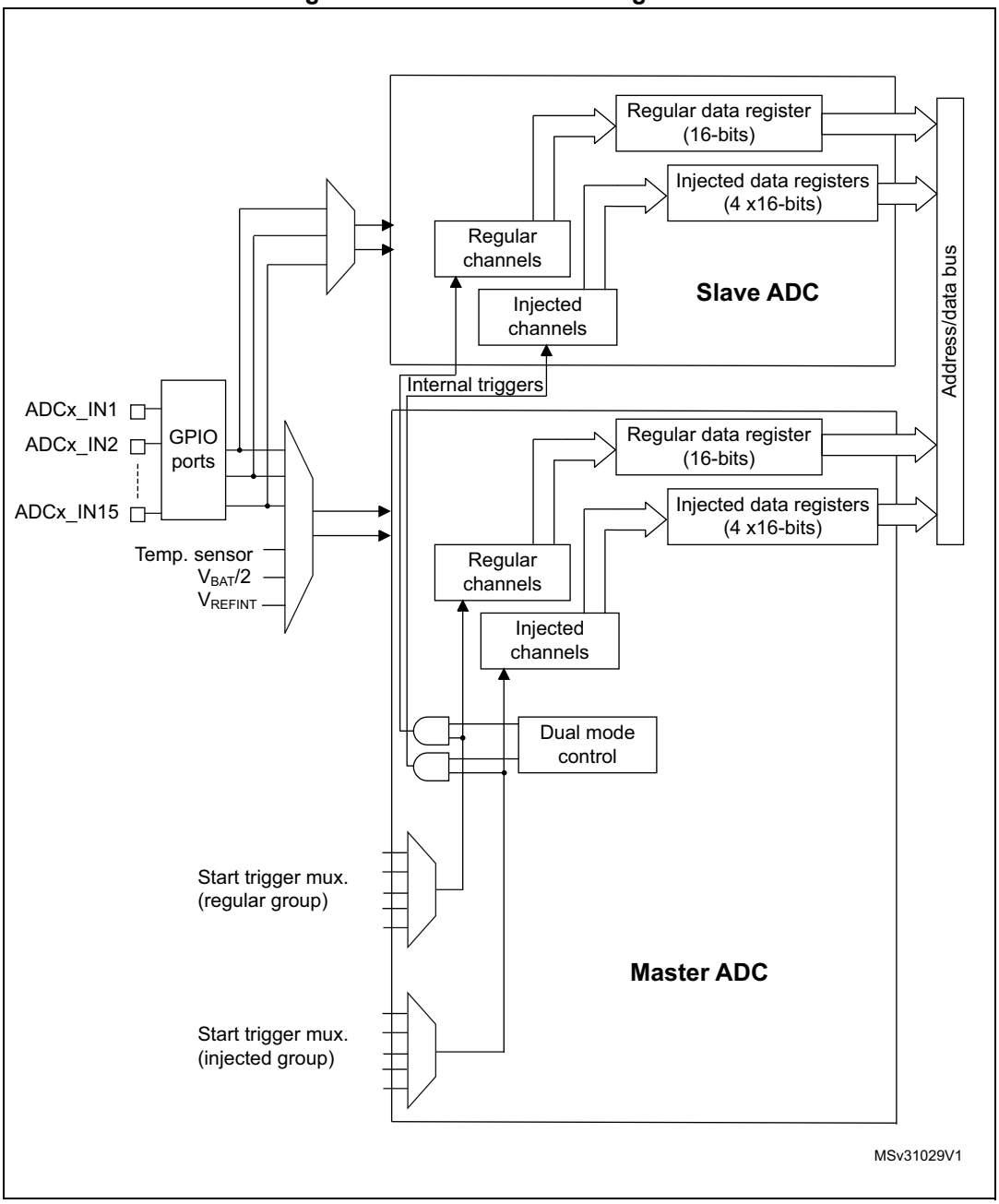
To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don't care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).

Figure 67. Dual ADC block diagram(1)



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL[3:0] bits in the ADCx_JSQR register).

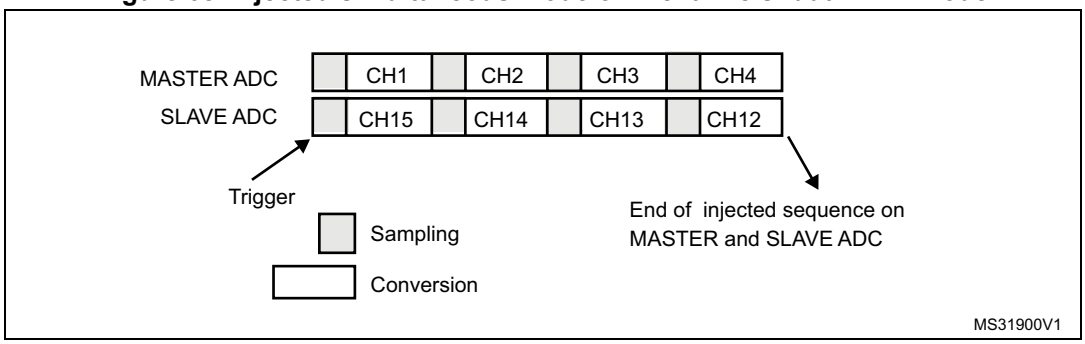
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADCx_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADCx_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 68*), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADCx_JDRy and slave ADCx_JDRy registers).

Figure 68. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADCx_DR) has been read. Any new regular trigger events occurring for the master ADC during the

ongoing regular sequence and the associated delay phases are ignored.
There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL[3:0] bits in the ADCx_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) will abort the current simultaneous conversions, which are re-started once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

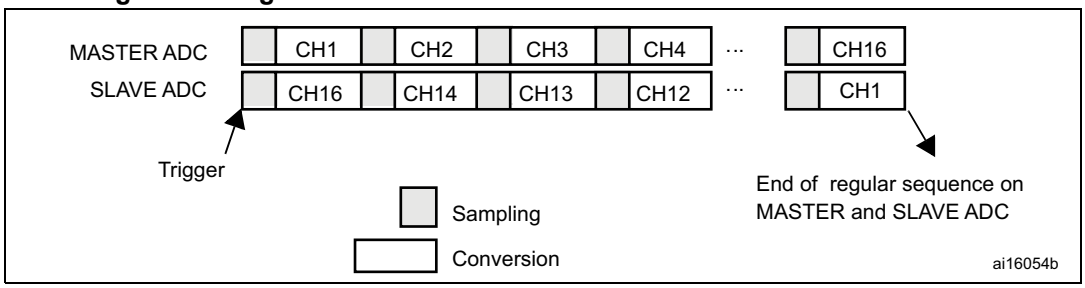
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 69*), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADCx_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADCx_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Note: In MDMA mode (MDMA[1:0]=0b10 or 0b11), the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions will not generate a DMA request.

Figure 69. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN=1 then each "n" simultaneous conversions of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADCx_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCx_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master's sequence is equal to the number of conversions in the slave's.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts to count after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

Software is notified by interrupts when it can read the data:

- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the slave ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts.

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using the two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADCx_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADCx_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which allows to save one DMA channel:
  - Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 70. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

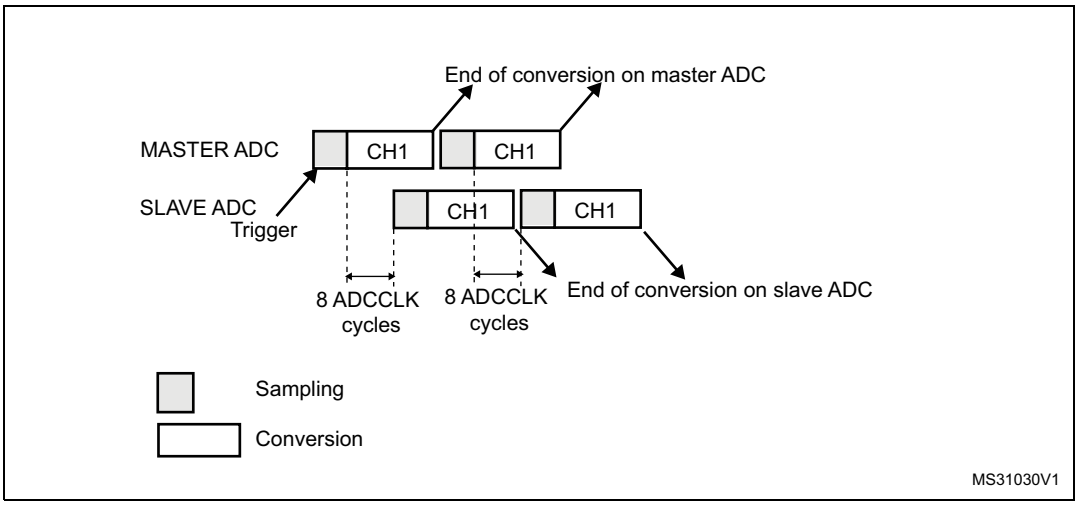
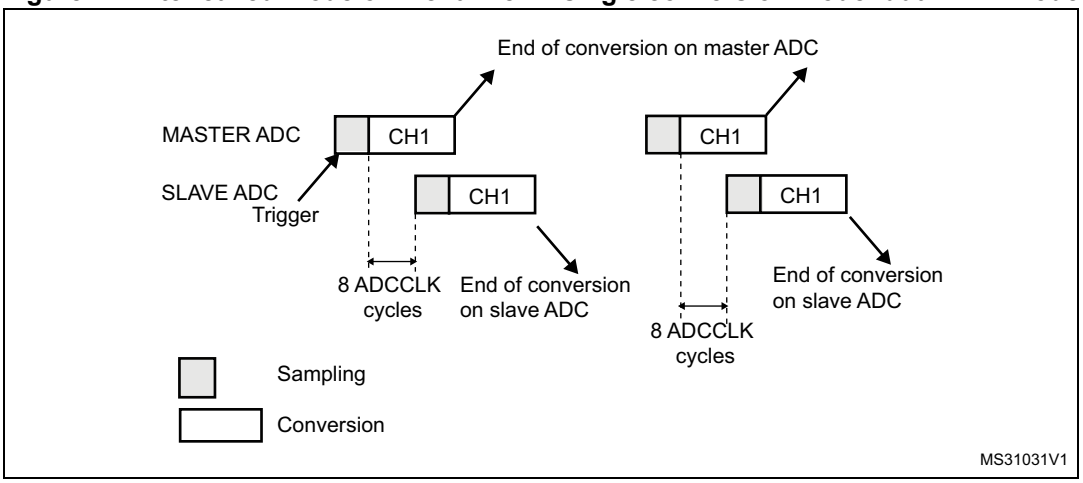


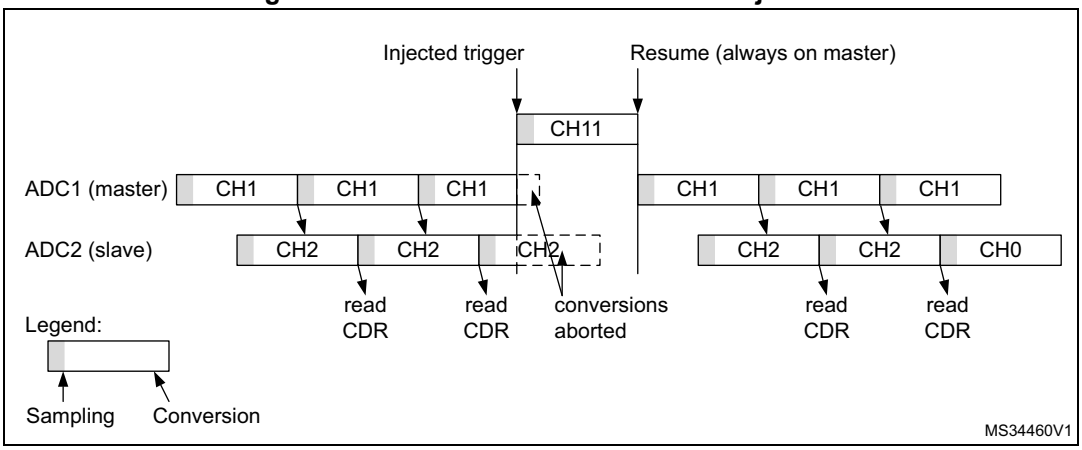
Figure 71. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is re-started from the master (see *Figure 72* below).

Figure 72. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

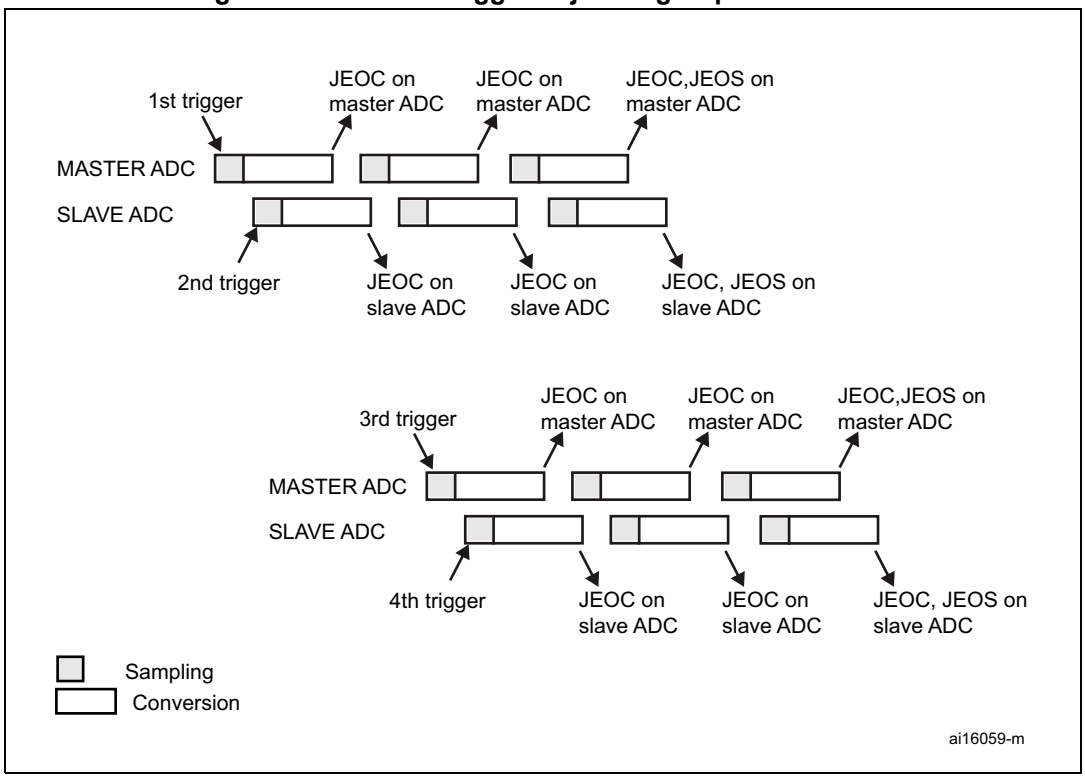
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 73. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

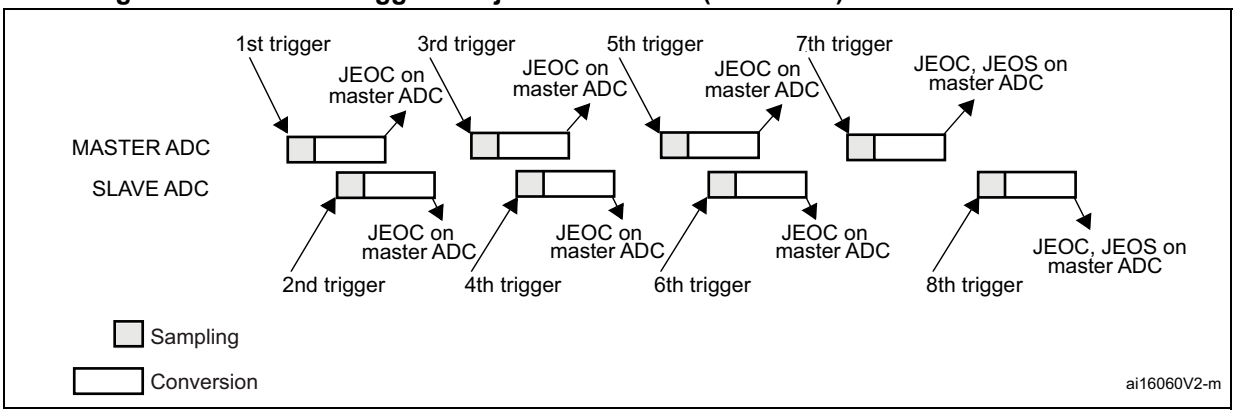
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 74. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

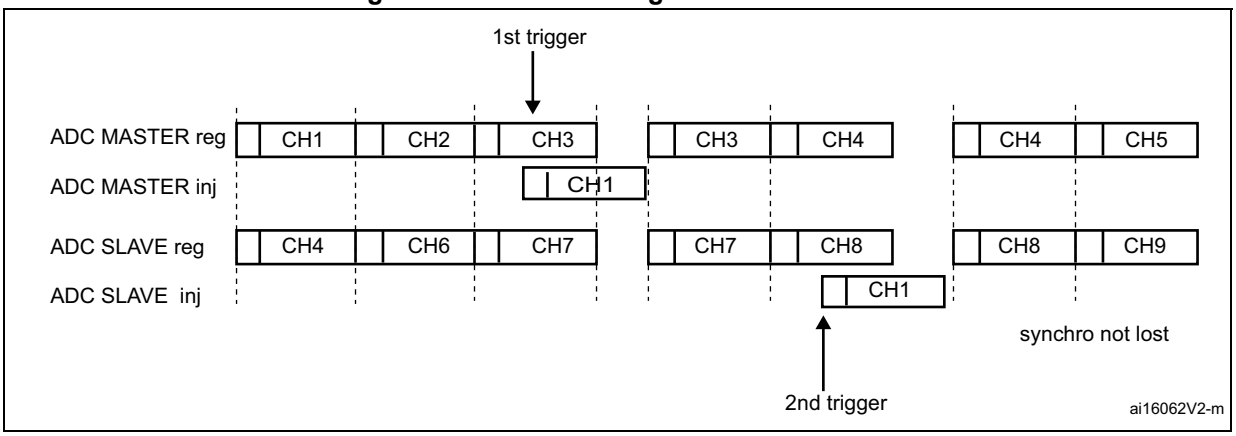
This mode is selected by programming bits DUAL[4:0]=00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 75* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

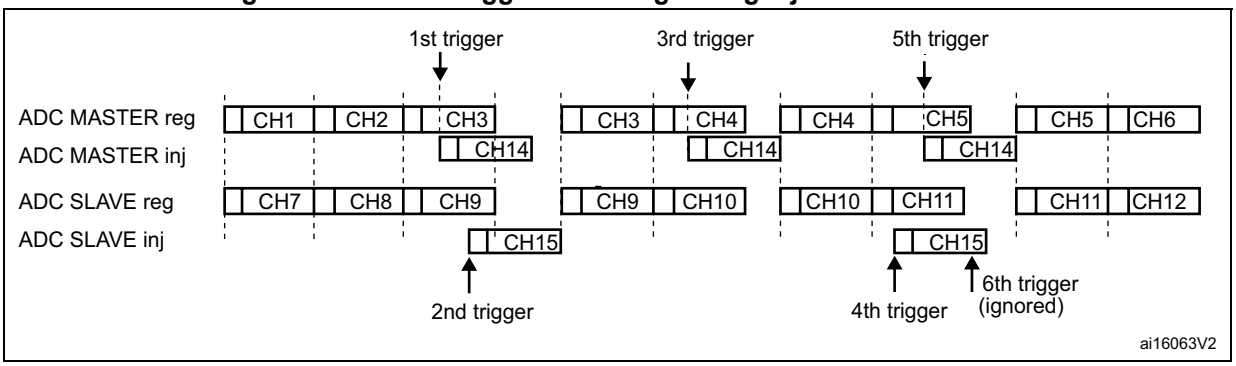
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 75. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 76* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

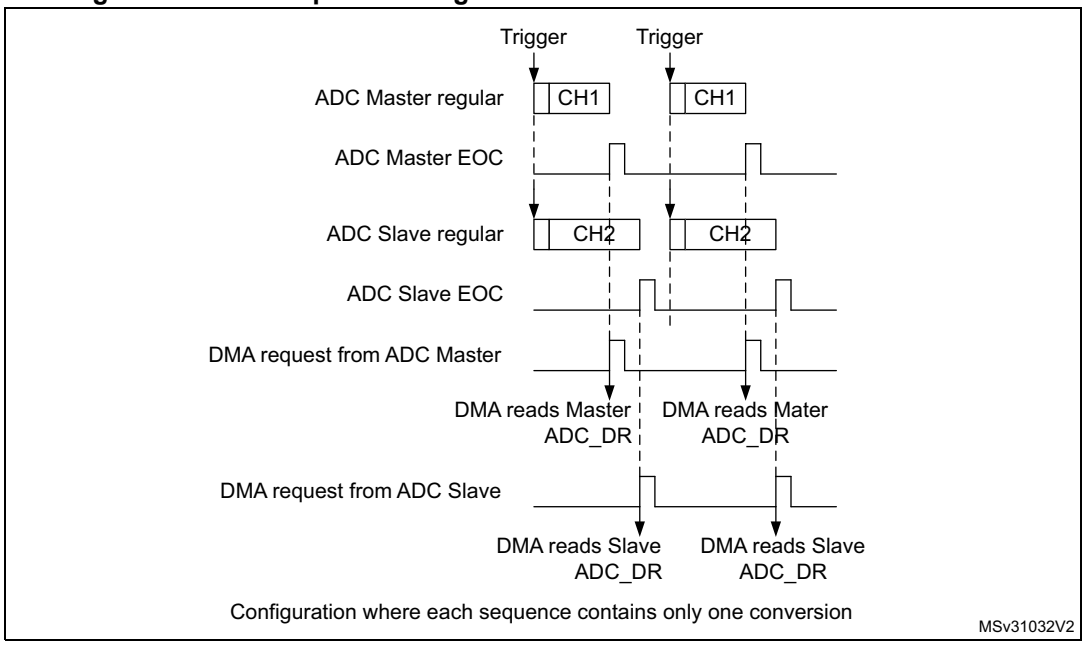
Figure 76. Case of trigger occurring during injected conversion



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 77: DMA Requests in regular simultaneous mode when MDMA=0b00*).

Figure 77. DMA Requests in regular simultaneous mode when MDMA=0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADCx_CCR register:

- **MDMA=0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.
  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[31:0] = SLV_ADCx_DR[15:0] | MST_ADCx_DR[15:0]

  2nd DMA request: ADCx_CDR[31:0] = SLV_ADCx_DR[15:0] | MST_ADCx_DR[15:0]

Figure 78. DMA requests in regular simultaneous mode when MDMA=0b10

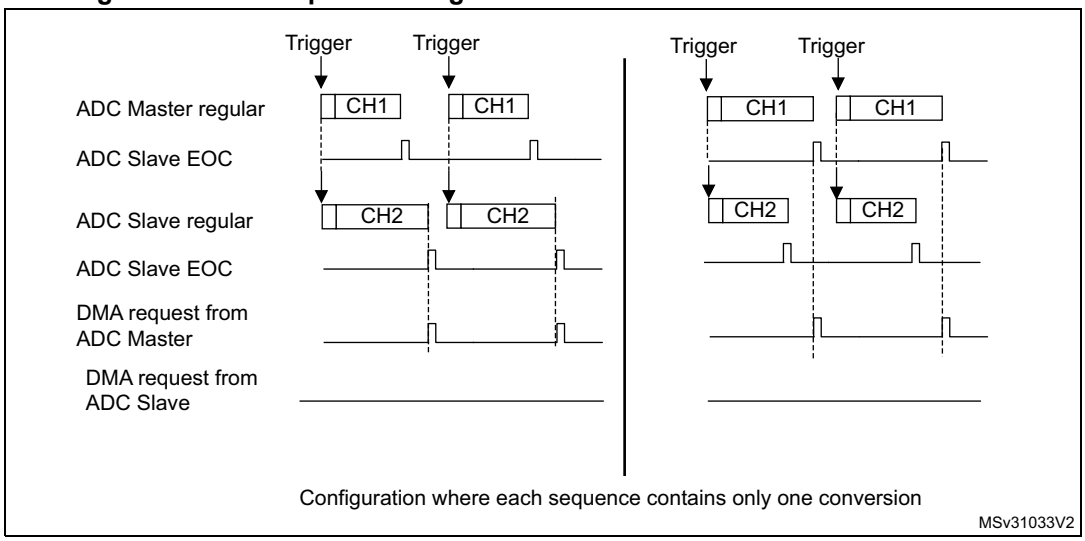
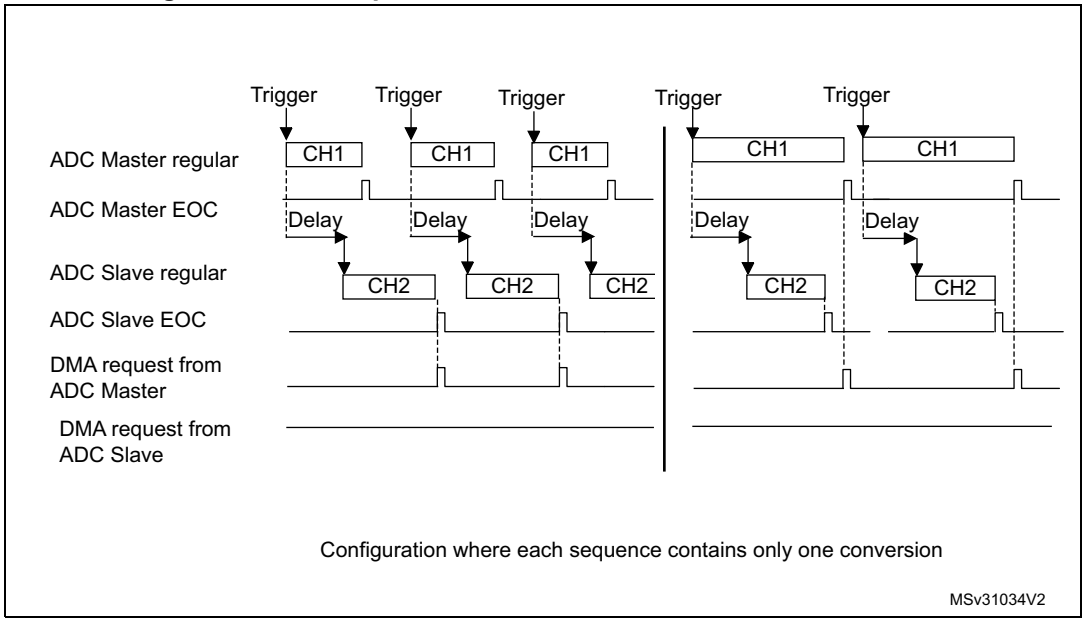


Figure 79. DMA requests in interleaved mode when MDMA=0b10



Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA=0b11**: This mode is similar to the MDMA=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[15:0] = SLV_ADCx_DR[7:0] | MST_ADCx_DR[7:0]

  2nd DMA request: ADCx_CDR[15:0] = SLV_ADCx_DR[7:0] | MST_ADCx_DR[7:0]

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when MDMA mode is selected

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADCx_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA* (bits DMACFG of master and slave ADCx_CFGR are not relevant).

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

13.3.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature (TJ) of the device. The temperature sensor is internally connected to the input channels which are used to convert the sensor output voltage to a digital value. When not in use, the sensor can be put in power down mode.

Figure 80 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the

temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the STM32F334xx datasheet for additional information.

Main features

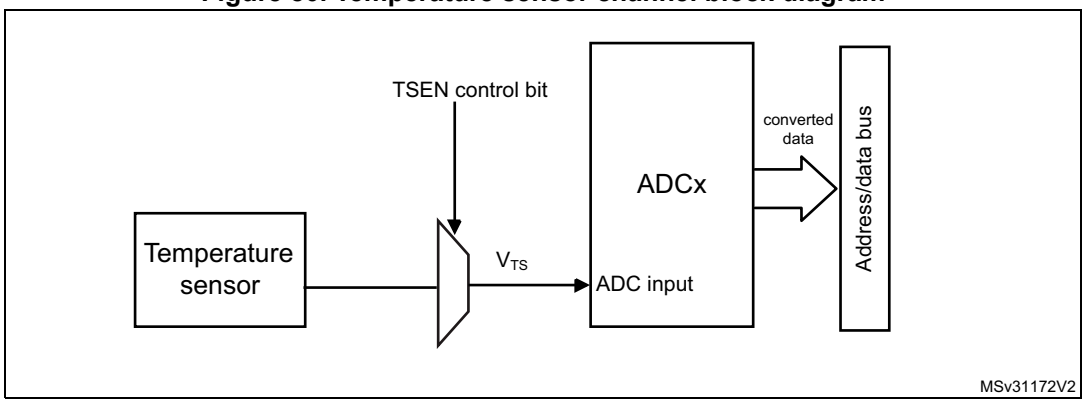
- Supported temperature range: –40 to 125 °C
- Precision: ±2 °C

The temperature sensor is internally connected to the ADC1_IN16 input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of STM32F334xx datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 80 shows the block diagram of the temperature sensor.

Figure 80. Temperature sensor channel block diagram



Note: The TSEN bit must be set to enable the conversion of the temperature sensor voltage V_{TS}.

Reading the temperature

To use the sensor:

1. Select the ADC1_IN16 input channel (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the STM32F334xx datasheet).
3. Set the TSEN bit in the ADC1_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{TS} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

   Temperature (in °C) = {(V_{25} – V_{TS}) / Avg_Slope} + 25

   Where:

   – V_{25} = V_{TS} value for 25° C
   – Avg_Slope = average slope of the temperature vs. V_{TS} curve (given in mV/°C or µV/°C)

   Refer to the datasheet electrical characteristics section for the actual values of V_{25} and Avg_Slope.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{TS} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.

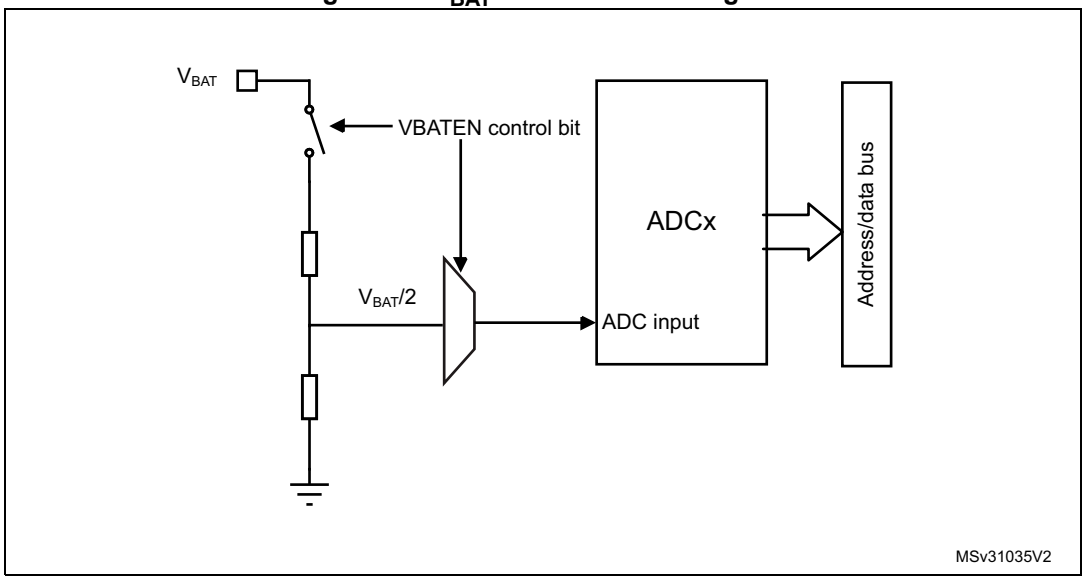
13.3.31 V_{BAT} supply monitoring

The VBATEN bit in the ADC12_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 2. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/2$ to the ADC1_IN17 input channel. As a consequence, the converted digital value is half the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the STM32F334xx datasheet for the sampling time value to be applied when converting the $V_{BAT}/2$ voltage.

Figure 81 shows the block diagram of the V_{BAT} sensing feature.

Figure 81. V_{BAT} channel block diagram



Note: The VBATEN bit must be set to enable the conversion of internal channel ADC1_IN17 (V_{BATEN}).

13.3.32 Monitoring the internal voltage reference

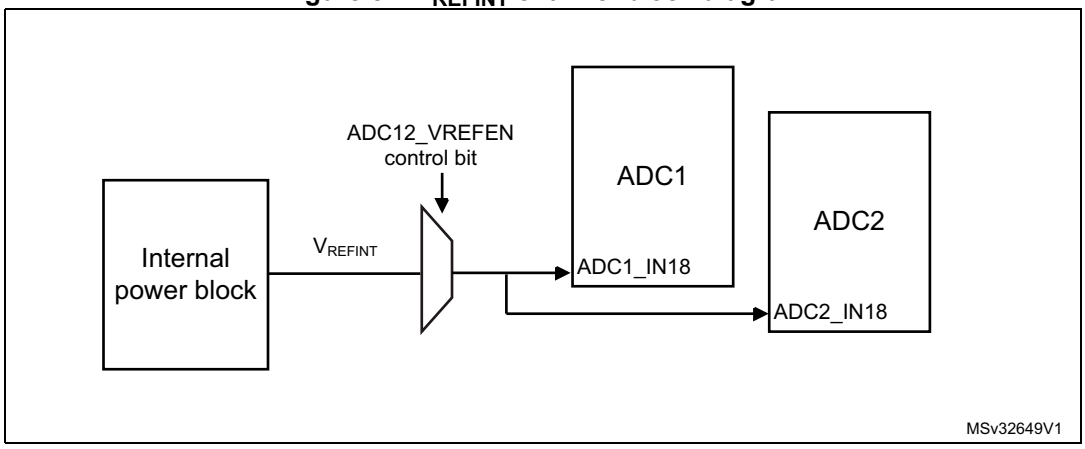
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal voltage reference is internally connected to the input channel 18 of the two ADCs (ADCx_IN18).

Refer to the electrical characteristics section of the STM32F334xx datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 81 shows the block diagram of the V_{REFINT} sensing feature.

Figure 82. V_{REFINT} channel block diagram



Note: The VREFEN bit into ADC12_CCR register must be set to enable the conversion of internal channels ADC1_IN18 or ADC2_IN18 (V_{REFINT}).

Calculating the actual V_{DDA} voltage using the internal reference voltage

The V_{DDA} power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference (V_{REFINT}) and its calibration data acquired by the ADC during the manufacturing process at V_{DDA} = 3.3 V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{DDA} = 3.3\ V\ x\ VREFINT\_CAL / VREFINT\_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V_{DDA}. For applications where V_{DDA} is known and ADC converted values are right-aligned user can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL\_SCALE} \times ADCx\_DATA$$

For applications where V_{DDA} value is not known, user must use the internal voltage reference and V_{DDA} can be replaced by the expression provided in the section *Calculating the actual V_{DDA} voltage using the internal reference voltage*, resulting in the following formula:

$$V_{CHANNELx} = \frac{3.3\ V \times VREFINT\_CAL \times ADCx\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- ADCx_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it will be 2^{12} - 1 = 4095 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

13.4 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 47. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

13.5 ADC registers (for each ADC)

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

13.5.1 ADC interrupt and status register (ADCx_ISR, x=1..2)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 13.3.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADCx_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADCx_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADCx_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADCx_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADCx_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADCx_DR register. It is cleared by software writing 1 to it or by reading the ADCx_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

13.5.2 ADC interrupt enable register (ADCx_IER, x=1..2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQ OVFIE	AWD3 IE	AWD2 IE	AWD1 IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMP IE	ADRDY IE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

13.5.3 ADC control register (ADCx_CR, x=1..2)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	ADVREGEN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: Software is allowed to launch a calibration by setting ADCAL only when ADEN=0.

Note: Software is allowed to update the calibration factor by writing ADCx_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL will launch a calibration in Single-ended inputs Mode.

1: Writing ADCAL will launch a calibration in Differential inputs Mode.

Note: Software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 29:28 **ADVREGEN[1:0]**: ADC voltage regulator enable

These bits are set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

00: Intermediate state required when moving the ADC voltage regulator from the enabled to the disabled state or from the disabled to the enabled state.

01: ADC Voltage regulator enabled.

10: ADC Voltage regulator disabled (Reset state)

11: reserved

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 13.3.6: ADC voltage regulator (ADVREGEN)*.

Note: The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: Software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: Software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Note: In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: Software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: Software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: Software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: Software is allowed to set ADEN only when all bits of ADCx_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

13.5.4 ADC configuration register (ADCx_CFGR, x=1..2)

Address offset: 0x0C

Reset value: 0x0000 00000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1 EN	AWD1 EN	AWD1S GL	JQM	JDISC EN	DISCNUM[2:0]			DISC EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUT DLY	CONT	OVR MOD	EXTEN[1:0]		EXTSEL[3:0]				ALIGN	RES[1:0]		Res.	DMA CFG	DMA EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: reserved (analog input channel 0 is not mapped)

00001: ADC analog input channel-1 monitored by AWD1

.....

10010: ADC analog input channel-18 monitored by AWD1

others: reserved, must not be used

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR Mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR Mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 13.3.21: Queue of context for injected conversions* for more information.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Note: It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun Mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADCx_DR register is preserved with the old data when an overrun is detected.

1: ADCx_DR register is overwritten with the last conversion result when an overrun is detected.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: Event 0
0001: Event 1
0010: Event 2
0011: Event 3
0100: Event 4
0101: Event 5
0110: Event 6
0111: Event 7
...
1111: Event 15

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Figure : Data register, data alignment and offset (ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN)*

0: Right alignment
1: Left alignment

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN=1.

0: DMA One Shot Mode selected

1: DMA Circular Mode selected

For more details, refer to *Section : Managing conversions using the DMA*

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADCx_CCR register.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the GP-DMA to manage automatically the converted data. For more details, refer to *Section : Managing conversions using the DMA*.

0: DMA disabled

1: DMA enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADCx_CCR register.

13.5.5 ADC sample time register 1 (ADCx_SMPR1, x=1..2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		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]			Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

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

Bits 29:3 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 4.5 ADC clock cycles
011: 7.5 ADC clock cycles
100: 19.5 ADC clock cycles
101: 61.5 ADC clock cycles
110: 181.5 ADC clock cycles
111: 601.5 ADC clock cycles

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bites 2:0 Reserved

13.5.6 ADC sample time register 2 (ADCx_SMPR2, x=1..2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 26:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 4.5 ADC clock cycles
011: 7.5 ADC clock cycles
100: 19.5 ADC clock cycles
101: 61.5 ADC clock cycles
110: 181.5 ADC clock cycles
111: 601.5 ADC clock cycles

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

13.5.7 ADC watchdog threshold register 1 (ADCx_TR1, x=1..2)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 1.
Refer to *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*
Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 1.
Refer to *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*
Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

13.5.8 ADC watchdog threshold register 2 (ADCx_TR2, x = 1..2)

Address offset: 0x24

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 2.
Refer to *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*
Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 2.
Refer to *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*
Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

13.5.9 ADC watchdog threshold register 3 (ADCx_TR3, x=1..2)

Address offset: 0x28

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 13.3.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

13.5.10 ADC regular sequence register 1 (ADCx_SQR1, x=1..2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 4th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 3rd in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 2nd in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 1st in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

13.5.11 ADC regular sequence register 2 (ADCx_SQR2, x=1..2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 9th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 8th in the regular conversion sequence

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 7th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 6th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 5th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

13.5.12 ADC regular sequence register 3 (ADCx_SQR3, x=1..2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 14th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 13th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 12th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 11th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 10th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

13.5.13 ADC regular sequence register 4 (ADCx_SQR4, x=1..2)

Address offset: 0x3C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

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

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 16th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 15th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

13.5.14 ADC regular Data Register (ADCx_DR, x=1..2)

Address offset: 0x40

Reset value: 0x0000 0000

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

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

Bits 15:0 **RDATA[15:0]**: Regular Data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 13.3.26: Data management*.

13.5.15 ADC injected sequence register (ADCx_JSQR, x=1..2)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:2]		
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[1:0]		Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[3:0]				JL[1:0]	
rw	rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 4th in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 3rd in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 2nd in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 1st in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bits 7:6 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 13.3.21: Queue of context for injected conversions)

Bits 5:2 **JEXTSEL[3:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

0000: Event 0
0001: Event 1
0010: Event 2
0011: Event 3
0100: Event 4
0101: Event 5
0110: Event 6
0111: Event 7
...
1111: Event 15

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion
01: 2 conversions
10: 3 conversions
11: 4 conversions

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

13.5.16 ADC offset register (ADCx_OFRy, x=1..2) (y=1..4)

Address offset: 0x60, 0x64, 0x68, 0x6C

Reset value: 0x0000 0000

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

Bit 31 **OFFSETy_EN:** Offset y Enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSETy_CH[4:0]:** Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] will apply.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: Analog input channel 0 is not mapped: value “00000” should not be used

Bits 25:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSETy[11:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADCx_DR (regular conversion) or from in the ADCx_JDRyi registers (injected conversion).

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.
Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

13.5.17 ADC injected data register (ADCx_JDRy, x=1..2, y= 1..4)

Address offset: 0x80 - 0x8C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 13.3.26: Data management*.

13.5.18 ADC Analog Watchdog 2 Configuration Register (ADCx_AWD2CR, x=1..2)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:1]															Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bits 18:1 **AWD2CH[18:1]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel-i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel-i is monitored by AWD2

When AWD2CH[18:1] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 0 Reserved, must be kept at reset value.

13.5.19 ADC Analog Watchdog 3 Configuration Register (ADCx_AWD3CR, x=1..2)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:16]		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:1]															Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bits 18:1 **AWD3CH[18:1]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel-i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel-i is monitored by AWD3

When AWD3CH[18:1] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 0 Reserved, must be kept at reset value.

13.5.20 ADC Differential Mode Selection Register (ADCx_DIFSEL, x=1..2)

Address offset: 0xB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:16]		
													r	r	r

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

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

Bits 18:16 **DIFSEL[18:16]**: Differential mode for channels 18 to 16.

These bits are read only. These channels are forced to single-ended input mode (either connected to a single-ended I/O port or to an internal channel).

Bits 15:1 **DIFSEL[15:1]**: Differential mode for channels 15 to 1

These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Note: It is mandatory to keep cleared ADC1_DIFSEL[15] (connected to an internal single ended channel)

Bit 0 Reserved, must be kept at reset value.

13.5.21 ADC Calibration Factors (ADCx_CALFACT, x=1..2)

Address offset: 0xB4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new differential calibration is launched.

Note: Software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In Single-Ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new single-ended calibration is launched.

Note: Software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

13.6 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

13.6.1 ADC Common status register (ADCx_CSR, x=12)

Address offset: 0x00 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADCx_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
							Slave ADC								
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							Master ADC								
					r	r	r	r	r	r	r	r	r	r	r

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

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADCx_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADCx_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADCx_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC
This bit is a copy of the EOS bit in the corresponding ADCx_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADCx_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADCx_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx_ISR register.

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

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADCx_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADCx_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADCx_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADCx_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADCx_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADCx_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx_ISR register.

13.6.2 ADC common control register (ADCx_CCR, x=12)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBAT EN	TS EN	VREF EN	Res.	Res.	Res.	Res.	CKMODE[1:0]	
							rw	rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDMA[1:0]		DMA CFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled
1: V_{BAT} channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 23 **TSEN**: Temperature sensor enable

This bit is set and cleared by software to enable/disable the temperature sensor channel.

0: Temperature sensor channel disabled
1: Temperature sensor channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

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

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK_ADCx (x=123) (Asynchronous clock mode), generated at product level (refer to *Section 8: Reset and clock control (RCC)*)

01: HCLK/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: HCLK/2 (Synchronous clock mode)

11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:14 **MDMA[1:0]:** Direct memory access mode for dual ADC mode

This bit-field is set and cleared by software. Refer to the DMA controller section for more details.

00: MDMA mode disabled

01: reserved

10: MDMA mode enabled for 12 and 10-bit resolution

11: MDMA mode enabled for 8 and 6-bit resolution

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG:** DMA configuration (for dual ADC mode)

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN=1.

0: DMA One Shot Mode selected

1: DMA Circular Mode selected

For more details, refer to *Section : Managing conversions using the DMA*

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY:** Delay between 2 sampling phases

Set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 48* for the value of ADC resolution versus DELAY bits values.

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.

All the ADCs independent:

00000: Independent mode

00001 to 01001: Dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined Interleaved mode + injected simultaneous mode
00100: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
All other combinations are reserved and must not be programmed

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Table 48. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}	1 * T_{ADC_CLK}
0001	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}	2 * T_{ADC_CLK}
0010	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}	3 * T_{ADC_CLK}
0011	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}	4 * T_{ADC_CLK}
0100	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}	5 * T_{ADC_CLK}
0101	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}	6 * T_{ADC_CLK}
0110	7 * T_{ADC_CLK}	7 * T_{ADC_CLK}	7 * T_{ADC_CLK}	6 * T_{ADC_CLK}
0111	8 * T_{ADC_CLK}	8 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1000	9 * T_{ADC_CLK}	9 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1001	10 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1010	11 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
1011	12 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}
others	12 * T_{ADC_CLK}	10 * T_{ADC_CLK}	8 * T_{ADC_CLK}	6 * T_{ADC_CLK}

13.6.3 ADC common regular data register for dual mode (ADCx_CDR, x=12)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 13.3.29: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 13.3.29: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

13.7 ADC register map

The following table summarizes the ADC registers.

Table 49. ADC global register map(1)

Offset	Register
0x000 - 0x04C	Master ADC1
0x050 - 0x0FC	Reserved
0x100 - 0x14C	Slave ADC2
0x118 - 0x1FC	Reserved
0x200 - 0x24C	Reserved
0x250 - 0x2FC	Reserved
0x300 - 0x308	Master and slave ADCs common registers (ADC12)

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

Table 50. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADCx_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	**ADCx_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	**ADCx_CR**	ADCAL	ADCALDIF	ADVREGEN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x0C	**ADCx_CFGR**	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL [3:0]				ALIGN	RES [1:0]		Res.	DMACFG	DMAEN
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x10	Reserved	Res.																															
0x14	**ADCx_SMPR1**	Res.	Res.	SMP9 [2:0]			SMP8 [2:0]			SMP7 [2:0]			SMP6 [2:0]			SMP5 [2:0]			SMP4 [2:0]			SMP3 [2:0]			SMP2 [2:0]			SMP1 [2:0]			Res.	Res.	Res.
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x18	**ADCx_SMPR2**	Res.	Res.	Res.	Res.	Res.	SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [2:0]		
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**ADCx_TR1**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	Res.	Res.	Res.	LT1[11:0]												
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADCx_TR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	**ADCx_TR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x2C	Reserved	Res.																															
0x30	**ADCx_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]						Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[3:0]		
	Reset value				0	0	0	0	0		0	0	0	0	0			0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x34	**ADCx_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]						Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	Res.	SQ5[4:0]		
	Reset value				0	0	0	0	0		0	0	0	0	0			0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x38	**ADCx_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]						Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	Res.	SQ10[4:0]		
	Reset value				0	0	0	0	0		0	0	0	0	0			0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x3C	**ADCx_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADCx_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x4C	**ADCx_JSQR**	Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]			Res.	JSQ1[4:0]				JEXTEN[1:0]	JEXTSEL [3:0]				JL[1:0]					
	Reset value		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x50-0x5C	Reserved	Res																															

Table 50. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	**ADCx_OFR1**	OFFSET1_EN	OFFSET1_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x64	**ADCx_OFR2**	OFFSET2_EN	OFFSET2_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x68	**ADCx_OFR3**	OFFSET3_EN	OFFSET3_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x6C	**ADCx_OFR4**	OFFSET4_EN	OFFSET4_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															
0x80	**ADCx_JDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**ADCx_JDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**ADCx_JDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**ADCx_JDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C-0x9C	Reserved	Res.																															
0xA0	**ADCx_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:1]																		Res.
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xA4	**ADCx_AWD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:1]																		Res.
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xA8-0xAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xB0	**ADCx_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:1]																		Res.
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xB4	**ADCx_CALFACT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0

Table 51. ADC register map and reset values (master and slave ADC common registers) offset =0x300, x=1)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADCx_CSR**	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
									slave ADC2															master ADC1									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	**ADCx_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.	Res.	Res.	Res.	CKMODE[1:0]		MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0					0	0	0	0	0		0	0	0	0				0	0	0	0	0
0x0C	**ADCx_CDR**	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

14 Digital-to-analog converter (DAC1 and DAC2)

14.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. An input reference voltage, V_{REF+}(shared with ADC), is available. The output can optionally be buffered for higher current drive.

14.2 DAC1/2 main features

The devices integrate three 12-bit DAC channels:

- DAC1 integrates two DAC channels:
  - DAC1 channel 1 which output is DAC1_OUT1
  - DAC1 channel 2 which output is DAC1_OUT2

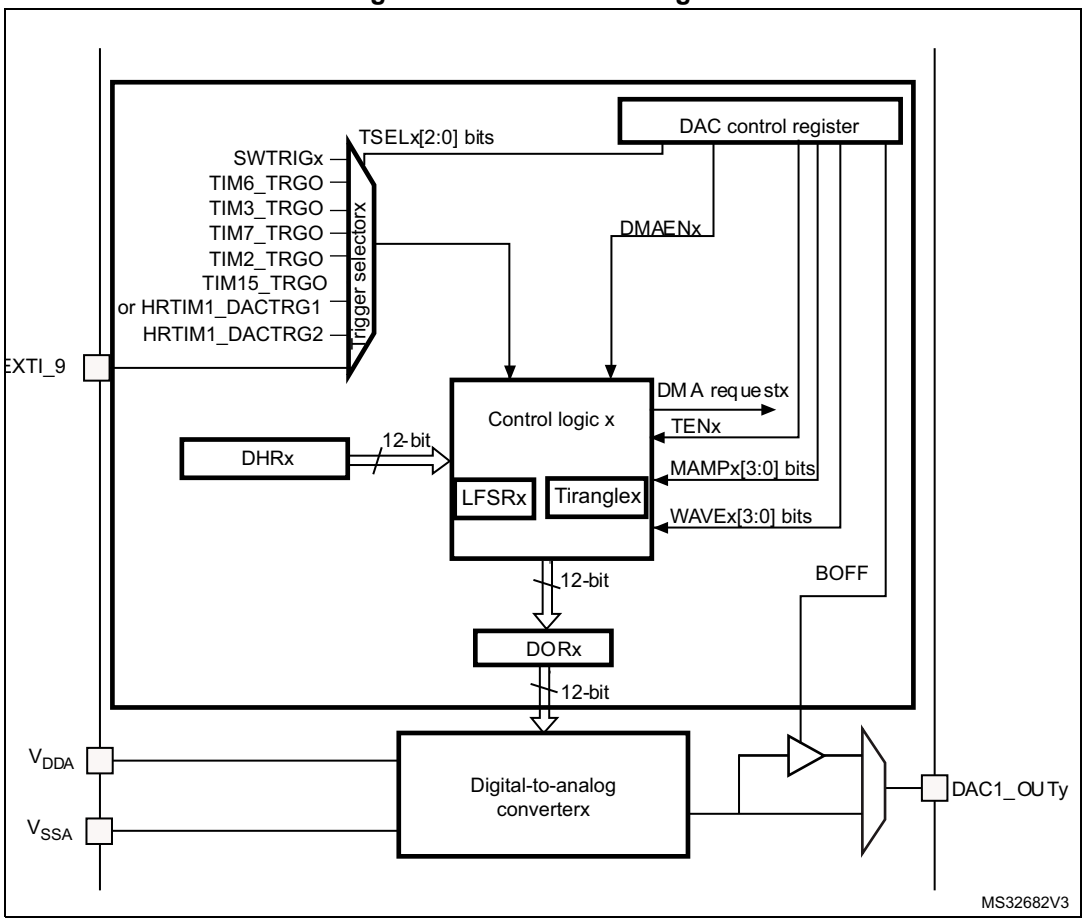
  The two channels can be used independently or simultaneously when both channels are grouped together for synchronous update operations (dual mode).
- DAC2 integrates only one channel, DAC2 channel 1 which output is DAC2_OUT1 .

The DAC main features are the following:

- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation (DAC1 only)
- Triangular-wave generation (DAC1 only)
- Independent or simultaneous conversions (dual mode only)
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Programmable internal buffer
- Input voltage reference, V_{DDA}

Figure 83 and *Figure 84* show the block diagram of a DAC1 and DAC2 channel and *Table 52* gives the pin description.

Figure 83. DAC1 block diagram



1. On STM32F334, there is no output buffer on the DAC1 channel 2. There is instead a switch allowing to connect the DAC1_OUT2 to the corresponding I/O (PA5) (refer to DAC2 block diagram).

Figure 84. DAC2 block diagram

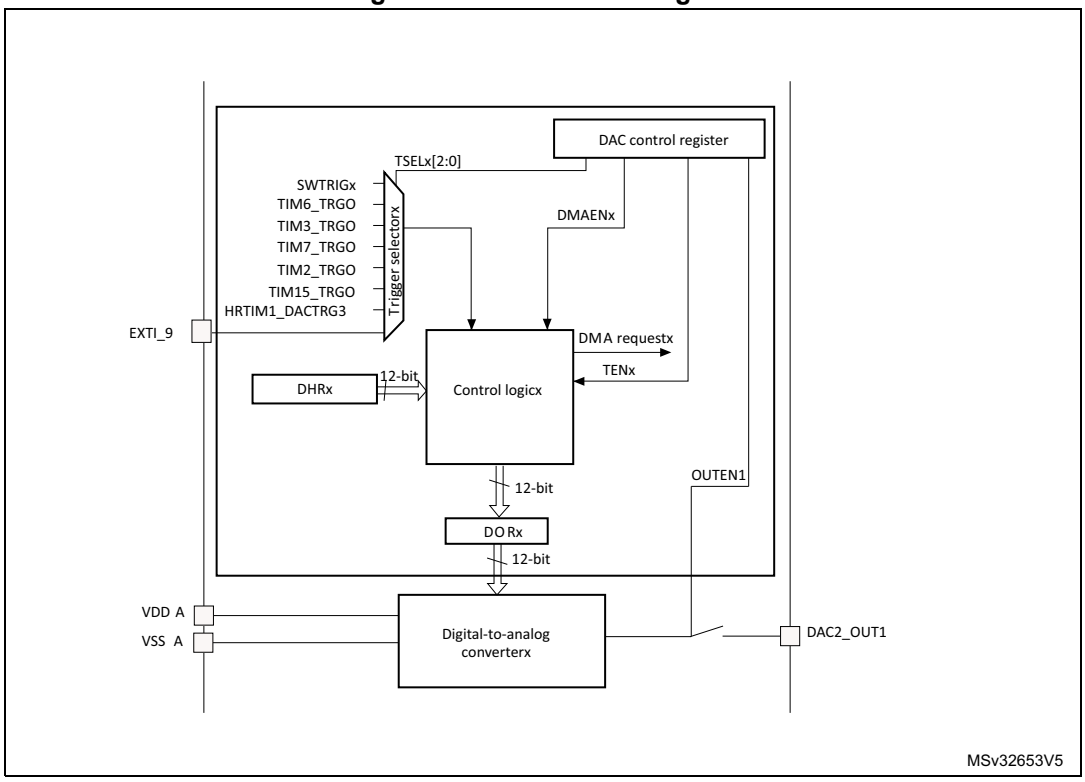


Table 52. DACx pins

Name	Signal type	Remarks
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC1_OUT1/2 DAC2_OUT1	Analog output signal	DACx channel y analog output

Note: Once the DACx channel y is enabled, the corresponding GPIO pin (PA4, PA5 or PA6) is automatically connected to the analog converter output (DACx_OUTy). In order to avoid parasitic consumption, the PA4, PA5 or PA6 pin should first be configured to analog (AIN).

14.3 DAC output buffer enable/DAC output switch

The DAC1 channel 1 comes with an output buffer that can be used to reduce the output impedance on DAC1_OUT1 output, and to drive external loads directly without having to add an external operational amplifier.

In the STM32F334xx, the DAC1 channel 1 comes with an output buffer. The DAC1 channel2 does not have an output buffer, it has instead a switch allowing to connect the DAC1_OUT2 to the corresponding I/O (PA5). The switch can be enabled and disabled through the OUTEN2 bit in the DAC_CR register. The DAC2 channel1 does not have an output buffer, it has instead a switch allowing to connect the DAC2_OUT1 to the

corresponding I/O (PA6). The switch can be enabled and disabled through the OUTEN1 bit in the DAC_CR register.

The DAC1 channel output buffer can be enabled and disabled through the BOFF1 bit in the DAC_CR register.

14.4 DAC channel enable

Each DAC channel can be powered on by setting the corresponding ENx bit in the DAC_CR register. Each 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.

14.5 Single mode functional description

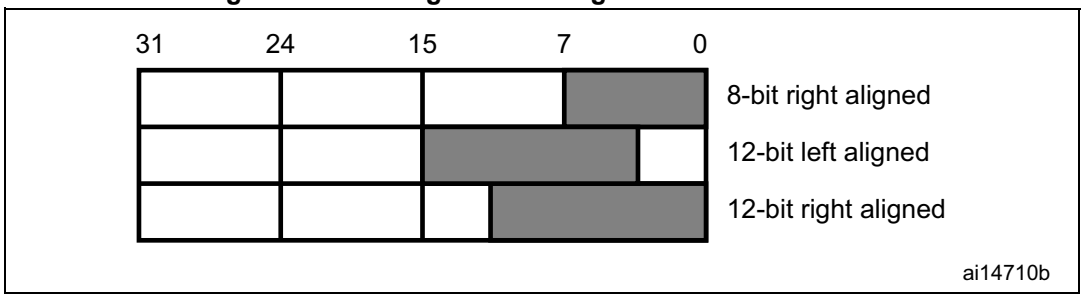
14.5.1 DAC data format

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 85. Data registers in single DAC channel mode



14.5.2 DAC channel conversion

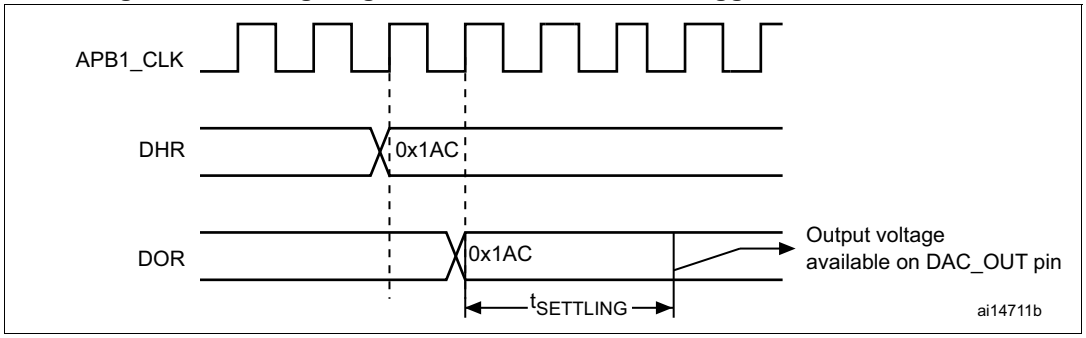
The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR

register is set) and a trigger occurs, the transfer is performed three PCLK1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 86. Timing diagram for conversion with trigger disabled TEN = 0



Independent trigger with single LFSR generation

To configure the DAC in this conversion mode (see *Section 14.7: Noise generation*), the following sequence is required:

1. Set the DAC channel trigger enable bit TENx.
2. Configure the trigger source by setting TSELx[2:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
4. Load the DAC channel data into the desired DAC_DHRx register (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the LFSRx counter, with the same mask, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). Then the LFSRx counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode (see *Section 14.8: Triangle-wave generation*), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure the trigger source by setting TSELx[2:0] bits.
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC_DHRx register. (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

14.5.3 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{DDA}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{DDA} \times \frac{DOR}{4096}$$

14.5.4 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 possible events will trigger conversion as shown in *Table 53*.

Table 53. External triggers (DAC1)

Source	Type	TSEL[2:0]
TIM6_TRGO event	Internal signal from on-chip timers	000
TIM3_TRGO event(1)		001
TIM7_TRGO event		010
TIM15_TRGO event or HRTIM1_DACTRG1 event(2)		011(2)
TIM2_TRGO event		100
HRTIM1_DACTRG2 event(3)		101(3)
EXTI line9	External pin	110
SWTRIG	Software control bit	111

1. To select TIM3_TRGO event as DAC1 trigger source, the DAC_ TRIG_RMP bit must be set in SYSCFG_CFGR1 register.
2. When TSEL=011, the DAC trigger is selected using the DAC1_TRIG3_RMP bit in SYSCFG_CFGR3 register.
3. When TSEL=101, the DAC trigger is selected using the DAC1_TRIG5_RMP bit in SYSCFG_CFGR3 register.

Table 54. External triggers (DAC2)

Source	Type	TSEL[2:0]
TIM6_TRGO event	Internal signal from on-chip timers	000
TIM3_TRGO event(1)		001
TIM7_TRGO event		010
TIM15_TRGO event		011
TIM2_TRGO event		100
HRTIM1_DACTRG3 event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

1. To select TIM3_TRGO event as DAC1 trigger source, the DAC_ TRIG_RMP bit must be set in SYSCFG_CFGR1 register.

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[2:0] bit cannot be changed when the ENx bit is set. When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one APB1 clock cycle.

14.6 Dual-mode functional description

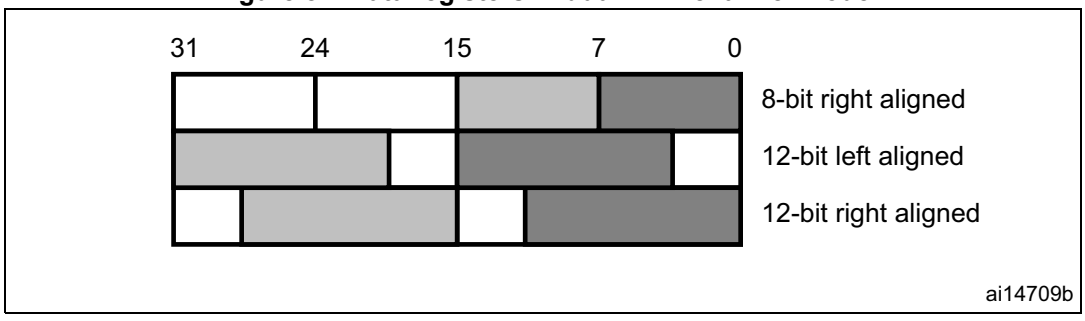
14.6.1 DAC data format

In Dual DAC channel mode, there are three possibilities:

- 8-bit right alignment: data for DAC channel1 to be loaded in the DAC_DHR8RD [7:0] bits (stored in the DHR1[11:4] bits) and data for DAC channel2 to be loaded in the DAC_DHR8RD [15:8] bits (stored in 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 in the DHR2[11:0] bits)
- 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored in the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [27:16] bits (stored in the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored in 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 87. Data registers in dual DAC channel mode



14.6.2 DAC channel conversion in dual mode

The DAC channel conversion in dual mode is performed in the same way as in single mode (refer to *Section 14.5.2*) except that the data have to be loaded by writing to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12RD.

14.6.3 Description of dual conversion modes

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 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.

Refer to *Section 14.5.2: DAC channel conversion* for details on the APB bus (APB or APB1) that clocks the DAC conversions.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode (refer to *Section 14.7: Noise generation*), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB 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 APB clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode (refer to *Section 14.7: Noise generation*), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB 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 APB clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode (refer to *Section 14.8: Triangle-wave generation*), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure different trigger sources by setting different values in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC_DHRx register.

Refer to *Section 14.5.2: DAC channel conversion* for details on the APB bus (APB or APB1) that clocks the DAC conversions.

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode (refer to *Section 14.8: Triangle-wave generation*), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure different trigger sources by setting different values in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC_DHRx register.

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with a triangle amplitude configured by MAMPx[3:0], is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

1. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB clock cycles).

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. 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 APB clock cycles).

Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode (refer to *Section 14.7: Noise generation*), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
4. Load the dual DAC channel data 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 APB 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 APB clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode (refer to *Section 14.7: Noise generation*), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a 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 APB 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 APB clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode (refer to *Section 14.8: Triangle-wave generation*), the following sequence is required:

1. Set the DAC channelx trigger enable TEN1x bits.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSELx[2:0] bits.
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC_DHRx registers.

When a trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode ‘refer to *Section 14.8: Triangle-wave generation*), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure the same trigger source for DAC channelx by setting the same value in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMPx[3:0] bits.
4. Load the DAC channelx data into the desired DAC_DHRx registers.

When a trigger arrives, the DAC channelx triangle counter, with a triangle amplitude configured by MAMPx[3:0], is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). Then the DAC channelx triangle counter is updated.

14.6.4 DAC output voltage

Refer to *Section 14.5.3: DAC output voltage*.

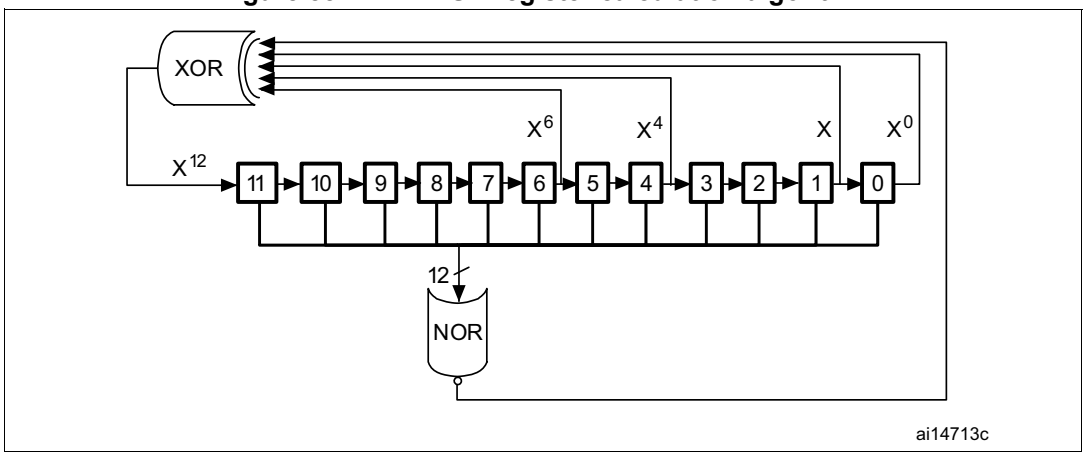
14.6.5 DAC trigger selection

Refer to *Section 14.5.4: DAC trigger selection*

14.7 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in LFSR is 0xAAA. This register is updated three APB clock cycles after each trigger event, following a specific calculation algorithm.

Figure 88. DAC LFSR register calculation algorithm

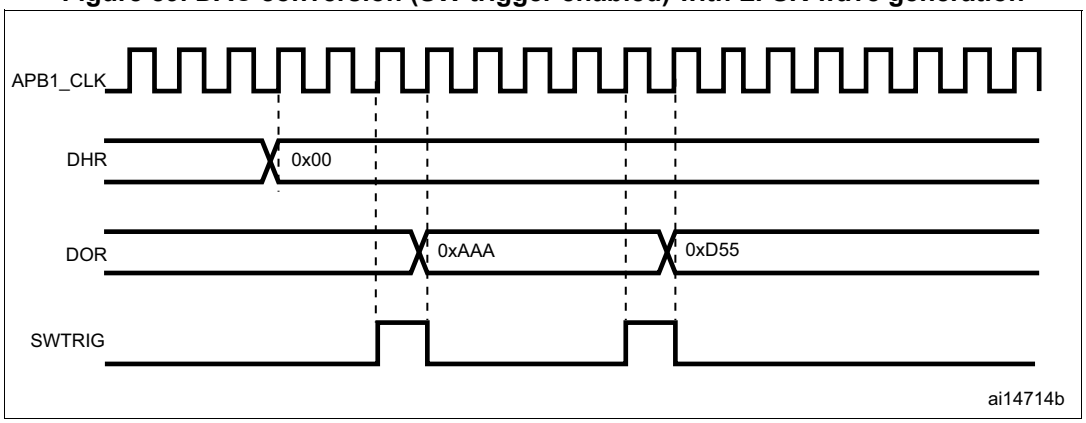


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

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 89. 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.

14.8 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 90. DAC triangle wave generation

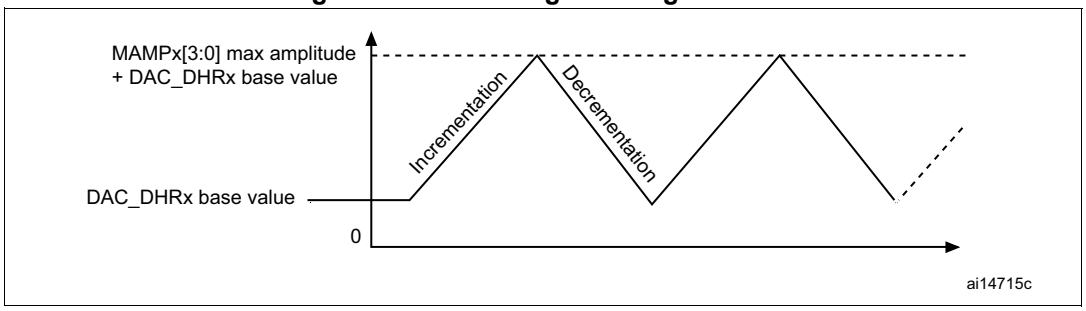
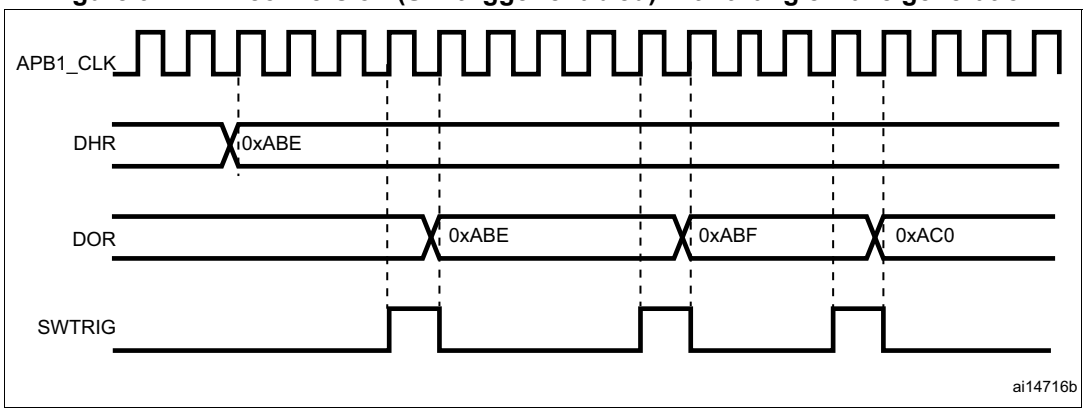


Figure 91. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

14.9 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred to the DAC_DORx register.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, 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 acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

The software should clear the DMAUDRx flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channel, an interrupt is also generated if the corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

14.10 DAC registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

14.10.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2/OUTEN2	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
Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1/OUTEN1	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.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available in dual mode only. It is reserved in single mode.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available in dual mode only. It is reserved in single mode.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only in dual mode when wave generation is supported. Otherwise, they are reserved and must be kept at reset value.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable
These bits are set/reset by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only in dual mode when wave generation is supported. Otherwise, they are reserved and must be kept at reset value.

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 3 TRGO event
010: Timer 7 TRGO event
011: Timer 15 TRGO or HRTM1_DACTRG1 event
100: Timer 2 TRGO event
101: HRTIM1_DACTRG2 event
110: EXTI line9
111: Software trigger
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available in dual mode only. They are reserved in single mode.

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 APB1clock 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 reg*ister to the DAC_DOR2 register takes only one APB1 clock cycle.*
Note: This bit is available in dual mode only. It is reserved in single mode.

Bit 17 **OUTEN2:** DAC channel2 output switch enable
This bit is set and cleared by software to enable/disable DAC channel2 output switch.
0: DAC channel2 output switch disabled
1: DAC channel2 output switch enabled

Bit 16 **EN2**: DAC channel2 enable
This bit is set and cleared by software to enable/disable DAC channel2.
0: DAC channel2 disabled
1: DAC channel2 enabled
Note: This bit is available in dual mode only. It is reserved in single mode.

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 3 TRGO event
010: Timer 7 TRGO event
011: Timer15 TRGO or HRTM1_DACTRG1 event (DAC1 only)
100: Timer 2 TRGO event
101: HRTIM1_DACTRG2 (DAC1) or HRTM1_DACTRG3 (DAC2) event
110: EXTI line9
111: Software trigger

Note: When TSEL=011, the DAC trigger is selected using the DAC1_TRIG3_RMP bit in SYSCFG_CFGR3register.

Note: When TSEL=101, the DAC trigger is selected using the DAC1_TRIG5_RMP bit in SYSCFG_CFGR3register.

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 **In DAC1:**

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

In DAC2: (STM32F334xx only)

OUTEN1: DAC channel1 output switch enable

This bit is set and cleared by software to enable/disable DAC channel1 output switch.

0: DAC channel1 output switch disabled

1: DAC channel1 output switch enabled

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

14.10.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.

This bit is available in dual mode only. It is reserved in single mode.

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.

14.10.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

14.10.4 DAC channel1 12-bit left-aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												v	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

14.10.5 DAC channel1 8-bit right-aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

14.10.6 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

14.10.7 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

14.10.8 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

14.10.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	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.

14.10.10 Dual DAC 12-bit left-aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	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]												Res.	Res.	Res.	Res.
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.

14.10.11 Dual DAC 8-bit right-aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

14.10.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

14.10.13 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

14.10.14 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
Res.	Res.	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		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)

Note: This bit is available in dual mode only. It is reserved in single mode.

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.

14.10.15 DAC register map

Table 55 summarizes the DAC registers.

Table 55. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	Res.	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_ SWTRIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	DAC_ DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_ DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_ DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	DAC_ DHR12R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_ DHR12L2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_ DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20	DAC_ DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_ DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	DAC_ DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 55. DAC register map (continued)and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**DAC_SR**	Res.	Res.	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0																0													

Refer to *Section 2.2 on page 47* for the register boundary addresses.

15 Comparator (COMP)

15.1 Introduction

STM32F334xx devices embed three ultra-fast comparators, COMP2, COMP4 and COMP6 that can be used either as standalone devices (all terminals are available on I/Os) or combined with the timers.

The comparators can be used for a variety of functions including:

- Wakeup from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with the DAC and a PWM output from a timer.

15.2 COMP main features

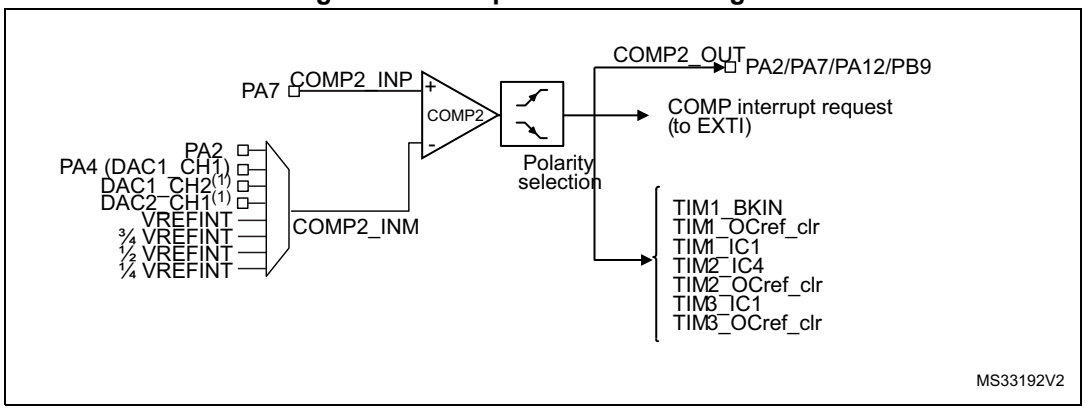
- Rail-to-rail comparators
- Each comparator has positive and configurable negative inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC1 channel 1, DAC1 channel 2, DAC2 channel1
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Capture events
  - OCREF_CLR events (for cycle-by-cycle current control)
  - Break events for fast PWM shutdowns
- .Comparator outputs with blanking source
- Each comparator has interrupt generation capability with wakeup from Sleep and Stop modes (through the EXTI controller)

15.3 COMP functional description

15.3.1 COMP block diagram

The block diagram of the comparators is shown in *Figure 92: Comparator 2 block diagram*.

Figure 92. Comparator 2 block diagram



1. In STM32F334xx devices, DAC1_CH2 and DAC2_CH1 outputs are connected directly, thus PA5 and PA6 are not available as COMP2_INM inputs.

15.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The table below summarizes the I/Os that can be used as comparators inputs and outputs.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

Table 56. STM32F334xx comparator input/outputs summary

	Comparator input/outputs		
	COMP2	**COMP4**	**COMP6**
Comparator inverting Input: connection to internal signals	DAC1_CH1/DAC1_CH2 DAC2_CH1 Vrefint ¾ Vrefint ½ Vrefint ¼ Vrefint		
Comparator Inputs connected to I/Os (+: non inverting input; -: inverting input)	+: PA7 -: PA2	+: PB0 -: PB2	+: PB11 -: PB15
Comparator outputs (motor control protection)	T1BKIN T1BKIN2		
Outputs on I/Os	PA2 PA12	PB1	PA10 PC6
Outputs to internal signals	TIM1_OCREF_CLR TIM1_IC1 TIM2_IC4 TIM2_OCREF_CLR TIM3_IC1 TIM3_OCrefClear HRTIM_EEV1, HRTIM_EEV6(1)	TIM3_IC3 TIM3_OCrefClear TIM15_OCREF_CLR TIM15_IC2 HRTIM_EEV2, HRTIM_EEV7(1)	TIM2_IC2 TIM2_OCREF_CLR TIM16_OCREF_CLR TIM16_IC1 HRTIM_EEV3, HRTIM_EEV8(1)

1. COMP2/4/6 output is connected directly to HRTIM1 peripheral in order to speed-up the propagation delays.

15.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the PCLK2 (APB2 clock).

There is no clock enable control bit provided in the RCC controller. To use a clock source for the comparator, the SYSCFG clock enable control bit must be set in the RCC controller.

Important: The polarity selection logic and the output redirection to the port works independently from the *PCLK2* clock. This allows the comparator to work even in Stop mode.

15.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to

insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

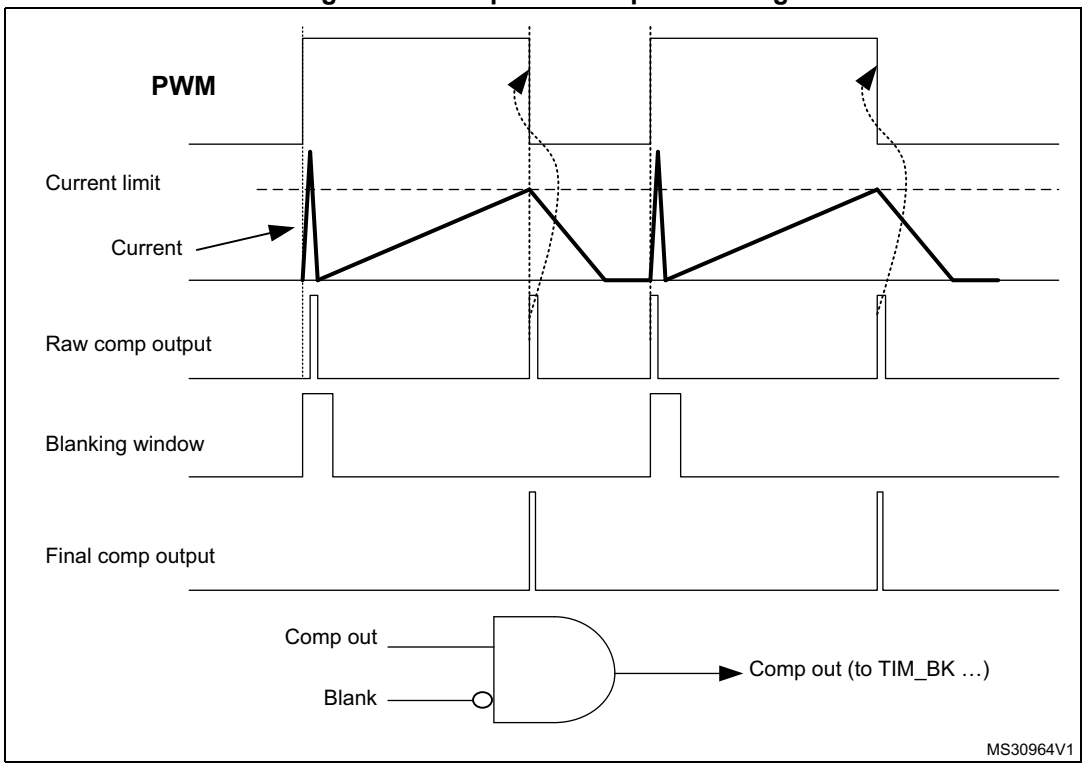
Once the programming is completed, using bits 30:0 of COMPx_CSR, the COMPx LOCK bit can be set to 1. This causes the whole COMPx_CSR register to become read-only, including the COMPx LOCK bit.

The write protection can only be reset by a MCU reset.

15.3.5 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes).It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 93. Comparator output blanking



15.4 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

15.5 COMP registers

15.5.1 COMP2 control and status register (COMP2_CSR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2LOCK	COMP2OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2INMSEL[3]	Res.	COMP2_BLANKING[2:0]			Res.	
rwo	r								rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2POL	Res.	COMP2OUTSEL[3:0]				Res.	Res.	Res.	COMP2INMSEL[2:0]			Res.		Res.	COMP2EN
rw		rw	rw	rw	rw				rw	rw	rw				rw

Bit 31 **COMP2LOCK**: Comparator 2 lock
This bit is write-once. It is set by software. It can only be cleared by a system reset.
It allows to have COMP2_CSR register as read-only.
0: COMP2_CSR is read-write.
1: COMP2_CSR is read-only.

Bit 30 **COMP2OUT**: Comparator 2 output
This read-only bit is a copy of comparator 1output state.
0: Output is low (non-inverting input below inverting input).
1: Output is high (non-inverting input above inverting input).

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **COMP2INMSEL[3]**: Comparator 2 inverting input selection. It is used with Bits [6..4] to configure the Comp inverting input.

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **COMP2_BLANKING[2:0]**: Comparator 2 output blanking source
These bits select which Timer output controls the comparator 1 output blanking.
000: No blanking
001: TIM1 OC5 selected as blanking source
010: TIM2 OC3 selected as blanking source
011: TIM3 OC3 selected as blanking source
Other configurations: reserved

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **COMP2POL**: Comparator 2 output polarity
This bit is used to invert the comparator 2 output.
0: Output is not inverted
1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP2OUTSEL[3:0]**: Comparator 2 output selection
These bits select which Timer input must be connected with the comparator2 output.
0000: No selection
0001: (BRK_ACTH) Timer 1 break input
0010: (BRK2) Timer 1 break input 2
0011: Reserved
0100: Reserved
0101: Timer 1 break input2
0110: Timer 1 OCREF_CLR input
0111: Timer 1 input capture 1
1000: Timer 2 input capture 4
1001: Timer 2 OCREF_CLR input
1010: Timer 3 input capture 1
1011: Timer 3 OCrefclear input
Remaining combinations: reserved.

Note: COMP2 output is connected directly to HRTIM1 to speed-up the propagation delay.

Bits 9:7 Reserved, must be kept at reset value.

Bits 6:4 **COMP2INMSEL[3:0]**: Comparator 2 inverting input selection
These bits, together with bit 22, allows to select the source connected to the inverting input of the comparator 2.
0000: 1/4 of Vrefint
0001: 1/2 of Vrefint
0010: 3/4 of Vrefint
0011: Vrefint
0100: PA4 or DAC1_CH1 output if enabled
0101: DAC1_CH2 output
0110: PA2
1000 DAC2_CH1 output
Remaining combinations: reserved.

Bit 0 **COMP2EN**: Comparator 2 enable
This bit switches COMP2 ON/OFF.
0: Comparator 2 disabled
1: Comparator 2 enabled

15.5.2 COMP4 control and status register (COMP4_CSR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP4LOCK	COMP4OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP4INMSEL[3]	Res.	COMP4_BLANKING[2:0]			Res.	
rwo	r								rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP 4POL	Res.	COMP4OUTSEL[3:0]				Res.	Res.	Res.	COMP4INMSEL[2:0]			Res.		Res.	COMP4 EN
rw		rw	rw	rw	rw				rw	rw	rw				rw

Bit 31 **COMP4LOCK**: Comparator 4 lock
This bit is write-once. It is set by software. It can only be cleared by a system reset.
It allows to have COMP4_CSR register as read-only.
0: COMP4_CSR is read-write.
1: COMP4_CSR is read-only.

Bit 30 **COMP4OUT**: Comparator 4 output
This read-only bit is a copy of comparator 4 output state.
0: Output is low (non-inverting input below inverting input).
1: Output is high (non-inverting input above inverting input).

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **COMP4INMSEL[3]:** Comparator 4 inverting input selection. It is used with Bits [6..4] to configure the Comp inverting input.

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **COMP4_BLANKING**: Comparator 4 blanking source
These bits select which Timer output controls the comparator 4 output blanking.
000: No blanking
001: TIM3 OC4 selected as blanking source
010: Reserved
011: TIM15 OC1 selected as blanking source
Other configurations: reserved, must be kept at reset value

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **COMP4POL**: Comparator 4 output polarity
This bit is used to invert the comparator 4 output.
0: Output is not inverted
1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP4OUTSEL[3:0]**: Comparator 4 output selection
These bits select which Timer input must be connected with the comparator4 output.
0000: No timer input selected
0001: (BRK) Timer 1 break input
0010: (BRK2) Timer 1 break input 2
0011: Reserved
0100: Reserved
0101: Timer 1 break input 2
0110: Timer 3 input capture 3
0111: Reserved
1000: Timer 15 input capture 2
1001: Reserved
1010: Timer 15 OCREF_CLR input
1011: Timer 3 OCrefclear input
Remaining combinations: reserved.

Note: COMP4 output is connected directly to HRTIM1 to speed-up the propagation delay.

Bits 9:7 Reserved, must be kept at reset value.

Bits 6:4 **COMP4INMSEL[3:0]**: Comparator 4 inverting input selection

These bits, together with bit 22, allows to select the source connected to the inverting input of the comparator 4.

0000: 1/4 of Vrefint
0001: 1/2 of Vrefint
0010: 3/4 of Vrefint
0011: Vrefint
0100: PA4 or DAC1_CH1 output if enabled
0101: DAC1_CH2 output
0110: Reserved
0111: PB2
1000: DAC2_CH1 output
Remaining combinations: reserved.

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

Bit 0 **COMP4EN**: Comparator 4 enable

This bit switches COMP4 ON/OFF.

0: Comparator 4 disabled
1: Comparator 4 enabled

15.5.3 COMP6 control and status register (COMP6_CSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP 6LOCK	COMP 6OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP 6INMS EL[3]	Res.	COMP6_BLANKING[2:0]			Res.	
rw	r								rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP 6POL	Res.	COMP6OUTSEL[3:0]				Res.	Res.	Res.	COMP6INMSEL[2:0]			Res.		Res.	COMP6 EN
rw		rw	rw	rw	rw				rw	rw	rw				rw

Bit 31 **COMP6LOCK**: Comparator 6 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

It allows to have COMP6_CSR register as read-only.

0: COMP6_CSR is read-write.
1: COMP6_CSR is read-only.

Bit 30 **COMP6OUT**: Comparator 6 output

This read-only bit is a copy of comparator 6 output state.

0: Output is low (non-inverting input below inverting input).
1: Output is high (non-inverting input above inverting input).

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **COMP6INMSEL[3]**: Comparator 6 inverting input selection. It is used with Bits [6..4] to configure the Comp inverting input.

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **COMP6_BLANKING**: Comparator 6 blanking source
These bits select which Timer output controls the comparator 6 output blanking.
000: No blanking
001: Reserved
010: Reserved
011: TIM2 OC4 selected as blanking source
100: TIM15 OC2 selected as blanking source
Other configurations: reserved
The blanking signal is active high (masking comparator output signal). It is up to the user to program the comparator and blanking signal polarity correctly.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **COMP6POL**: Comparator 6 output polarity
This bit is used to invert the comparator 6 output.
0: Output is not inverted
1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP6OUTSEL[3:0]**: Comparator 6 output selection
These bits select which Timer input must be connected with the comparator 6 output.
0000: No timer input
0001: (BRK_ACTH) Timer 1 break input
0010: (BRK2) Timer 1 break input 2
0101: Timer 1 break input 2
0110: Timer 2 input capture 2
1000: Timer 2 OCREF_CLR input
1001: Timer 16 OCREF_CLR input
1010: Timer 16 input capture 1

Remaining combinations: reserved.

Note: COMP6 output is connected directly to HRTIM1 to speed-up the propagation delay.

Bits 9:7 Reserved, must be kept at reset value.

Bits 6:4 **COMP6INMSEL[3:0]**: Comparator 6 inverting input selection
These bits, together with bit 22, allows to select the source connected to the inverting input of the comparator 6.
0000: 1/4 of Vrefint
0001: 1/2 of Vrefint
0010: 3/4 of Vrefint
0011: Vrefint
0100: PA4 or DAC1_CH1 output if enabled
0101: DAC1_CH2 output
0111: PB15
1000: DAC2_CH1
Remaining combinations: reserved.

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

Bit 0 **COMP6EN**: Comparator 6 enable
This bit switches COMP6 ON/OFF.
0: Comparator 6 disabled
1: Comparator 6 enabled

15.5.4 COMP register map

The following table summarizes the comparator registers.

Table 57. COMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	**COMP2_CSR**	COMP2LOCK	COMP2OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2INMSEL[3]	Res.	COMP2_BLANKING			Res.		COMP2POL	Res.	COMP2OUT SEL[3:0]				Res.	Res.	Res.	COMP2INMSEL[2:0]			Res.		Res.	COMP2EN
	Reset value	0	0								0		0	0	0			0		0	0	0	0				0	0	0				0
0x28	**COMP4_CSR**	COMP4LOCK	COMP4OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP4INMSEL[3]	Res.	COMP4_BLANKING			Res.		COMP4POL	Res.	COMP4OUT SEL[3:0]				Res.	Res.	Res.	COMP4INMSEL[2:0]			Res.		Res.	COMP4EN
	Reset value	0	0								0		0	0	0			0		0	0	0	0				0	0	0				0
0x30	**COMP6_CSR**	COMP6LOCK	COMP6OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP6INMSEL[3]	Res.	COMP6_BLANKING			Res.		COMP6POL	Res.	COMP6OUT SEL[3:0]				Res.	Res.	Res.	COMP6INMSEL[2:0]			Res.		Res.	COMP6EN
	Reset value	0	0								0		0	0	0			0		0	0	0	0				0	0	0				0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

16 Operational amplifier (OPAMP)

16.1 OPAMP introduction

STM32F334xx devices embed 1 operational amplifier OPAMP2. It can either be used as a standalone amplifier or as a follower / programmable gain amplifier.

The operational amplifier output is internally connected to an ADC channel for measurement purposes.

16.2 OPAMP main features

- Rail-to-rail input/output
- Low offset voltage
- Capability of being configured as a standalone operational amplifier or as a programmable gain amplifier (PGA)
- Access to all terminals
- Input multiplexer on inverting and non-inverting input
- Input multiplexer can be triggered by a timer and synchronized with a PWM signal.

16.3 OPAMP functional description

16.3.1 General description

On every OPAMP, there is one 4:1 multiplexer on the non-inverting input and one 2:1 multiplexer on the inverting input.

The inverting and non inverting inputs selection is made using the VM_SEL and VP_SEL bits respectively in the OPAMPx_CSR register.

The I/Os used as OPAMP input/outputs must be configured in analog mode in the GPIOs registers.

The connections with dedicated I/O are summarized in the table below and in *Figure 94*.

Table 58. Connections with dedicated I/O

OPAMP2 inverting input	OPAMP2 non inverting input
PA5 (VM1)	PA7 (VP0)
PC5 (VM0)	PD14 (VP1)
-	PB0 (VP2)

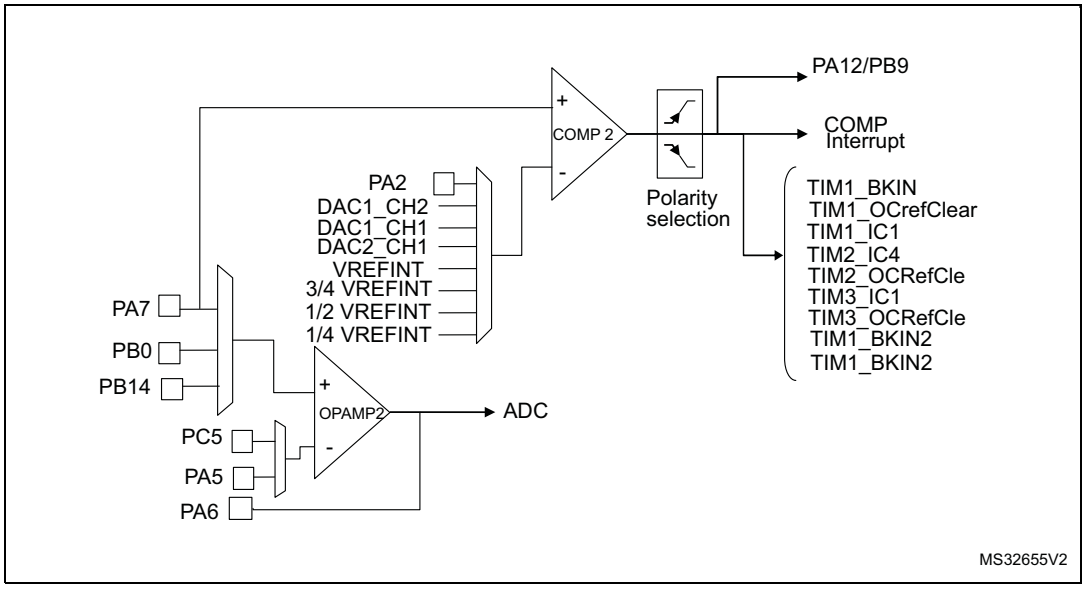
16.3.2 Clock

The OPAMP clock provided by the clock controller is synchronized with the PCLK2 (APB2 clock). There is no clock enable control bit provided in the RCC controller. To use a clock source for the OPAMP, the SYSCFG clock enable control bit must be set in the RCC controller.

16.3.3 Operational amplifiers and comparators interconnections

Internal connections between the operational amplifiers and the comparators are useful in motor control applications. These connections are summarized in the following figures.

Figure 94. STM32F334xx comparator and operational amplifier connections



16.3.4 Using the OPAMP outputs as ADC inputs

In order to use OPAMP outputs as ADC inputs, the operational amplifiers must be enabled and the ADC must use the OPAMP output channel number:

- For OPAMP2, ADC2 channel 3 is used.

16.3.5 Calibration

The OPAMP interface continuously sends trimmed offset values to the 4 operational amplifiers. At startup, these values are initialized with the preset ‘factory’ trimming value.

Furthermore each operational amplifier offset can be trimmed by the user.

The user can switch from the ‘factory’ values to the ‘user’ trimmed values using the USER_TRIM bit in the OPAMP control register. This bit is reset at startup (‘factory’ values are sent to the operational amplifiers).

The rail-to-rail input stage of the OPAMP is composed of two differential pairs:

- One pair composed of NMOS transistors
- One pair composed of PMOS transistors.

As these two pairs are independent, the trimming procedure calibrates each one separately. The TRIMOFFSETN bits calibrate the NMOS differential pair offset and the TRIMOFFSETP bits calibrate the PMOS differential pair offset.

To calibrate the NMOS differential pair, the following conditions must be met: CALON=1 and CALSEL=11. In this case, an internal high voltage reference (0.9 x V_{DDA}) is generated and applied on the inverting and non inverting OPAMP inputs connected together. The voltage

applied to both inputs of the OPAMP can be measured (the OPAMP reference voltage can be output through the TSTREF bit and connected internally to an ADC channel; refer to *Section 13: Analog-to-digital converters (ADC) on page 211*). The software should increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the OUTCAL bit to change from 1 to 0 in the OPAMP register. If the OUTCAL bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored.

The calibration of the PMOS differential pair is performed in the same way, with two differences: the TRIMOFFSETP bits-fields are used and the CALSEL bits must be programmed to '01' (an internal low voltage reference (0.1 x V_{DDA}) is generated and applied on the inverting and non inverting OPAMP inputs connected together).

Note: During calibration mode, to get the correct OUTCAL value, please make sure the OFFTRIMmax delay (specified in the datasheet electrical characteristics section) has elapsed between the write of a trimming value (TRIMOFFSETP or TRIMOFFSETN) and the read of the OUTCAL value,

To calibrate the NMOS differential pair, use the following software procedure:

1. Enable OPAMP by setting the OPAMPxEN bit
2. Enable the user offset trimming by setting the USERTRIM bit
3. Connect VM and VP to the internal reference voltage by setting the CALON bit
4. Set CALSEL to 11 (OPAMP internal reference =0.9 x V_{DDA})
5. In a loop, increment the TRIMOFFSETN value. To exit from the loop, the OUTCAL bit must be reset. In this case, the TRIMOFFSETN value must be stored.

The same software procedure must be applied for PMOS differential pair calibration with CALSEL = 01 (OPAMP internal reference = 0.1 V_{DDA}).

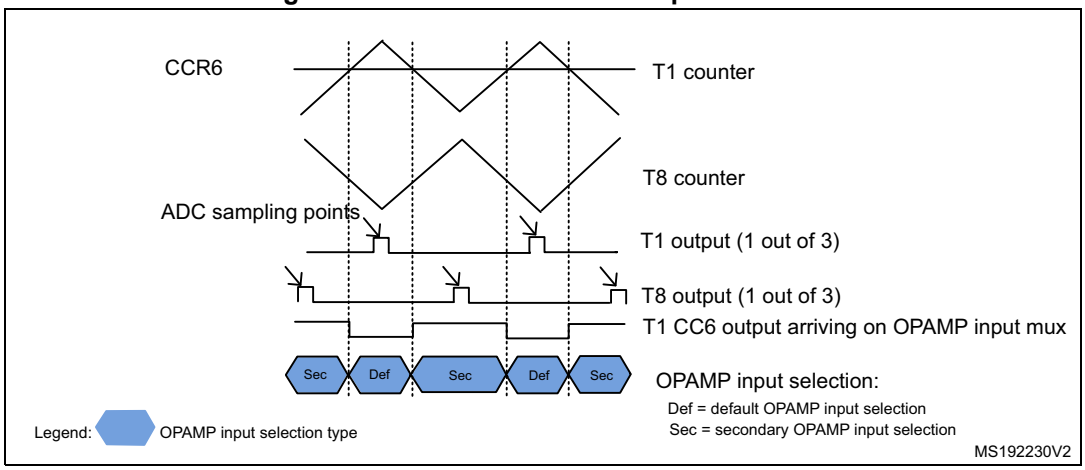
16.3.6 Timer controlled Multiplexer mode

The selection of the OPAMP inverting and non inverting inputs can be done automatically. In this case, the switch from one input to another is done automatically. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP input multiplexers.

This is useful for dual motor control with a need to measure the currents on the 3 phases instantaneously on a first motor and then on the second motor.

The automatic switch is enabled by setting the TCM_EN bit in the OPAMP control register. The inverting and non inverting inputs selection is performed using the VPS_SEL and VMS_SEL bit fields in the OPAMP control register. If the TCM_EN bit is cleared, the selection is done using the VP_SEL and VM_SEL bit fields in the OPAMP control register.

Figure 95. Timer controlled Multiplexer mode



16.3.7 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

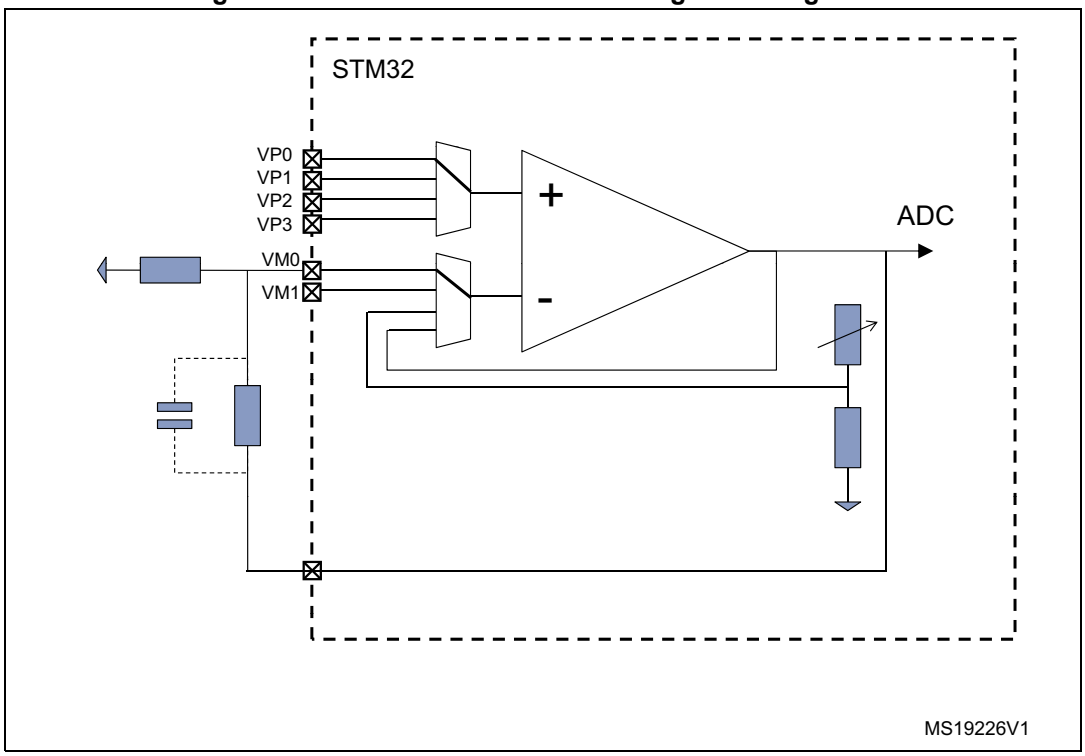
Important note: the amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.

Note: The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artefacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

Standalone mode (external gain setting mode)

The external gain setting mode gives full flexibility to choose the amplifier configuration and feedback networks. This mode is enabled by writing the VM_SEL bits in the OPAMPx_CR register to 00 or 01, to connect the inverting inputs to one of the two possible I/Os.

Figure 96. Standalone mode: external gain setting mode

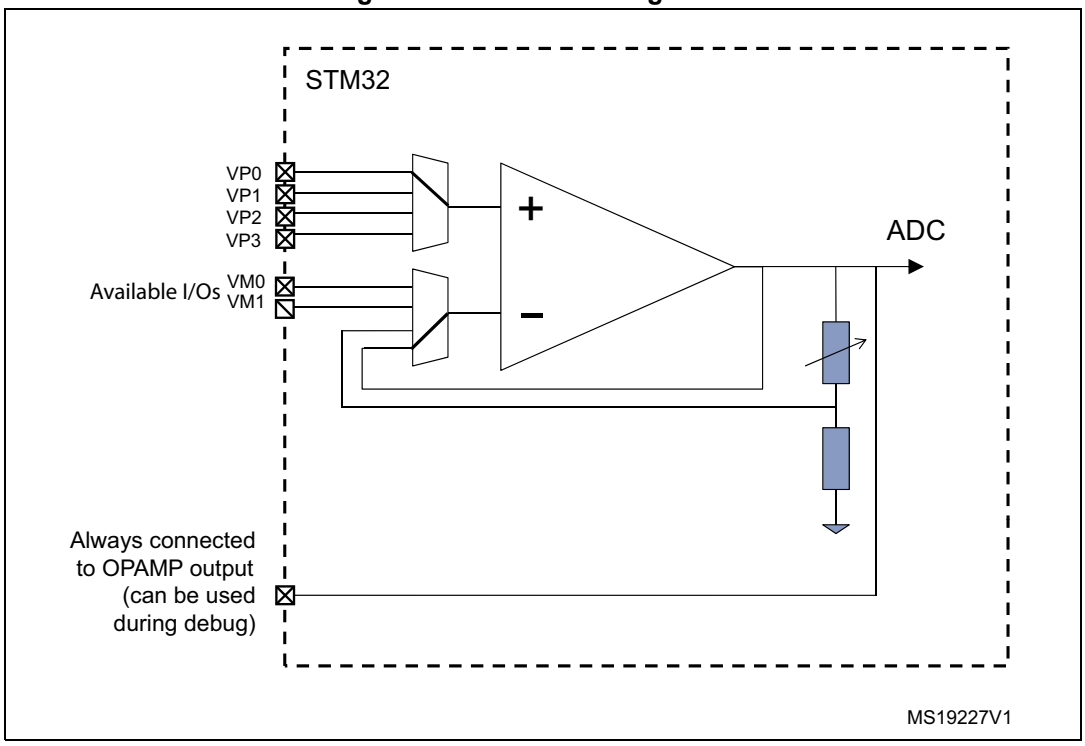


1. This figure gives an example in an inverting configuration. Any other option is possible, including comparator mode.

Follower configuration mode

The amplifier can be configured as a follower, by setting the VM_SEL bits to 11 in the OPAMPx_CR register. This allows you for instance to buffer signals with a relatively high impedance. In this case, the inverting inputs are free and the corresponding ports can be used as regular I/Os.

Figure 97. Follower configuration



1. This figure gives an example in an inverting configuration. Any other option is possible, including comparator mode.

Programmable Gain Amplifier mode

The Programmable Gain Amplifier (PGA) mode is enabled by writing the VM_SEL bits to 10 in the OPAMPx_CR register. The gain is set using the PGA_GAIN bits which must be set to 0x00..0x11 for gains ranging from 2 to 16.

In this case, the inverting inputs are internally connected to the central point of a built-in gain setting resistive network. *Figure 98: PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used* shows the internal connection in this mode.

An alternative option in PGA mode allows you to route the central point of the resistive network on one of the I/Os connected to the non-inverting input. This is enabled using the PGA_GAIN bits in OPAMPx_CR register:

- 10xx values are setting the gain and connect the central point to one of the two available inputs
- 11xx values are setting the gain and connect the central point to the second available input

This feature can be used for instance to add a low-pass filter to PGA, as shown in *Figure 99: PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering*. Please note that the cut-off frequency is changed if the gain is modified (refer to the electrical characteristics section of the datasheet for details on resistive network elements.

Figure 98. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used

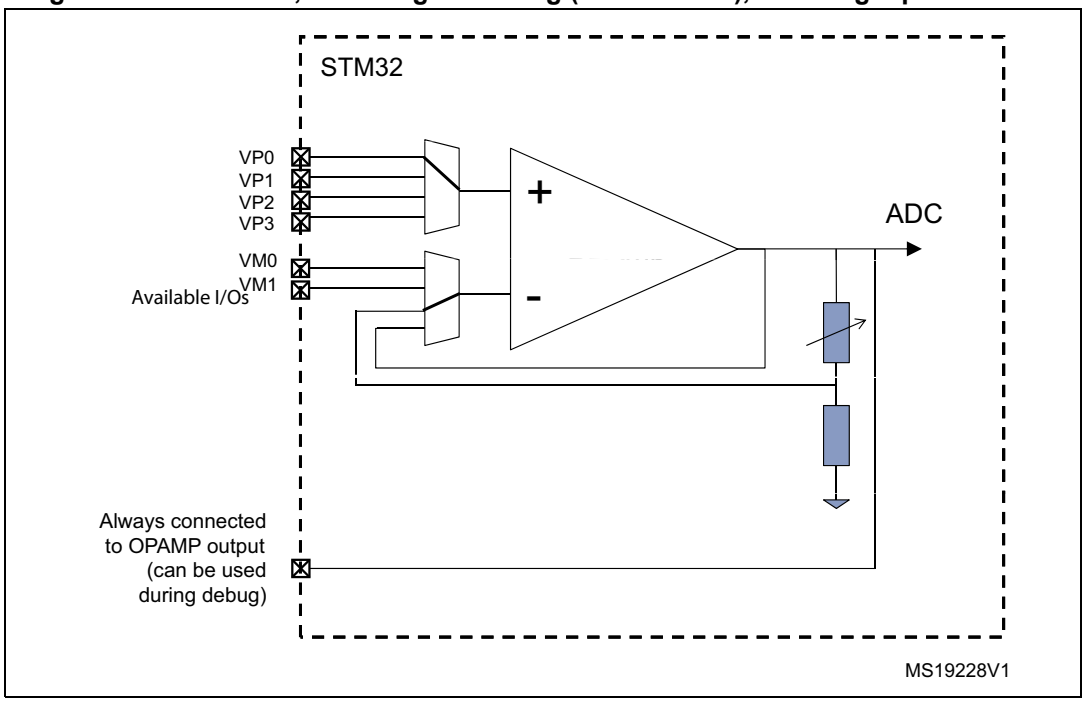
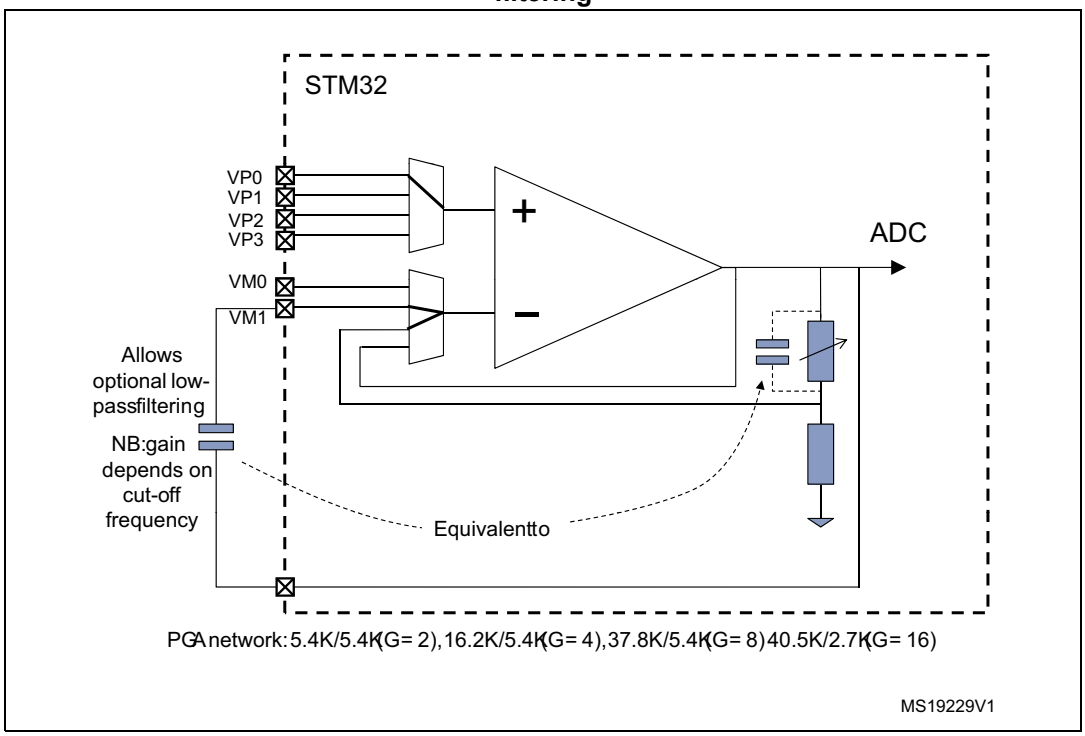


Figure 99. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



16.4 OPAMP registers

16.4.1 OPAMP2 control register (OPAMP2_CSR)

Address offset: 0x3C

Reset value: 0xXXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUT CAL	TSTR EF	TRIMOFFSETN					TRIMOFFSETP					USER_ TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CAL ON	VPS_SEL		VMS_ SEL	TCM_ EN	VM_SEL		Res.	VP_SEL		FORCE _VP	OPAMP 2EN
rw		rw		rw	rw		rw	rw	rw			rw		rw	rw

Bit 31 **LOCK**: OPAMP 2 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.
This bit is used to configure the OPAMP2_CSR register as read-only.
0: OPAMP2_CSR is read-write.
1: OPAMP2_CSR is read-only.

Bit 30 **OUTCAL:**

OPAMP output status flag, when the OPAMP is used as comparator during calibration.
0: Non-inverting < inverting
1: Non-inverting > inverting.

Bit 29 **TSTREF:**

This bit is set and cleared by software. It is used to output the internal reference voltage ($V_{REFOPAMP2}$).
0: $V_{REFOPAMP2}$ is output.
1: $V_{REFOPAMP2}$ is not output.

Bits 28:24 **TRIMOFFSETN:** Offset trimming value (NMOS)

Bits 23:19 **TRIMOFFSETP:** Offset trimming value (PMOS)

Bit 18 **USER_TRIM:** User trimming enable.

This bit is used to configure the OPAMP offset.
0: User trimming disabled.
1: User trimming enabled.

Bits 17:14 **PGA_GAIN:** gain in PGA mode

0X00 = Non-inverting gain = 2
0X01 = Non-inverting gain = 4
0X10 = Non-inverting gain = 8
0X11 = Non-inverting gain = 16
1000 = Non-inverting gain = 2 - Internal feedback connected to VM0
1001 = Non-inverting gain = 4 - Internal feedback connected to VM0
1010 = Non-inverting gain = 8 - Internal feedback connected to VM0
1011 = Non-inverting gain = 16 - Internal feedback connected to VM0
1100 = Non-inverting gain = 2 - Internal feedback connected to VM1
1101 = Non-inverting gain = 4 - Internal feedback connected to VM1
1110 = Non-inverting gain = 8 - Internal feedback connected to VM1
1111 = Non-inverting gain = 16 - Internal feedback connected to VM1

Bits 13:12 **CALSEL:** Calibration selection

This bit is set and cleared by software. It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00 = $V_{REFOPAMP}$ = 3.3% V_{DDA}

01 = $V_{REFOPAMP}$ = 10% V_{DDA}

10 = $V_{REFOPAMP}$ = 50% V_{DDA}

11 = $V_{REFOPAMP}$ = 90% V_{DDA}

Bit 11 **CALON:** Calibration mode enable

This bit is set and cleared by software. It is used to enable the calibration mode connecting VM and VP to the OPAMP internal reference voltage.

0: calibration mode disabled.

1: calibration mode enabled.

Bits 10:9 **VPS_SEL:** OPAMP2 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP2 non inverting input when TCM_EN = 1.

00: Reserved

01: PB14 used as OPAMP2 non inverting input

10: PB0 used as OPAMP2 non inverting input

11: PA7 used as OPAMP2 non inverting input

Bit 8 **VMS_SEL:** OPAMP2 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP2 inverting input when TCM_EN = 1.

0: PC5 (VM0) used as OPAMP2 inverting input

1: PA5 (VM1) used as OPAMP2 inverting input

Bit 7 **TCM_EN:** Timer controlled Mux mode enable.

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs.

Bits 6:5 **VM_SEL:** OPAMP2 inverting input selection.

Theses bits are set and cleared by software. They are used to select the OPAMP2 inverting input.

00: PC5 (VM0) used as OPAMP2 inverting input

01: PA5 (VM1) used as OPAMP2 inverting input

10: Resistor feedback output (PGA mode)

11: follower mode

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL:** OPAMP2 non inverting input selection.
Theses bits are set/reset by software. They are used to select the OPAMP2 non inverting input.
00: Reserved
01: PB14 used as OPAMP2 non inverting input
10: PB0 used as OPAMP2 non inverting input
11: PA7 used as OPAMP2 non inverting input

Bit 1 **FORCE_VP:**
This bit forces a calibration reference voltage on non-inverting input and disables external connections.
0: Normal operating mode. Non-inverting input connected to inputs.
1: Calibration mode. Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAMP2EN:** OPAMP2 enable.
This bit is set and cleared by software. It is used to select the OPAMP2.
0: OPAMP2 is disabled.
1: OPAMP2 is enabled.

16.4.2 OPAMP register map

The following table summarizes the OPAMP registers.

Table 59. OPAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C	**OPAMP2_CSR**	LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN				CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res	VP_SEL		FORCE_VP	OPAMP2EN
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

17 Touch sensing controller (TSC)

17.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

17.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 18 capacitive sensing channels
- Up to 6 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

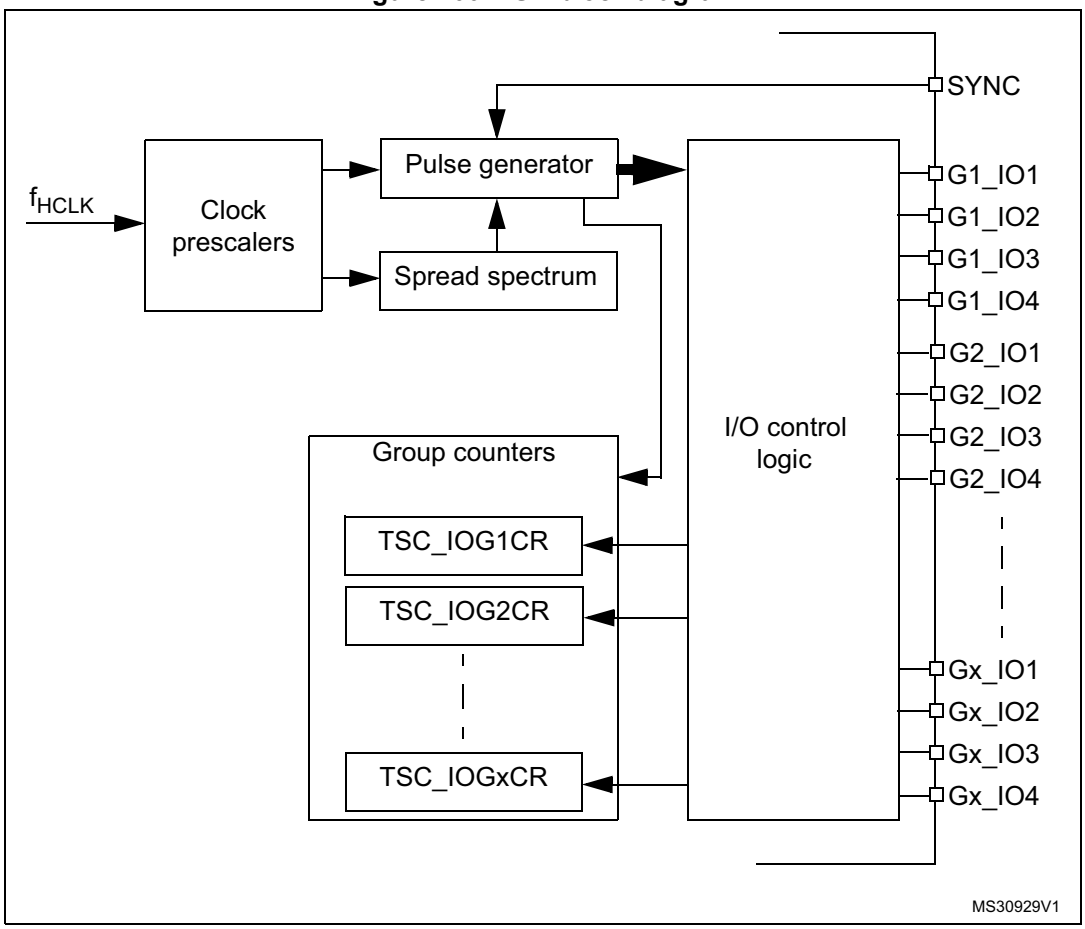
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.

17.3 TSC functional description

17.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in *Figure 100: TSC block diagram*.

Figure 100. TSC block diagram



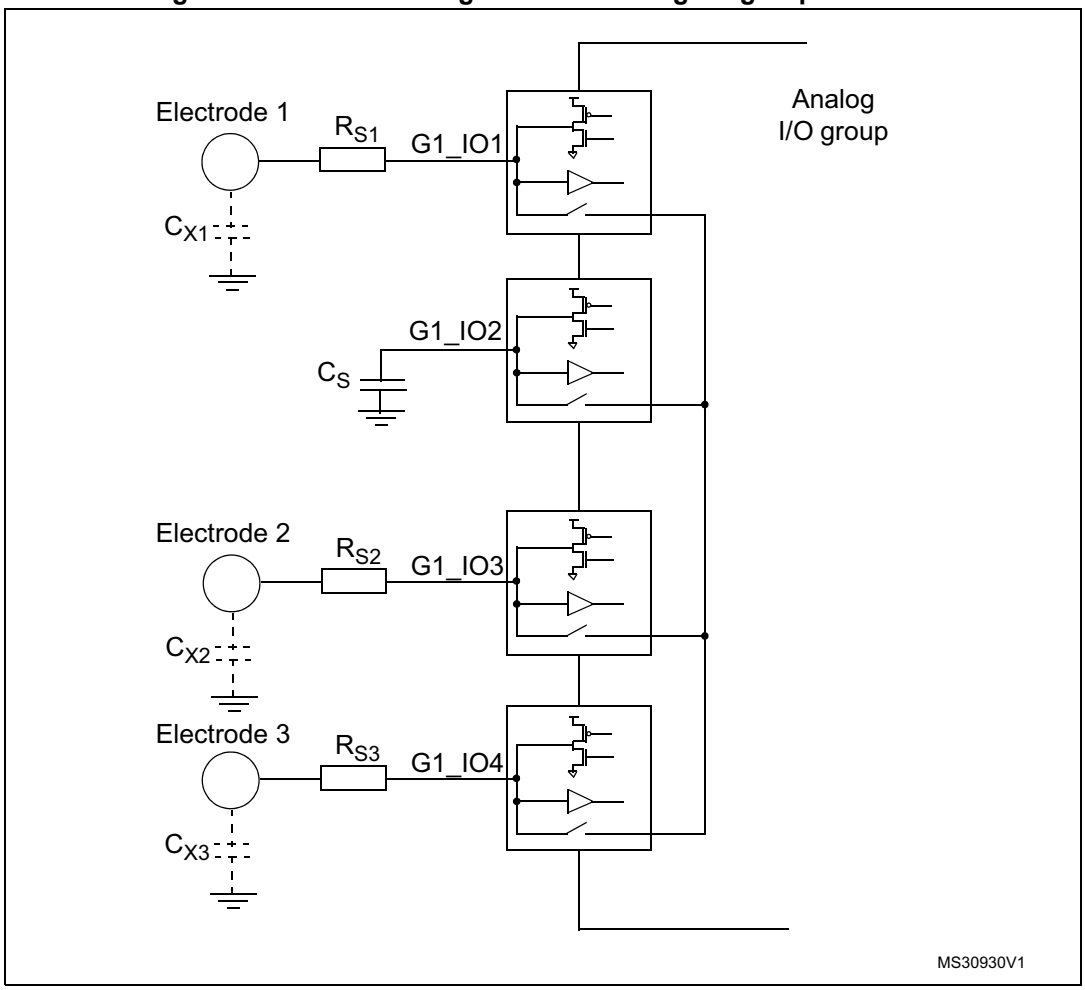
17.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group which is composed of up to four GPIOs (see *Figure 101*). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S. Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 101. Surface charge transfer analog I/O group structure



Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

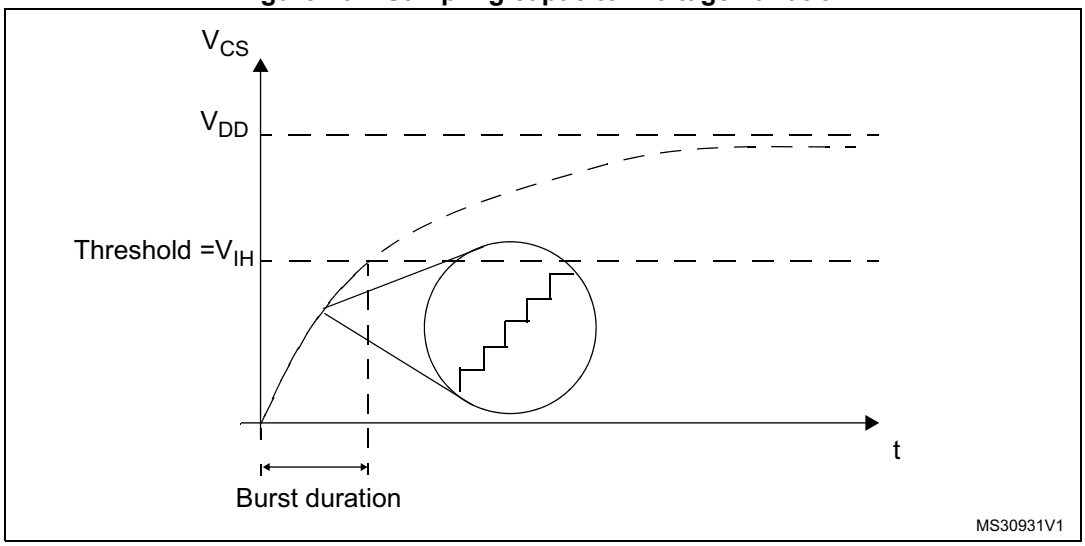
The *Table 60* details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.

Table 60. Acquisition sequence summary

State	G1_IO1 (channel)	G1_IO2 (sampling)	G1_IO3 (channel)	G1_IO4 (channel)	State description
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all C_X and C_S
#2	Input floating				Dead time
#3	Output push-pull high	Input floating			Charge C_{X1}
#4	Input floating				Dead time
#5	Input floating with analog switch closed		Input floating		Charge transfer from C_{X1} to C_S
#6	Input floating				Dead time
#7	Input floating				Measure C_S voltage

The voltage variation over the time on the sampling capacitor C_S is detailed below:

Figure 102. Sampling capacitor voltage variation



17.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

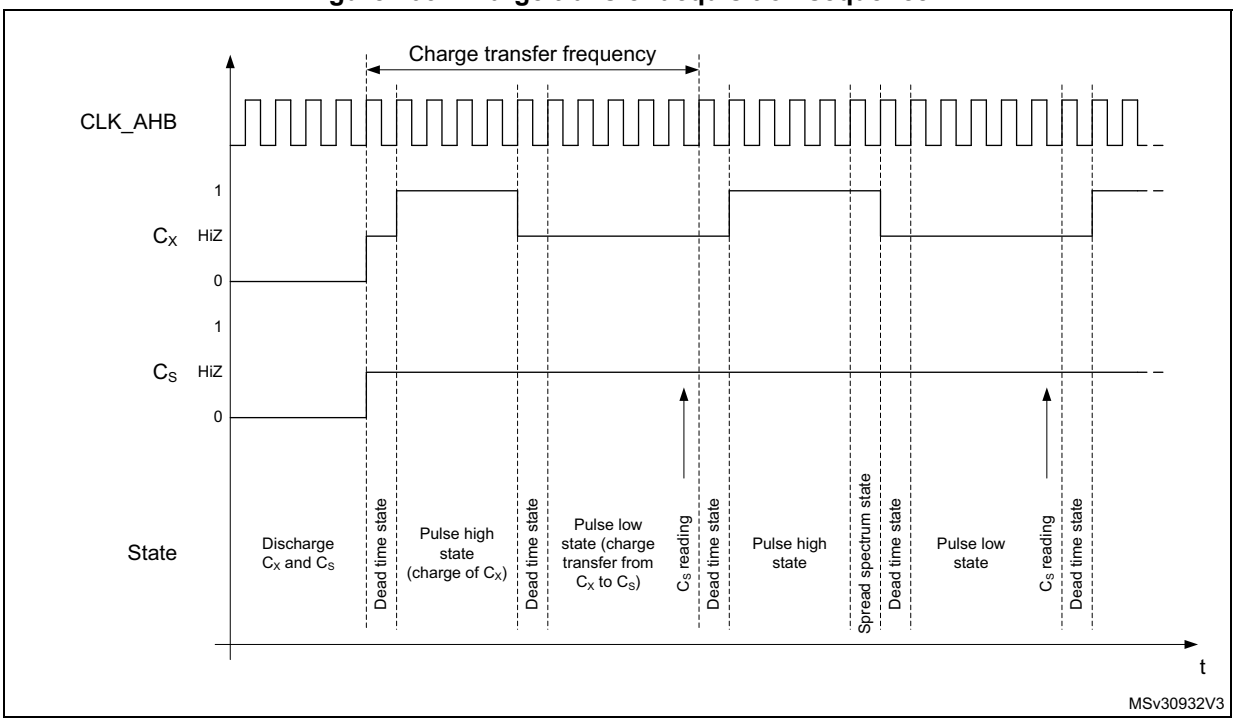
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to *Section 8: Reset and clock control (RCC)*.

17.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in *Figure 103*.

Figure 103. Charge transfer acquisition sequence



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_X) and the pulse low state (transfer of charge from C_X to C_S) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 µs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_X is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 2 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across C_S has reached the given threshold, is performed at the end of the pulse low state.

Note: The following TSC control register configurations are forbidden:

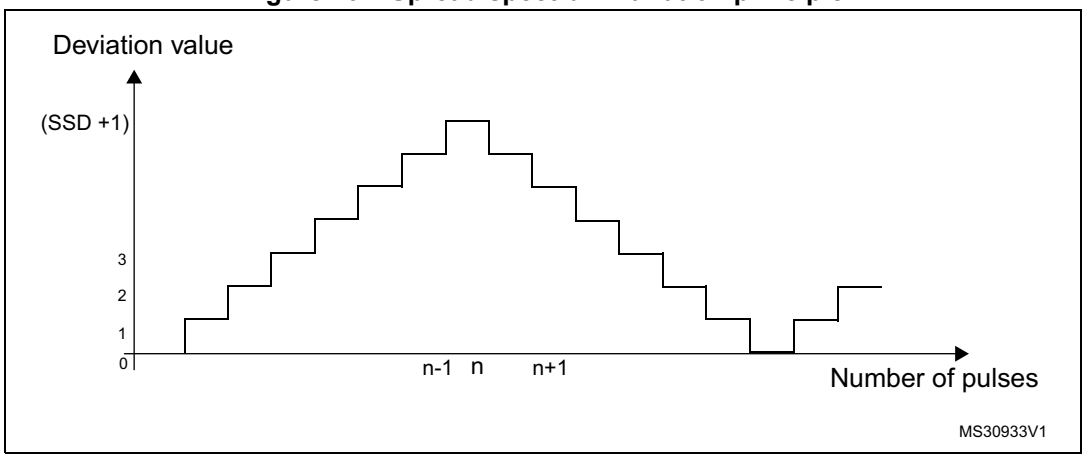
- *bits PGPSC are set to '000' and bits CTPL are set to '0000'*
- *bits PGPSC are set to '000' and bits CTPL are set to '0001'*
- *bits PGPSC are set to '001' and bits CTPL are set to '0000'*

17.3.5 Spread spectrum feature

The spread spectrum feature allows to generate a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 µs), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 104. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 61. Spread spectrum deviation versus AHB clock frequency

f_{HCLK}	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

17.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

17.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

Table 62. I/O state depending on its mode and IODEF bit value

IODEF bit	Acquisition status	Unused I/O mode	Channel I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	Ongoing	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	Ongoing	Input floating	-	-

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IOy bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

Note: During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

17.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAIC and MCEIC bits in the TSC_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

17.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller also allows to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_IOy bit in the TSC_IOHCR register.

17.4 TSC low-power modes

Table 63. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect TSC interrupts cause the device to exit Sleep mode.
Stop	TSC registers are frozen The TSC stops its operation until the Stop or Standby mode is exited.
Standby	

17.5 TSC interrupts

Table 64. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	Yes	No	No
Max count error	MCEIE	MCEIF	MCEIC	Yes	No	No

17.6 TSC registers

Refer to *Section 1.2 on page 43* of the reference manual for a list of abbreviations used in register descriptions.

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

17.6.1 TSC control register (TSC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of C_X).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from C_X to C_S).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 17.3.4: Charge transfer acquisition sequence for details.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: 1x t_{SSCLK}
0000001: 2x t_{SSCLK}
...
1111111: 128x t_{SSCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

Note: This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0: f_{HCLK}

1: f_{HCLK} /2

Note: This bit must not be modified when an acquisition is ongoing.

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software.They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000: f_{HCLK}

001: f_{HCLK} /2

010: f_{HCLK} /4

011: f_{HCLK} /8

100: f_{HCLK} /16

101: f_{HCLK} /32

110: f_{HCLK} /64

111: f_{HCLK} /128

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 17.3.4: Charge transfer acquisition sequence for details.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255

001: 511

010: 1023

011: 2047

100: 4095

101: 8191

110: 16383

111: reserved

Note: These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low

1: I/Os are in input floating

Note: This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only

1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

Note: This bit must not be modified when an acquisition is ongoing.

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSCE**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

Note: When the touch sensing controller is disabled, TSC registers settings have no effect.

17.6.2 TSC interrupt enable register (TSC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

17.6.3 TSC interrupt clear register (TSC_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a ‘0’ has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a ‘0’ has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC_ISR register

17.6.4 TSC interrupt status register (TSC_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAIC of the TSC_ICR register.

0: Acquisition is ongoing or not started

1: Acquisition is complete

17.6.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	Res. G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

17.6.6 TSC I/O analog switch control register (TSC_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **Gx_IOy**: Gx_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.

0: Gx_IOy analog switch disabled (opened)

1: Gx_IOy analog switch enabled (closed)

Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).

17.6.7 TSC I/O sampling control register (TSC_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **Gx_IOy**: Gx_IOy sampling mode

These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx_IOy unused

1: Gx_IOy used as sampling capacitor

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

17.6.8 TSC I/O channel control register (TSC_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **Gx_IOy**: Gx_IOy channel mode

These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.
0: Gx_IOy unused
1: Gx_IOy used as channel

Note: These bits must not be modified when an acquisition is ongoing.
During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

17.6.9 TSC I/O group control status register (TSC_IOGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6S	G5S	G4S	G3S	G2S	G1S
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6E	G5E	G4E	G3E	G2E	G1E
										rw	rw	rw	rw	rw	rw

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

Bits 21:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.
0: Acquisition on analog I/O group x is ongoing or not started
1: Acquisition on analog I/O group x is complete

Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.
0: Acquisition on analog I/O group x disabled
1: Acquisition on analog I/O group x enabled

17.6.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (x = 1..6)

Reset value: 0x0000 0000

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

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

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

Bits 13:0 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across C_S has reached the threshold).

17.6.11 TSC register map

Table 65. TSC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**TSC_CR**	CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE	SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV [2:0]			IODEF	SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x0004	**TSC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
	Reset value																															0	0
0x0008	**TSC_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
	Reset value																															0	0
0x000C	**TSC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
	Reset value																															0	0
0x0010	**TSC_IOHCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0014	Reserved																																
0x0018	**TSC_IOASCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	Reserved																																
0x0020	**TSC_IOSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	Reserved																																
0x0028	**TSC_IOCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																
0x0030	**TSC_IOGCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value											0	0	0	0	0	0											0	0	0	0	0	0
0x0034	**TSC_IOG1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	**TSC_IOG2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 65. TSC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x003C	**TSC_IOG3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	**TSC_IOG4CR**	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**TSC_IOG5CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**TSC_IOG6CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

18 Advanced-control timer (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

18.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

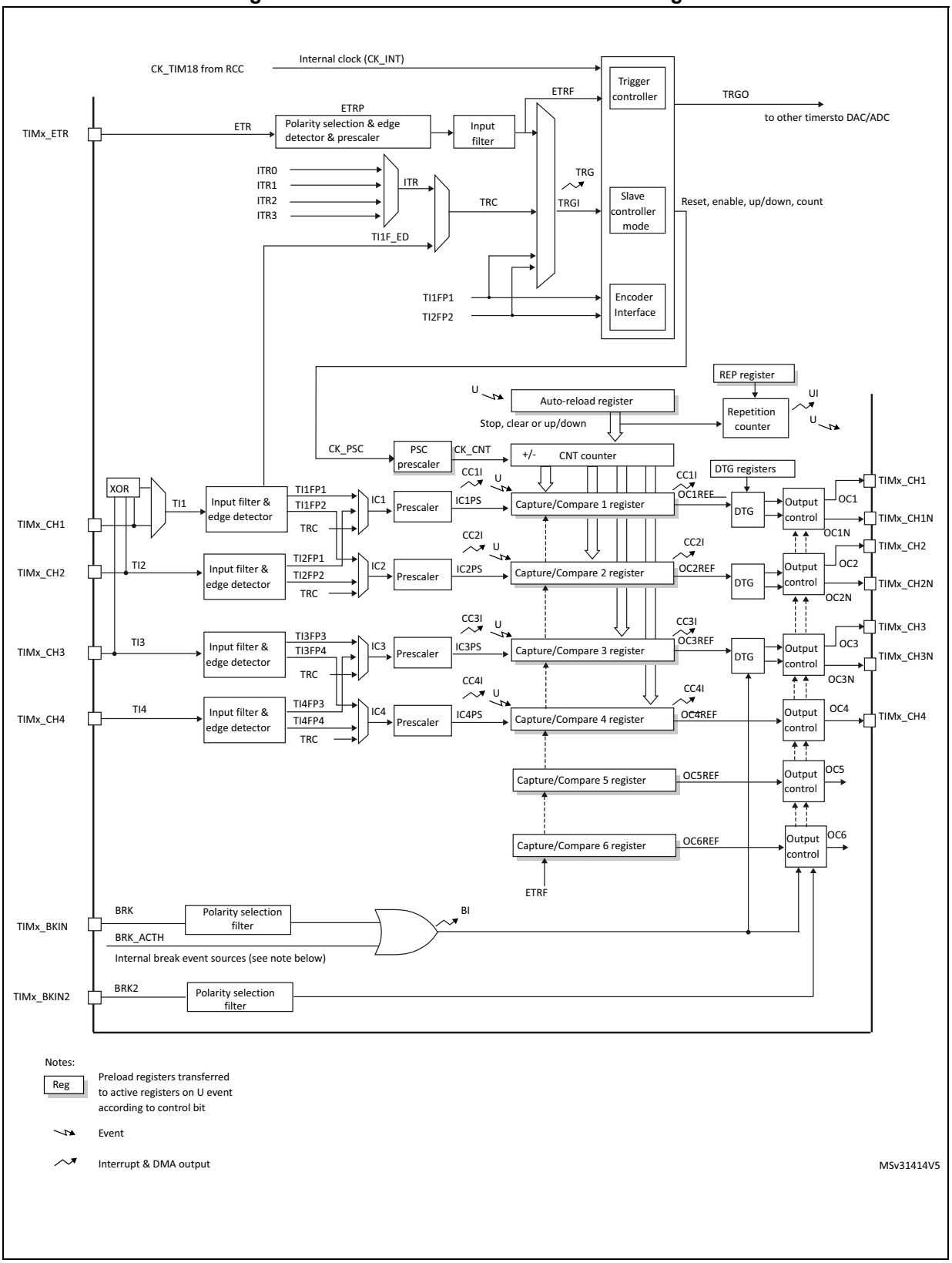
The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 18.3.25: Timer synchronization*.

18.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 105. Advanced-control timer block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 8.2.7: Clock security system (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M4 LOCKUP (Hardfault) output.
   - COMPx output, x = 1,2,3,5 and 6.

18.3 TIM1 functional description

18.3.1 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 106 and *Figure 107* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 106. Counter timing diagram with prescaler division change from 1 to 2

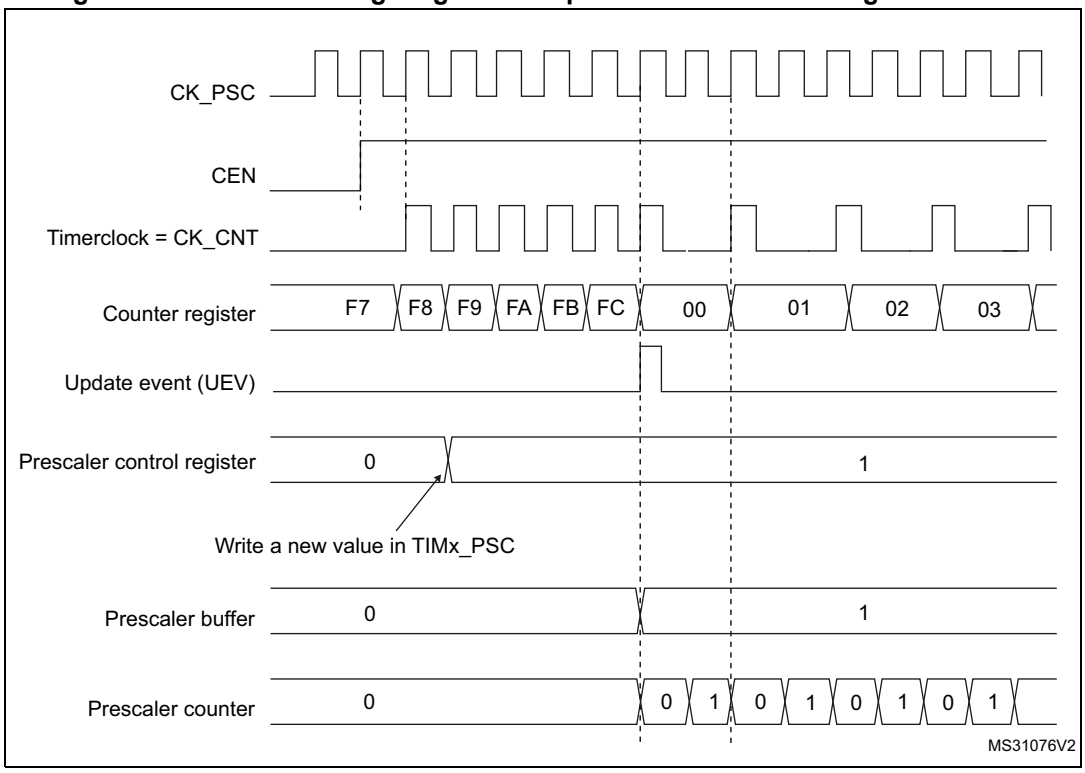
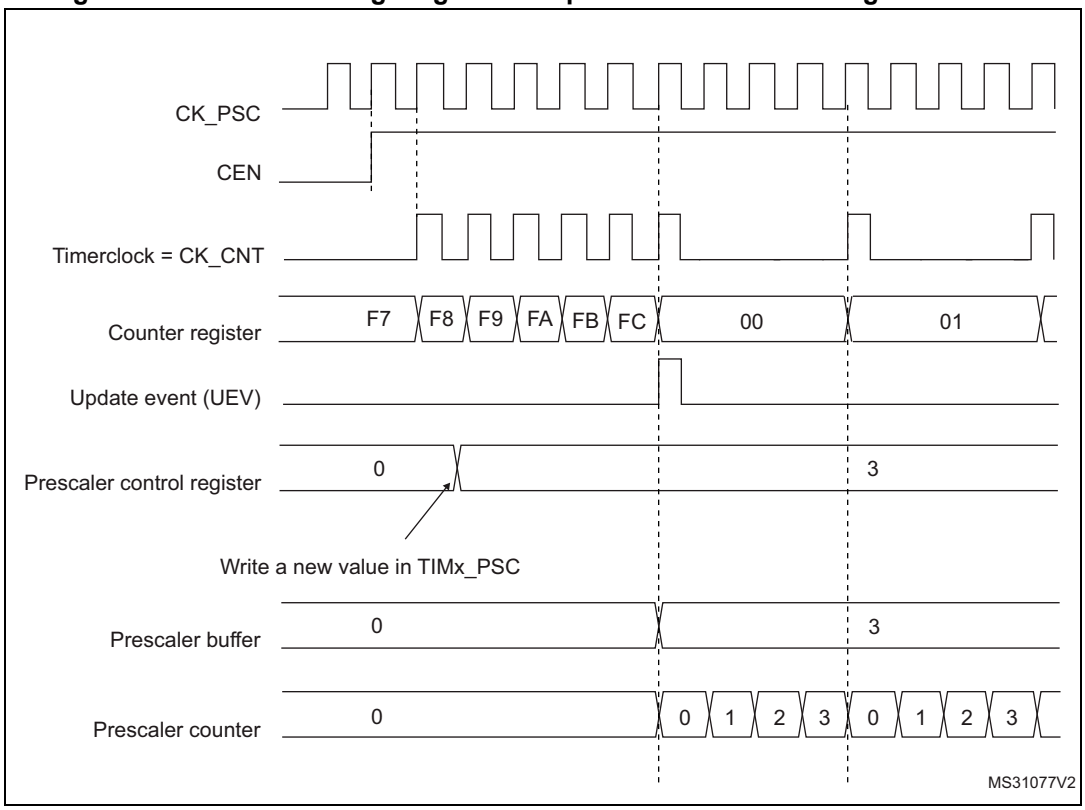


Figure 107. Counter timing diagram with prescaler division change from 1 to 4



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.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 108. Counter timing diagram, internal clock divided by 1

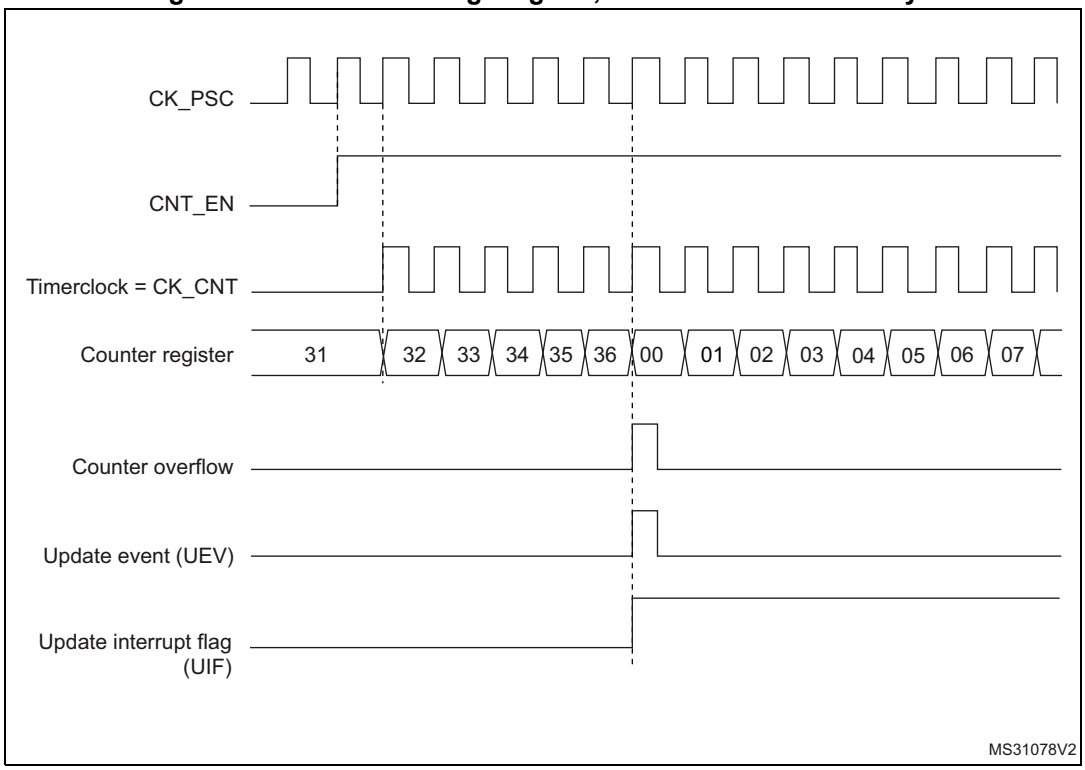


Figure 109. Counter timing diagram, internal clock divided by 2

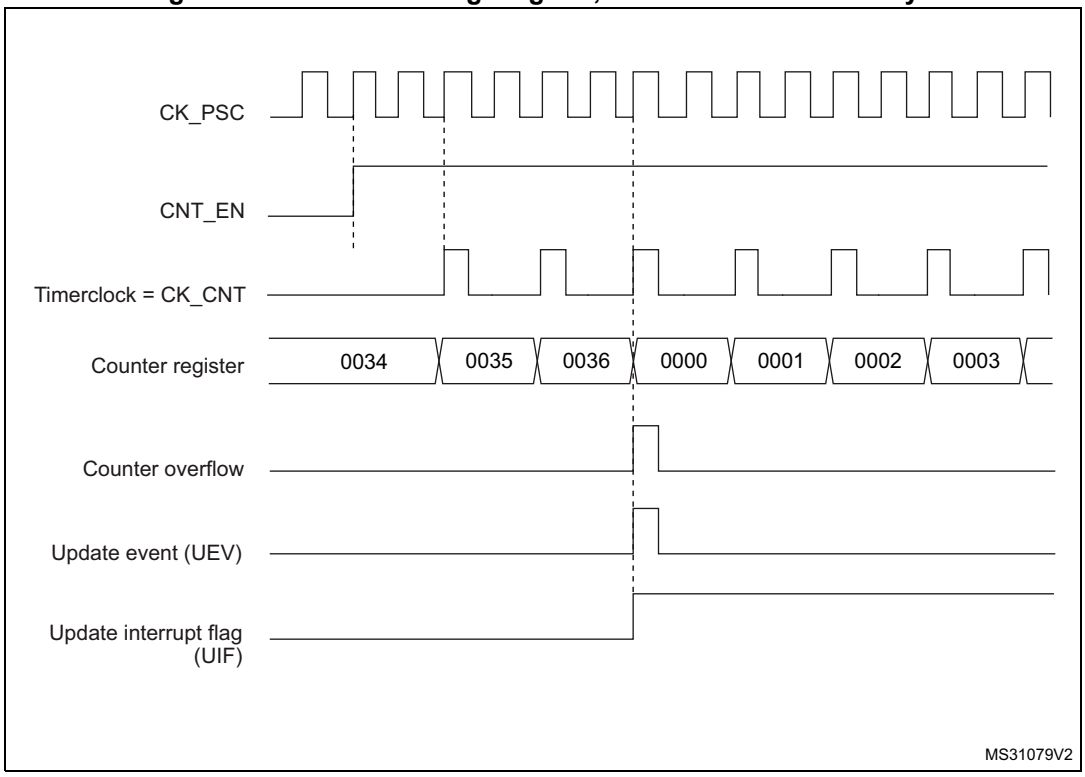


Figure 110. Counter timing diagram, internal clock divided by 4

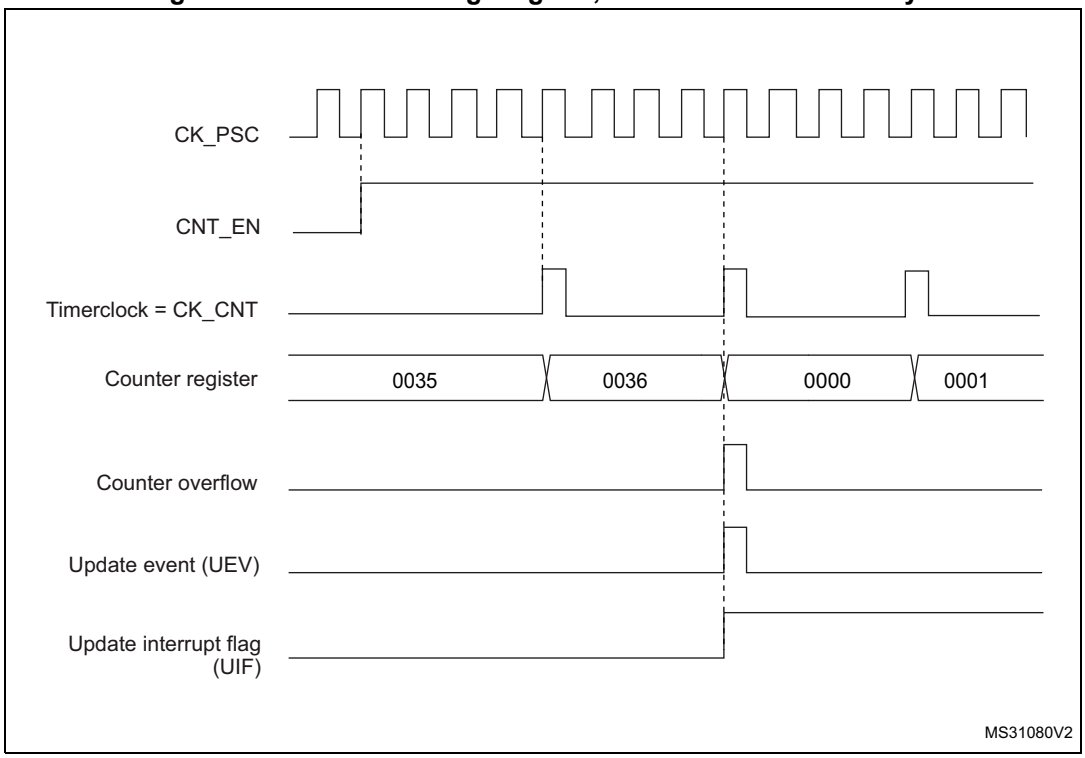


Figure 111. Counter timing diagram, internal clock divided by N

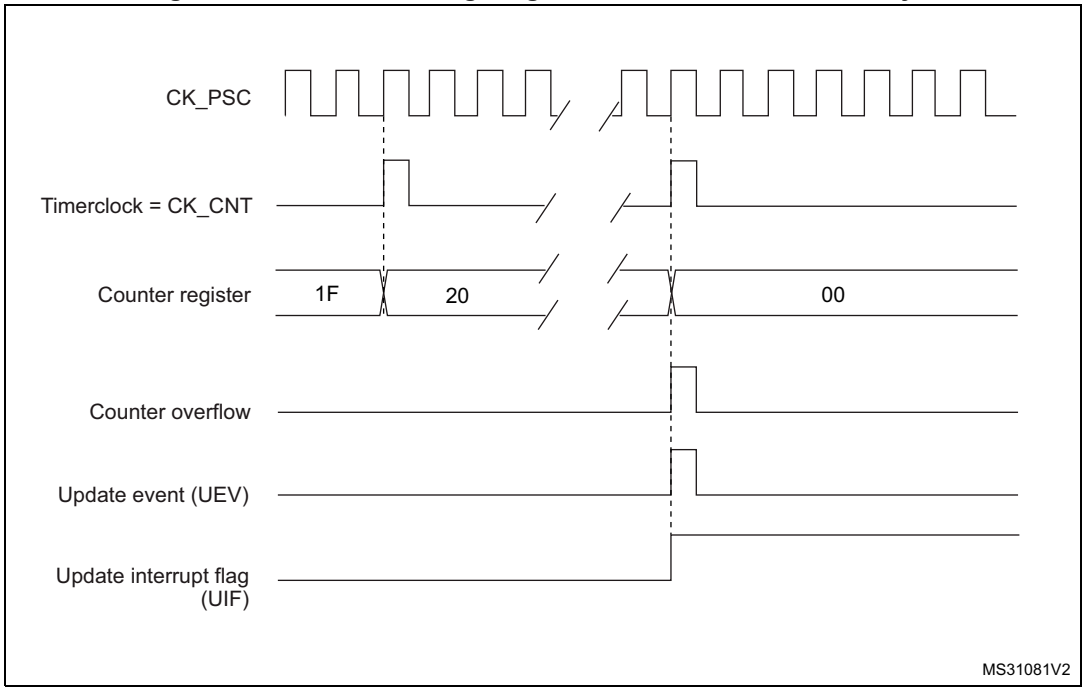


Figure 112. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

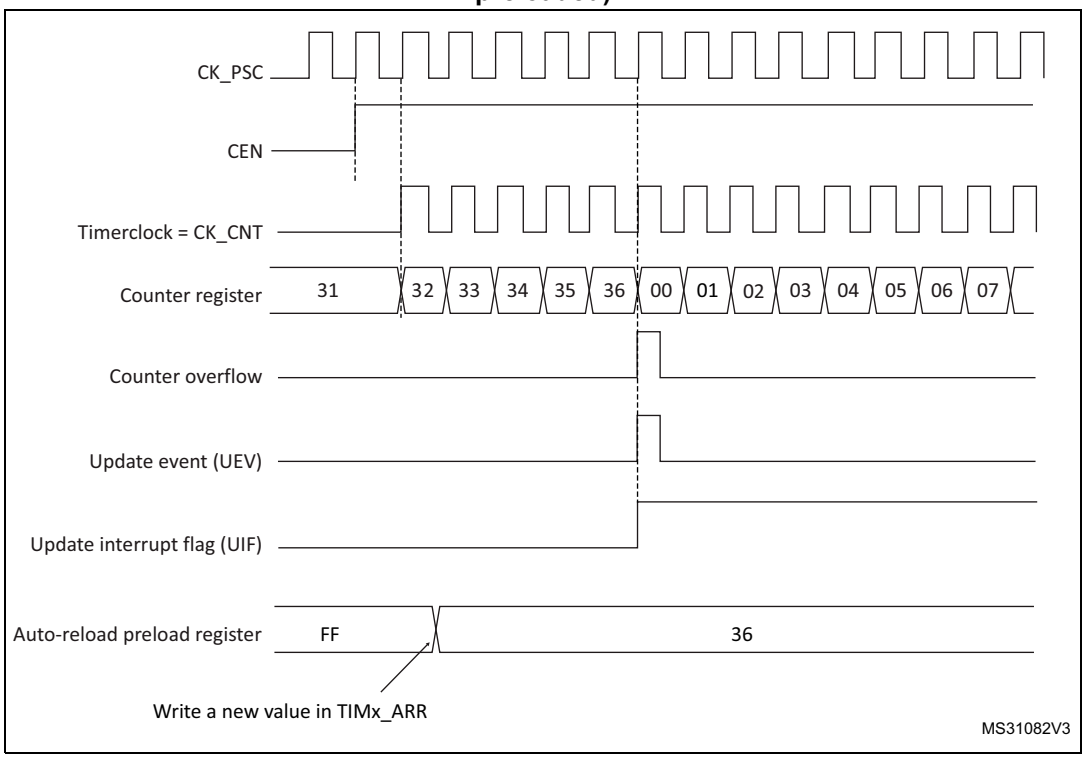
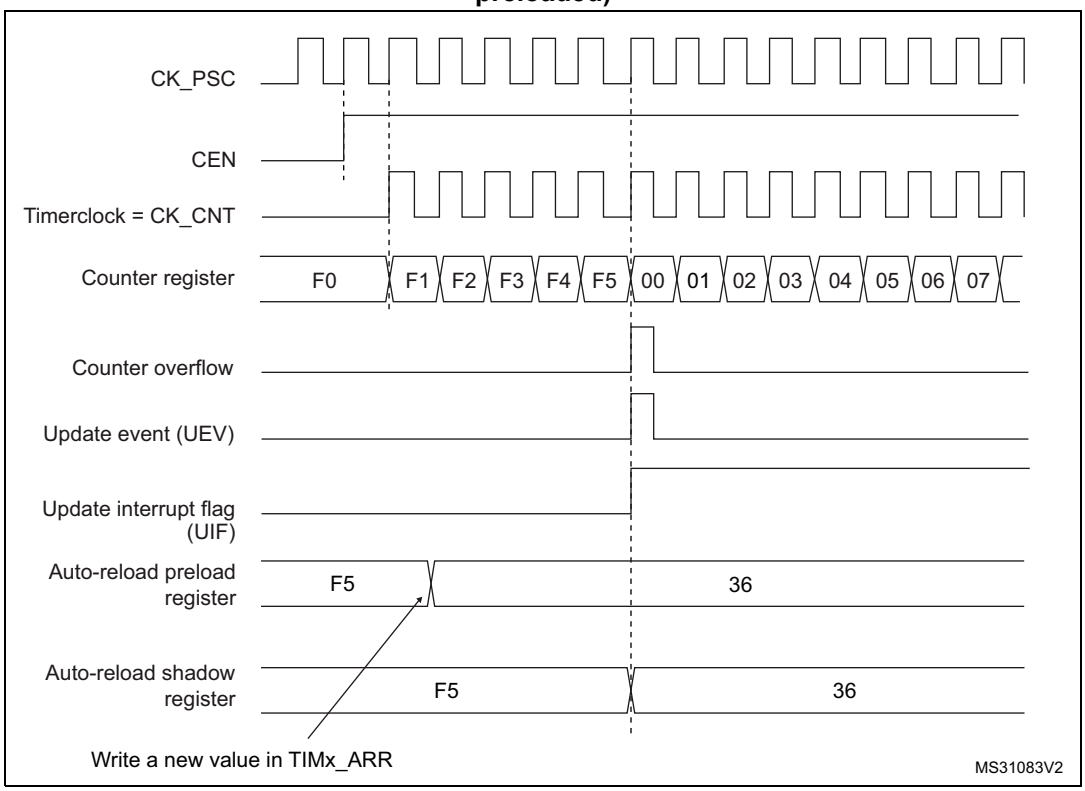


Figure 113. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 114. Counter timing diagram, internal clock divided by 1

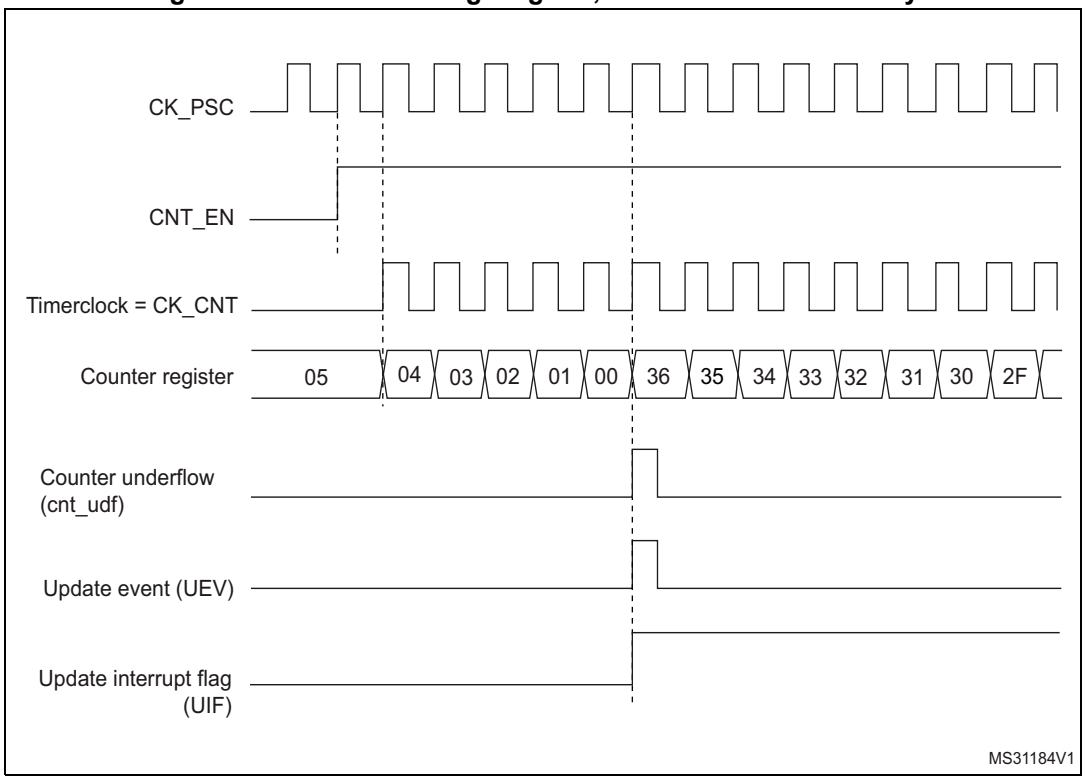


Figure 115. Counter timing diagram, internal clock divided by 2

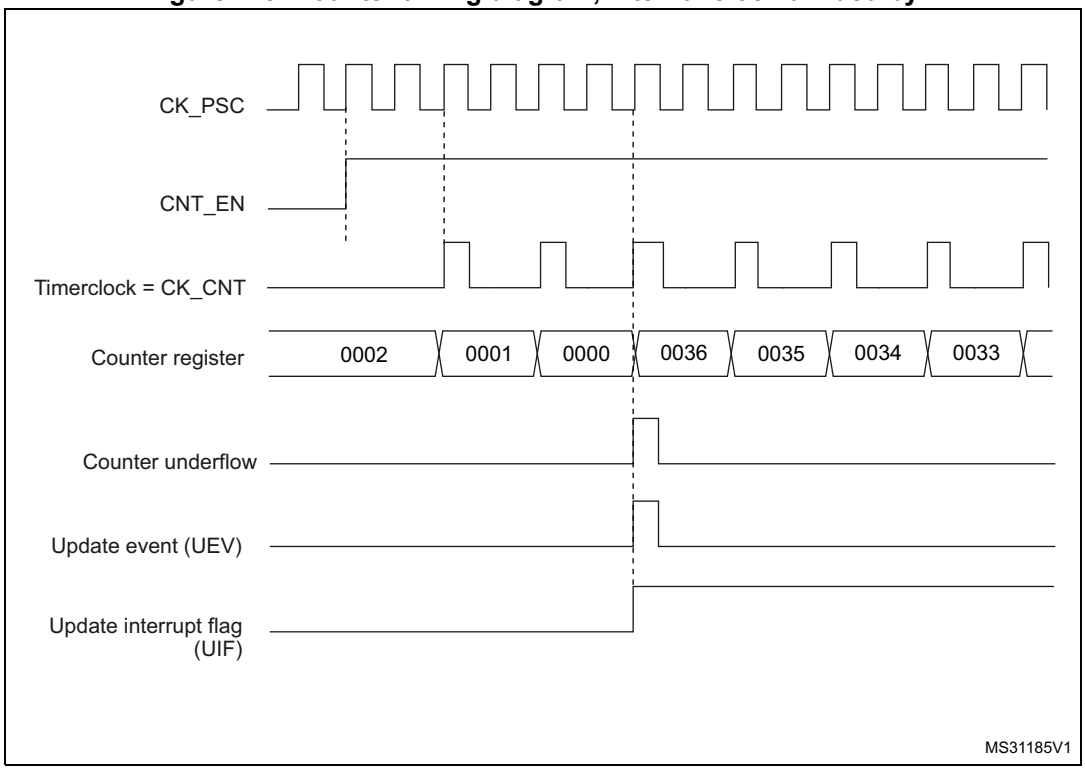


Figure 116. Counter timing diagram, internal clock divided by 4

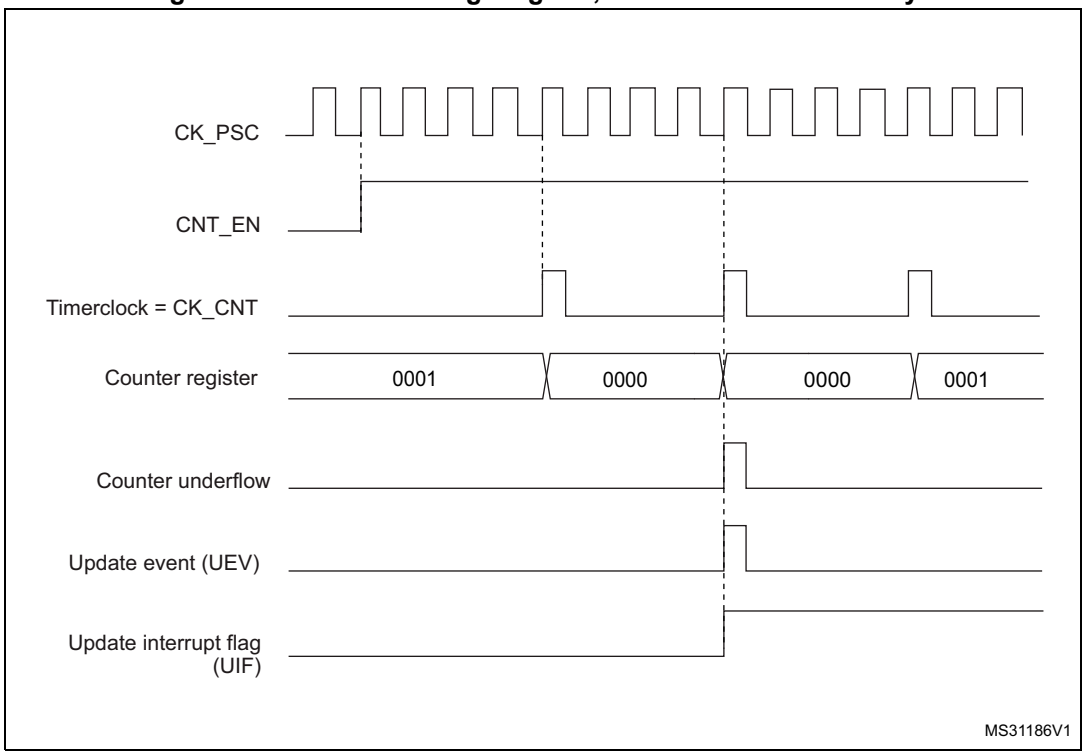


Figure 117. Counter timing diagram, internal clock divided by N

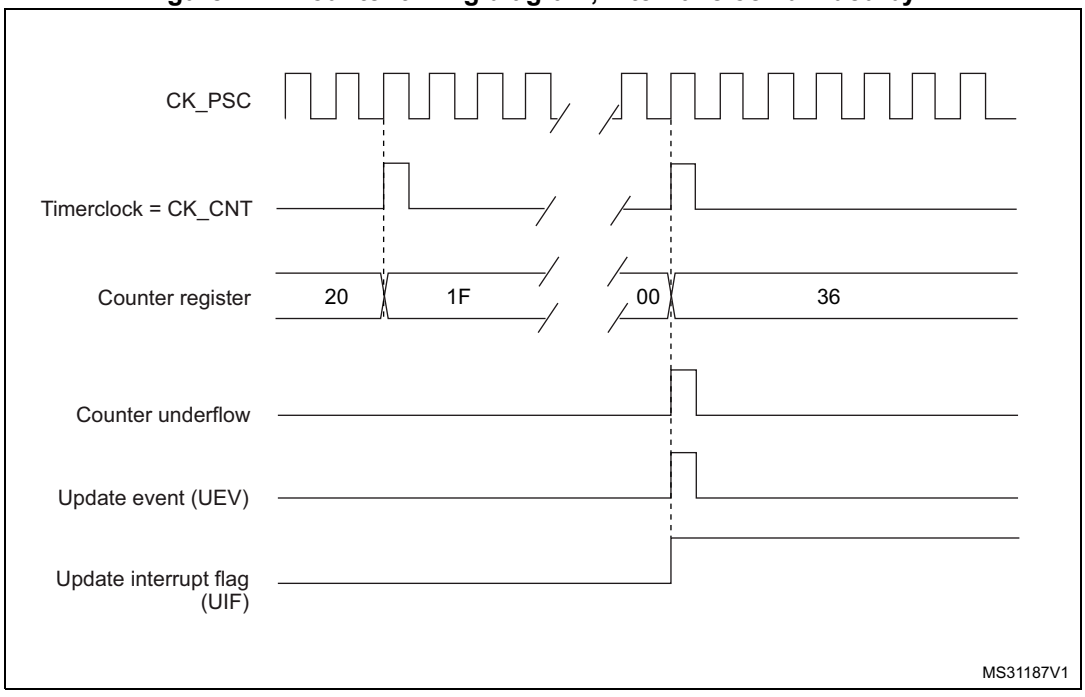
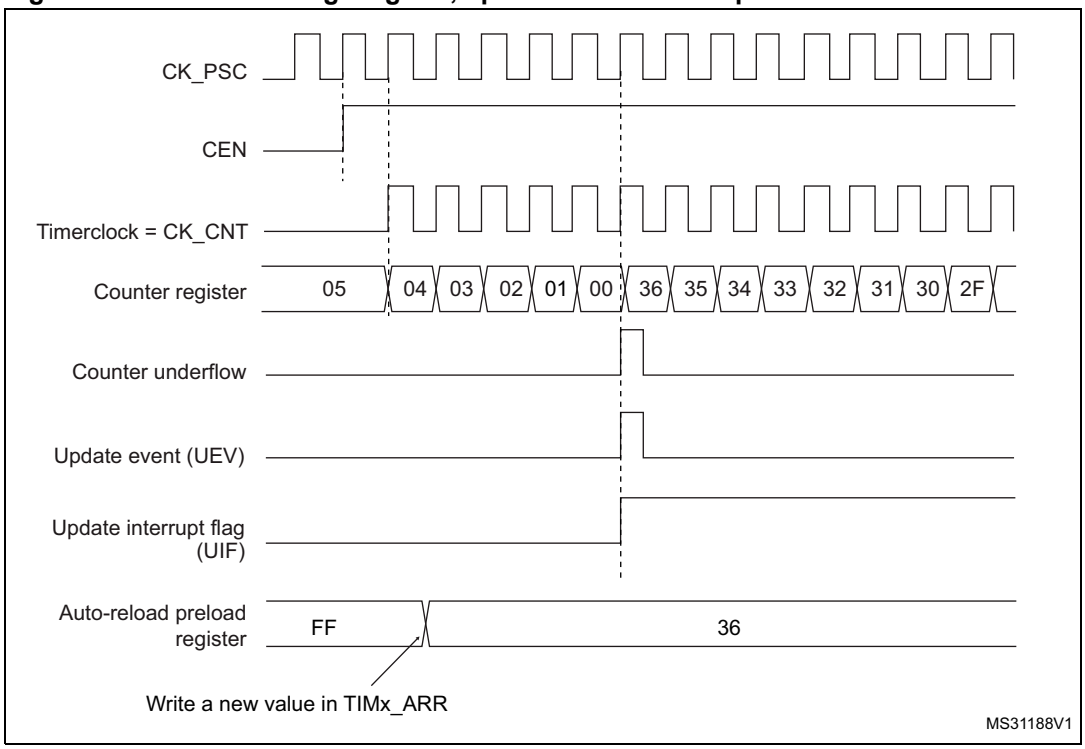


Figure 118. Counter timing diagram, update event when repetition counter is not used



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

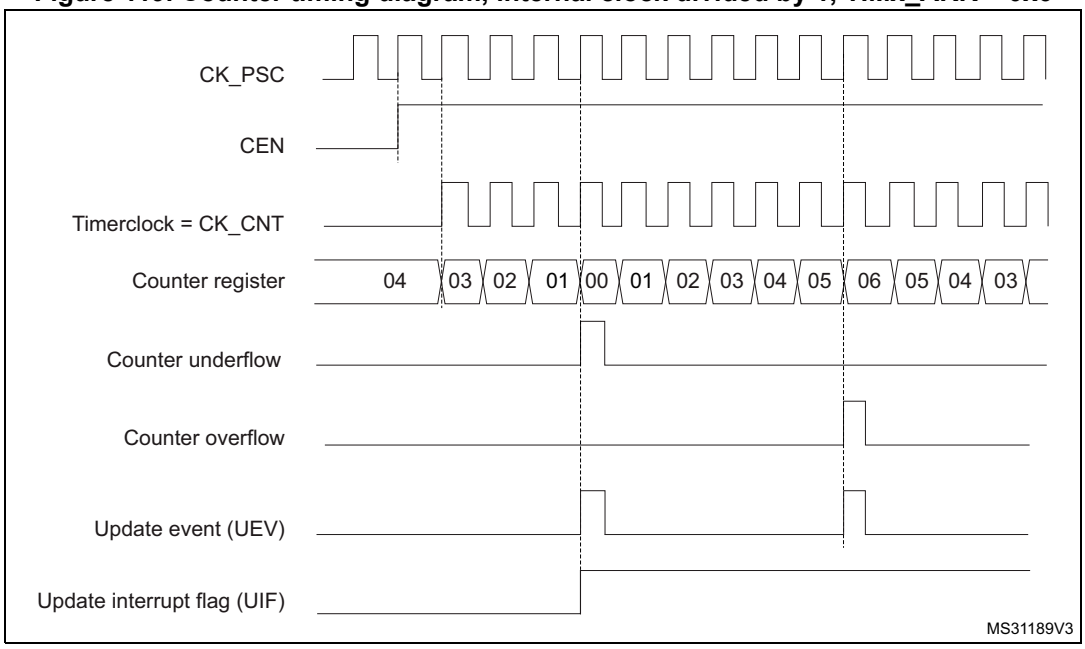
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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 119. 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: TIM1 registers*).

Figure 120. Counter timing diagram, internal clock divided by 2

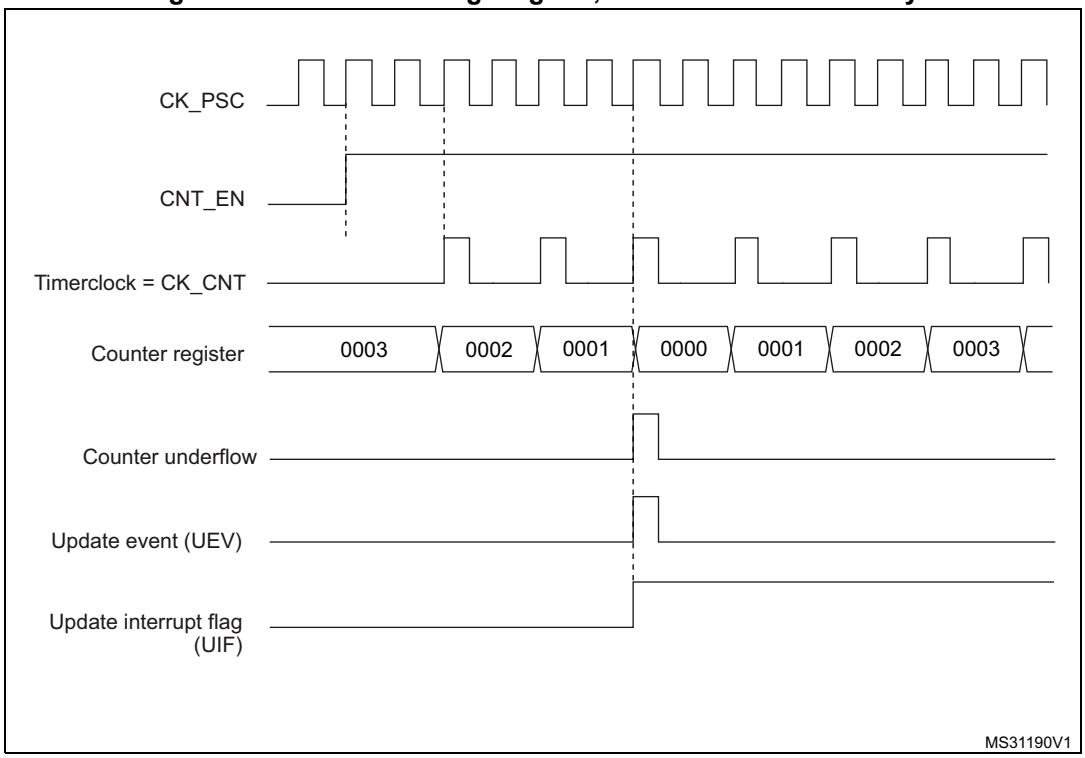


Figure 121. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

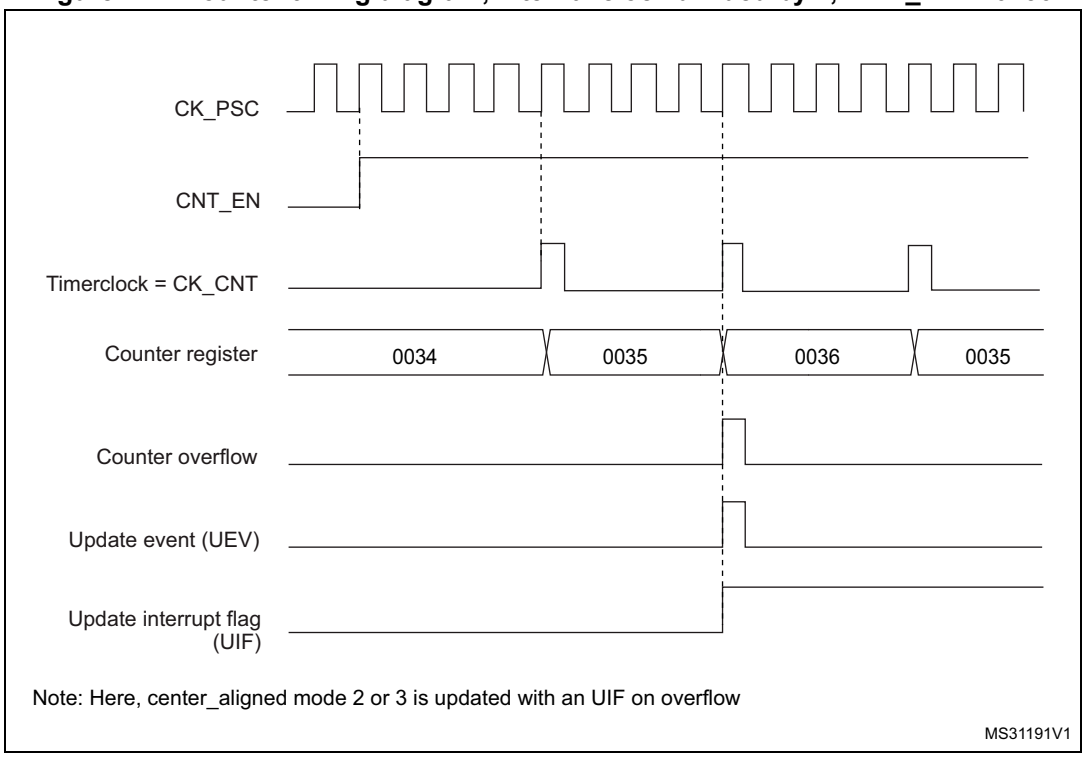


Figure 122. Counter timing diagram, internal clock divided by N

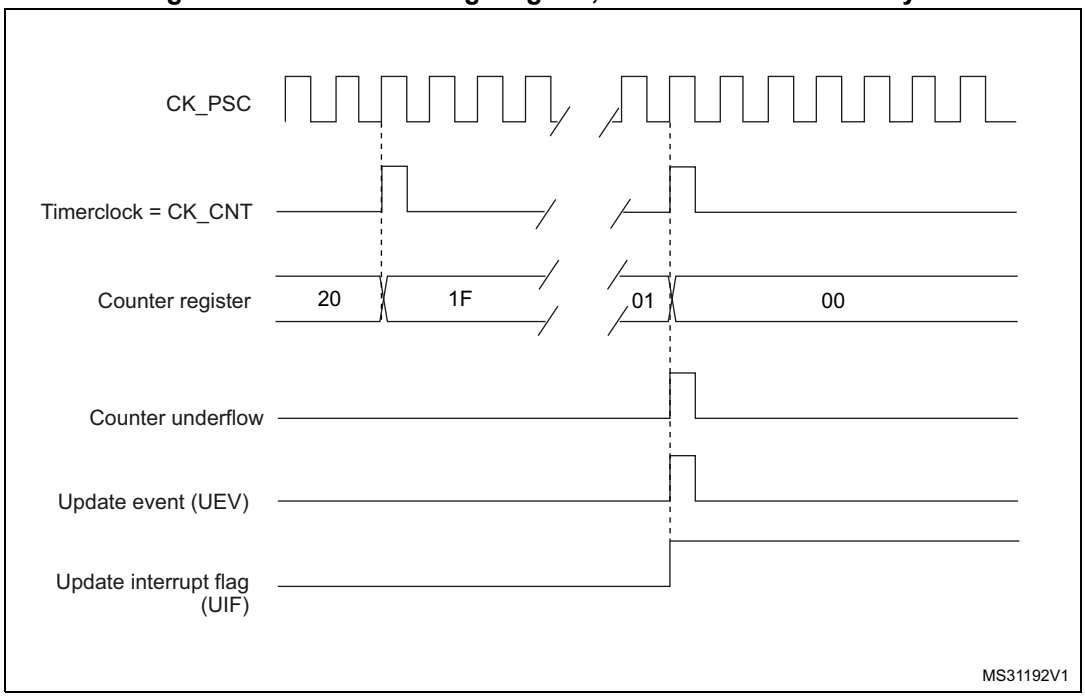


Figure 123. Counter timing diagram, update event with ARPE=1 (counter underflow)

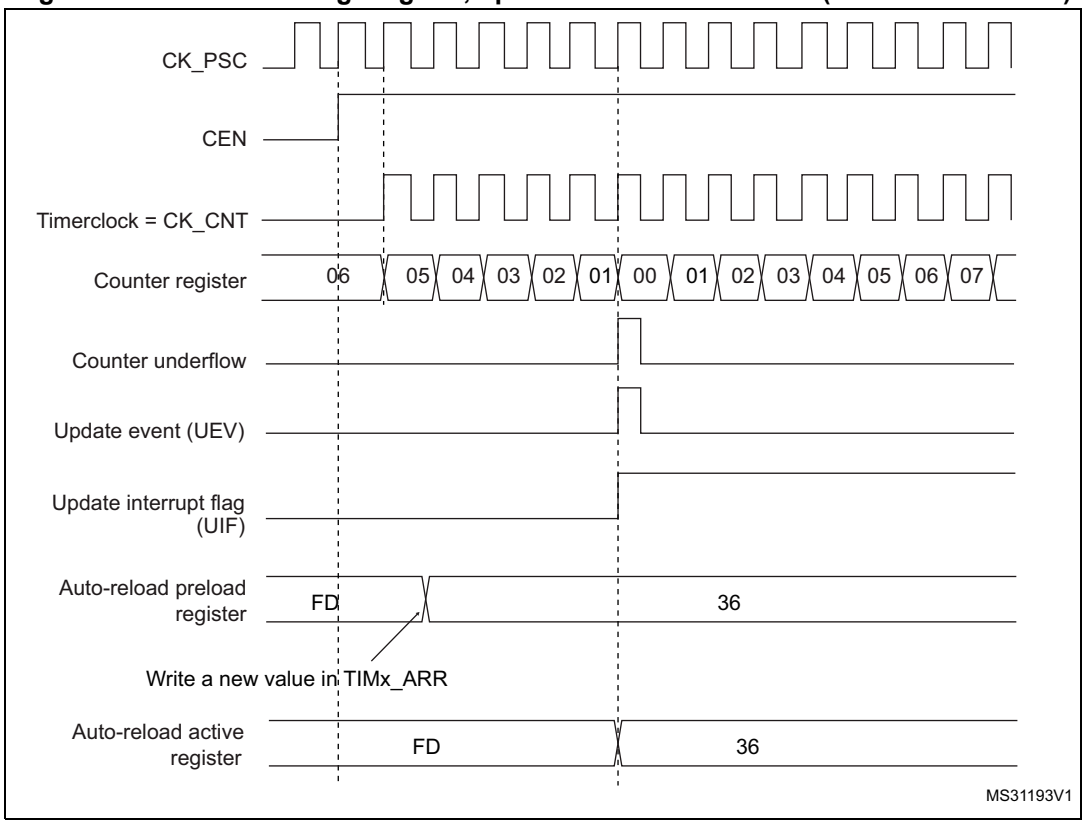
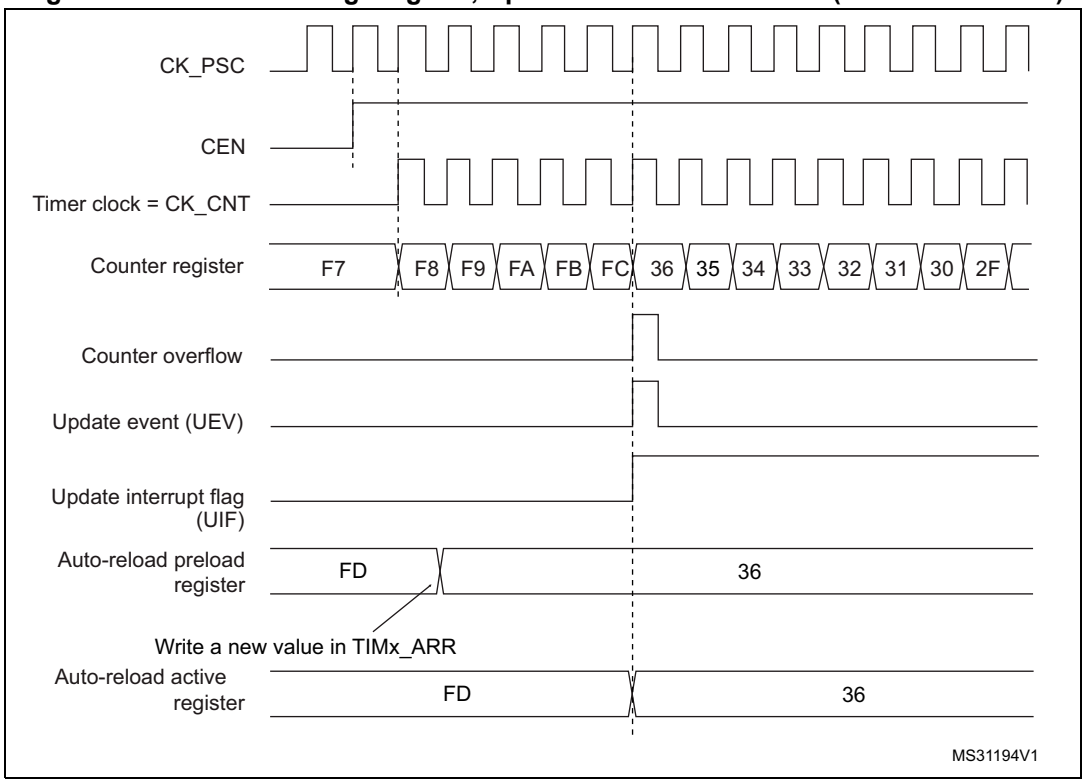


Figure 124. Counter timing diagram, Update event with ARPE=1 (counter overflow)



18.3.3 Repetition counter

Section 18.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

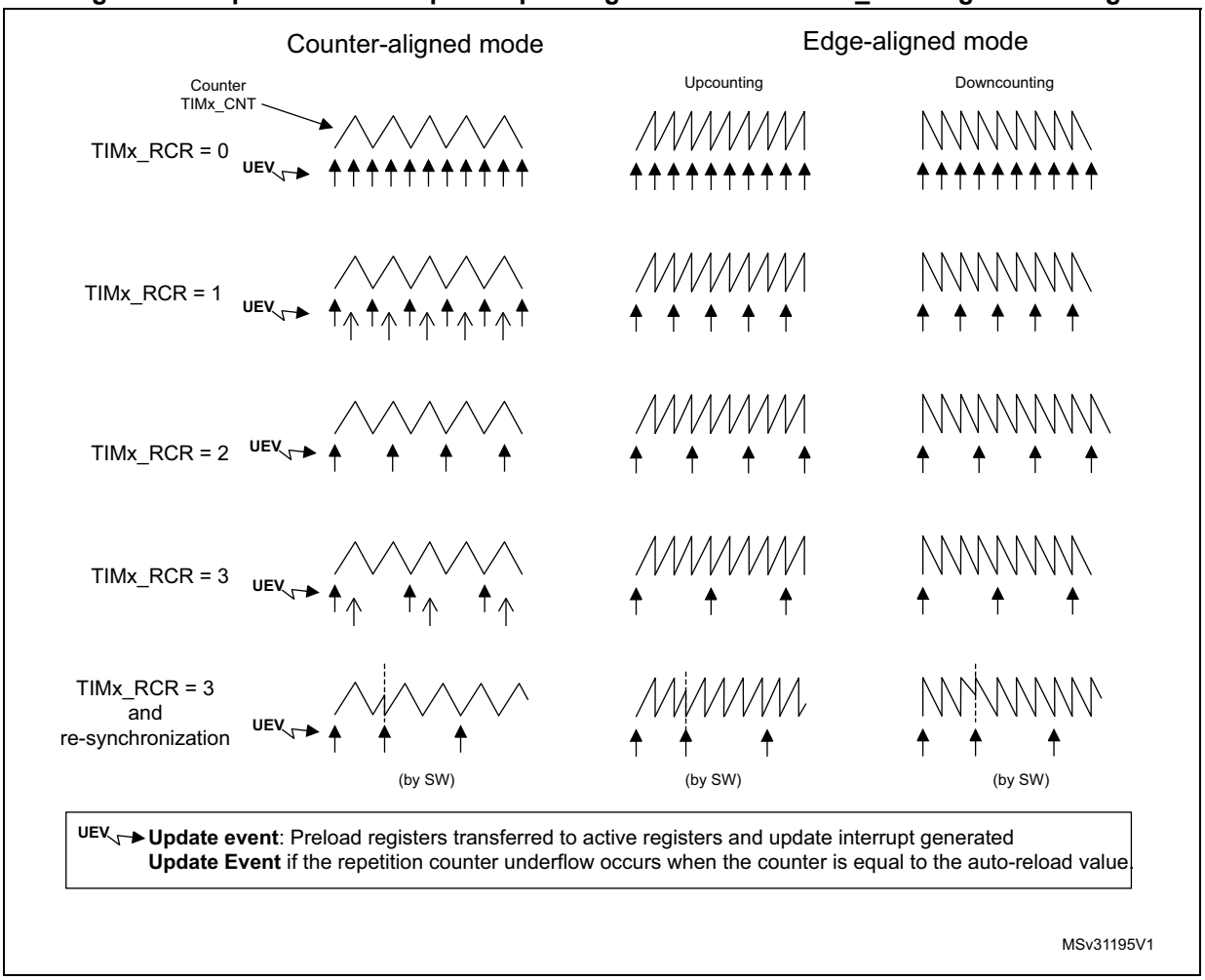
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 125*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 125. Update rate examples depending on mode and TIMx_RCR register settings



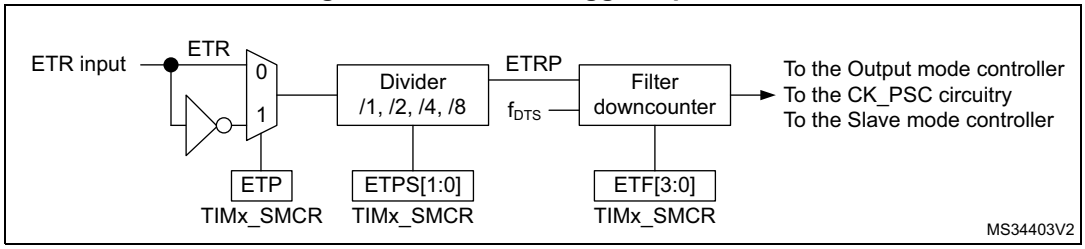
18.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 18.3.5*)
- trigger for the slave mode (see *Section 18.3.25*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 18.3.7*)

Figure 126 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 126. External trigger input block



18.3.5 Clock selection

The counter clock can be provided by the following clock sources:

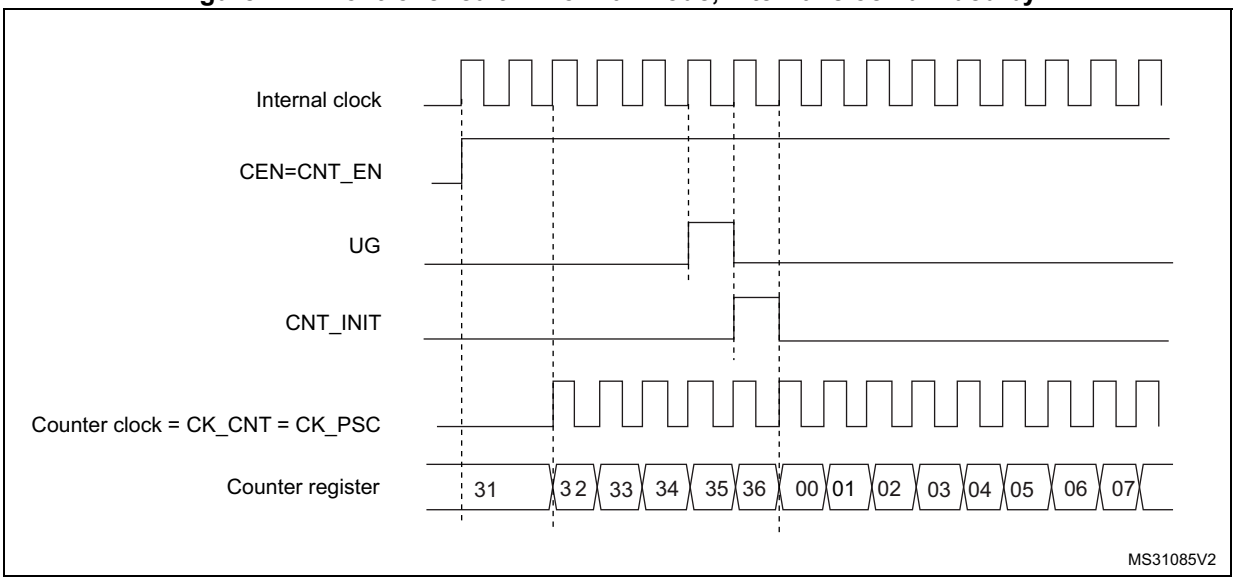
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 127 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

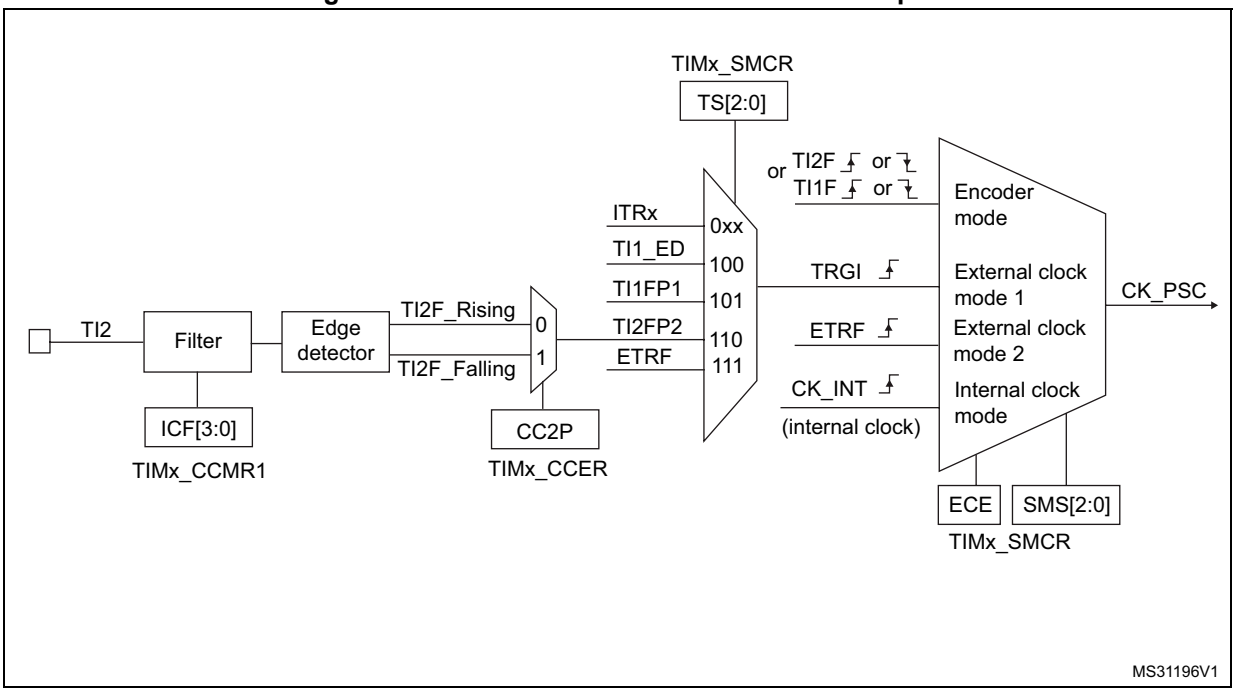
Figure 127. Control circuit in normal mode, internal clock divided by 1



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 128. 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:

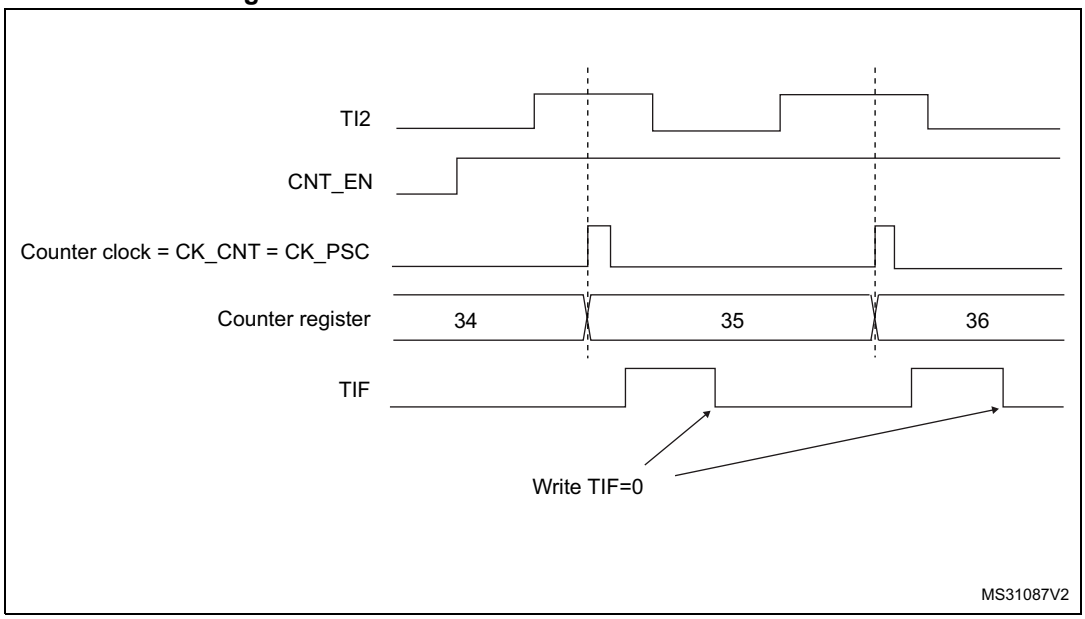
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 129. Control circuit in external clock mode 1



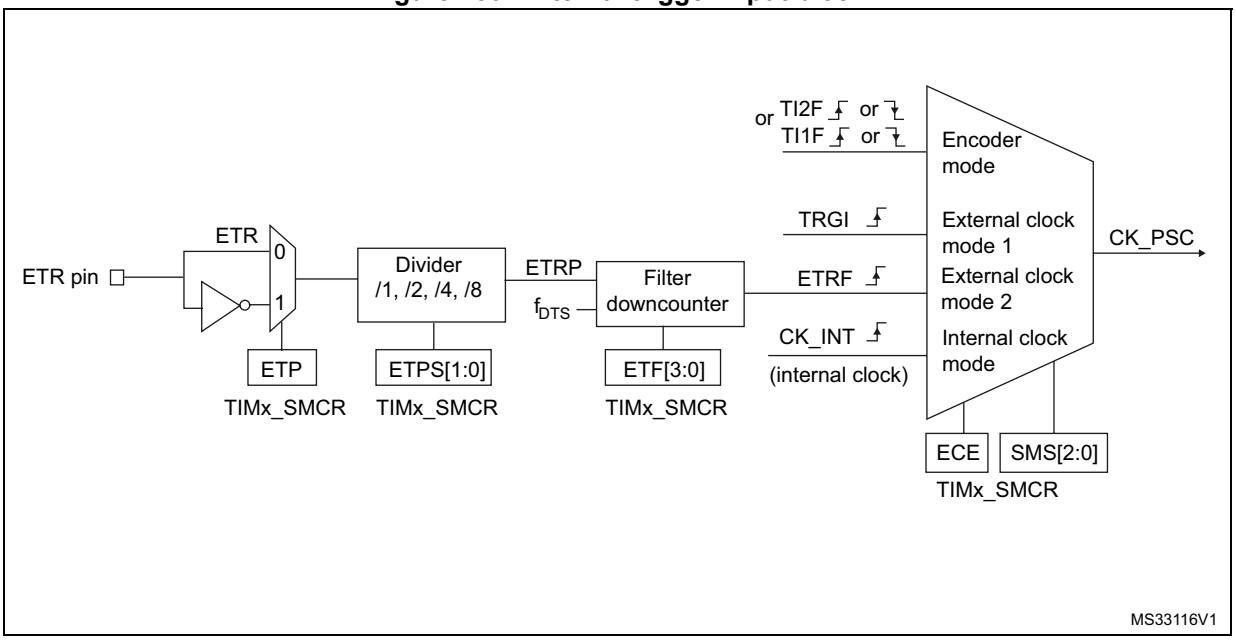
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 130* gives an overview of the external trigger input block.

Figure 130. External trigger input block



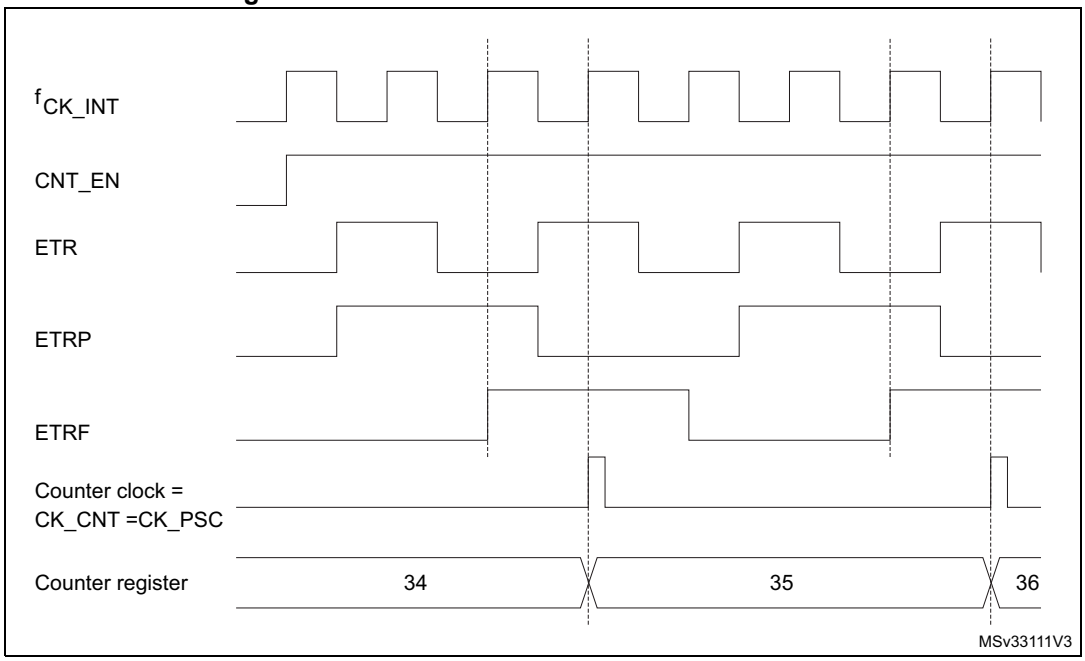
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

Figure 131. Control circuit in external clock mode 2



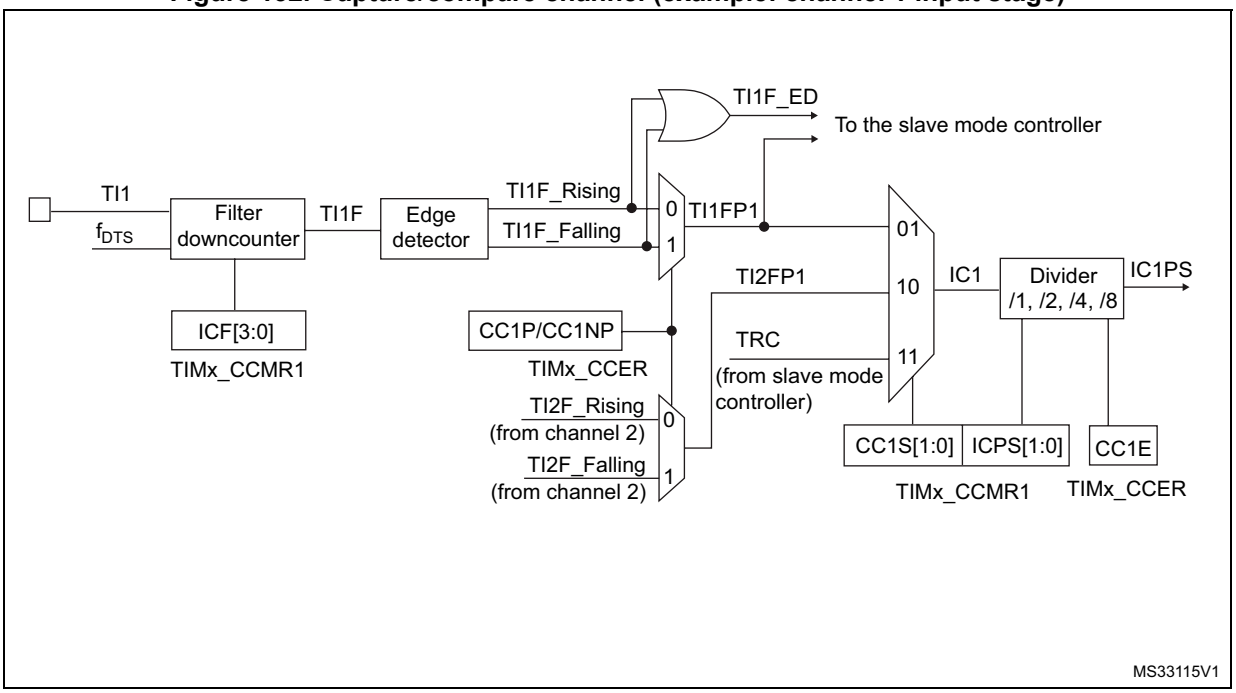
18.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 132 to *Figure 135* 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 132. 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 133. Capture/compare channel 1 main circuit

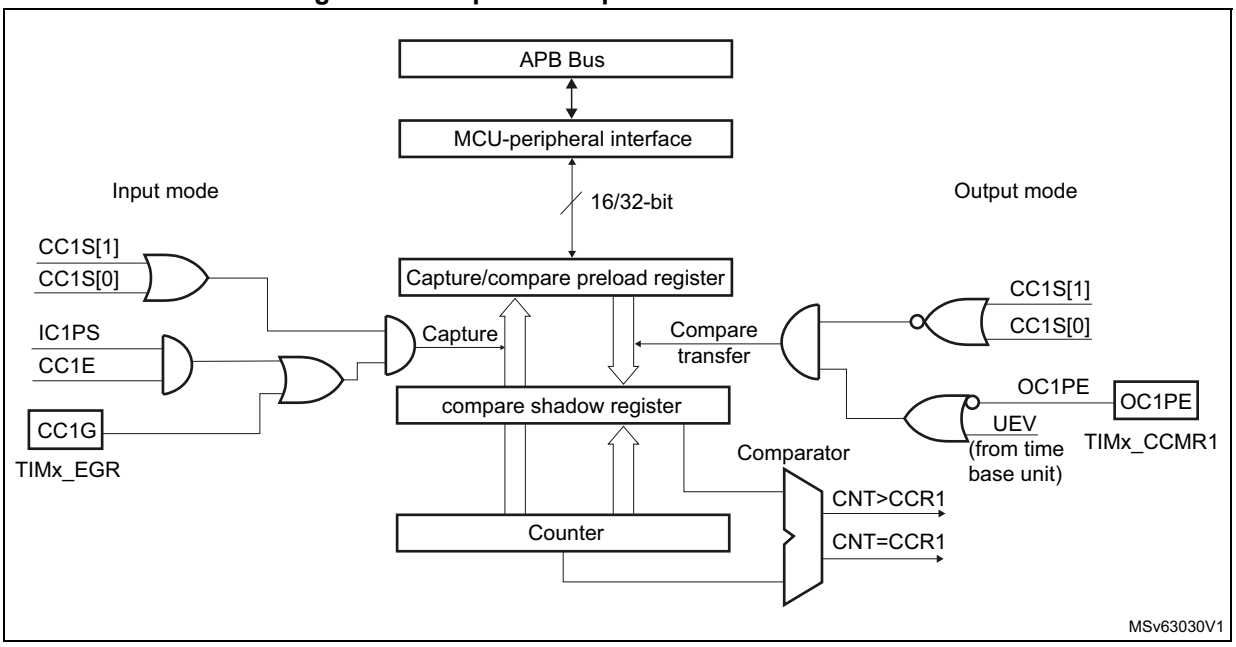
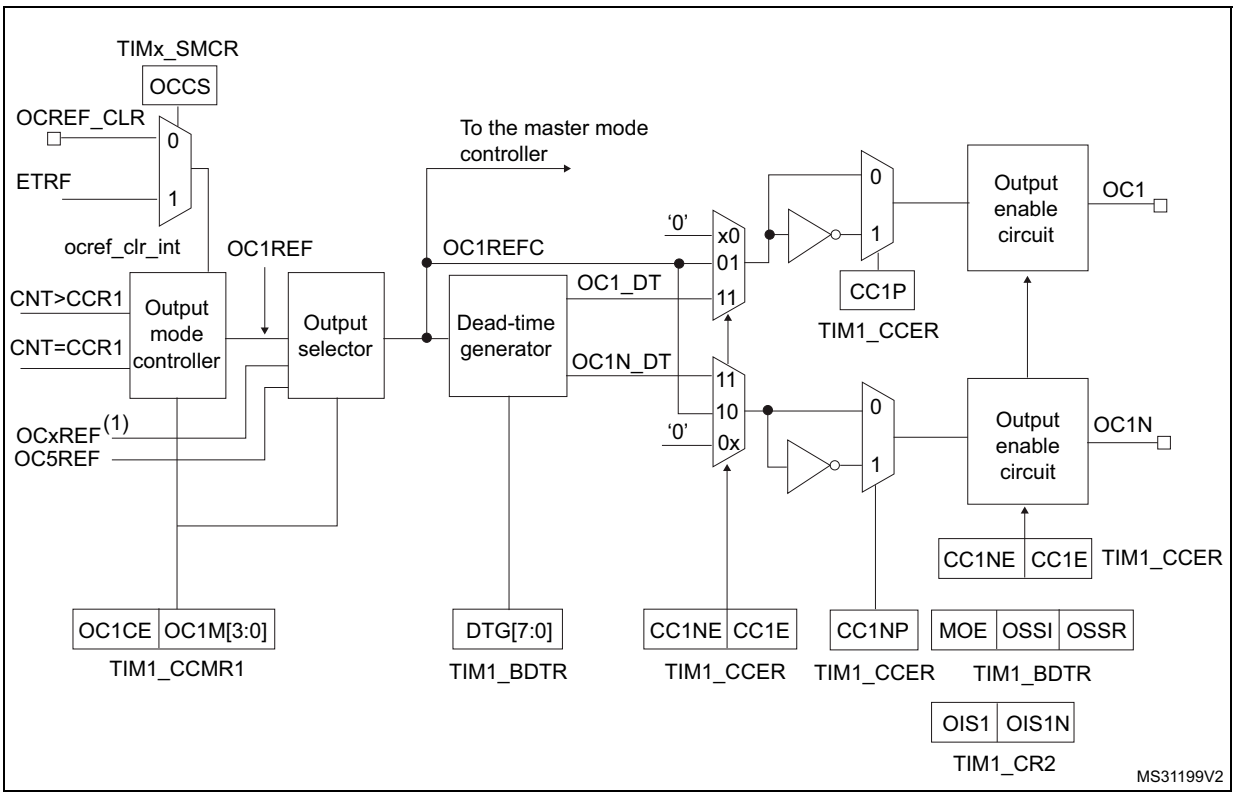


Figure 134. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 135. Output stage of capture/compare channel (channel 4)

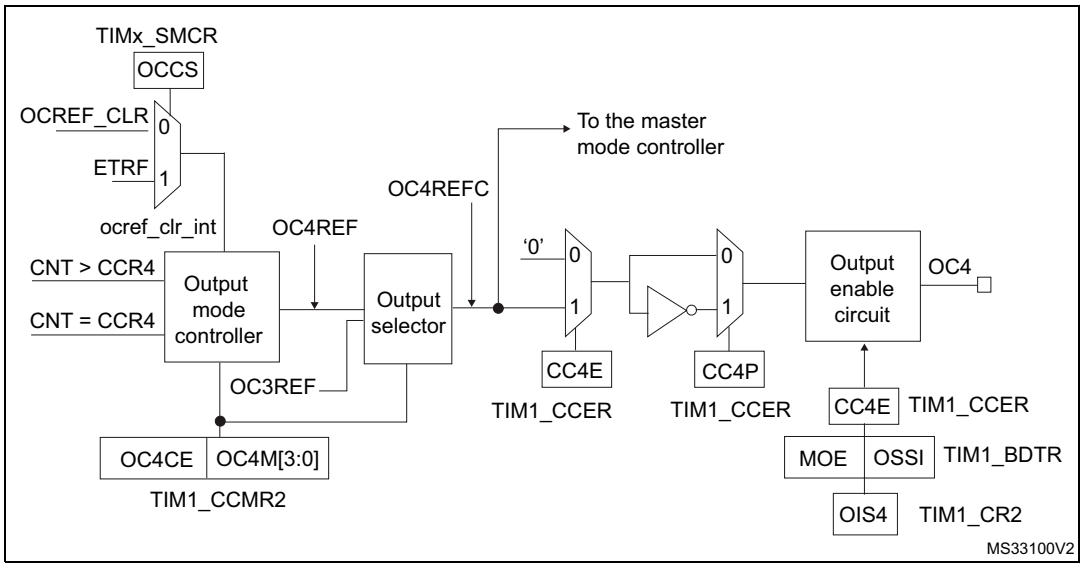
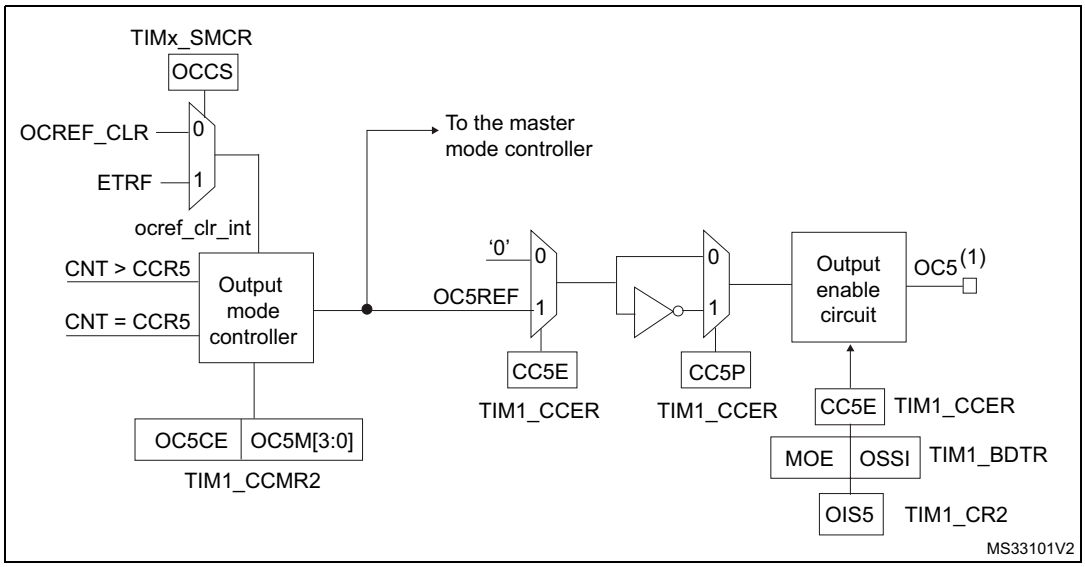


Figure 136. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

18.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at 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 writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

18.3.8 PWM input mode

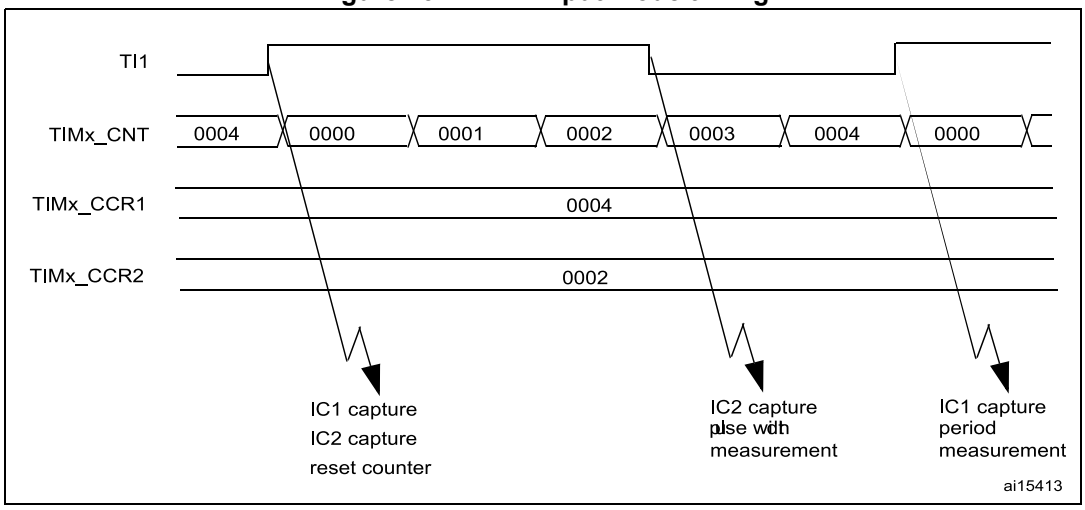
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 137. PWM input mode timing



18.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is

forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

18.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

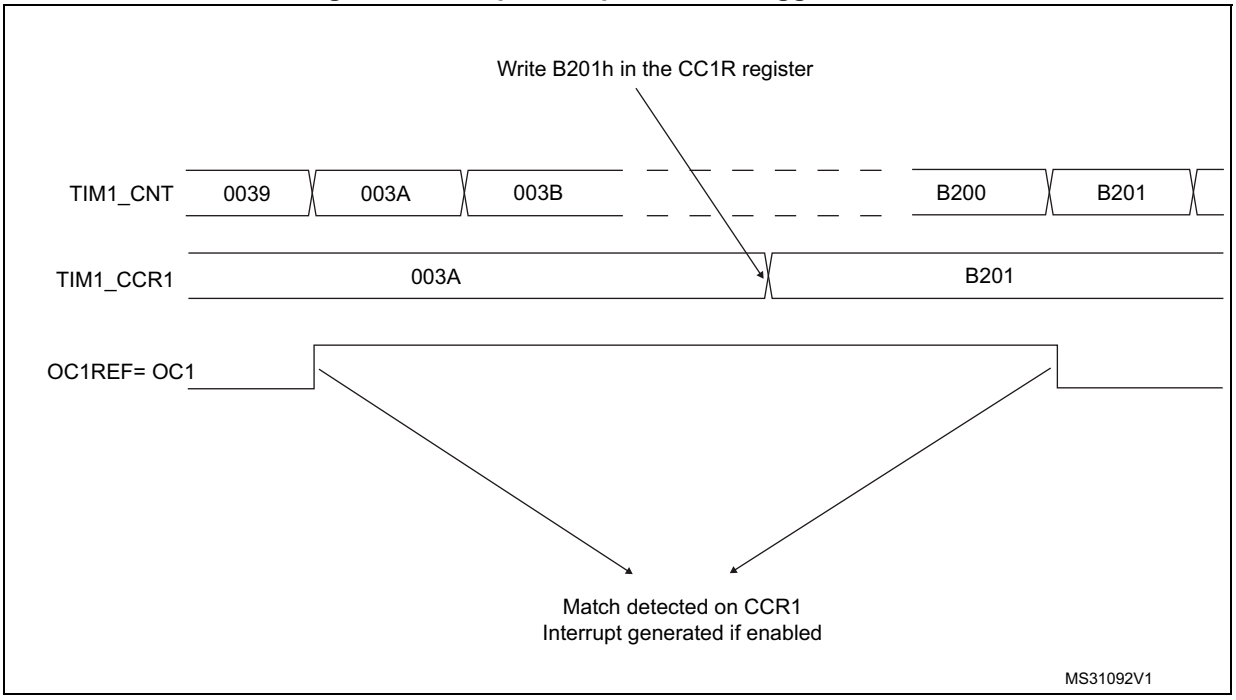
Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx

shadow register is updated only at the next update event UEV). An example is given in *Figure 138*.

Figure 138. Output compare mode, toggle on OC1



18.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, 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.

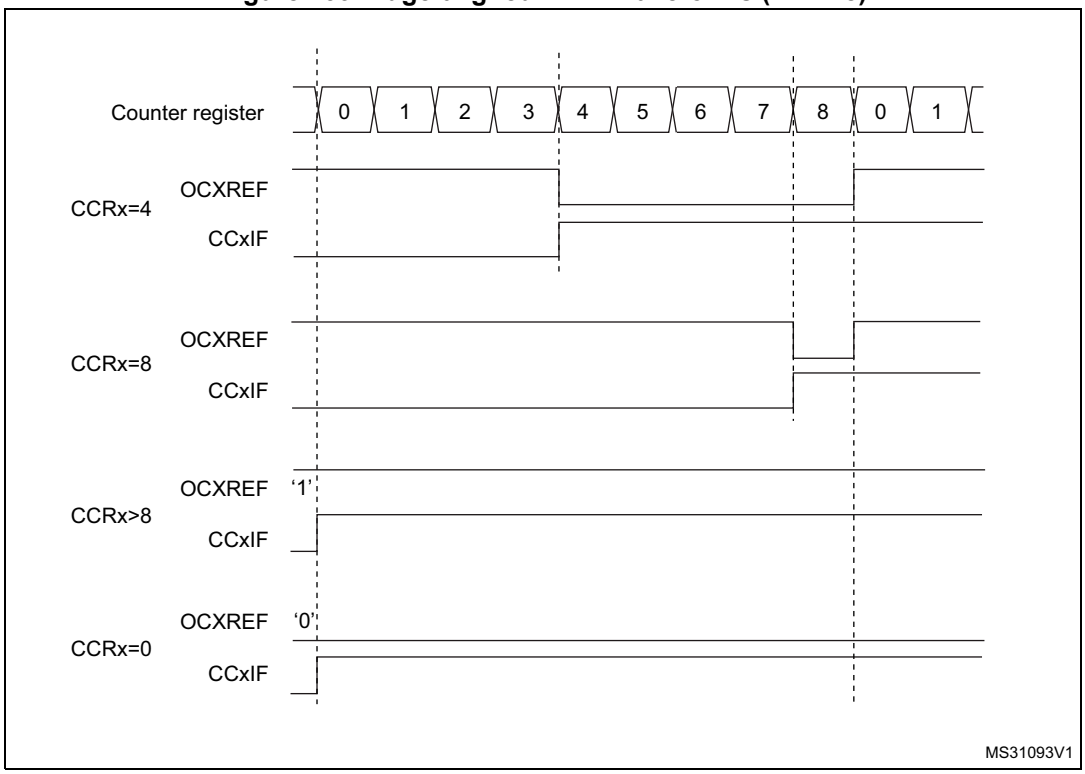
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 389*.

  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 139* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 139. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 393*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

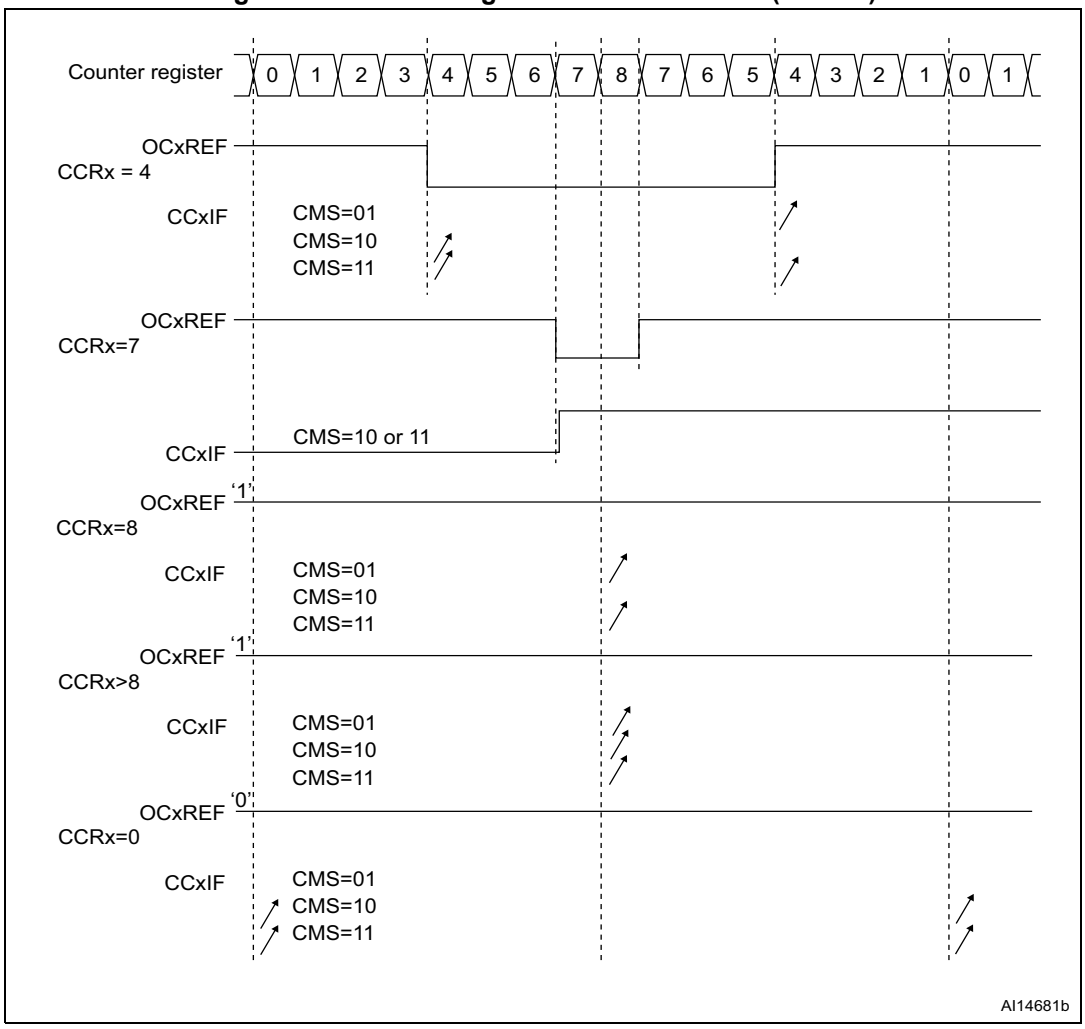
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 396*.

Figure 140 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 140. 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 a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

18.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

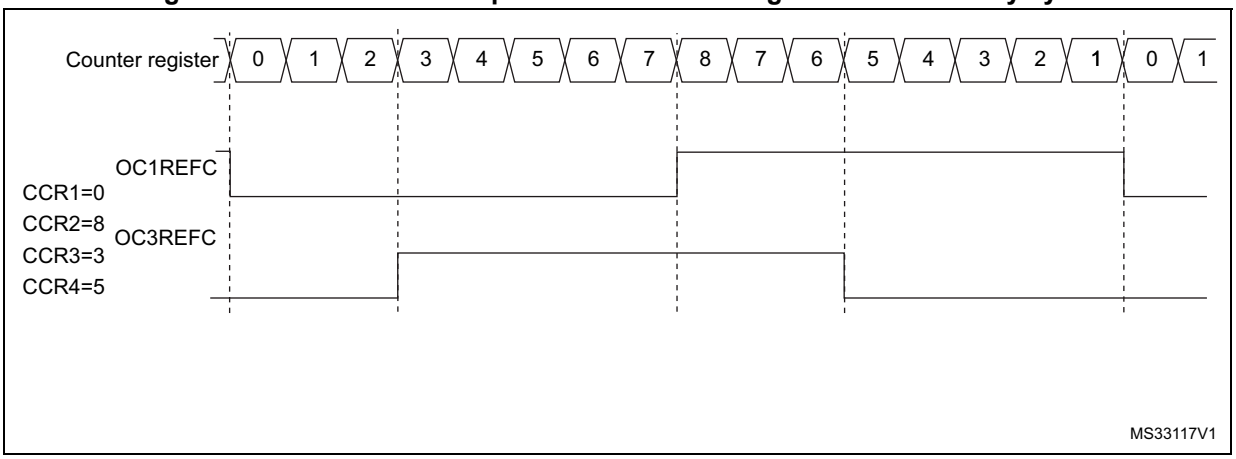
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 141 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 141. Generation of 2 phase-shifted PWM signals with 50% duty cycle



18.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

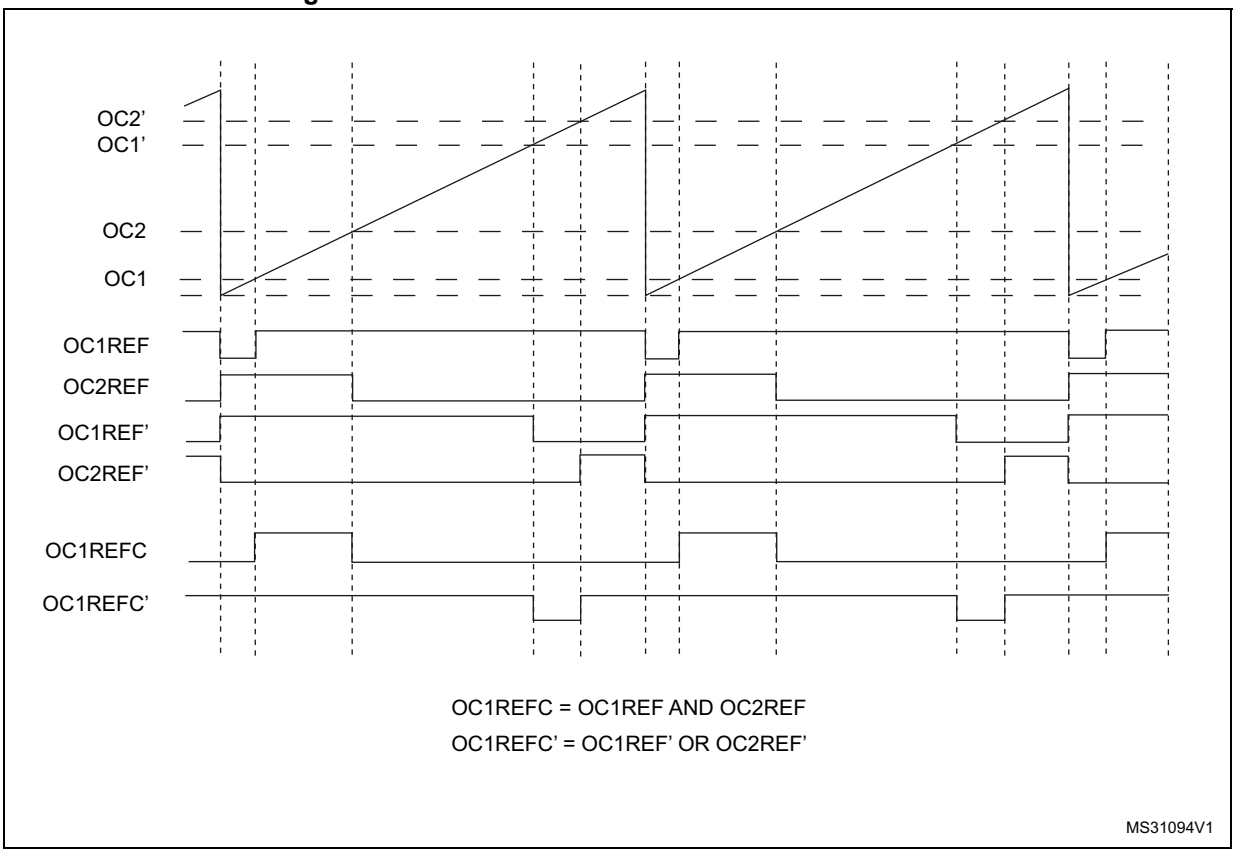
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 142 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 142. Combined PWM mode on channel 1 and 3



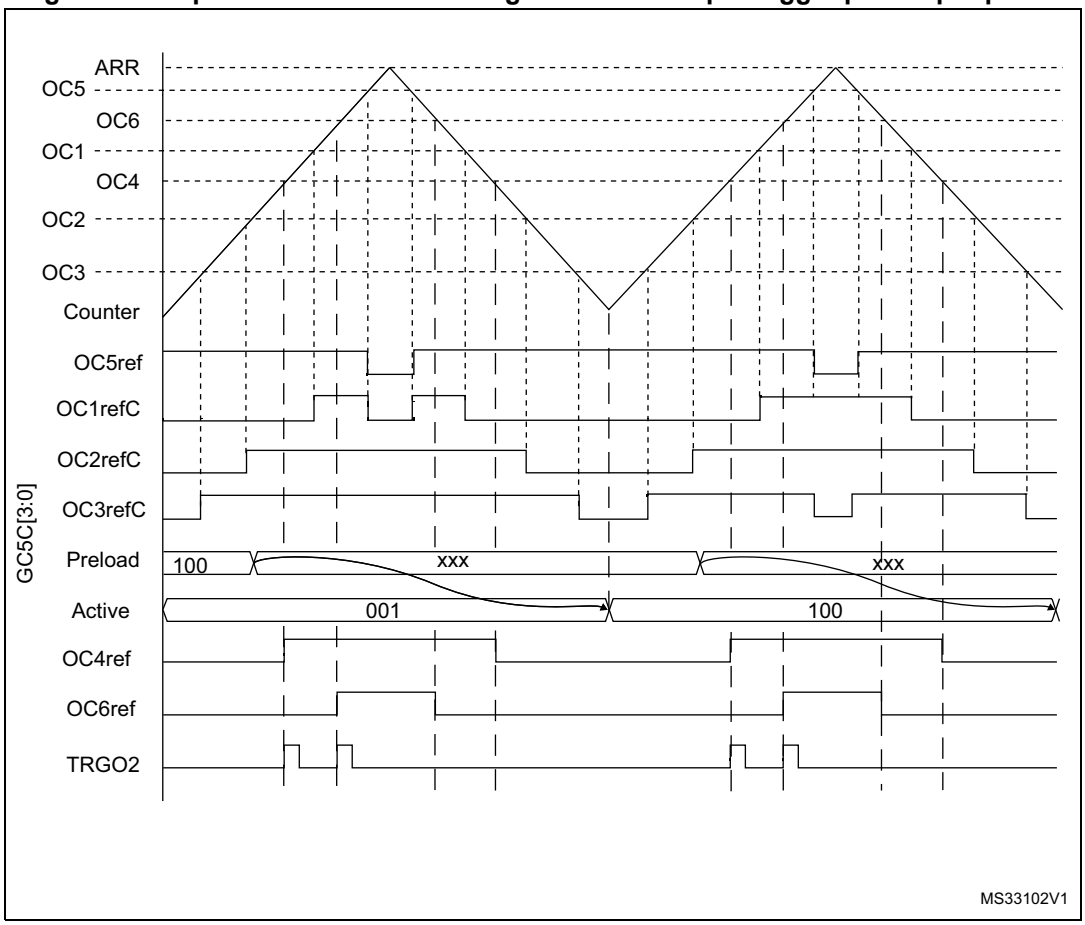
18.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 143. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 18.3.26: ADC synchronization* for more details.

18.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected 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 69: Output control bits for complementary OCx and OCxN channels with break feature on page 463* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 144. Complementary output with dead-time insertion

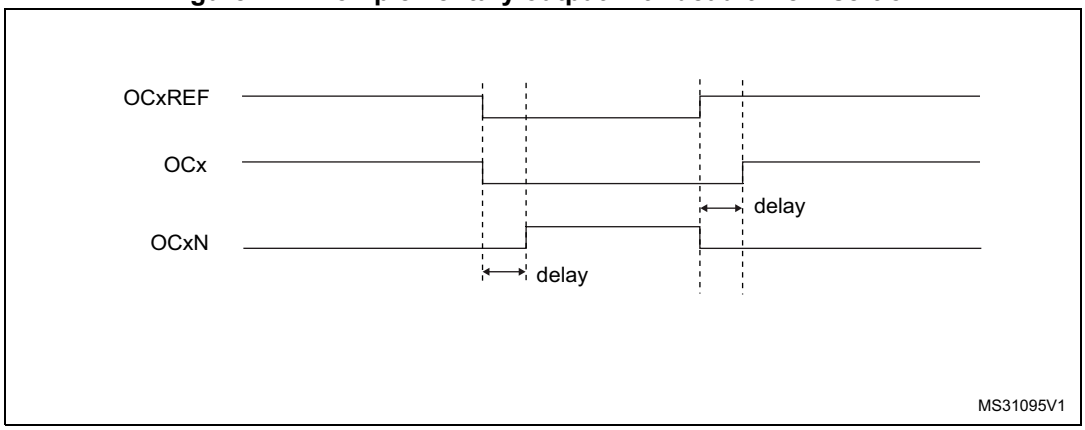


Figure 145. Dead-time waveforms with delay greater than the negative pulse

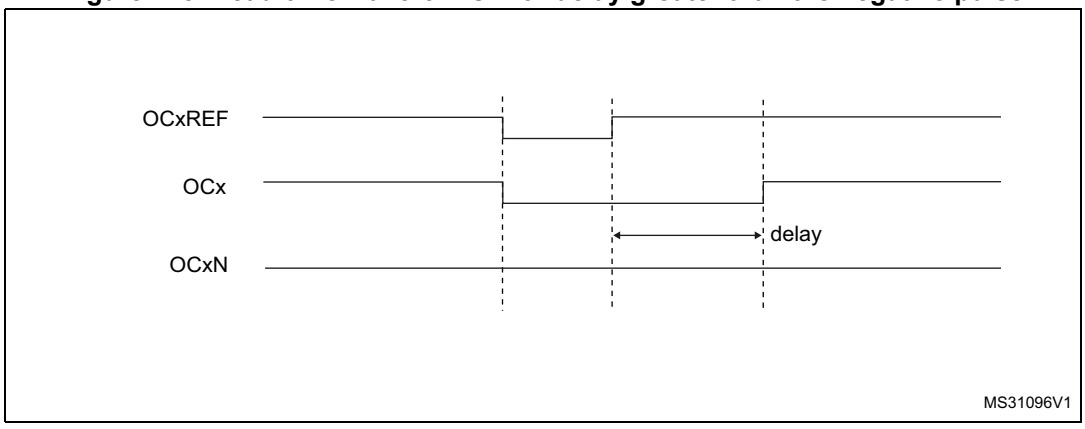
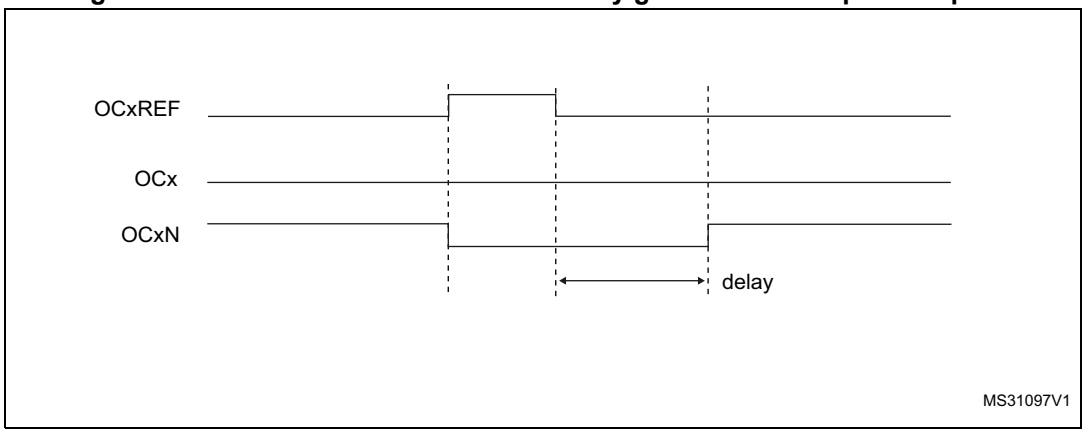


Figure 146. Dead-time waveforms with delay greater than the positive pulse



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 18.4.20: TIM1 break and dead-time register (TIM1_BDTR)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

18.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When using the break functions, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 69: Output control bits for complementary OCx and OCxN channels with break feature on page 463* for more details.

The source for BRK can be:

- An external source connected to the BKIN pin
- An internal source: COMP4 output

The source for BRK_ACTH can be internal only:

- A clock failure event generated by the CSS. For further information on the CSS, refer to *Section 8.2.7: Clock security system (CSS)*
- A PVD output
- SRAM parity error signal
- Cortex®-M4 LOCKUP (Hardfault) output
- COMPx output, x = 1,2,3,5 and 6

Caution: The internal sources protection is not available when the timer is in automatic output enable mode (AOE bit set in the TIMx_BDTR). The MOE bit is set again on the next update event, regardless of any pending error on the BRK_ACTH input.

The source for BRK2 can be:

- An external source connected to the BKIN2 pin
- An internal source coming from COMPx output, x = 1..7

If there are several break sources, the resulting break signal will be an OR between all the input signals.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated by any of the two break inputs (BRK, BRK2)and which has a:

- Programmable polarity (BKP/BK2P bit in the TIMx_BDTR register)
- Programmable enable bit (BKE/BK2E in the TIMx_BDTR register)
- Programmable filter (BKxF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

The digital filter feature is available on BRK and BRK2. It is not available on BRK_ACTH.

That means that the digital filter is:

- Available when the break source is external and comes from the external inputs BKIN/BKIN2.
- Available when the break source is internal and connected to BRK (COMP4 output) or BRK2 (all comparators' outputs)
- Not available when the break source is internal and connected to BRK_ACTH. (i.e. PVD output, SRAM parity error signal, Cortex®-M4 LOCKUP (Hardfault) output or COMPx output, x = 1, 2, 3, 5 and 6).

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

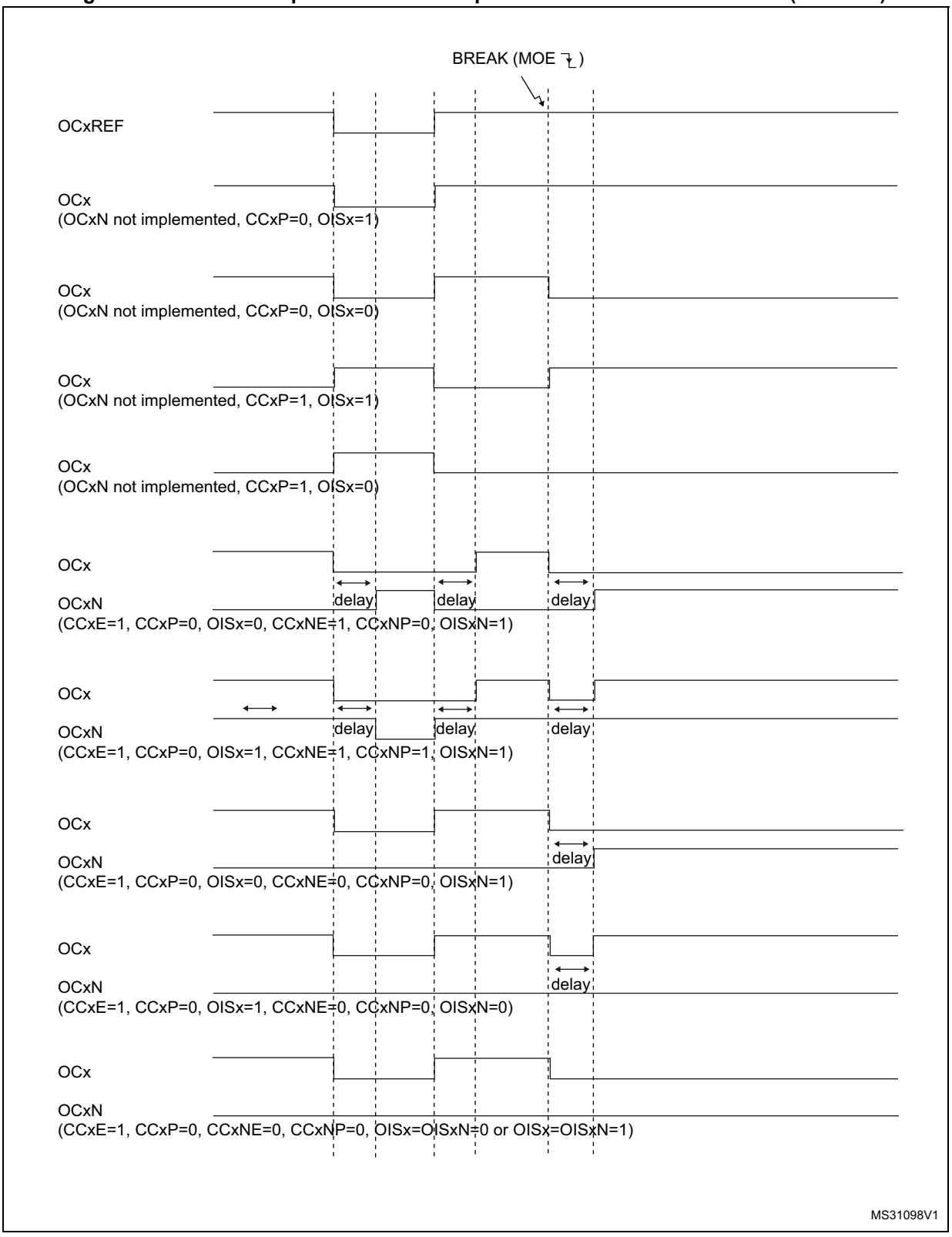
- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 18.4.20: TIM1 break and dead-time register (TIM1_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 147 shows an example of behavior of the outputs in response to a break.

Figure 147. Various output behavior in response to a break event on BKIN (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in *Table 66*.

Note: BRK2 must only be used with OSSR = OSSI = 1.

Table 66. Behavior of timer outputs versus BRK/BRK2 inputs

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 148 gives an example of OCx and OCxN output behavior in case of active signals on BKIN and BKIN2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 148. PWM output state following BKIN and BKIN2 pins assertion (OSSI=1)

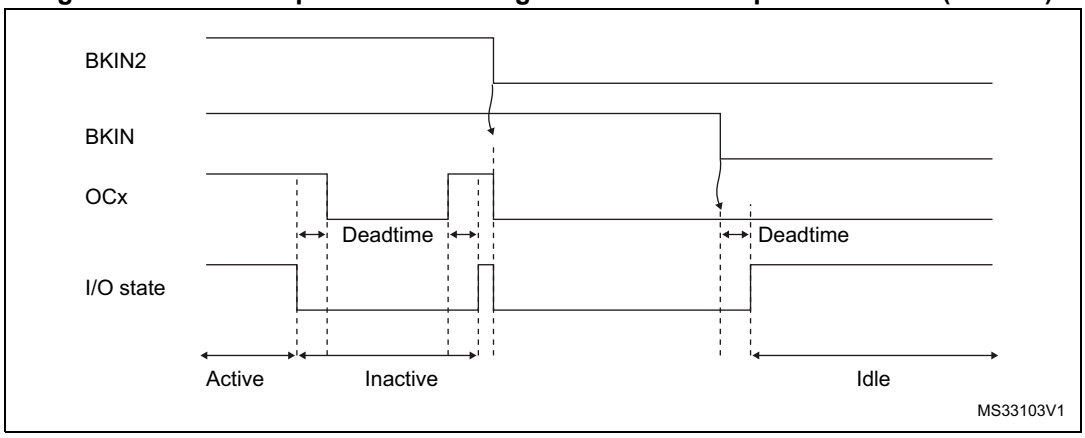
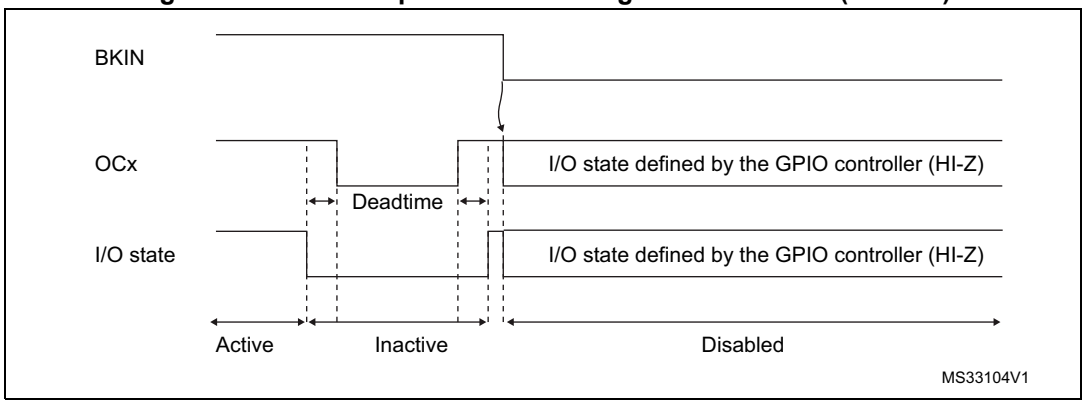


Figure 149. PWM output state following BKIN assertion (OSSI=0)



18.3.17 Clearing the OCxREF signal on an external event

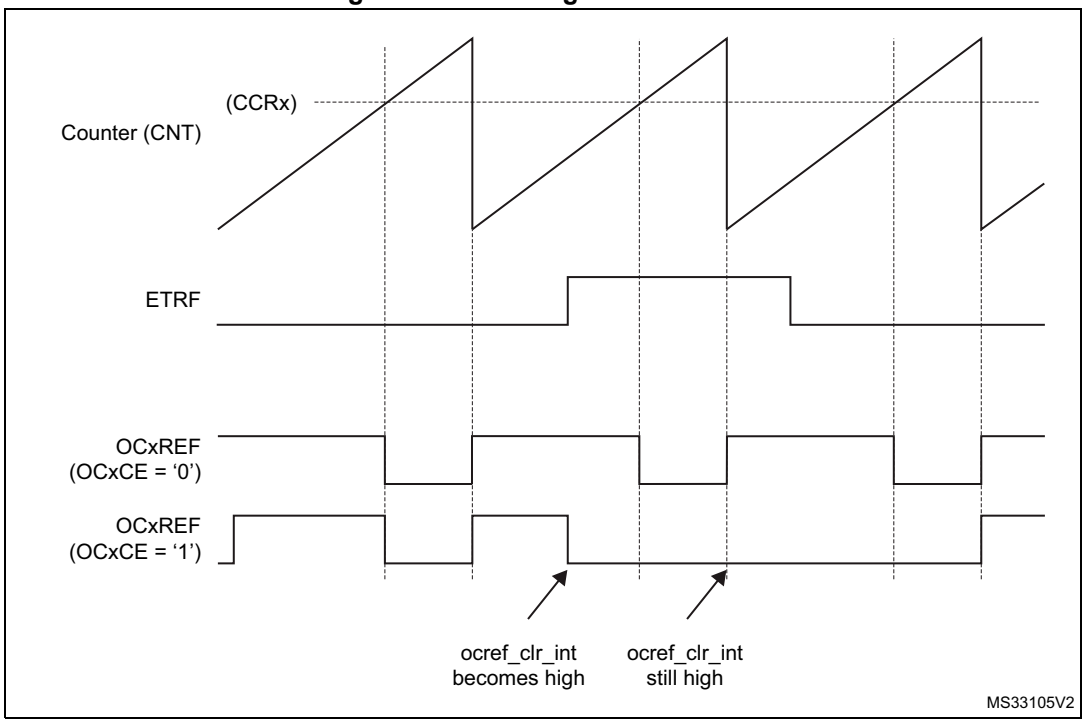
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. ocref_clr_int input can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 150 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 150. 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.

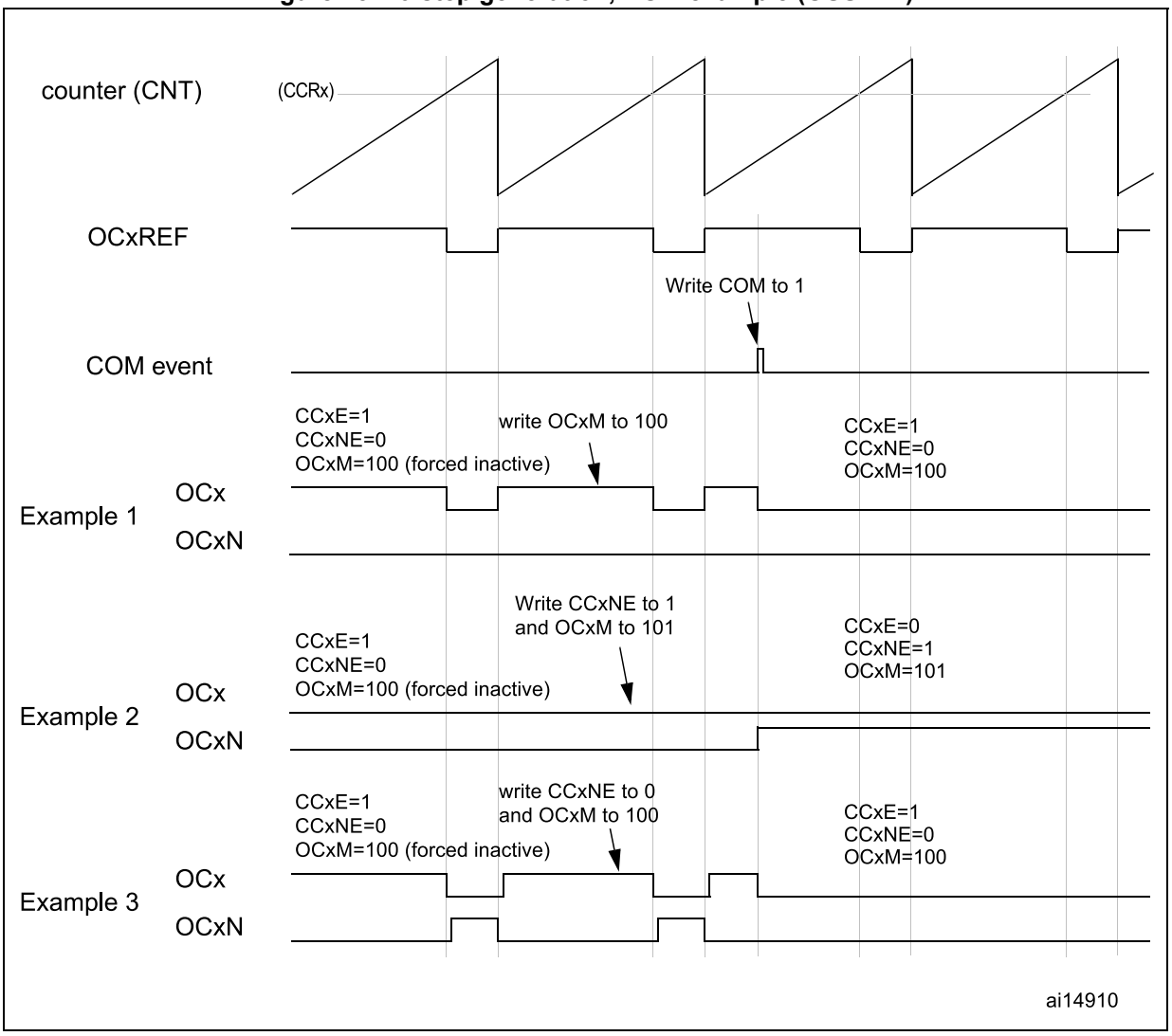
18.3.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 151* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 151. 6-step generation, COM example (OSSR=1)



18.3.19 One-pulse mode

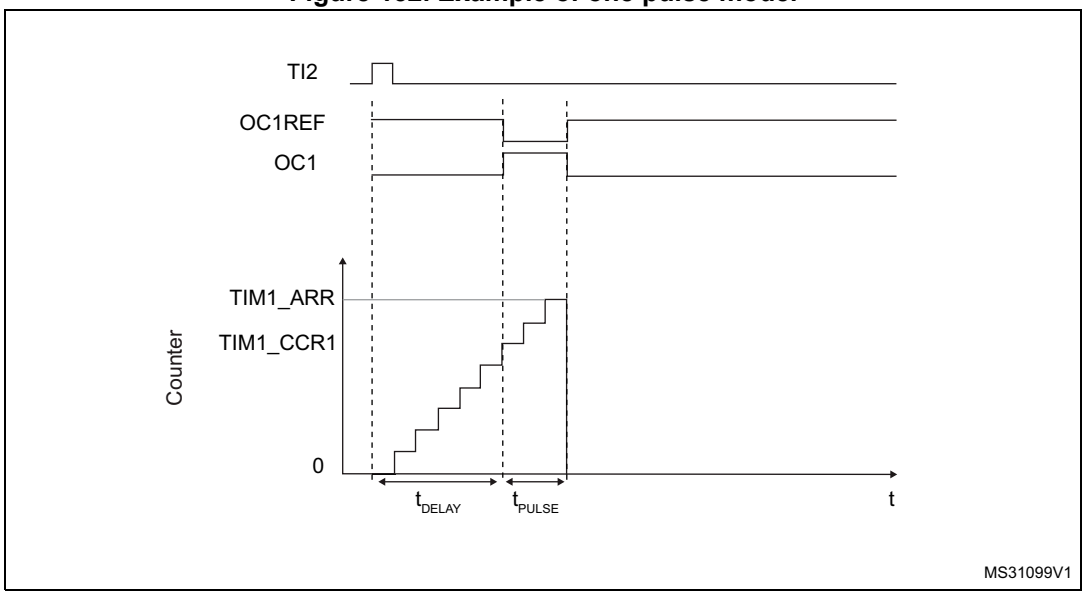
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 152. Example of one pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) 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.

18.3.20 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 18.3.19*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

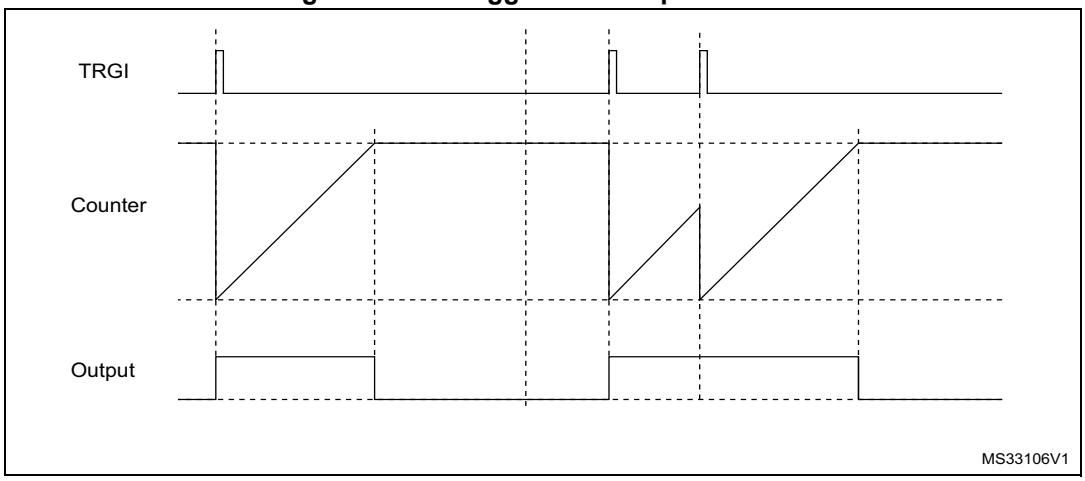
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 153. Retriggerable one pulse mode



18.3.21 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to *Table 67*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 67. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 154* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 154. Example of counter operation in encoder interface mode.

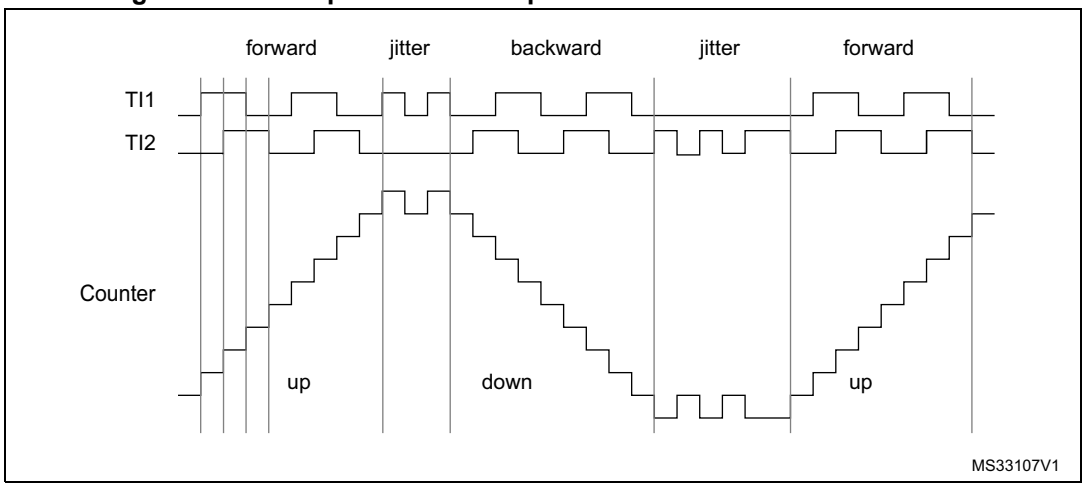
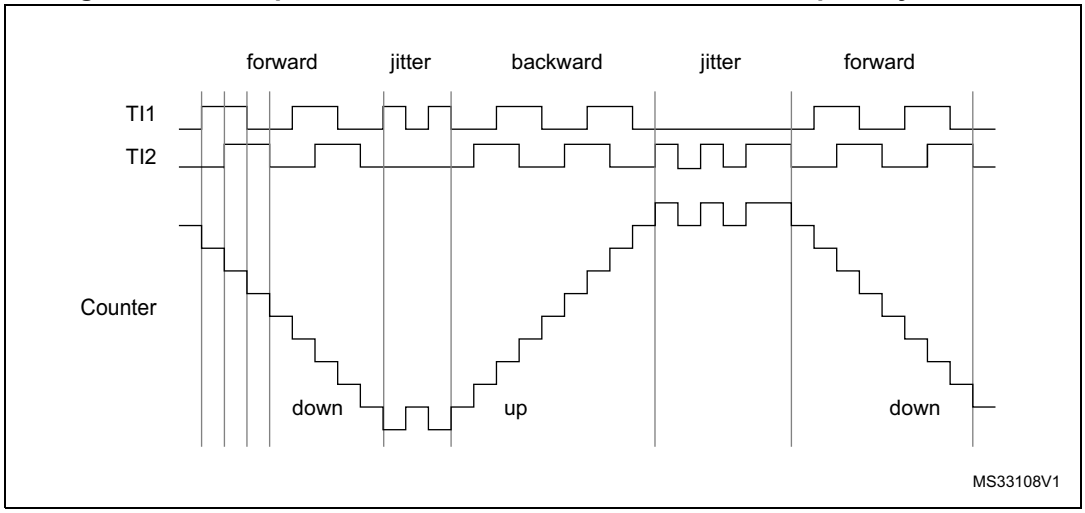


Figure 155 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 155. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

18.3.22 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

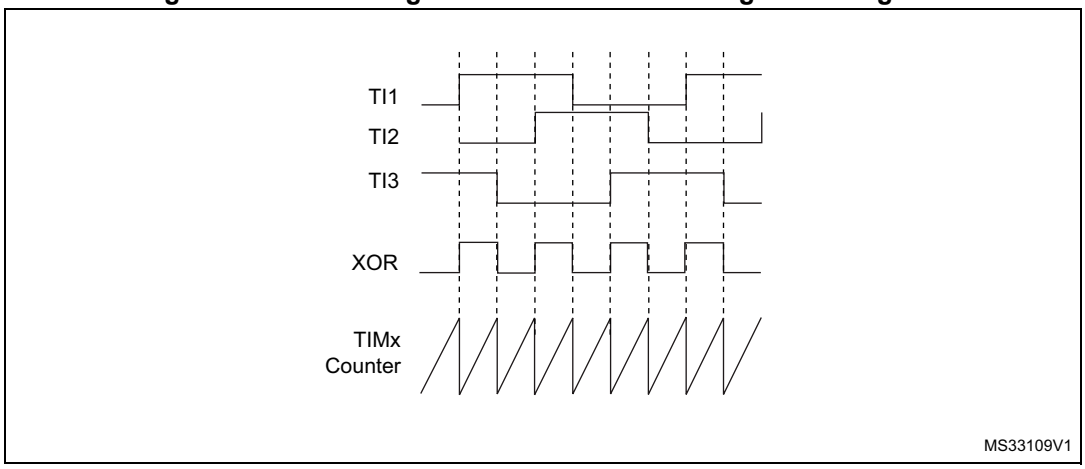
There is no latency between the UIF and UIFCPY flags assertion.

18.3.23 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 156* below.

Figure 156. Measuring time interval between edges on 3 signals



18.3.24 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3) referred to as “interfacing timer” in *Figure 157*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 132: Capture/compare channel (example: channel 1 input stage) on page 407*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

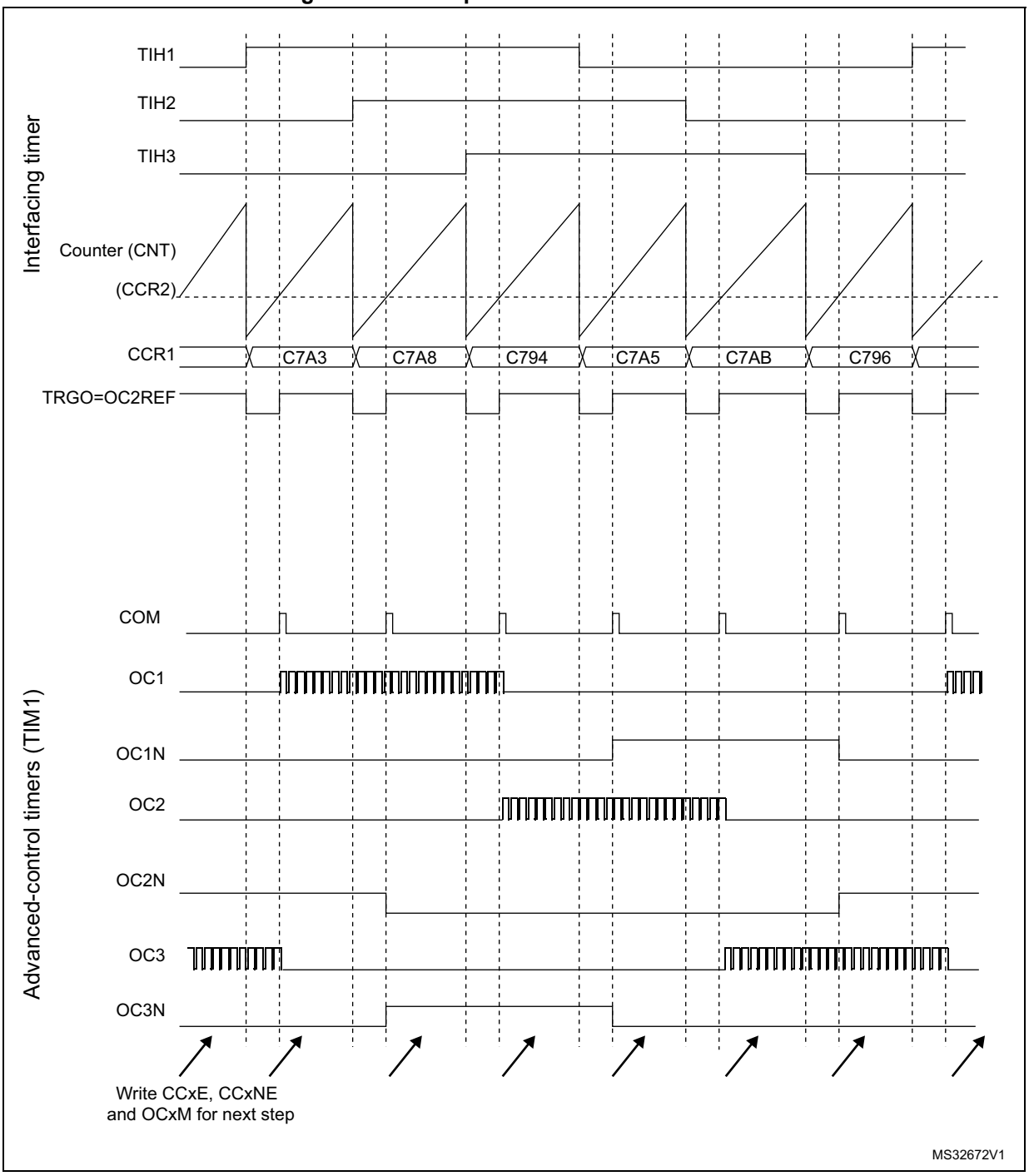
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 157* describes this example.

Figure 157. Example of Hall sensor interface



18.3.25 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 19.3.19: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

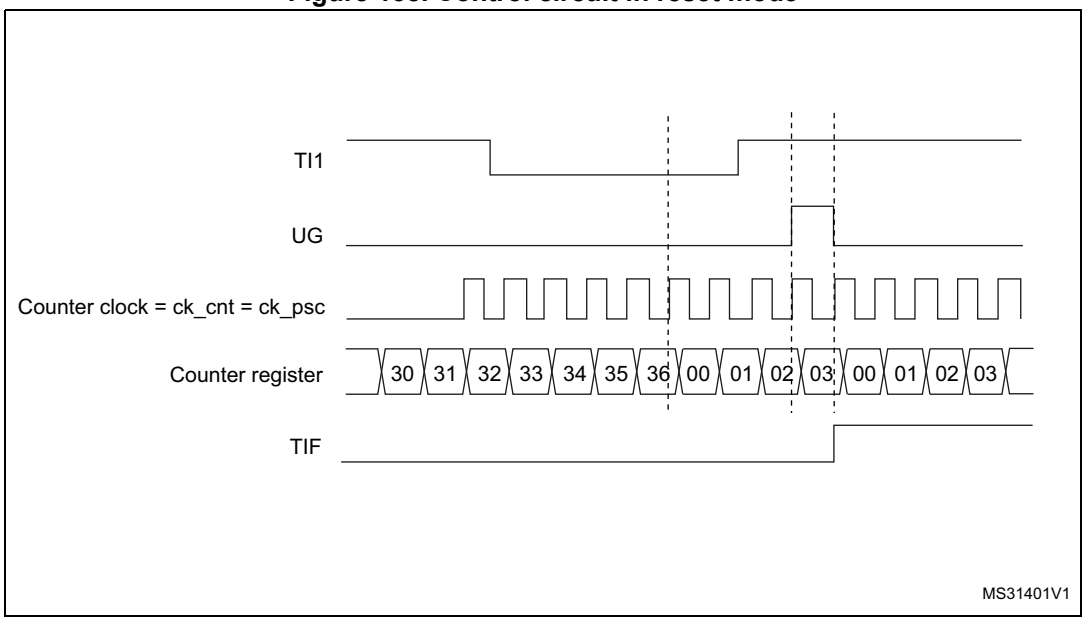
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=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 158. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

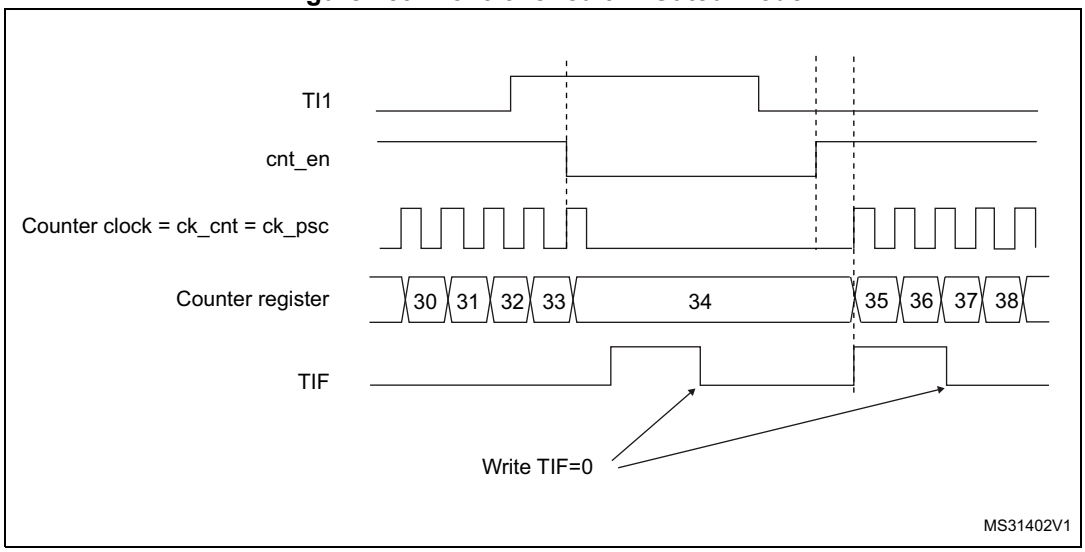
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=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 159. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1

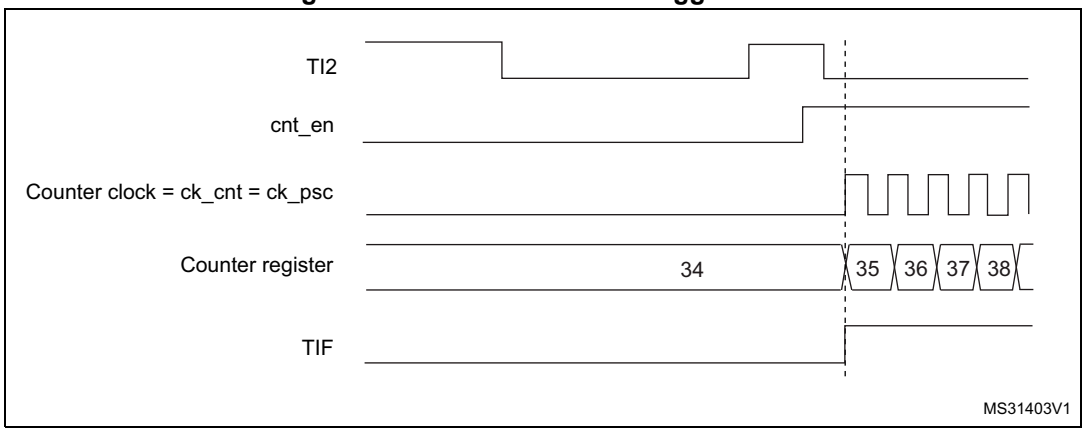
register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=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 160. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

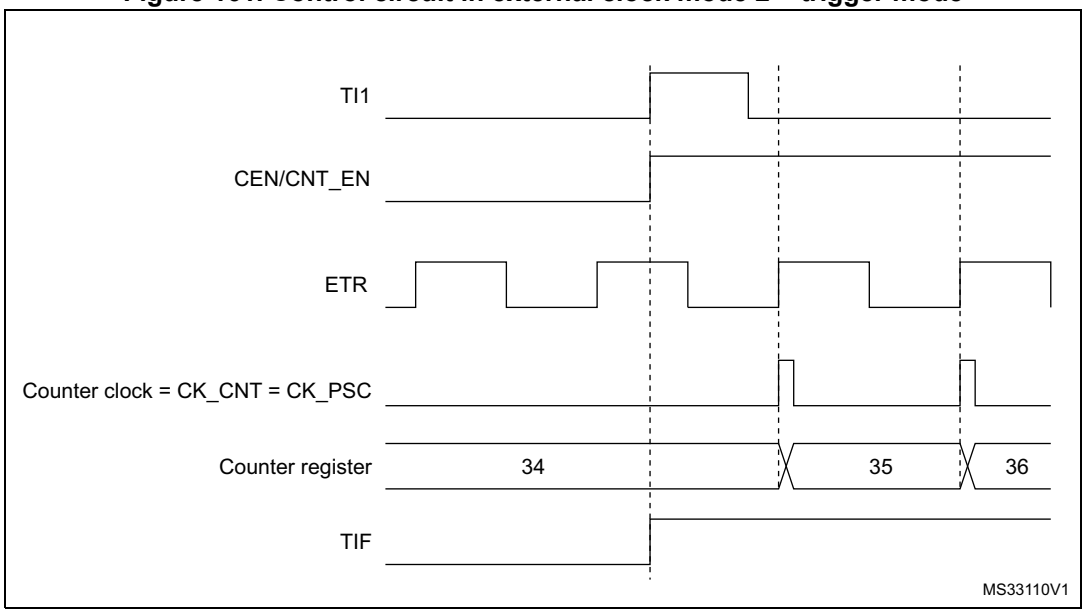
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in 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 161. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

18.3.26 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 143 on page 419*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

18.3.27 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

18.3.28 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

18.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

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

18.4.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx):

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Note: $t_{DTS} = 1/f_{DTS}$, $t_{CK\_INT} = 1/f_{CK\_INT}$.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

18.4.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)
Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)
Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

18.4.3 TIM1 slave mode control register (TIM1_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).
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).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of f_{CK_INT} frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 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 68: TIM1 internal trigger connection on page 449* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Note: The other bit is at position 16 in the same register

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: OCREF_CLR_INT is connected to the OCREF_CLR input
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 68. TIM1 internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM15	TIM2	TIM3	TIM17 OC1(1)

1. TIM1_ITR3 selection is made using bit 6 of the SYSCFG_CFGR1 register.

18.4.4 TIM1 DMA/interrupt enable register (TIM1_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

18.4.5 TIM1 status register (TIM1_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:13 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 **B2IF**: Break 2 interrupt flag
This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.
0: No break event occurred.
1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 18.4.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

18.4.6 TIM1 event generation register (TIM1_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

18.4.7 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Refer to IC1PSC[1:0] description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

18.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ocref_clr_int signal
1: OC1Ref is cleared as soon as a High level is detected on ocref_clr_int signal (OCREF_CLR input or ETRF input)

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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 decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

18.4.9 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

18.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

18.4.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high.
1: OC1N active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 69* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 69. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). **Note**: BRK2 can only be used if OSSI = OSSR = 1.	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

18.4.12 TIM1 counter (TIM1_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

18.4.13 TIM1 prescaler (TIM1_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

18.4.14 TIM1 auto-reload register (TIM1_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 18.3.1: Time-base unit on page 387* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

18.4.15 TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

18.4.16 TIM1 capture/compare register 1 (TIM1_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

18.4.17 TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

18.4.18 TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

18.4.19 TIM1 capture/compare register 4 (TIM1_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

18.4.20 TIM1 break and dead-time register (TIM1_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low

1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

0: Break input BRK2 disabled

1: Break input BRK2 enabled

Note: The BRK2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously

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

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously

0001: $f_{SAMPLING}=f_{CK\_INT}$, 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

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 18.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 18.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 18.4.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and 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}=125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

18.4.21 TIM1 DMA control register (TIM1_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

– If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.

– If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

18.4.22 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

18.4.23 TIM1 option registers (TIM1_OR)

Address offset: 0x50

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM1_ETR_ ADC2_RMP		TIM1_ETR_ ADC1_RMP	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **TIM1_ETR_ADC2_RMP[1:0]**: TIM1_ETR_ADC2 remapping capability

00: TIM1_ETR is not connected to any AWD (analog watchdog)
01: TIM1_ETR is connected to ADC2 AWD1
10: TIM1_ETR is connected to ADC2 AWD2
11: TIM1_ETR is connected to ADC2 AWD3

Bits 1:0 **TIM1_ETR_ADC1_RMP[1:0]**: TIM1_ETR_ADC1 remapping capability

00: TIM1_ETR is not connected to any AWD
01: TIM1_ETR is connected to ADC1 AWD1
10: TIM1_ETR is connected to ADC1 AWD2
11: TIM1_ETR is connected to ADC1 AWD3

Note: ADC1 and ADC2 AWDs are "ORed" with the other TIM1 ETR source signals. Consequently, the ETR alternate function inputs must be disabled when using AWDs features.

18.4.24 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode
Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable
Refer to OC1FE description.

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

Bit 7 **OC5CE:** Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode
Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable
Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.

18.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3
Distortion on Channel 3 output:
0: No effect of OC5REF on OC3REFC
1: OC3REFC is the logical AND of OC3REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group Channel 5 and Channel 2
Distortion on Channel 2 output:
0: No effect of OC5REF on OC2REFC
1: OC2REFC is the logical AND of OC2REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group Channel 5 and Channel 1
Distortion on Channel 1 output:
0: No effect of OC5REF on OC1REFC5
1: OC1REFC is the logical AND of OC1REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

18.4.26 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value
CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.

18.4.27 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 70. TIM1 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM1_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM1_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM1_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**TIM1_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM1_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TIM1_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM1_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM1_CCMR2** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM1_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	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	0	0	0	0	0

Table 70. TIM1 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIM1_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM1_PSC**	Res.	Res.	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM1_ARR**	Res.	Res.	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM1_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIM1_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM1_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM1_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM1_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM1_BDTR**	Res.	Res.	Res	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM1_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM1_DMAR**	DMAB[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM1_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM1_ETR_ADC2_RMP		TIM1_ETR_ADC1_RMP	
	Reset value																													0	0	0	0

Table 70. TIM1 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	**TIM1_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	**TIM1_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**TIM1_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

19 General-purpose timers (TIM2/TIM3)

19.1 TIM2/TIM3 introduction

The general-purpose timers consist of a 16-bit/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.

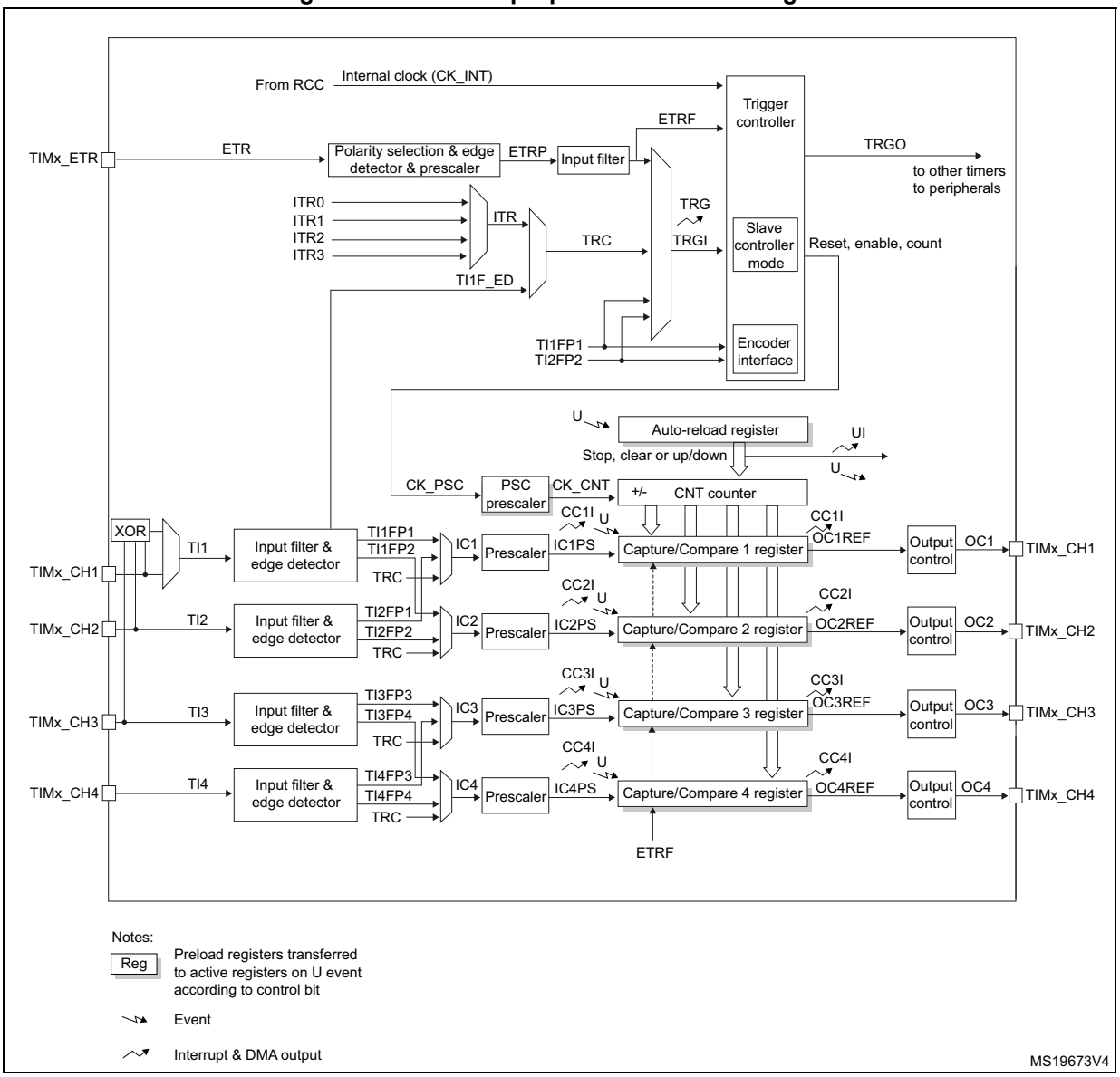
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 19.3.19: Timer synchronization*.

19.2 TIM2/TIM3 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3) or 32-bit (TIM2) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 162. General-purpose timer block diagram



19.3 TIM2/TIM3 functional description

19.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 163 and *Figure 164* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 163. Counter timing diagram with prescaler division change from 1 to 2

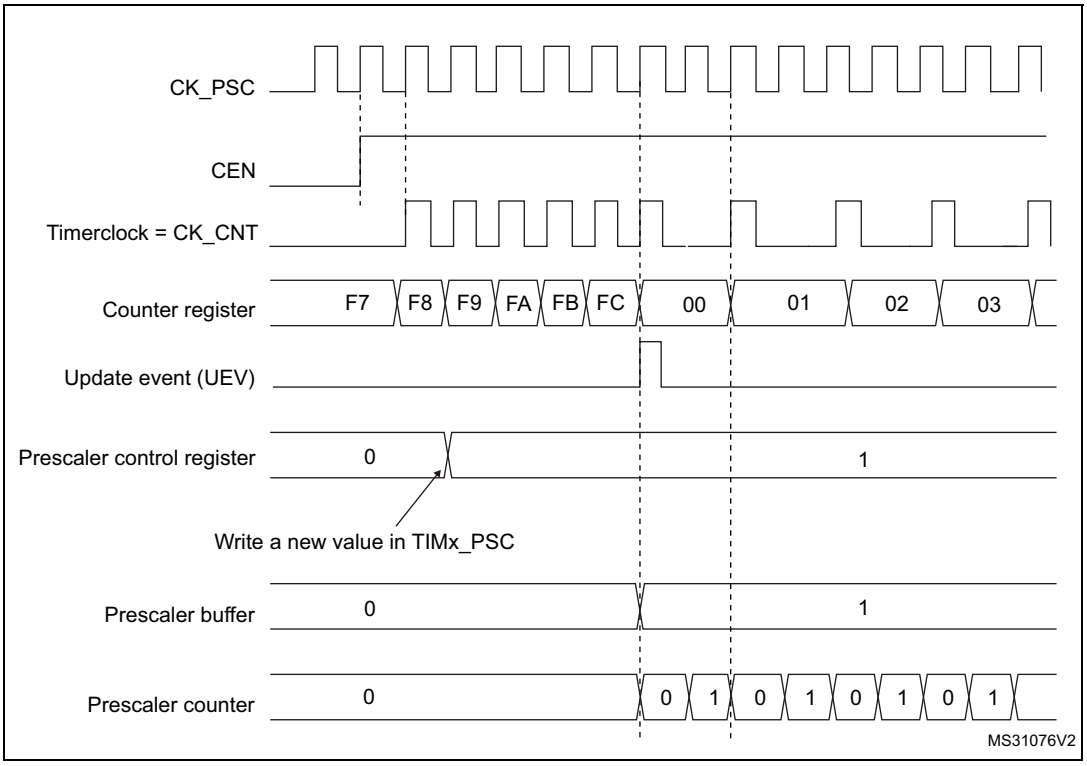
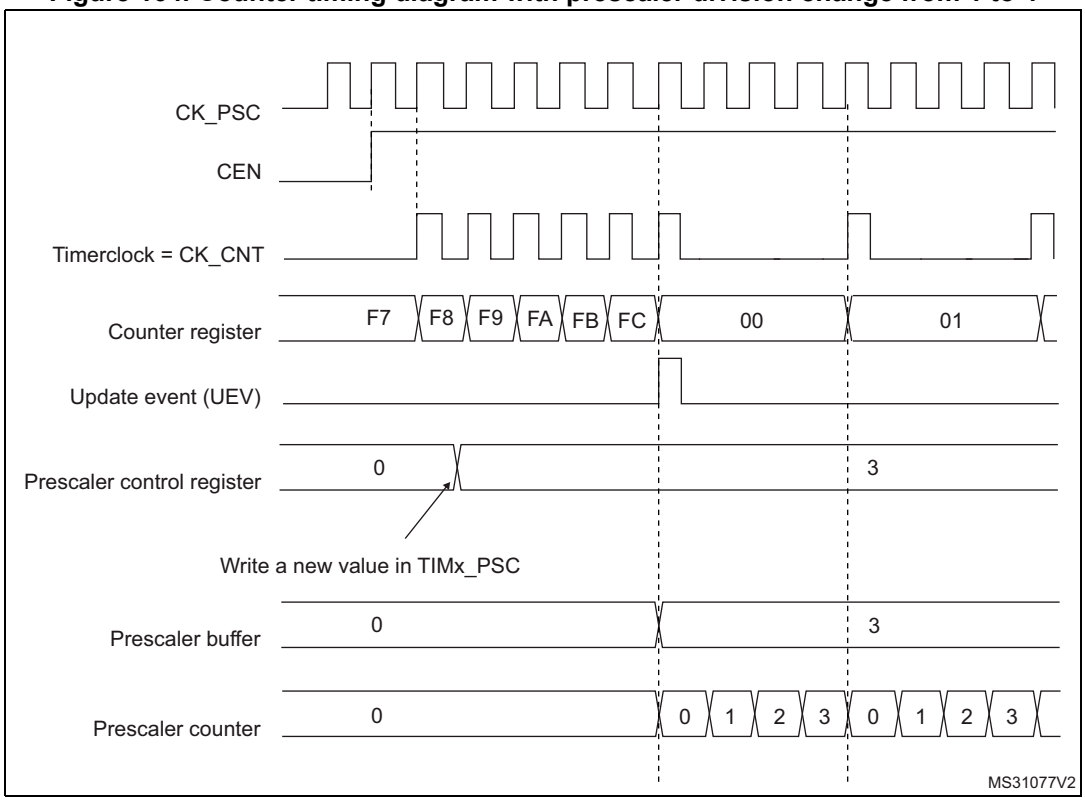


Figure 164. Counter timing diagram with prescaler division change from 1 to 4



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.

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 165. Counter timing diagram, internal clock divided by 1

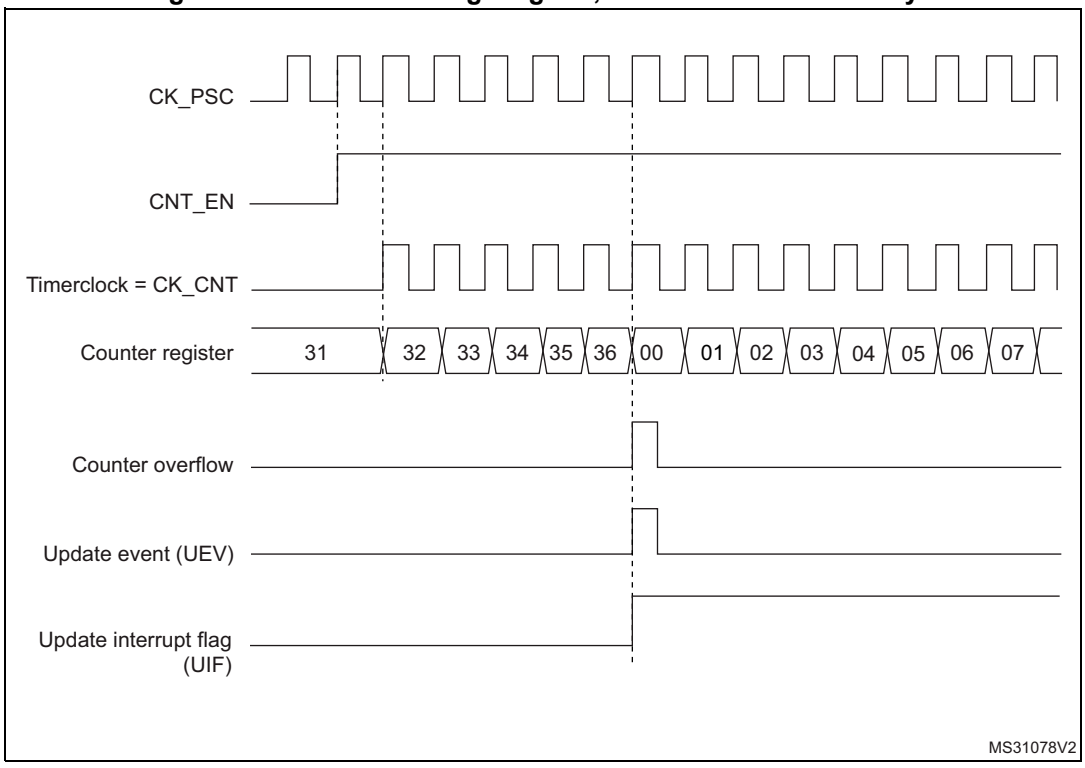


Figure 166. Counter timing diagram, internal clock divided by 2

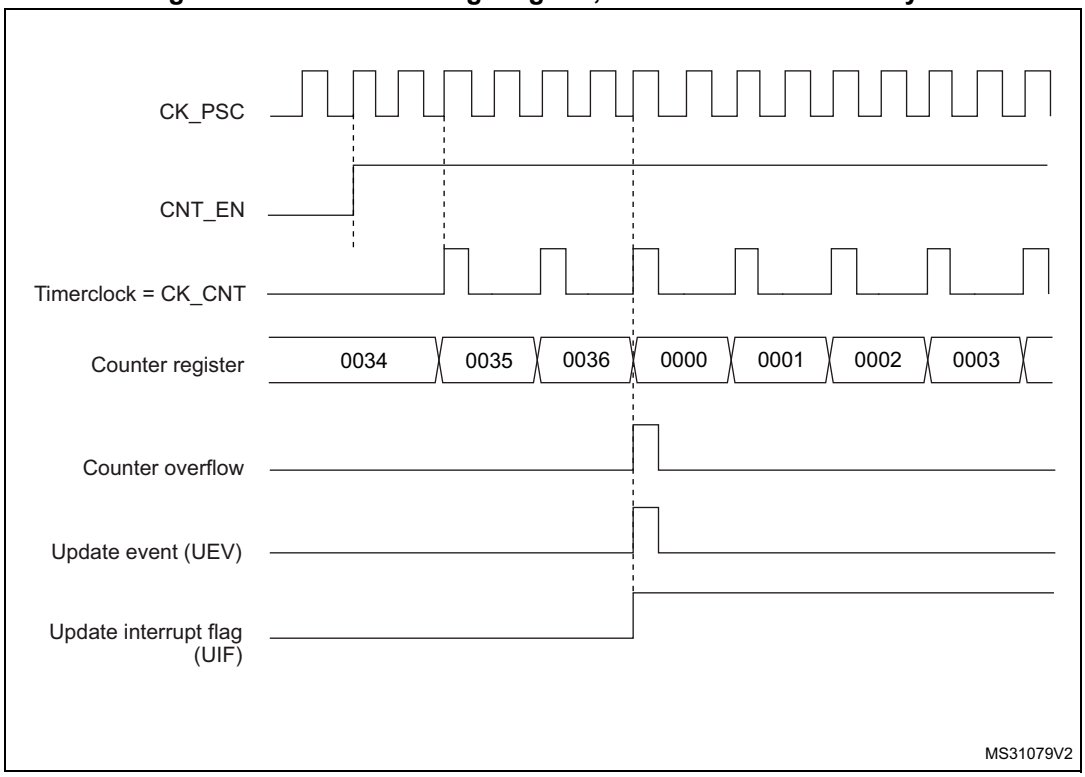


Figure 167. Counter timing diagram, internal clock divided by 4

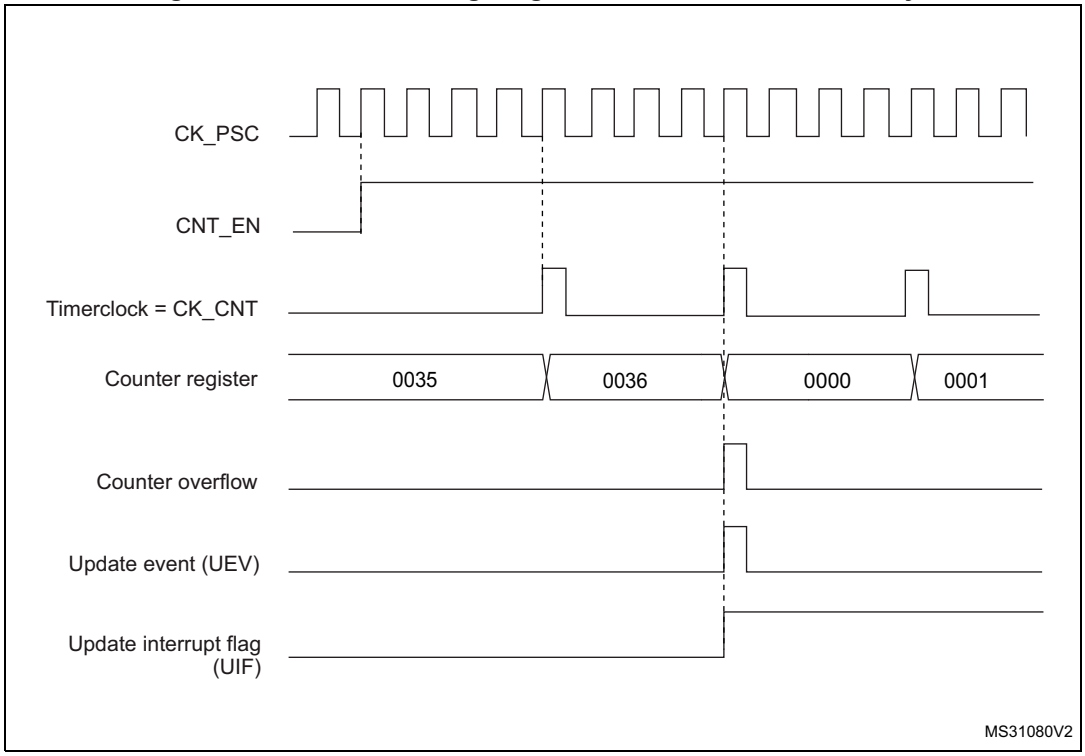


Figure 168. Counter timing diagram, internal clock divided by N

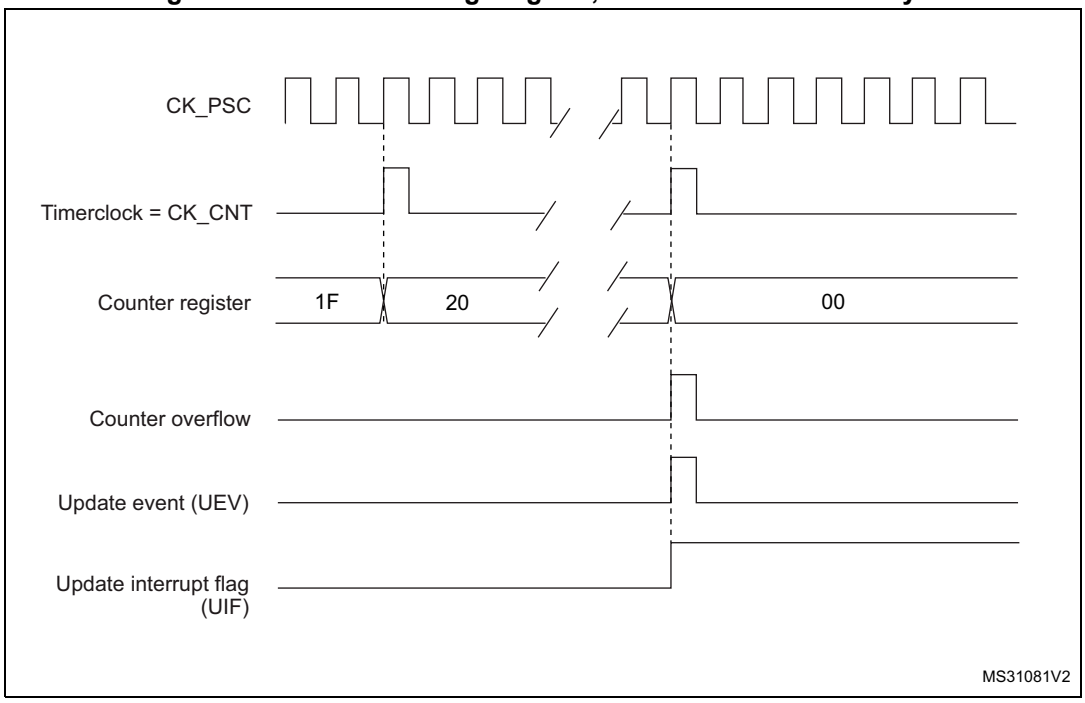


Figure 169. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

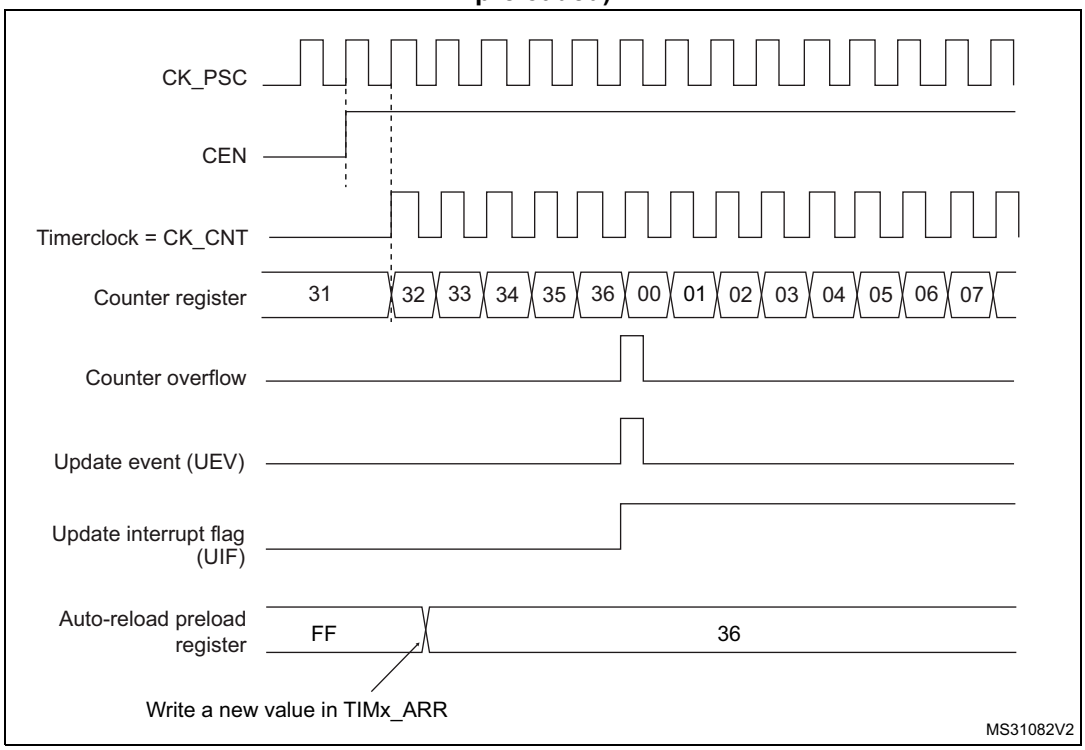
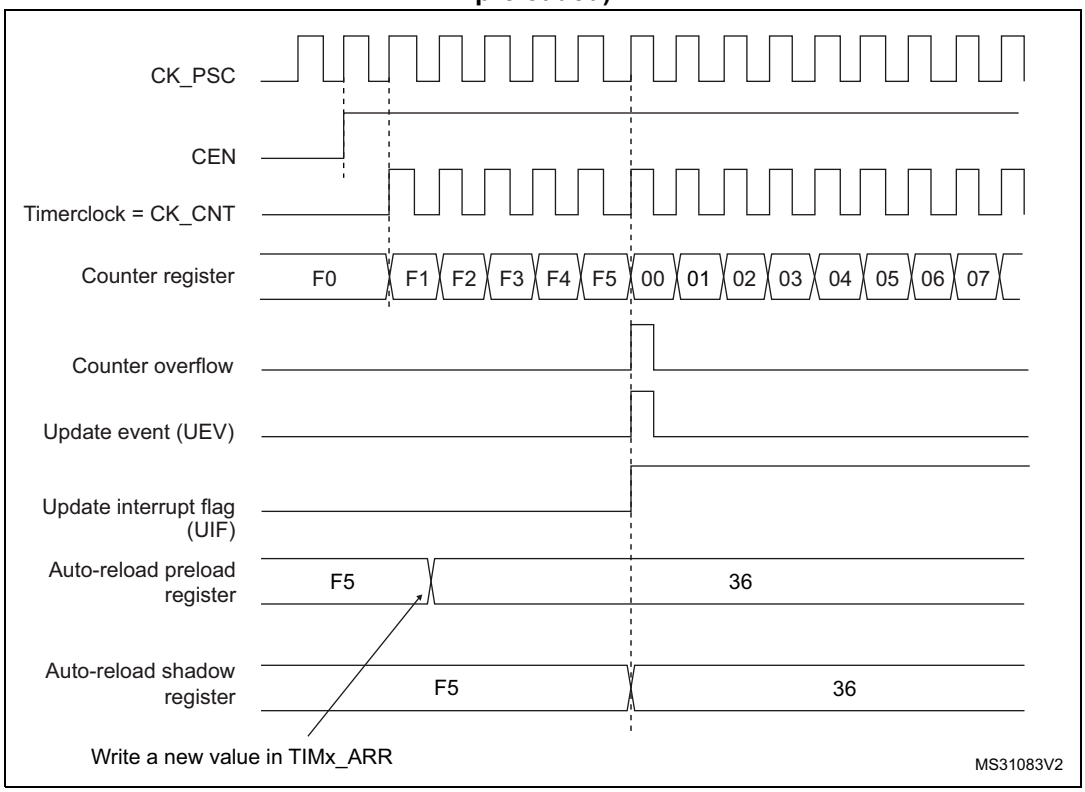


Figure 170. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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 171. Counter timing diagram, internal clock divided by 1

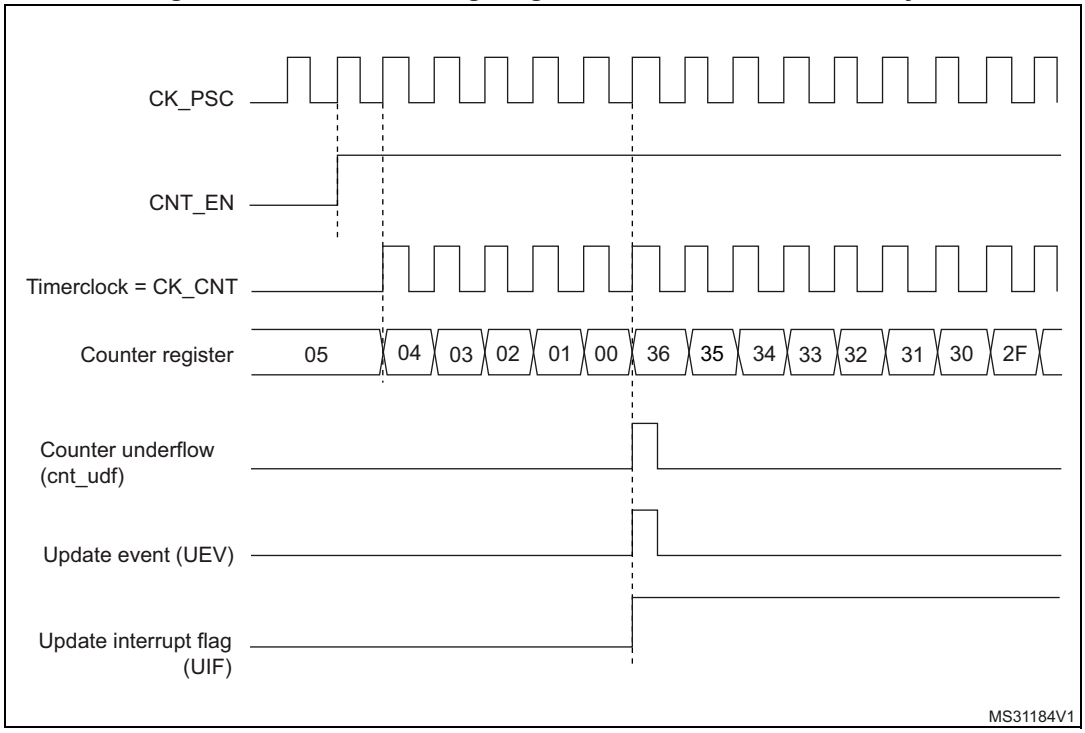


Figure 172. Counter timing diagram, internal clock divided by 2

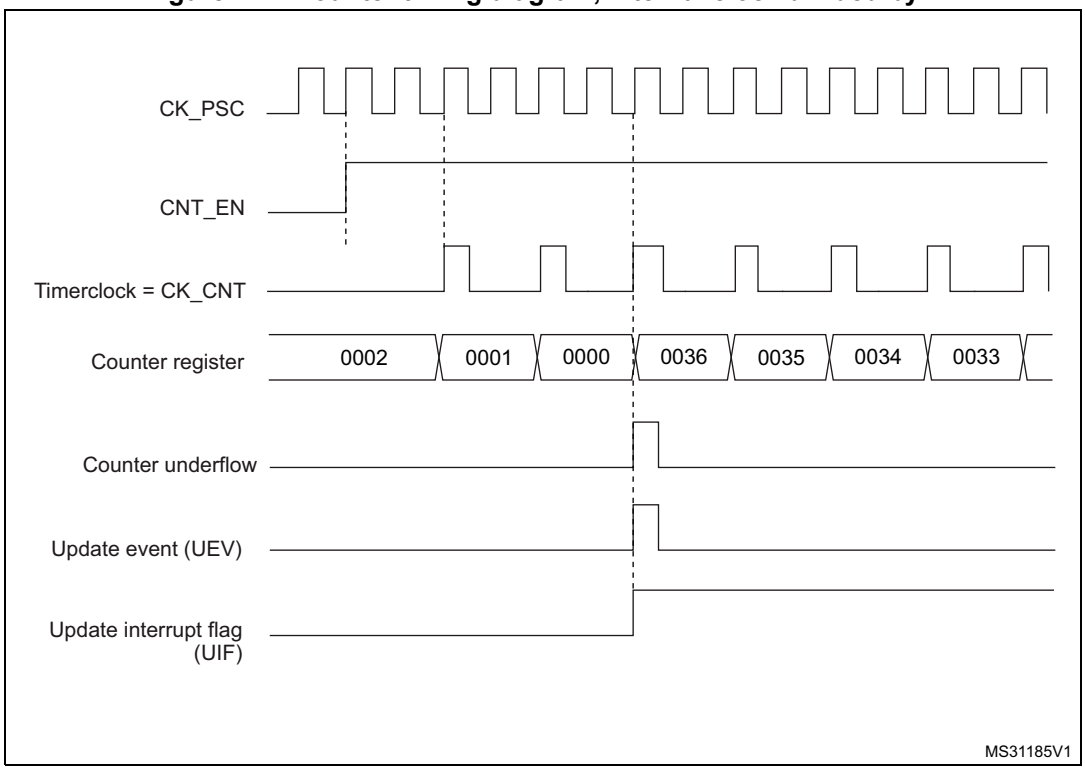


Figure 173. Counter timing diagram, internal clock divided by 4

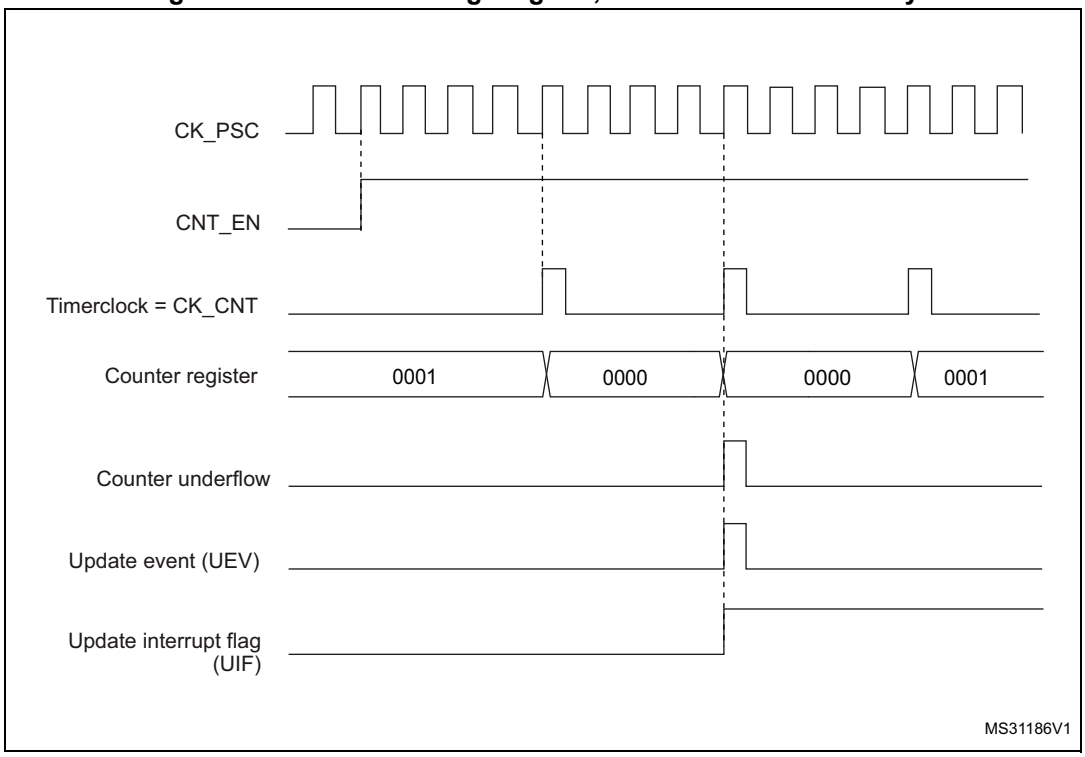


Figure 174. Counter timing diagram, internal clock divided by N

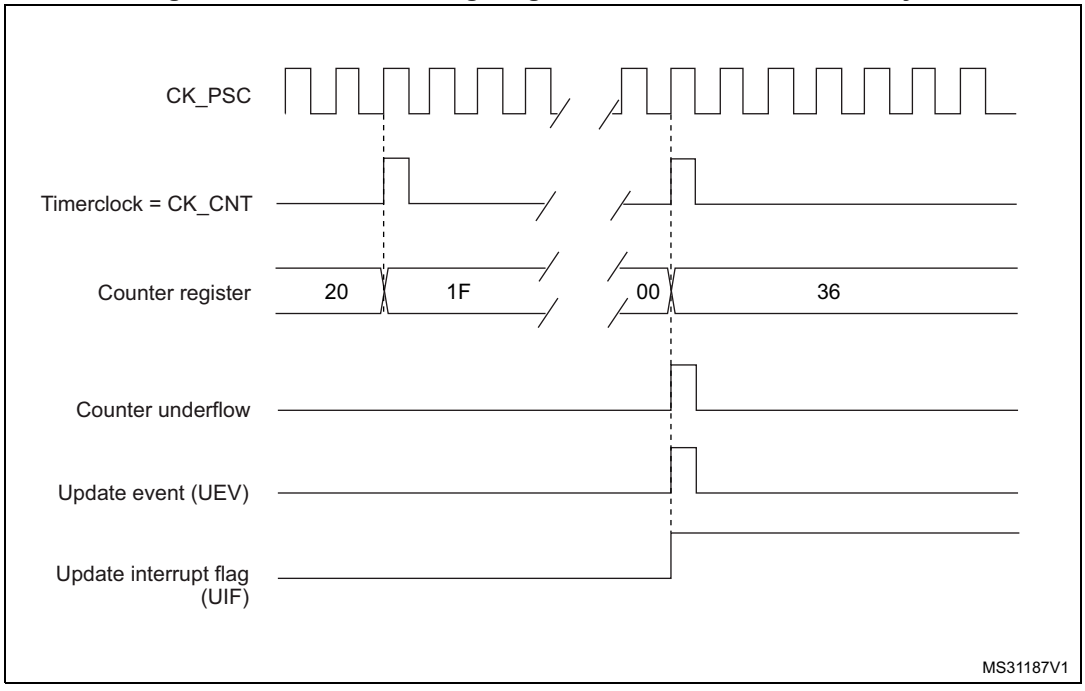
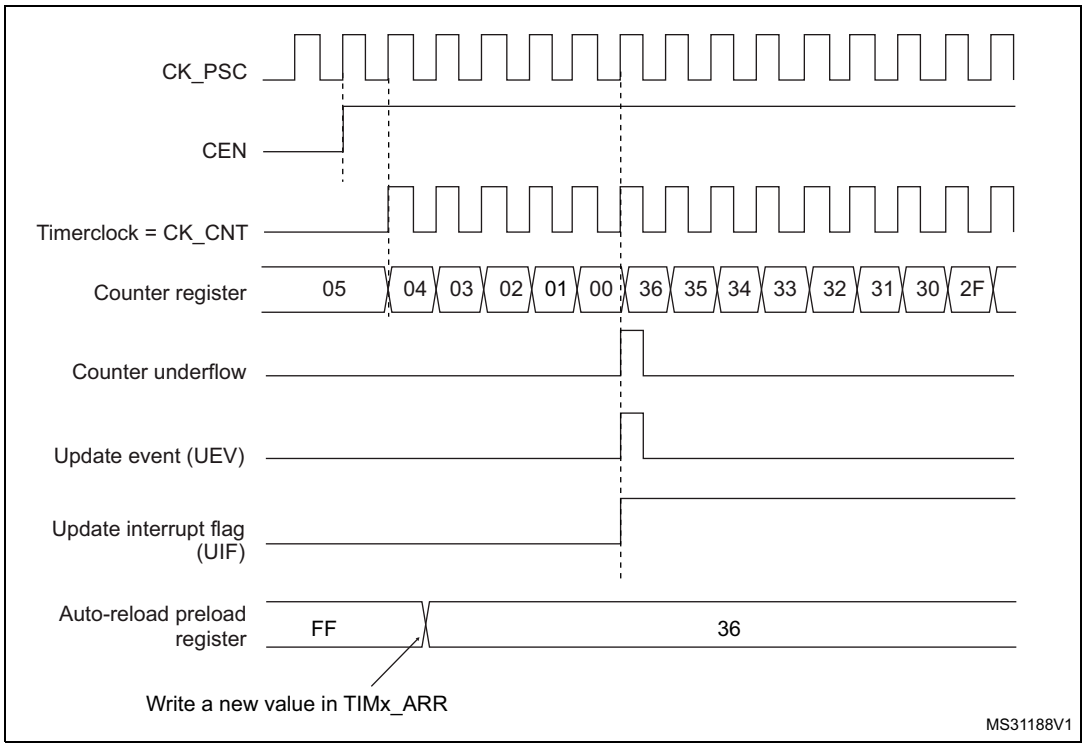


Figure 175. Counter timing diagram, Update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

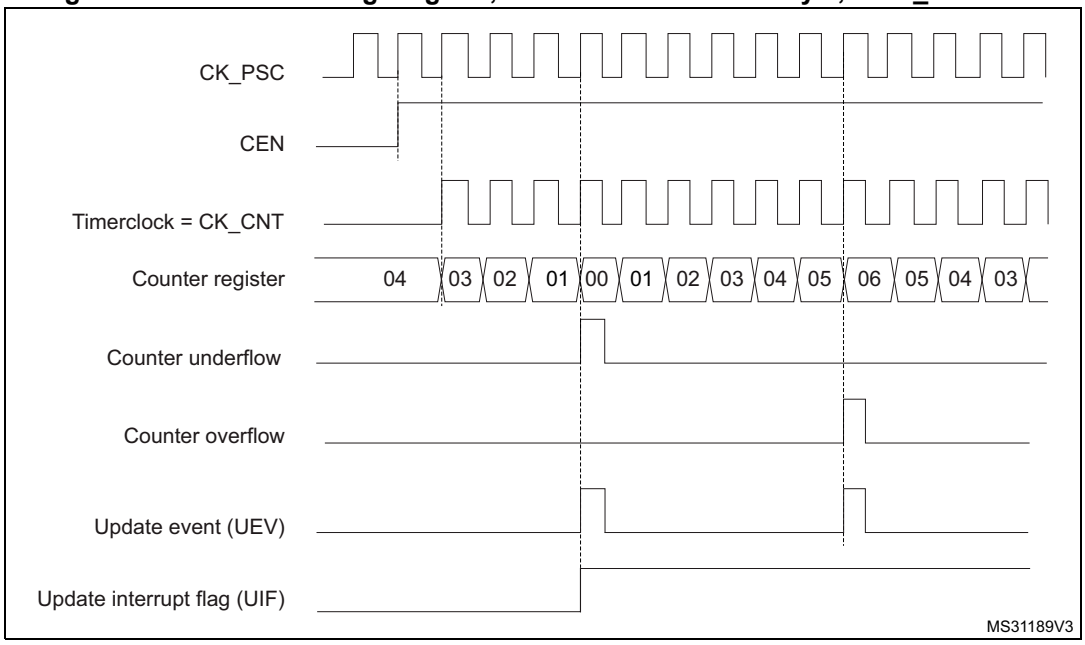
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 176. 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 19.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 3) on page 524*).

Figure 177. Counter timing diagram, internal clock divided by 2

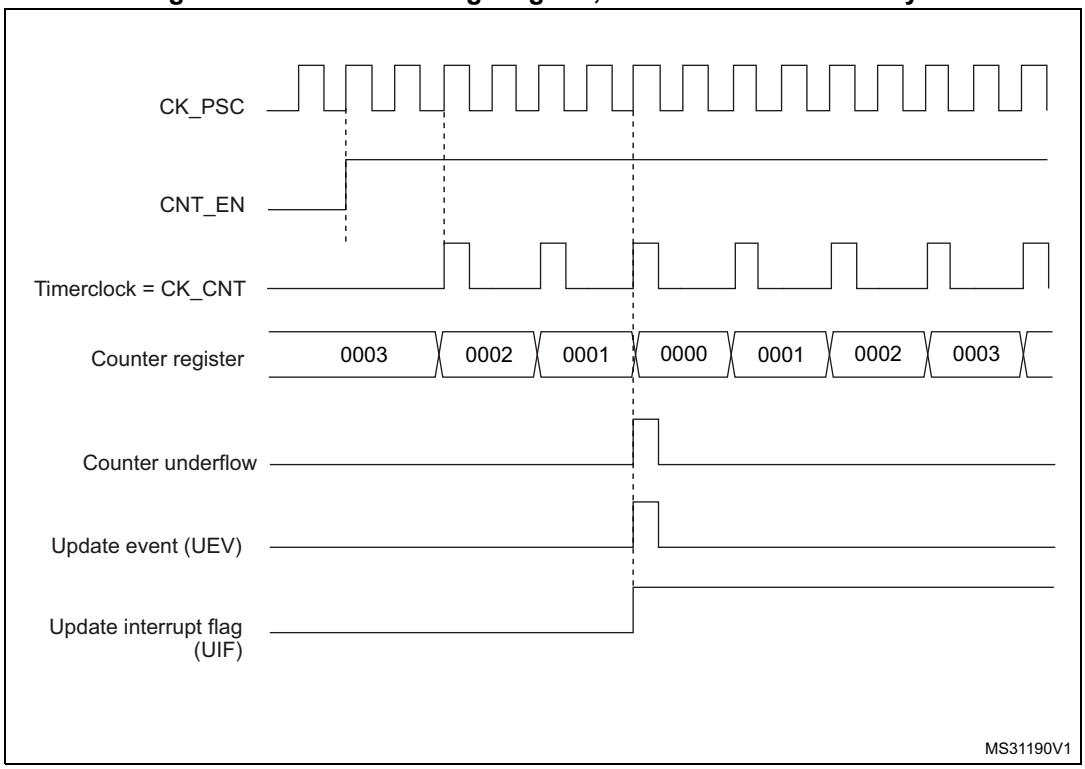
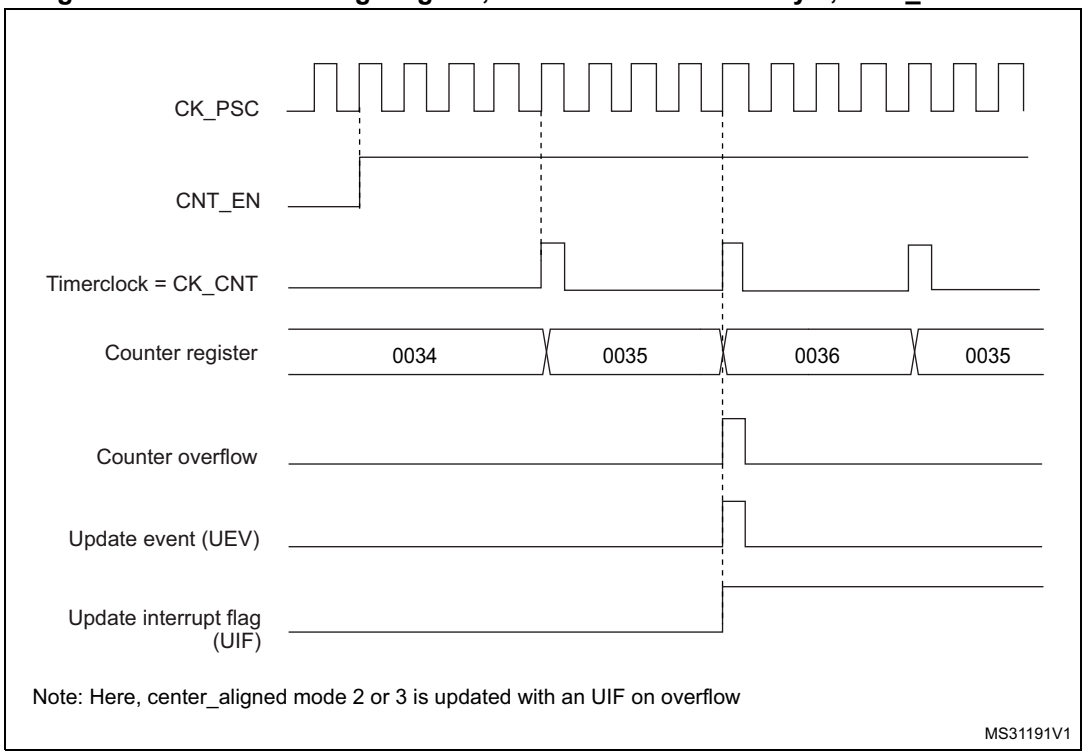


Figure 178. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 179. Counter timing diagram, internal clock divided by N

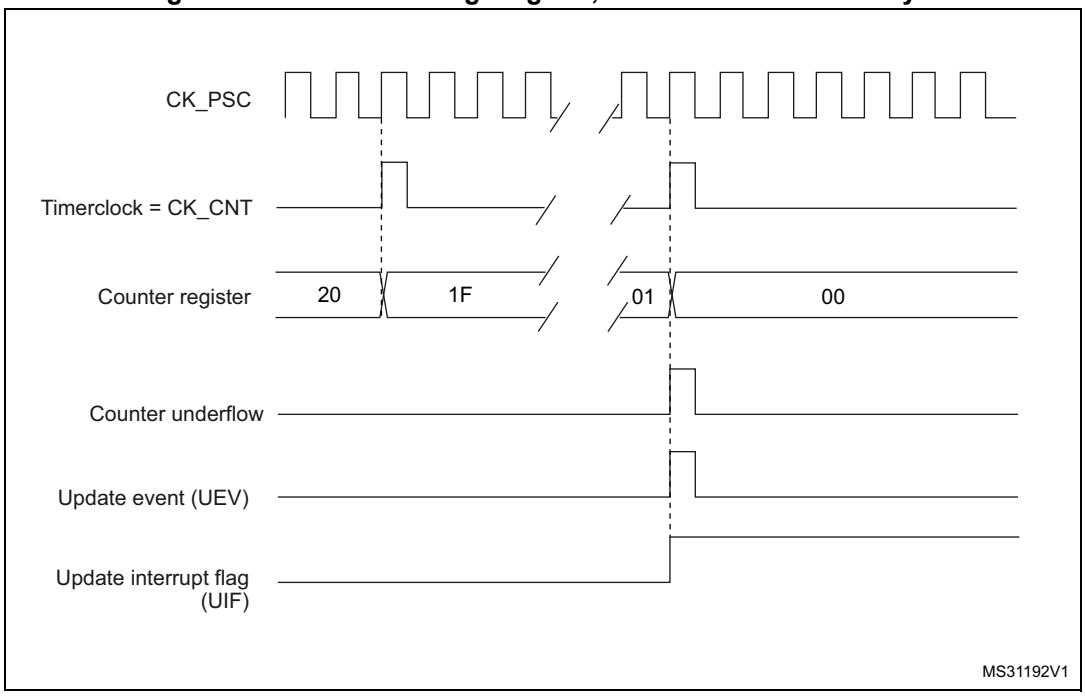


Figure 180. Counter timing diagram, Update event with ARPE=1 (counter underflow)

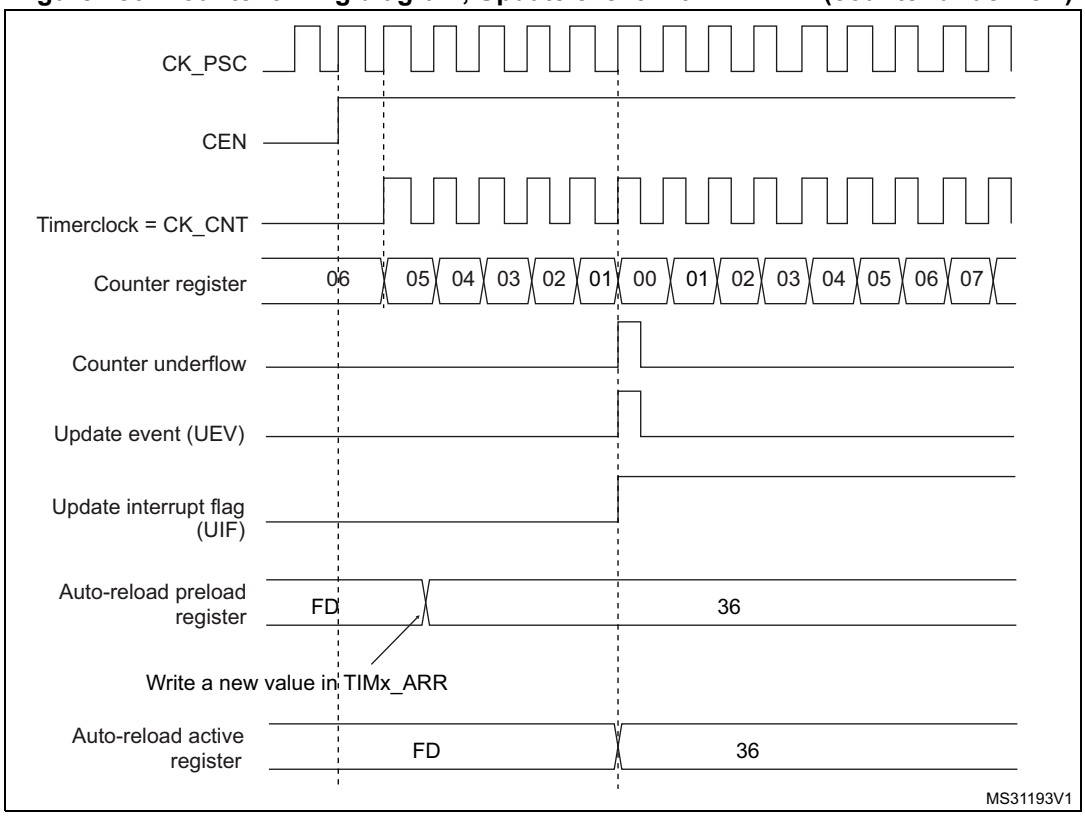
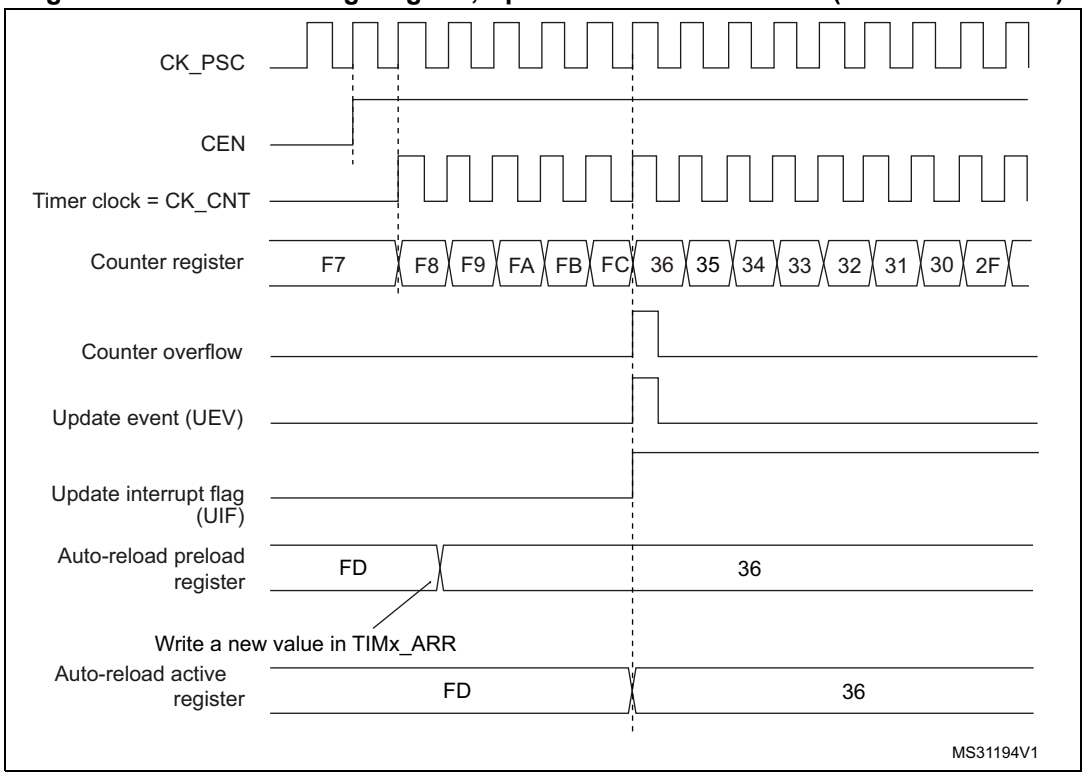


Figure 181. Counter timing diagram, Update event with ARPE=1 (counter overflow)



19.3.3 Clock selection

The counter clock can be provided by the following clock sources:

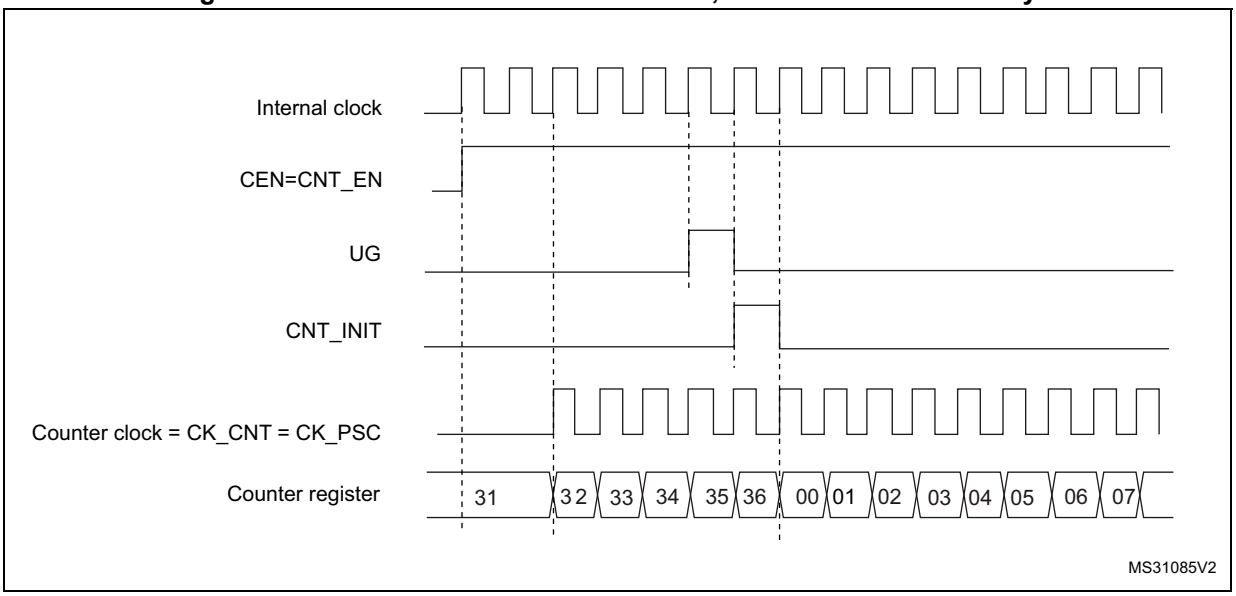
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to *: Using one timer as prescaler for another timer on page 518* 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 182 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

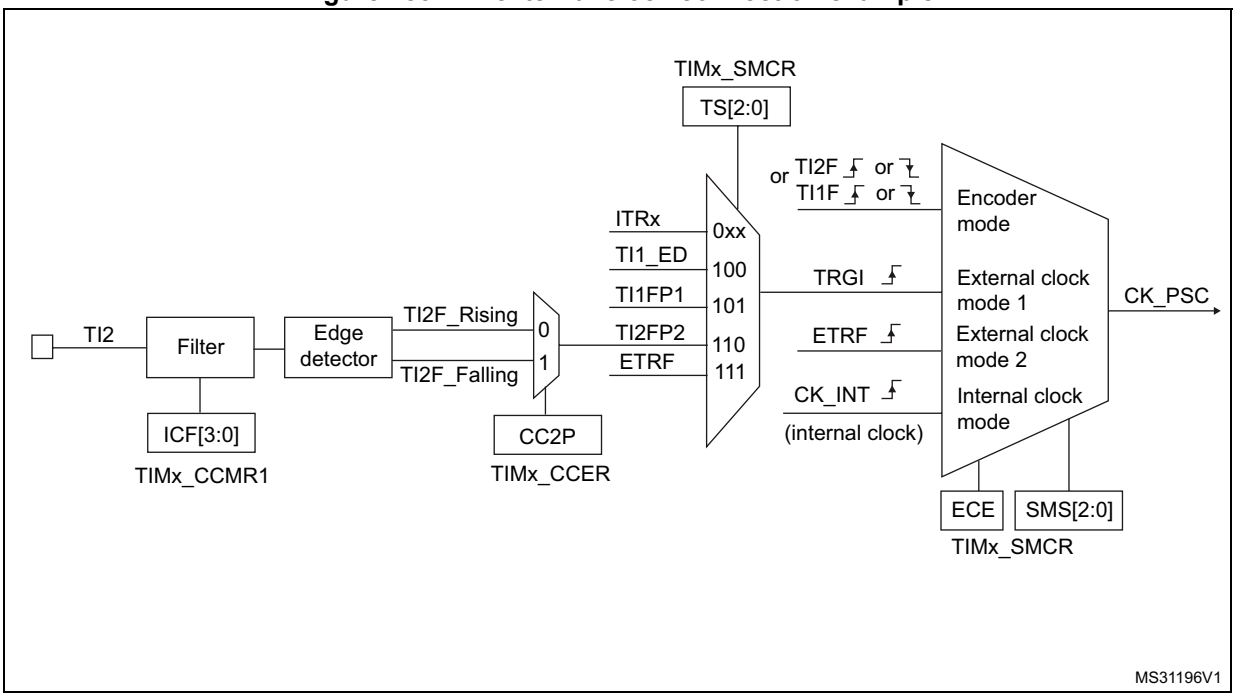
Figure 182. Control circuit in normal mode, internal clock divided by 1



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 183. 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).

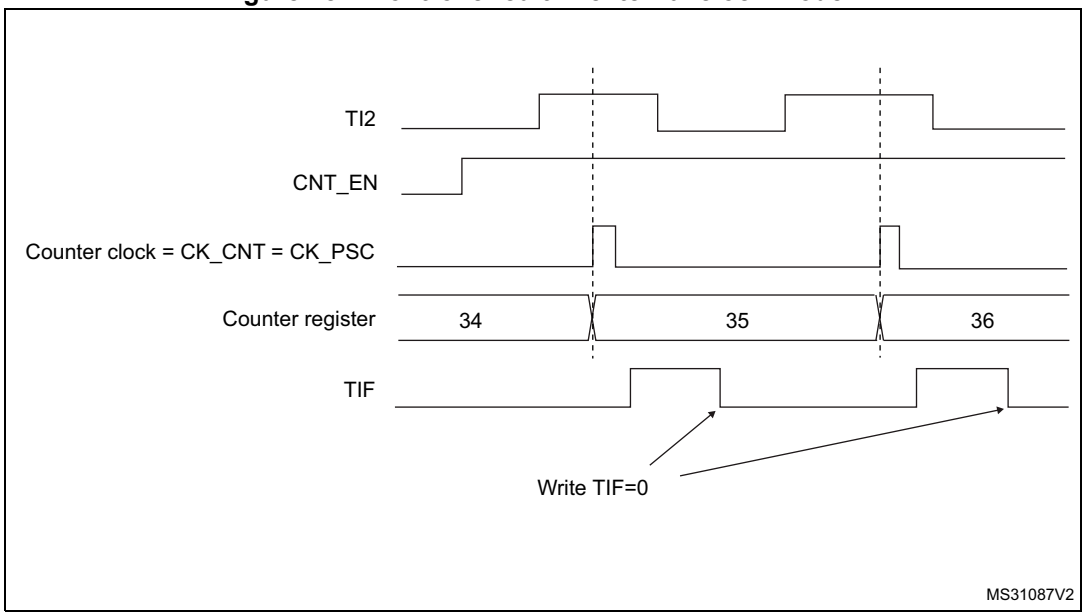
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 184. Control circuit in external clock mode 1



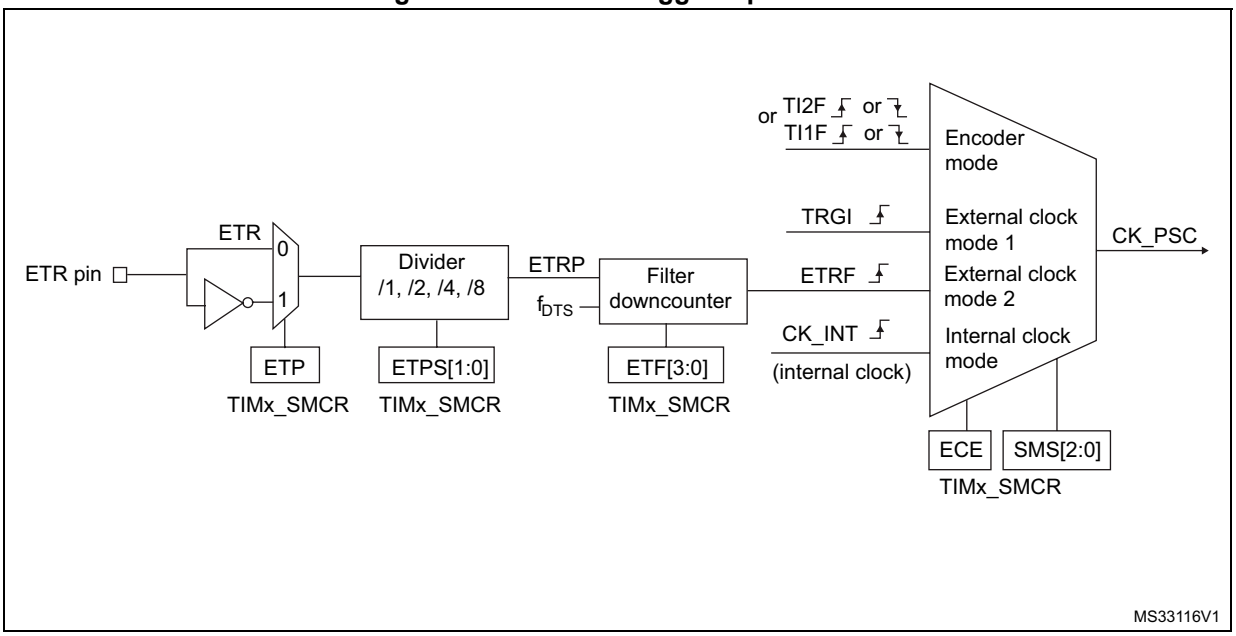
External clock source mode 2

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 185 gives an overview of the external trigger input block.

Figure 185. External trigger input block



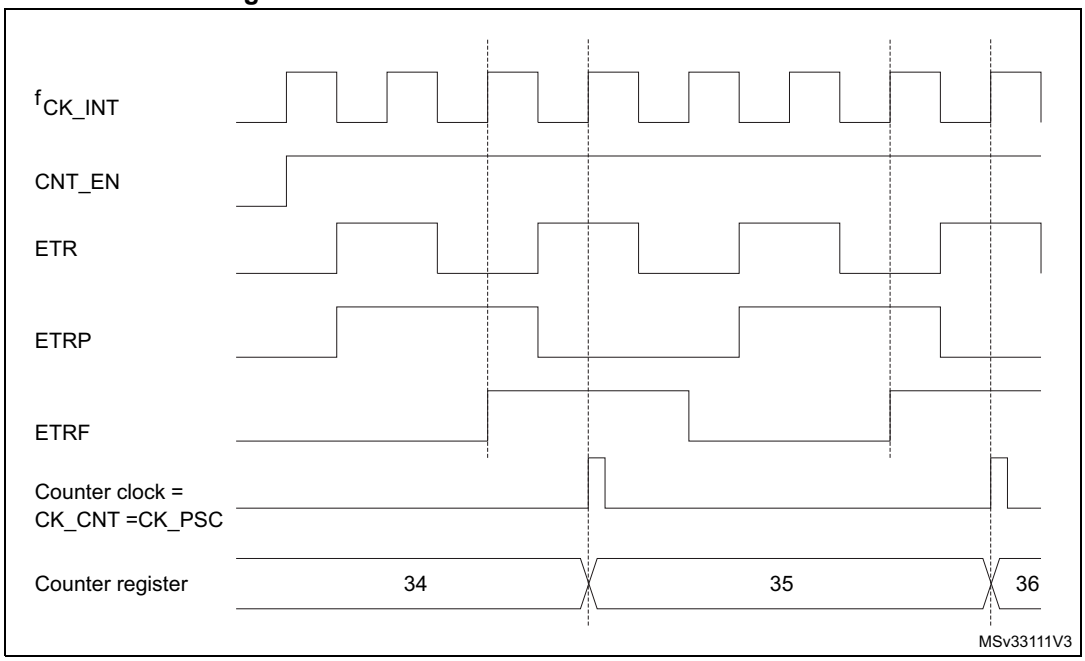
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 186. Control circuit in external clock mode 2



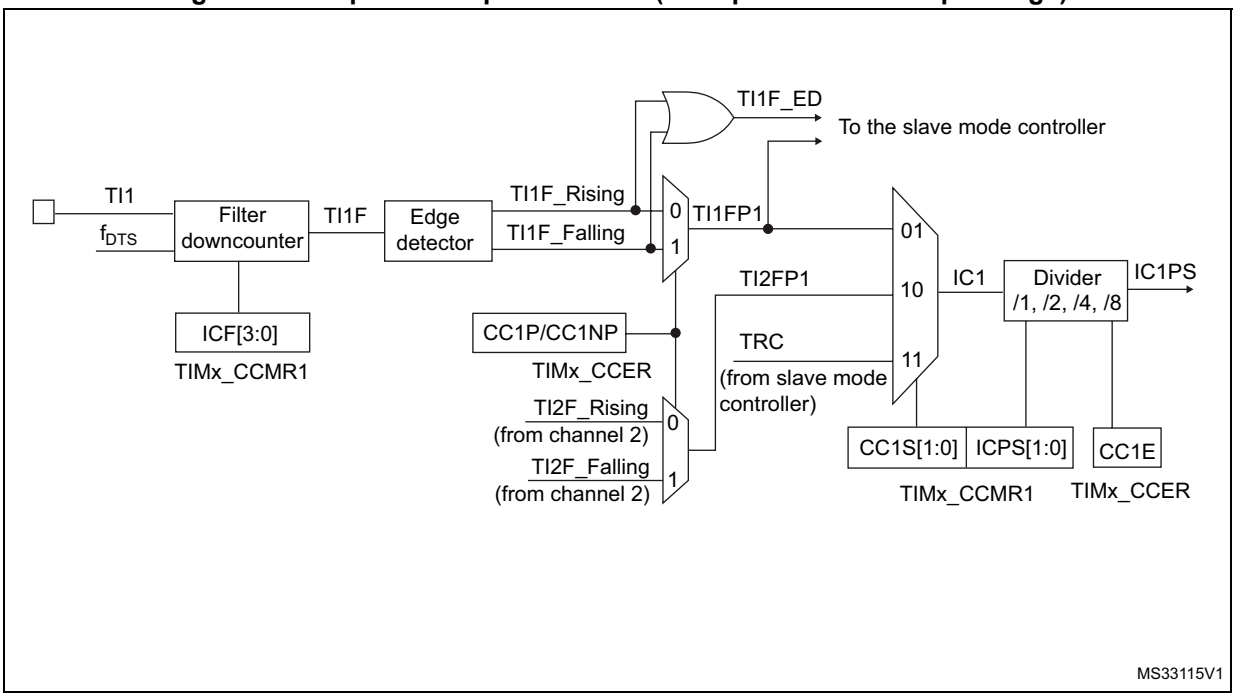
19.3.4 Capture/Compare channels

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).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 187. 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 188. Capture/Compare channel 1 main circuit

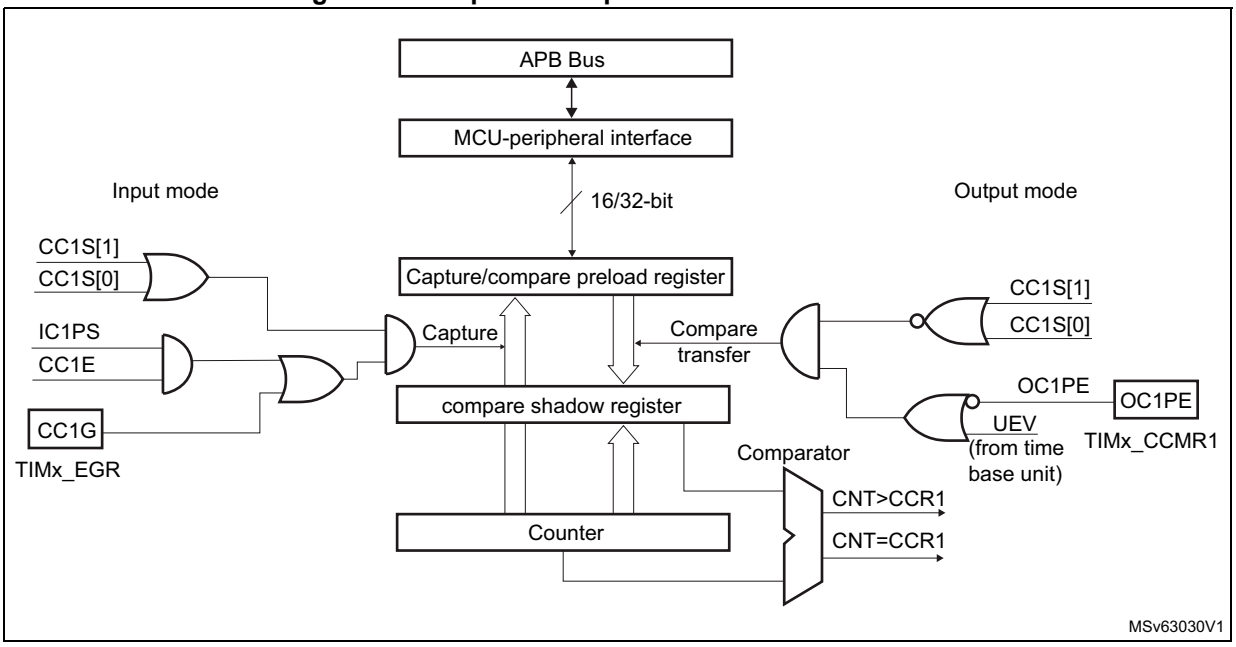
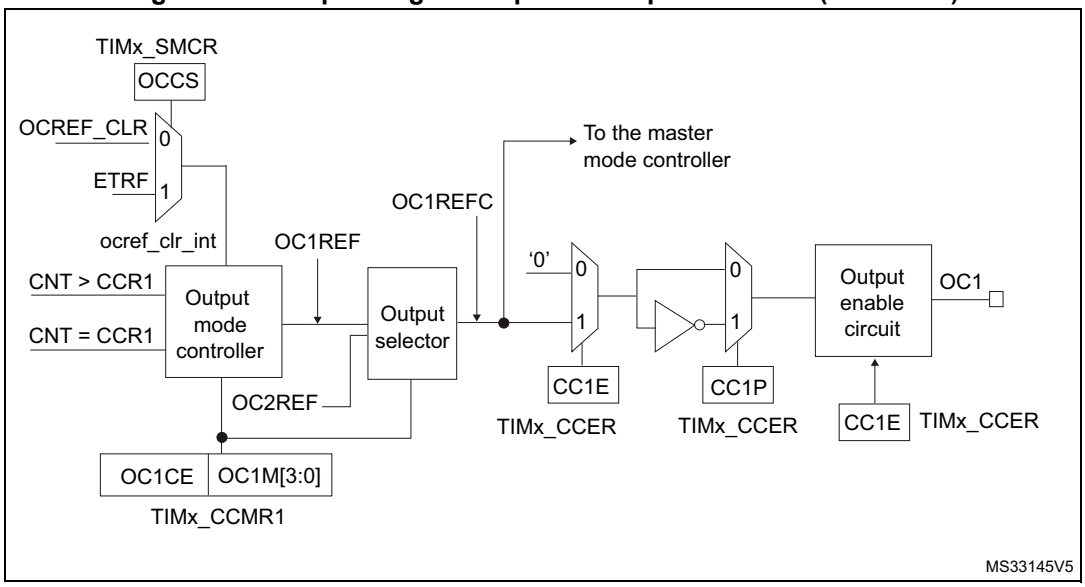


Figure 189. Output stage of Capture/Compare channel (channel 1)



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 it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at 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 writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

19.3.6 PWM input mode

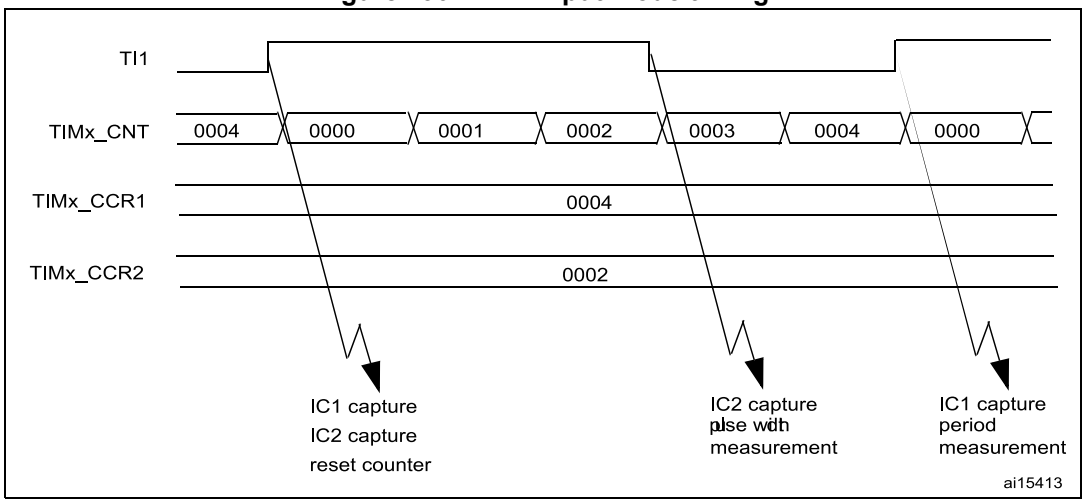
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1 in the TIMx_CCER register.

Figure 190. 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, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

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:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

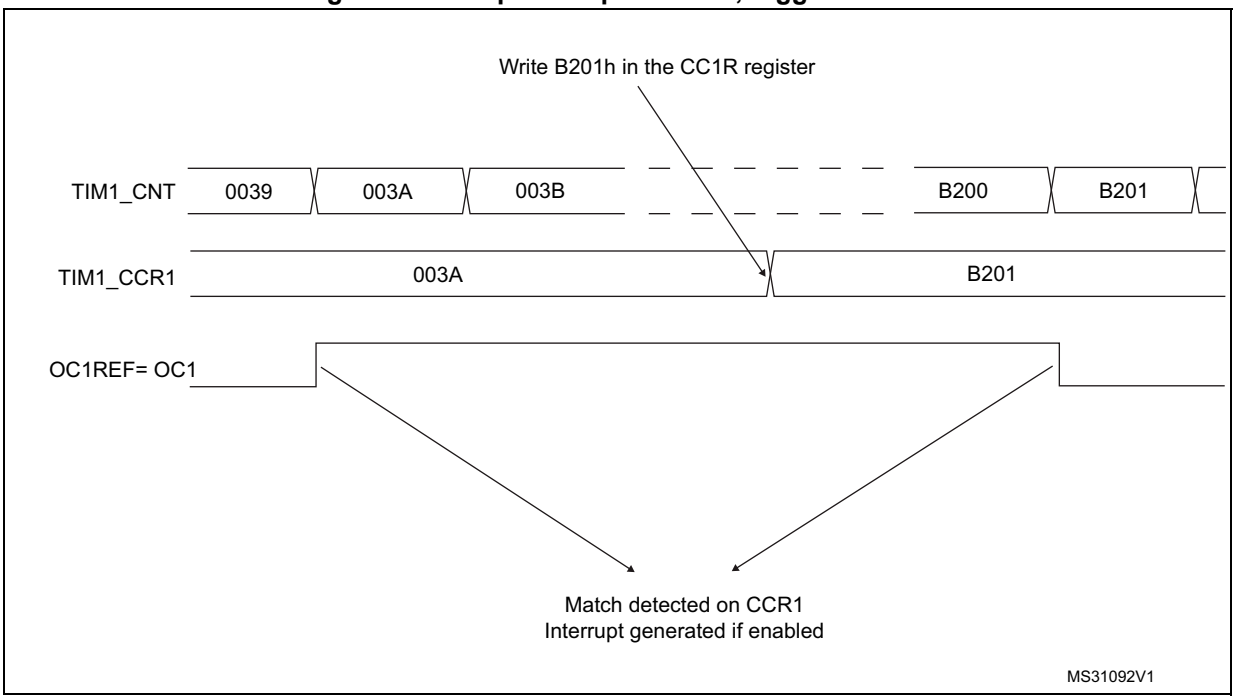
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 191*.

Figure 191. Output compare mode, toggle on OC1



19.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

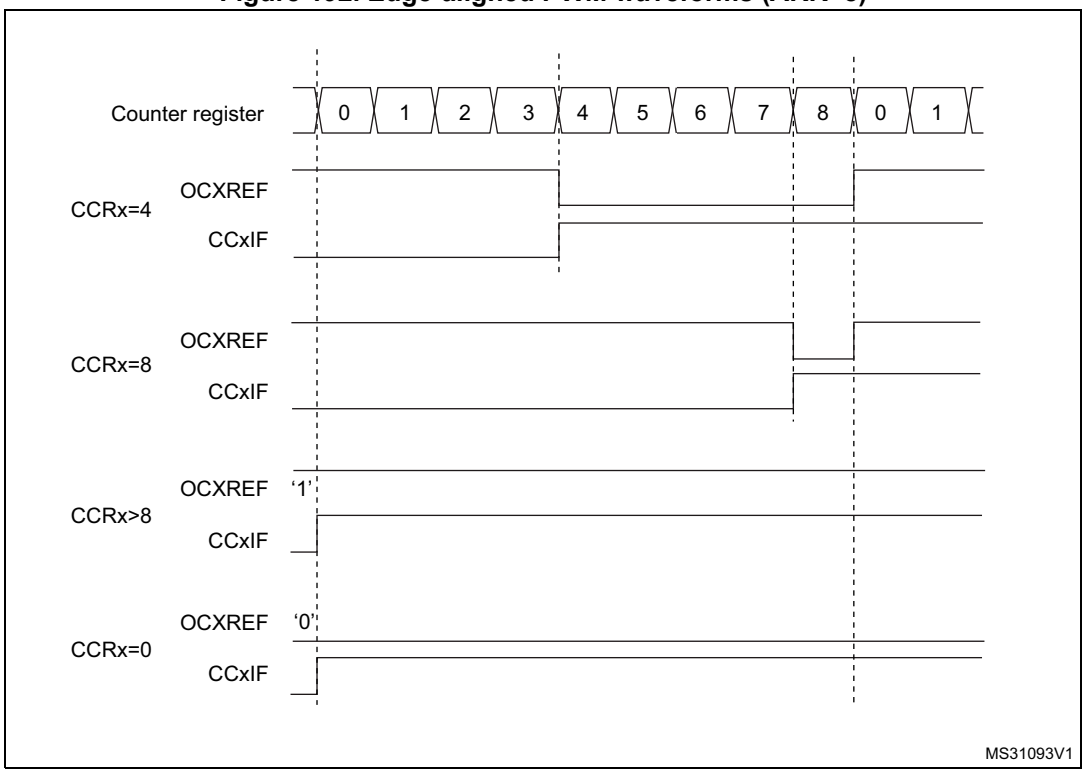
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 482*.

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 192* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 192. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 485*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

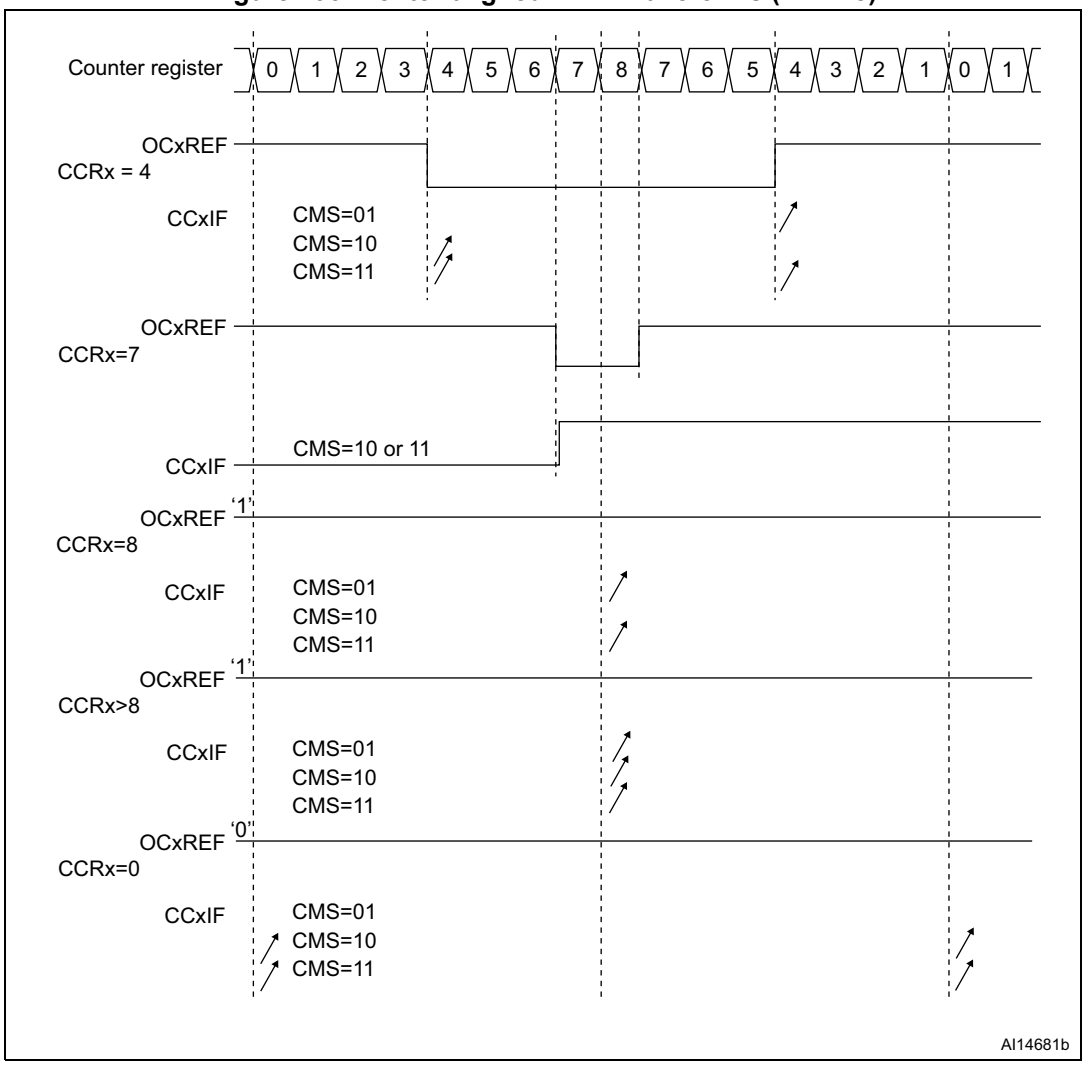
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The

compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 488*.

Figure 193 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 193. 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 a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

19.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

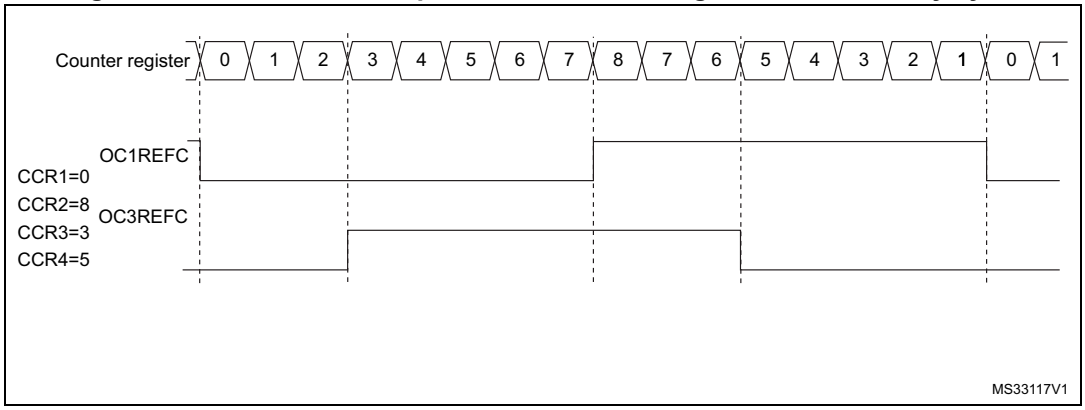
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 194 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

Figure 194. Generation of 2 phase-shifted PWM signals with 50% duty cycle



19.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2

– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

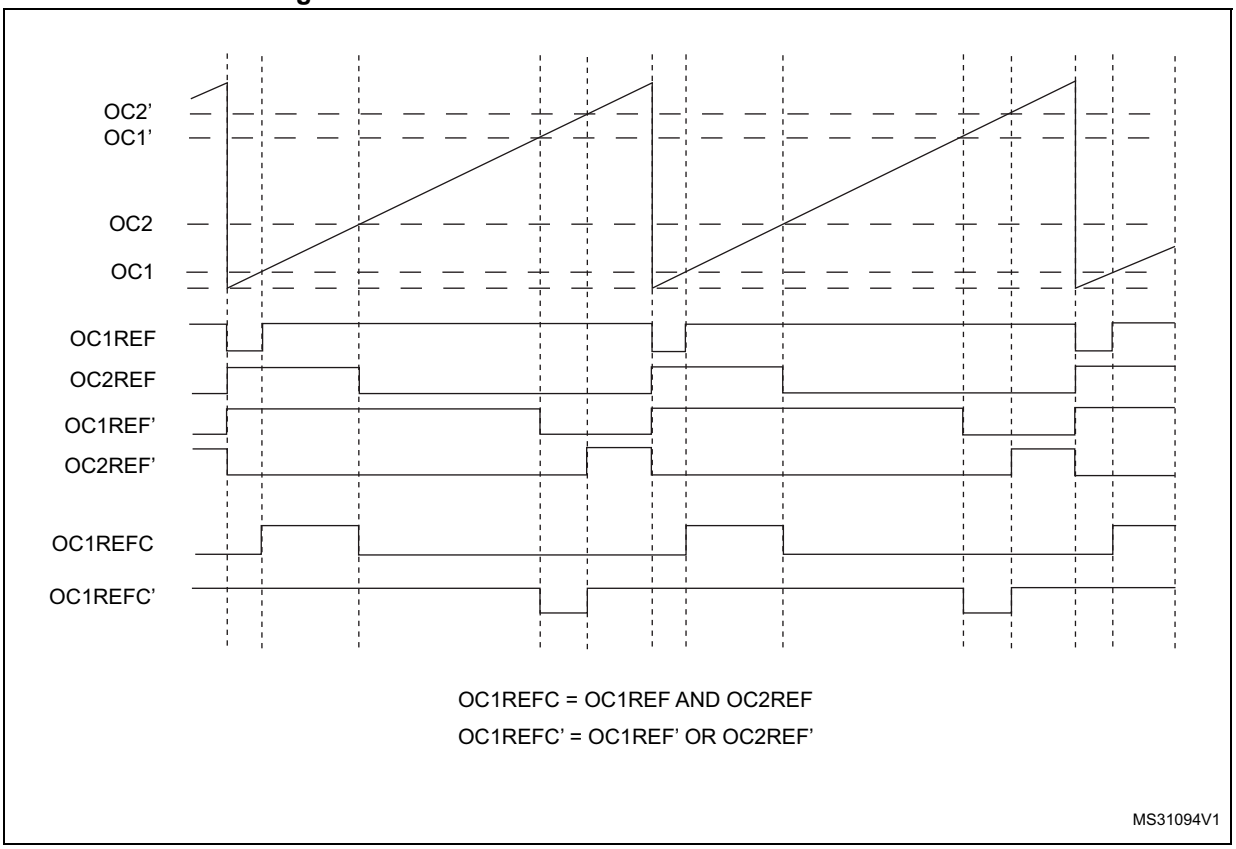
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 195 shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 195. Combined PWM mode on channels 1 and 3



19.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

OCREF_CLR_INPUT can be selected between the OCREF_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

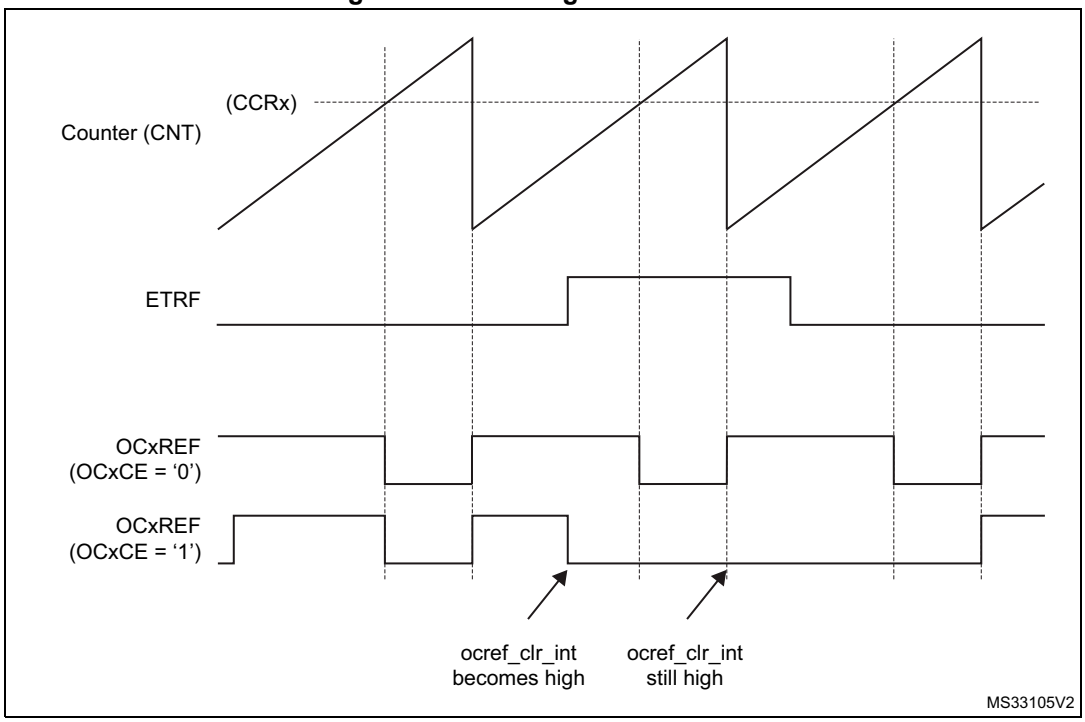
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 196 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 196. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

19.3.13 One-pulse mode

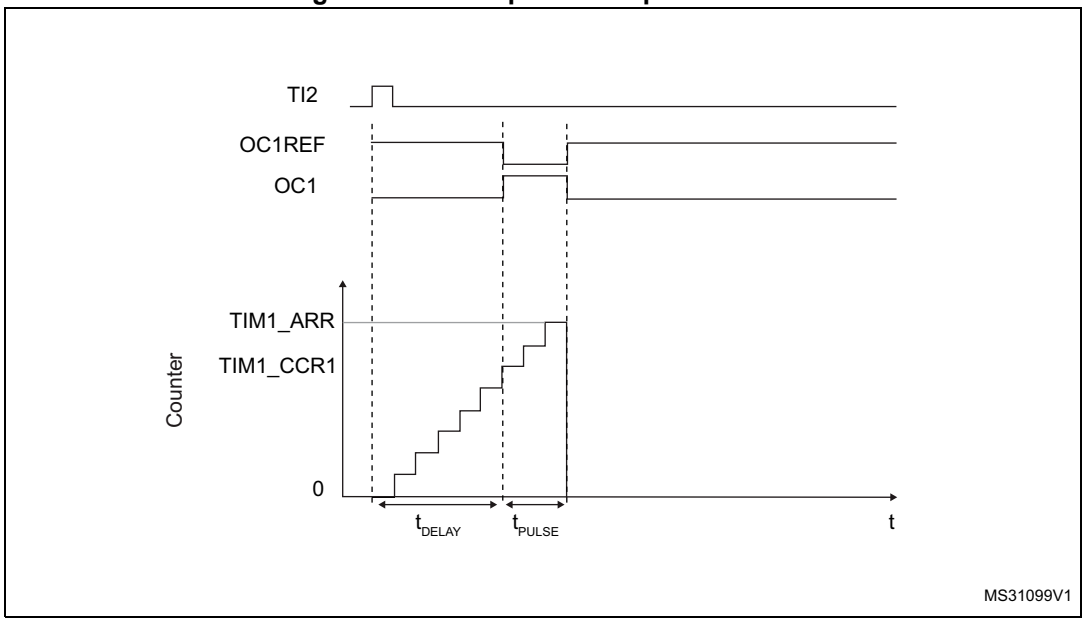
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 197. Example of one-pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

19.3.14 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 19.3.13*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

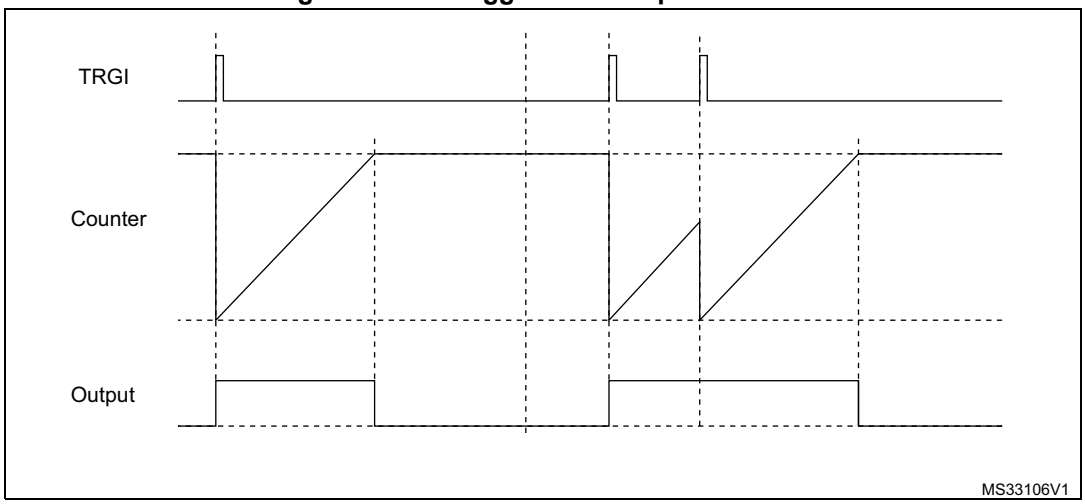
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 198. Retriggerable one-pulse mode.



19.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 71*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 71. 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 199 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 199. Example of counter operation in encoder interface mode

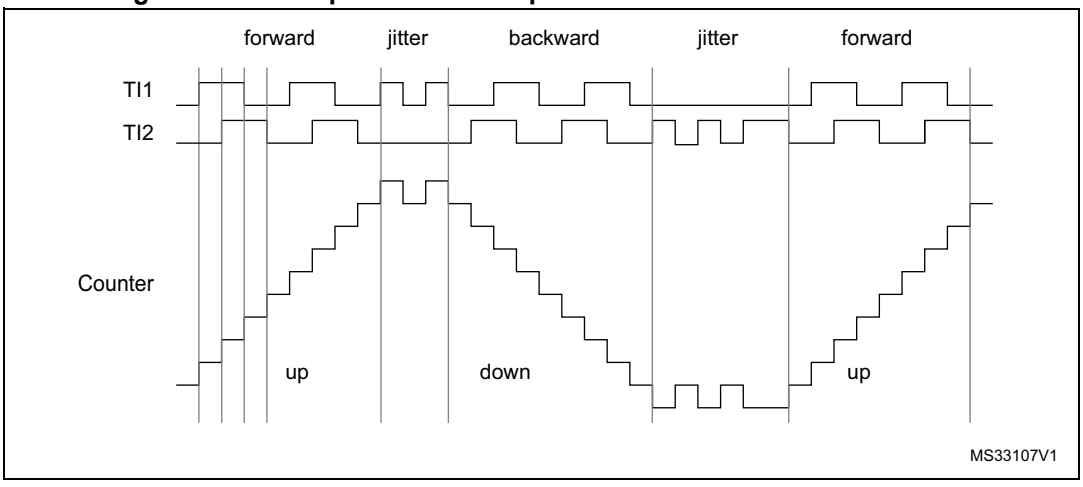
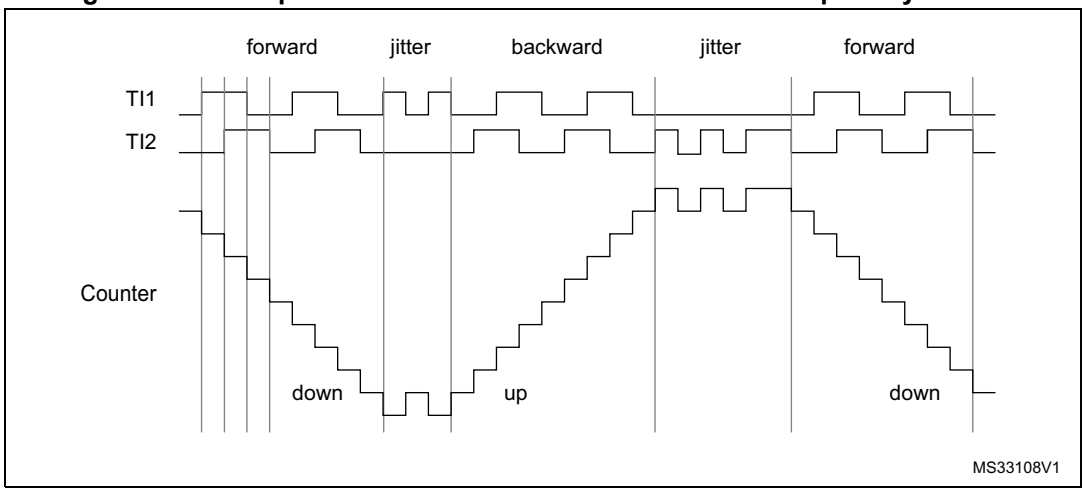


Figure 200 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 200. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

19.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

19.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 18.3.24: Interfacing with Hall sensors on page 434*.

19.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

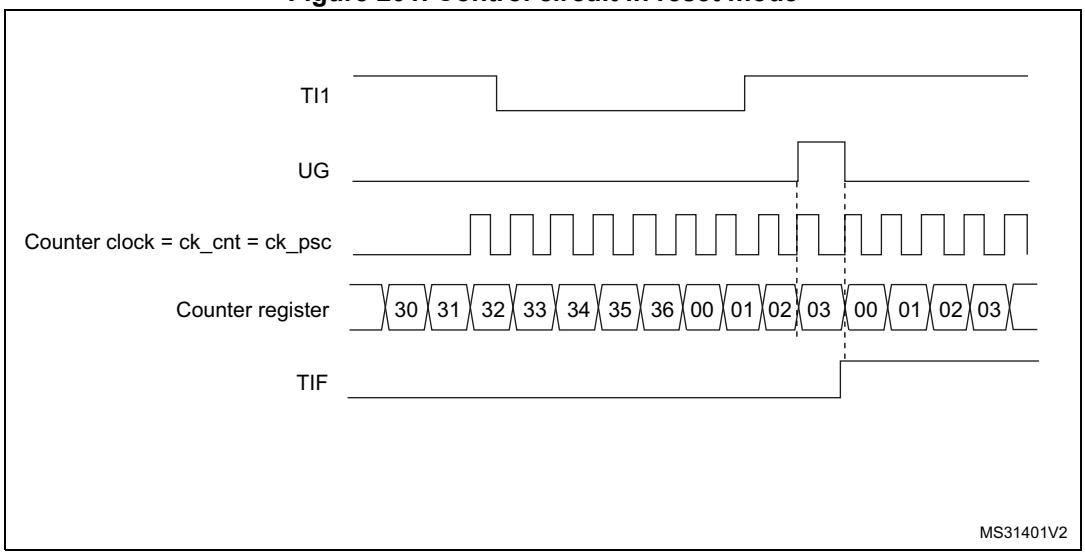
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 201. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

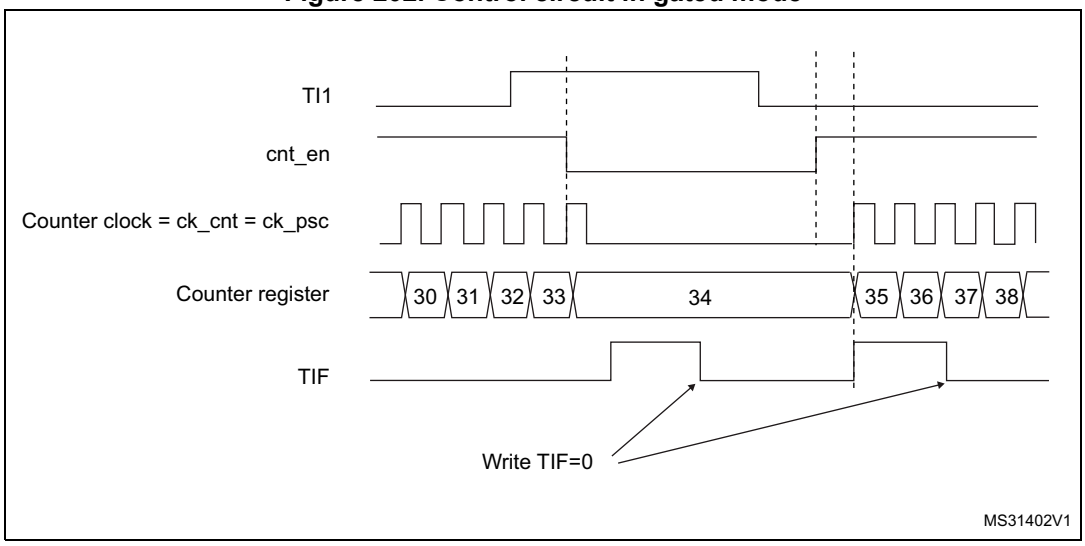
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=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 202. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

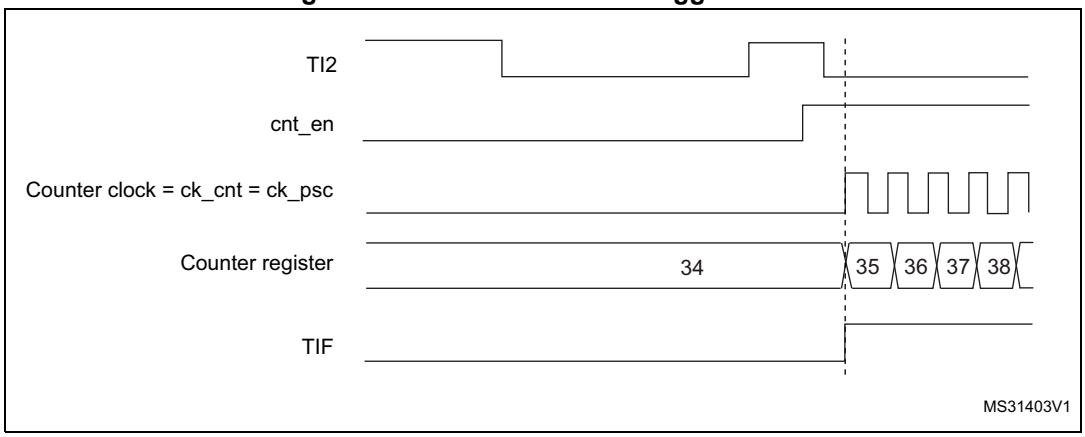
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

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 203. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

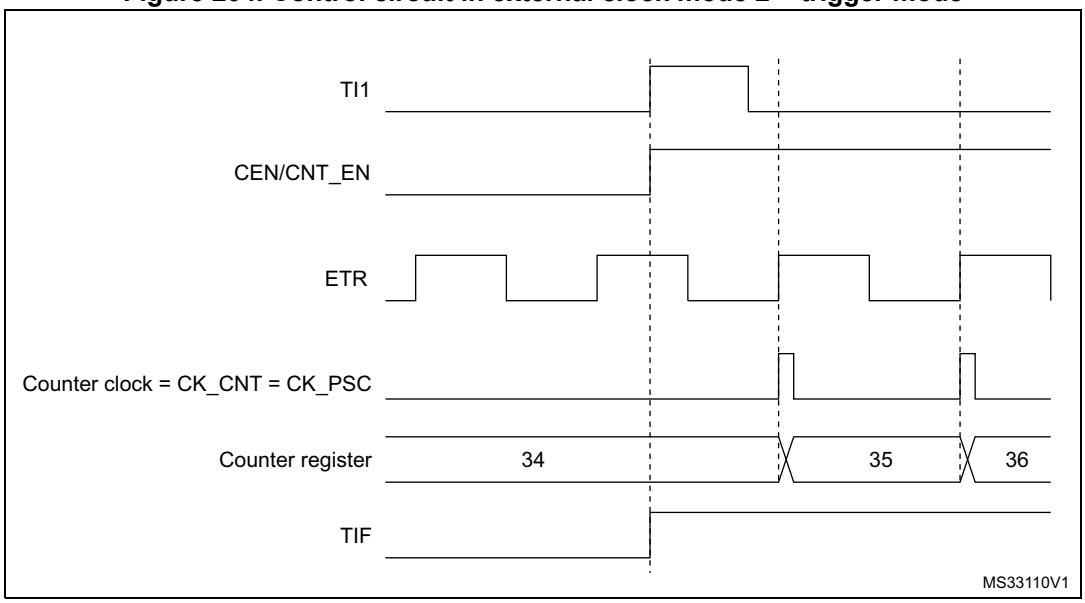
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in 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 204. Control circuit in external clock mode 2 + trigger mode



19.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 205: Master/Slave timer example and *Figure 206: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

Figure 205. Master/Slave timer example

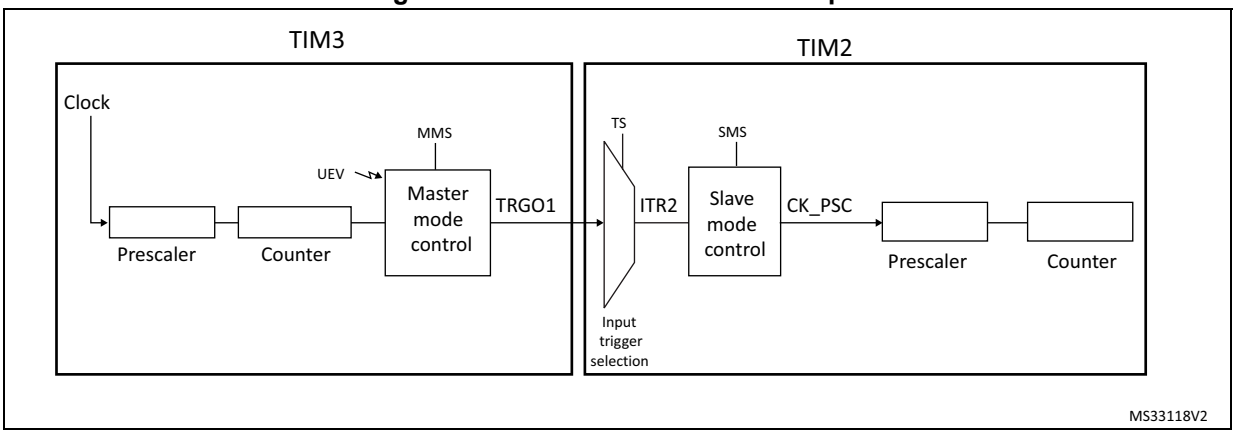
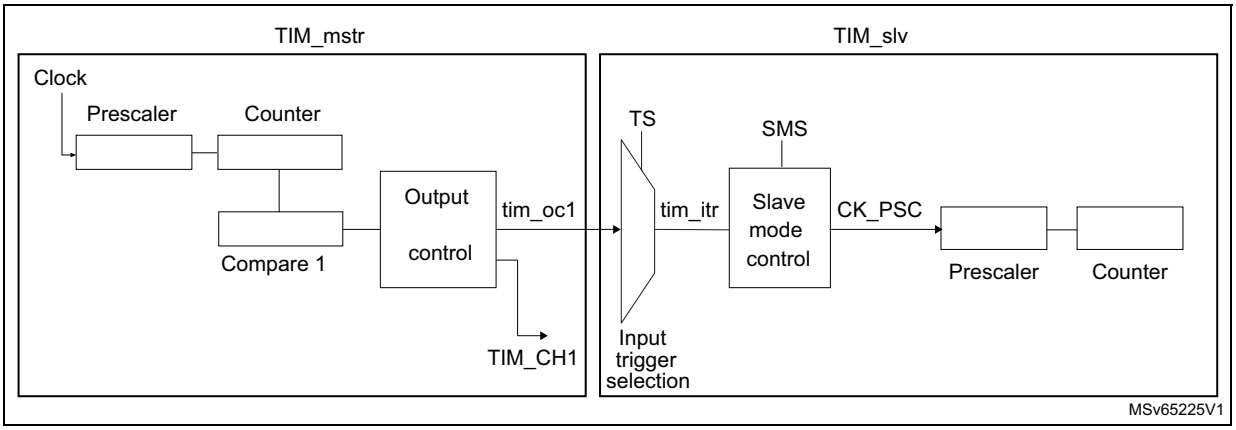


Figure 206. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 206) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.
The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.
For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM3 can be configured to act as a prescaler for TIM2. Refer to *Figure 205*. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

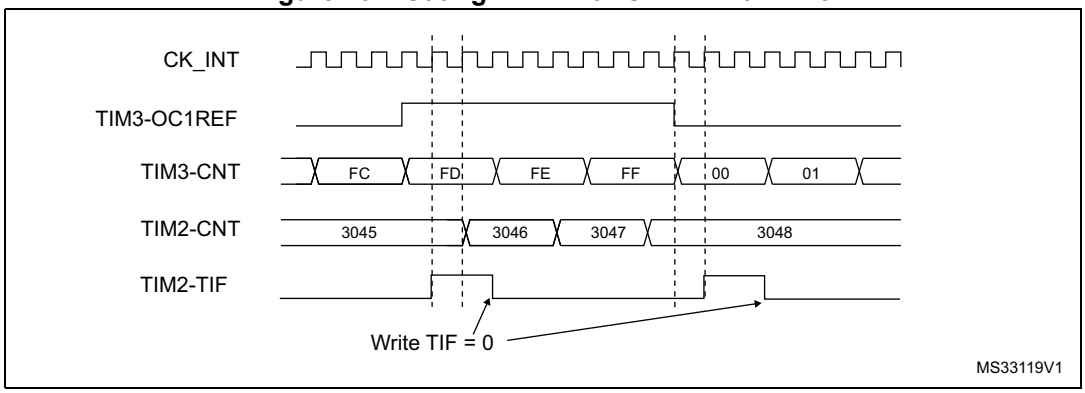
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to *Figure 205* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 207. Gating TIM2 with OC1REF of TIM3

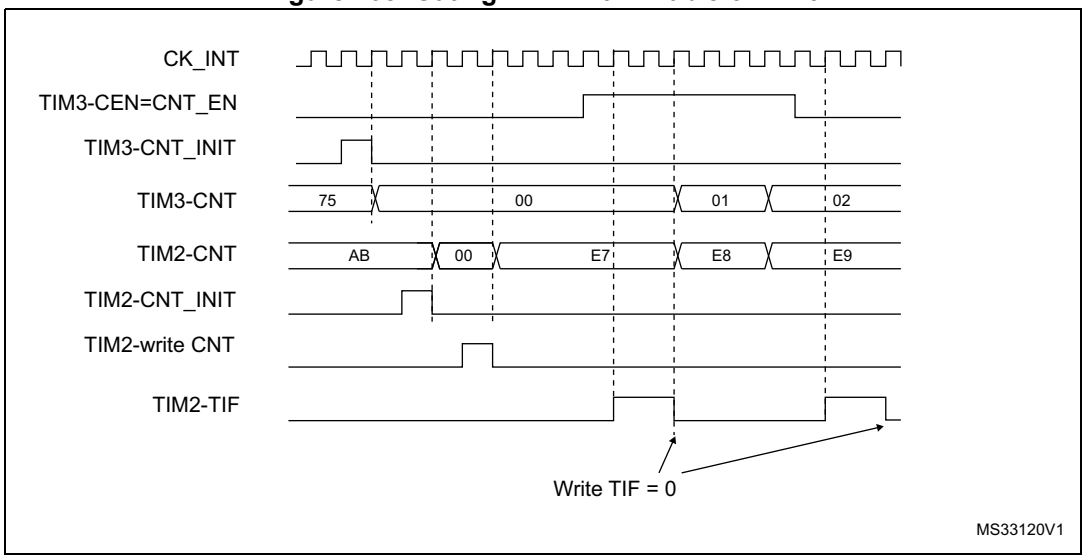


In the example in *Figure 207*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 208*), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 208. Gating TIM2 with Enable of TIM3

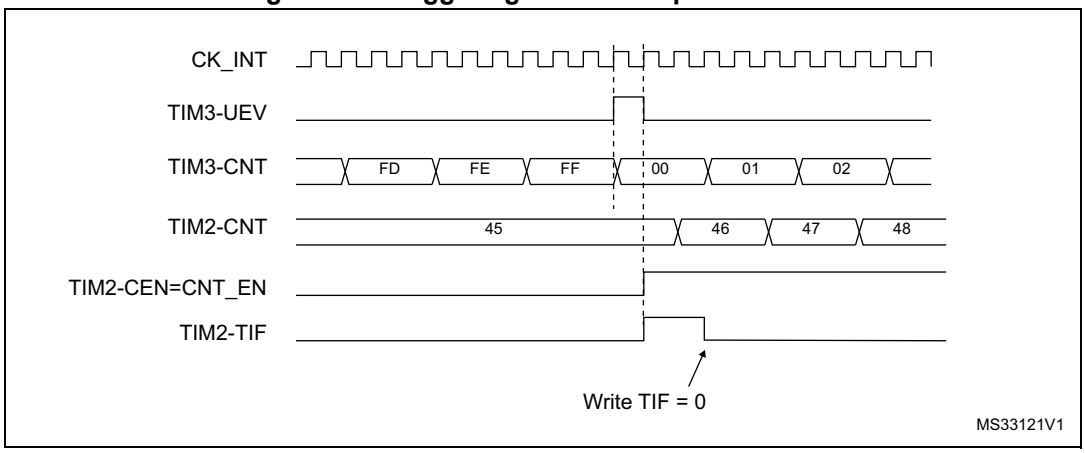


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 205* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

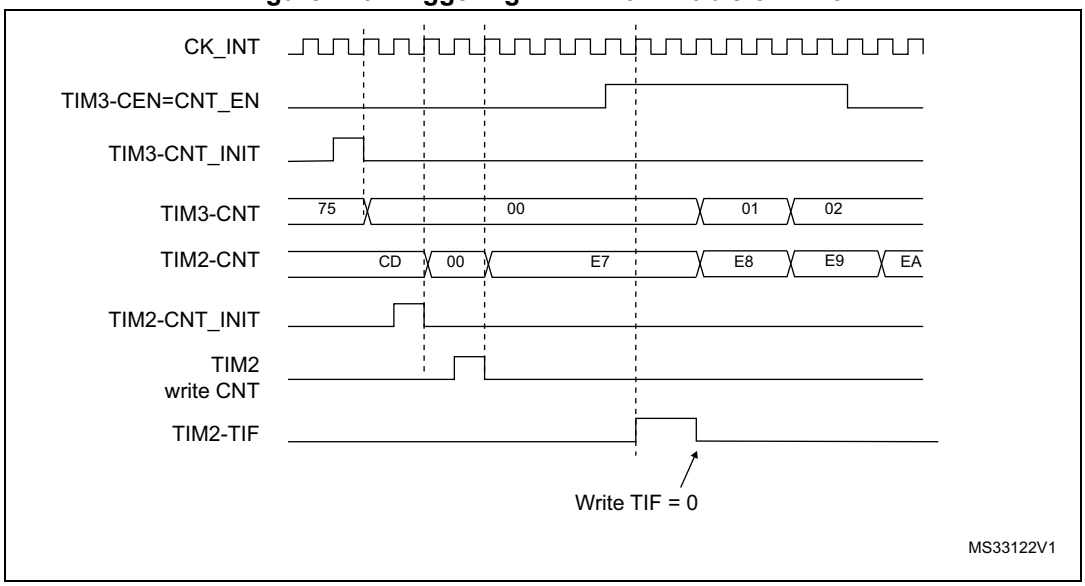
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 209. Triggering TIM2 with update of TIM3



As in the previous example, both counters can be initialized before starting counting. *Figure 210* shows the behavior with the same configuration as in *Figure 209* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 210. Triggering TIM2 with Enable of TIM3



Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 205* for connections. To ensure the counters are

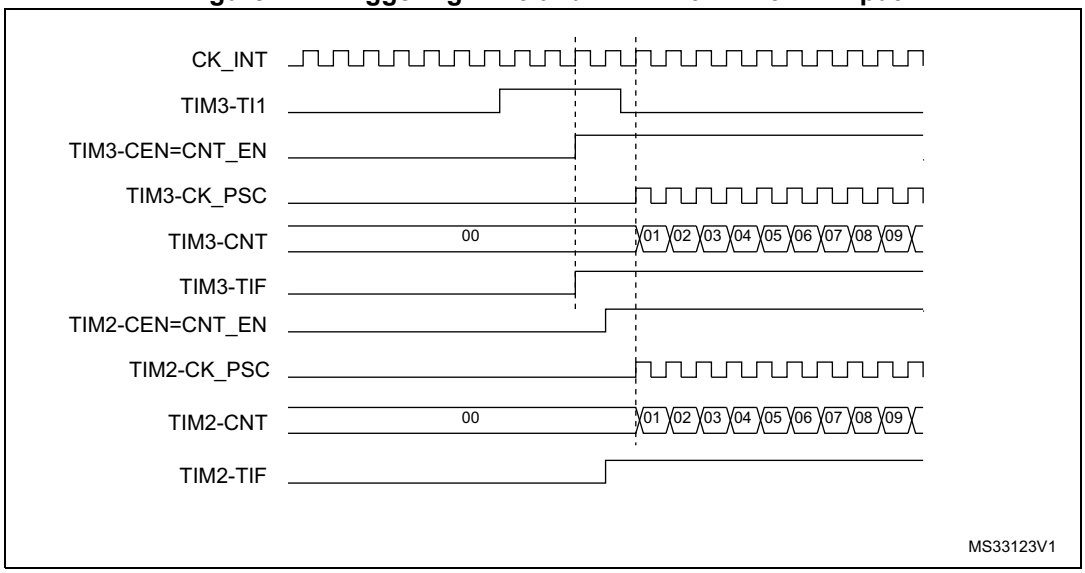
aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 211. Triggering TIM3 and TIM2 with TIM3 TI1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

19.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

19.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section 31.15.2: Debug support for timers, watchdog, bxCAN and I^2C*.

19.4 TIM2/TIM3 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

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

19.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 3)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

19.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 3)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 18.3.24: Interfacing with Hall sensors on page 434*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

19.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

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).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0).
001: Internal Trigger 1 (ITR1).
010: Internal Trigger 2 (ITR2).
011: Internal Trigger 3 (ITR3).
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)
See *Table 72: TIMx internal trigger connection on page 530* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: OCREF_CLR_INT is connected to the OCREF_CLR input
1: OCREF_CLR_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 72. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	Reserved	TIM3	Reserved
TIM3	TIM1	TIM2	Reserved	Reserved

19.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 3)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

19.4.5 TIMx status register (TIMx_SR)(x = 2 to 3)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

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

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow (for TIM2) 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.

19.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 3)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

19.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

19.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 3)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

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 decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

19.4.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 3)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

19.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 3)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

19.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 3)

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.*
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 73. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

19.4.12 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:16 **CNT[31:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

19.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 3)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 **CNT[30:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

19.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 3)

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”).

19.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 3)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

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

Bits 31:16 **ARR[31:16]**: High auto-reload value (TIM2)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 19.3.1: Time-base unit on page 480* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

19.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (TIM2)

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.

19.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (TIM2)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

19.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.
If channel CC3is 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.

19.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 3)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2)

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_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

19.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 3)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...
Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

19.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 3)

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).

19.4.22 TIMx register map

TIMx registers are mapped as described in the table below:

Table 74. TIM2/TIM3 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0		0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 74. TIM2/TIM3 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIMx_CNT**	CNT[31] or UIFCPY	CNT[30:16] (TIM2 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**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 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 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 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 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 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**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

20 General-purpose timers (TIM15/TIM16/TIM17)

20.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 20.4.21: Timer synchronization (TIM15)*.

20.2 TIM15 main features

TIM15 includes the following features:

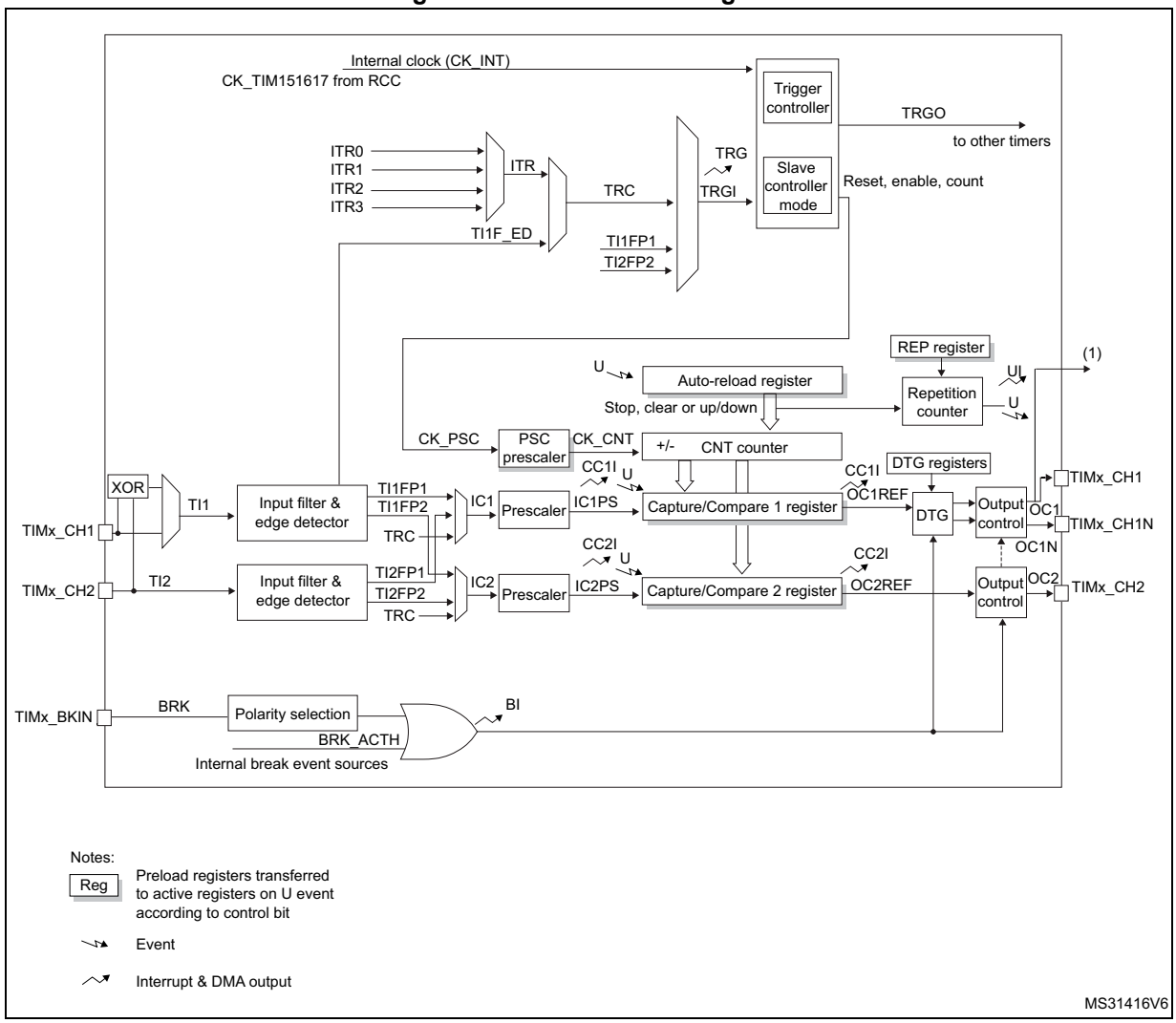
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

20.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

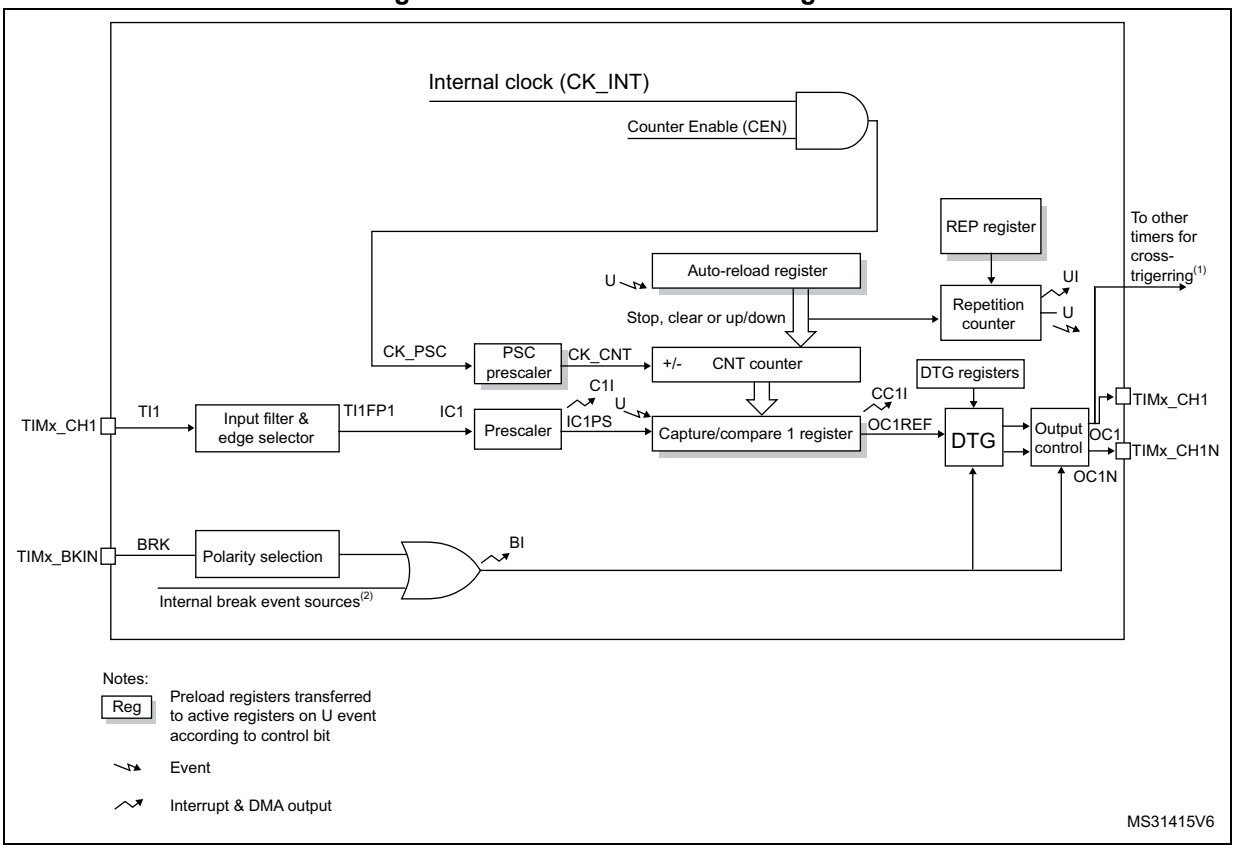
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 212. TIM15 block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 8.2.7: Clock security system (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M4 LOCKUP (Hardfault) output
   - COMP output

Figure 213. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see *Section 20.4.22: Using timer output as trigger for other timers (TIM16/TIM17)*.
2. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 8.2.7: Clock security system (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M4 LOCKUP (Hardfault) output
   - COMP output

20.4 TIM15/TIM16/TIM17 functional description

20.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 214 and *Figure 215* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 214. Counter timing diagram with prescaler division change from 1 to 2

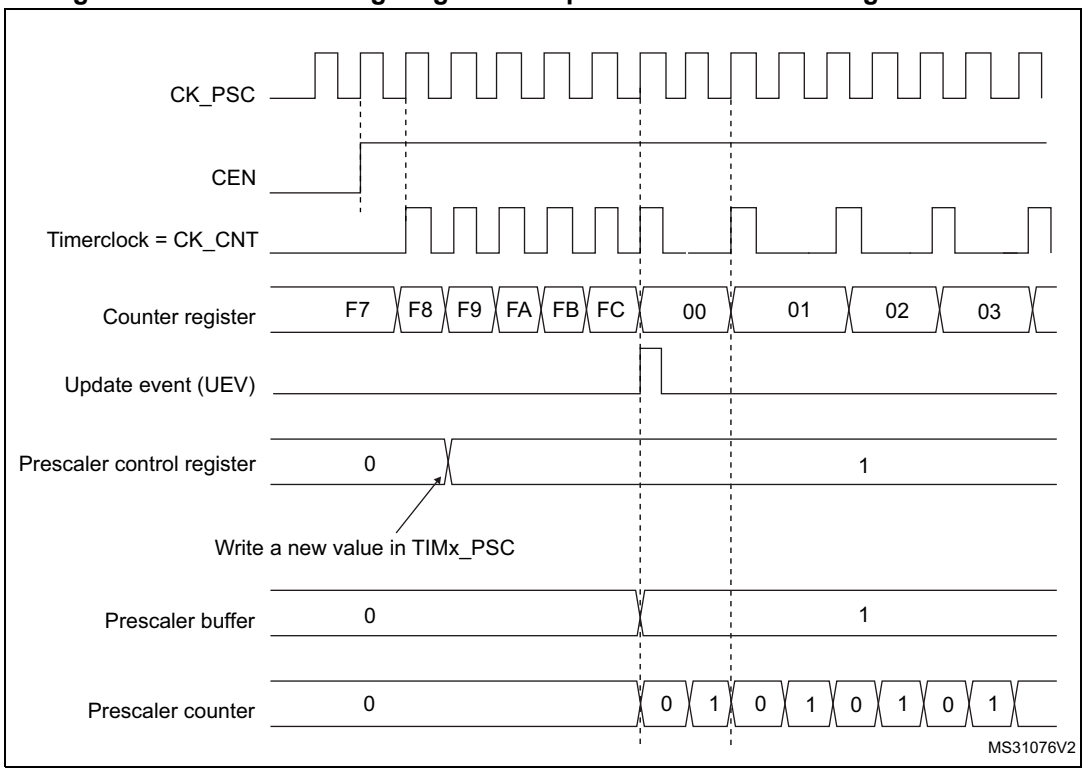
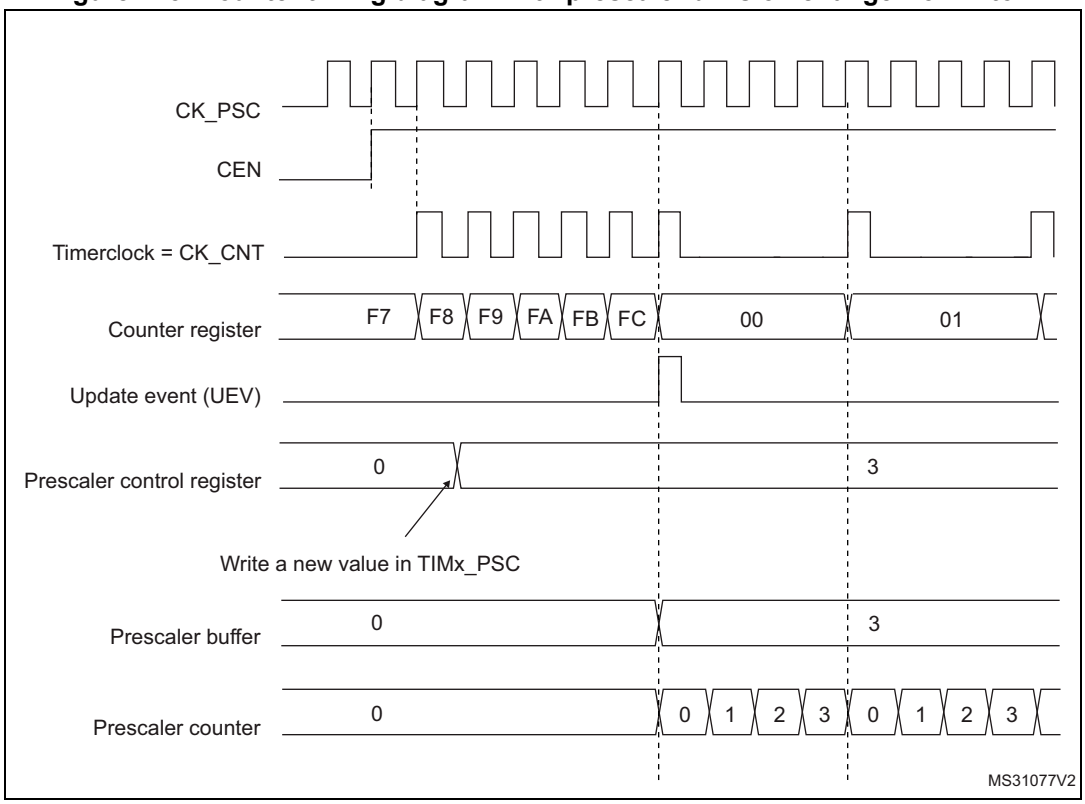


Figure 215. Counter timing diagram with prescaler division change from 1 to 4



20.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 216. Counter timing diagram, internal clock divided by 1

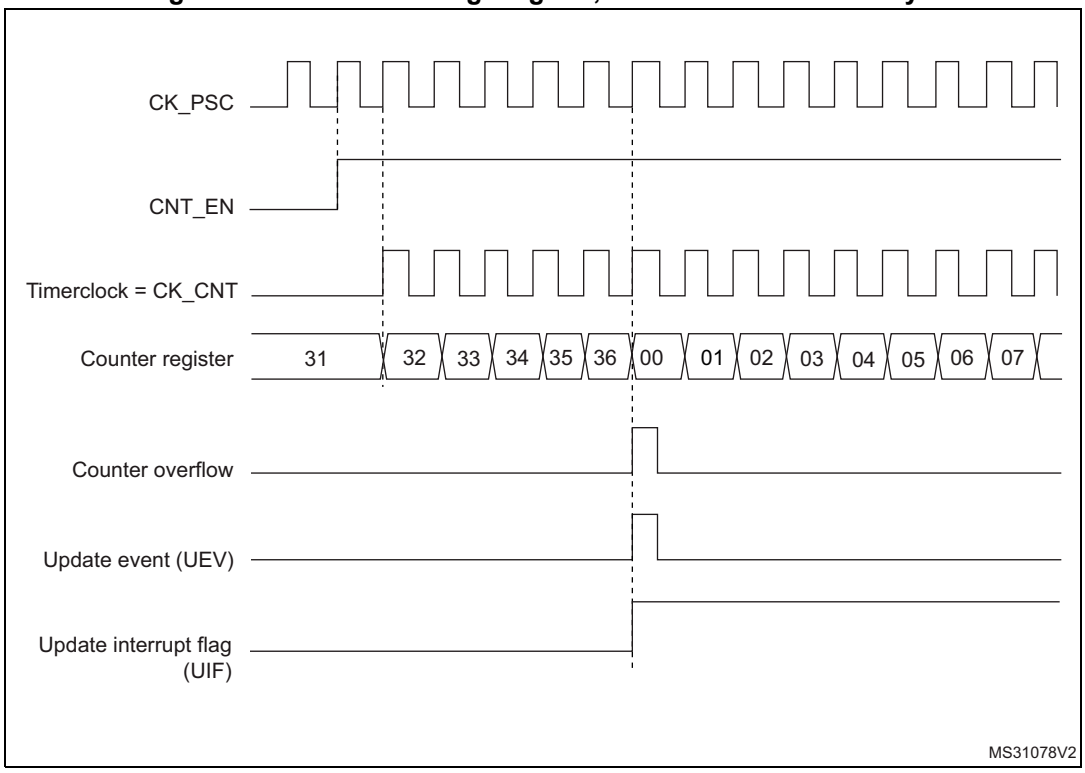


Figure 217. Counter timing diagram, internal clock divided by 2

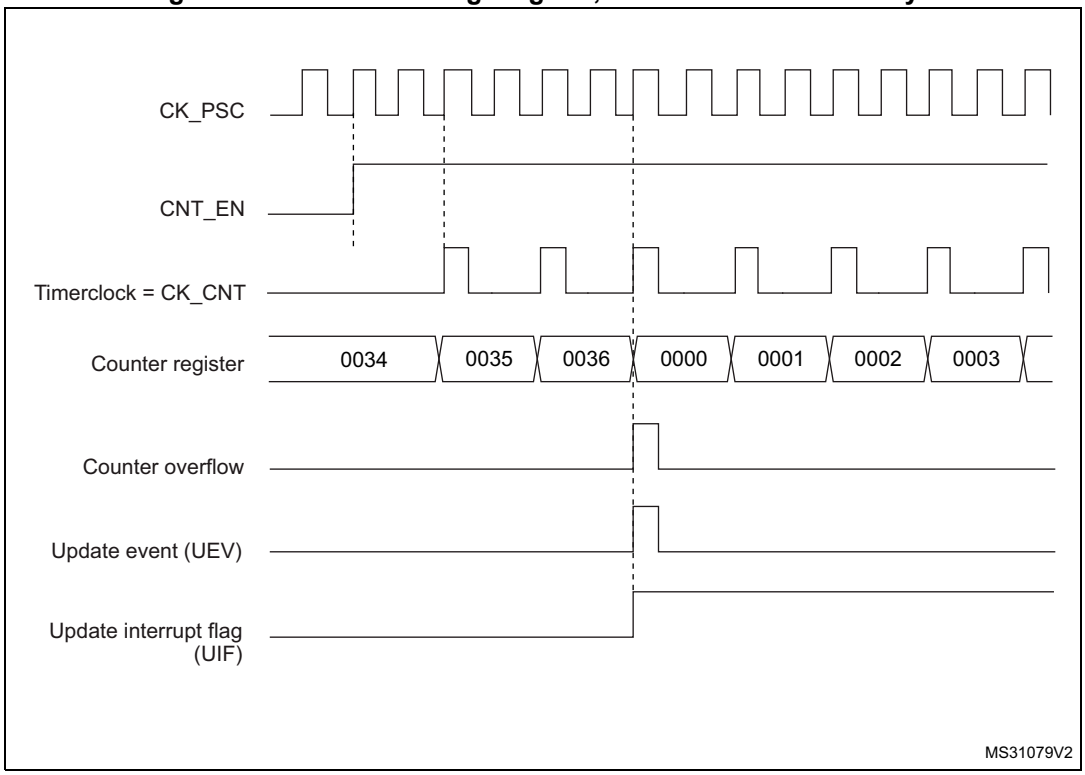


Figure 218. Counter timing diagram, internal clock divided by 4

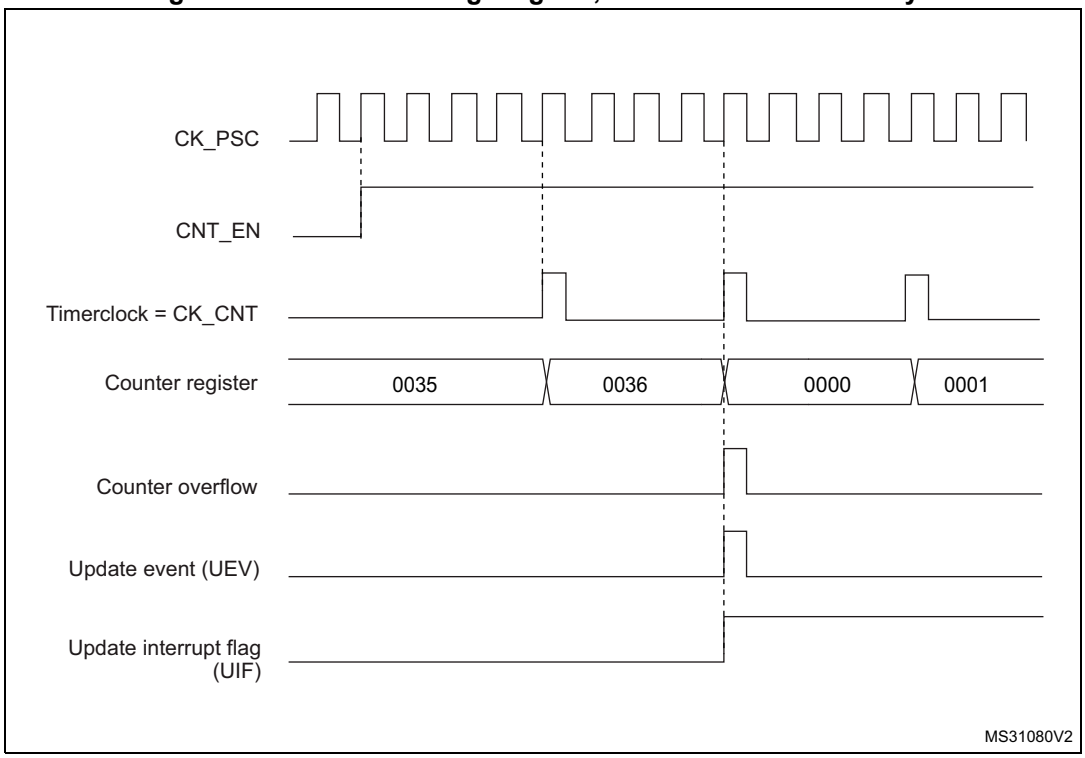


Figure 219. Counter timing diagram, internal clock divided by N

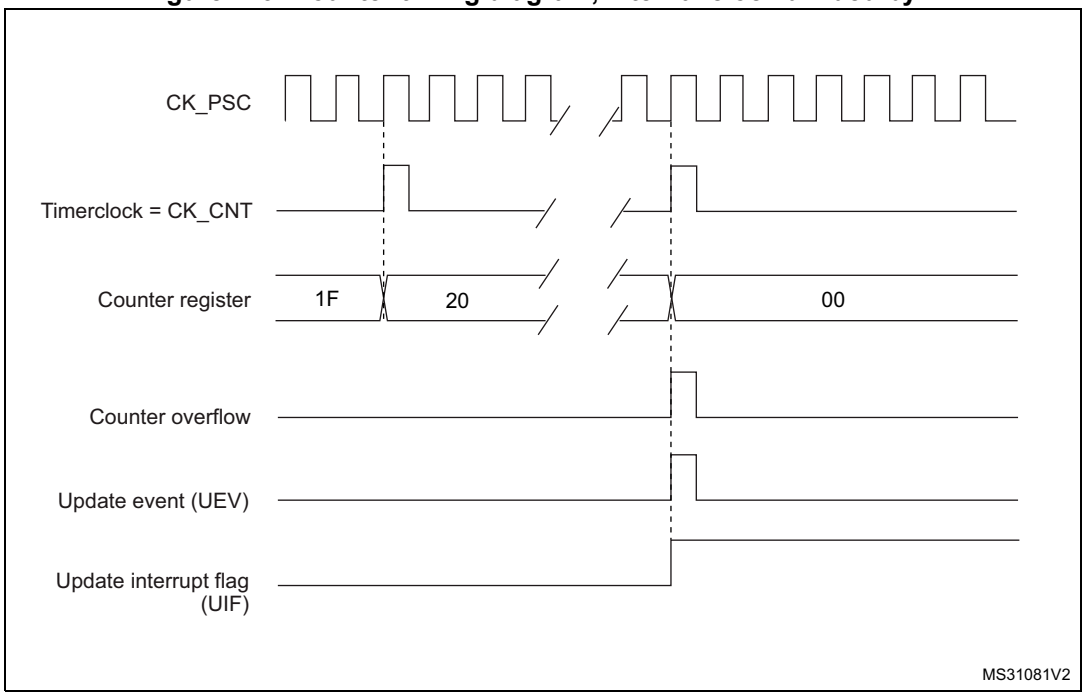


Figure 220. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

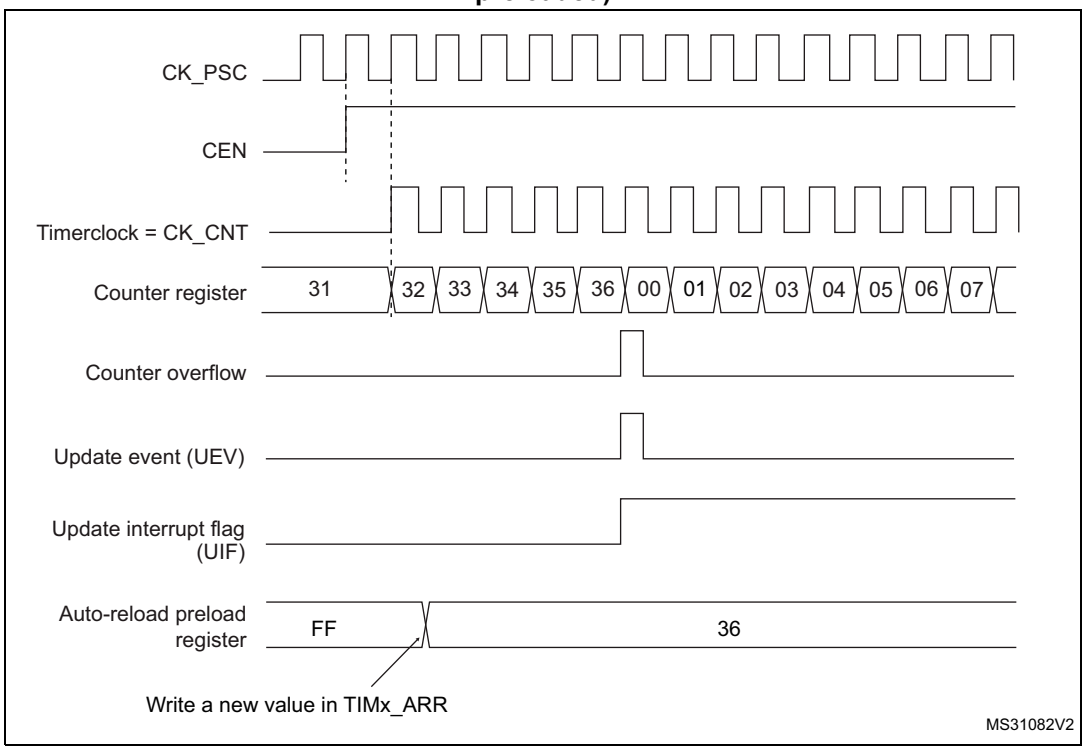
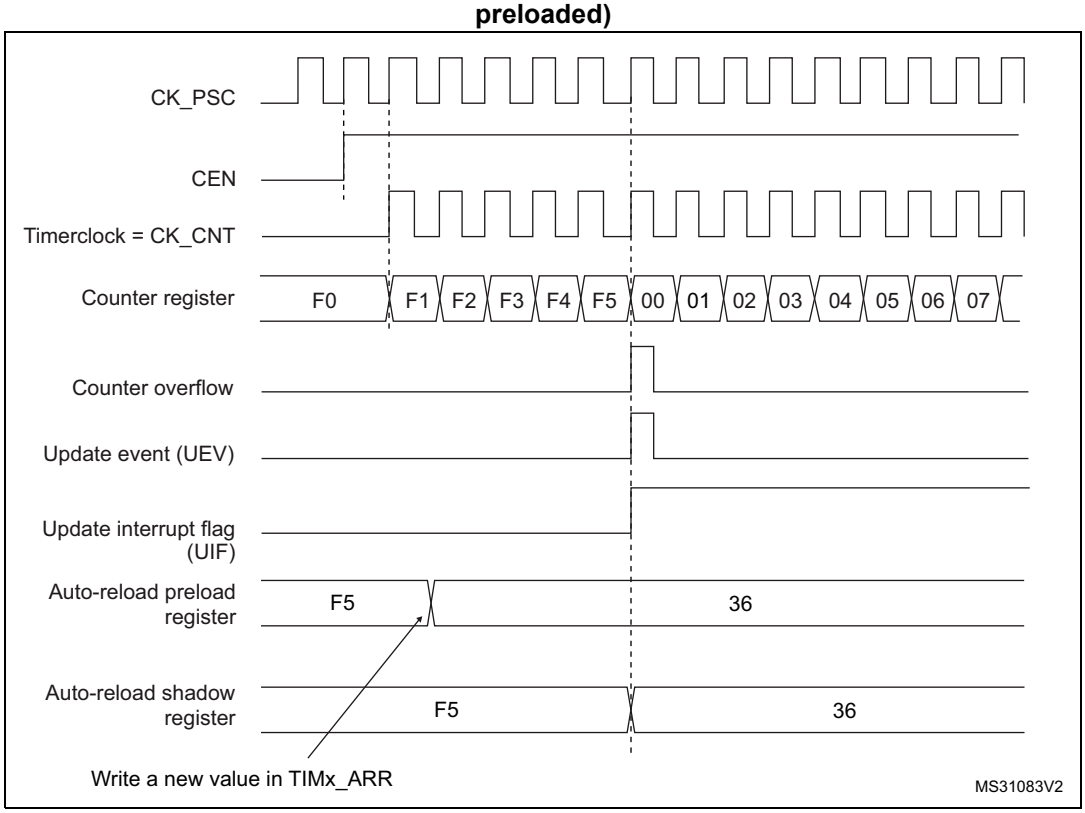


Figure 221. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



20.4.3 Repetition counter

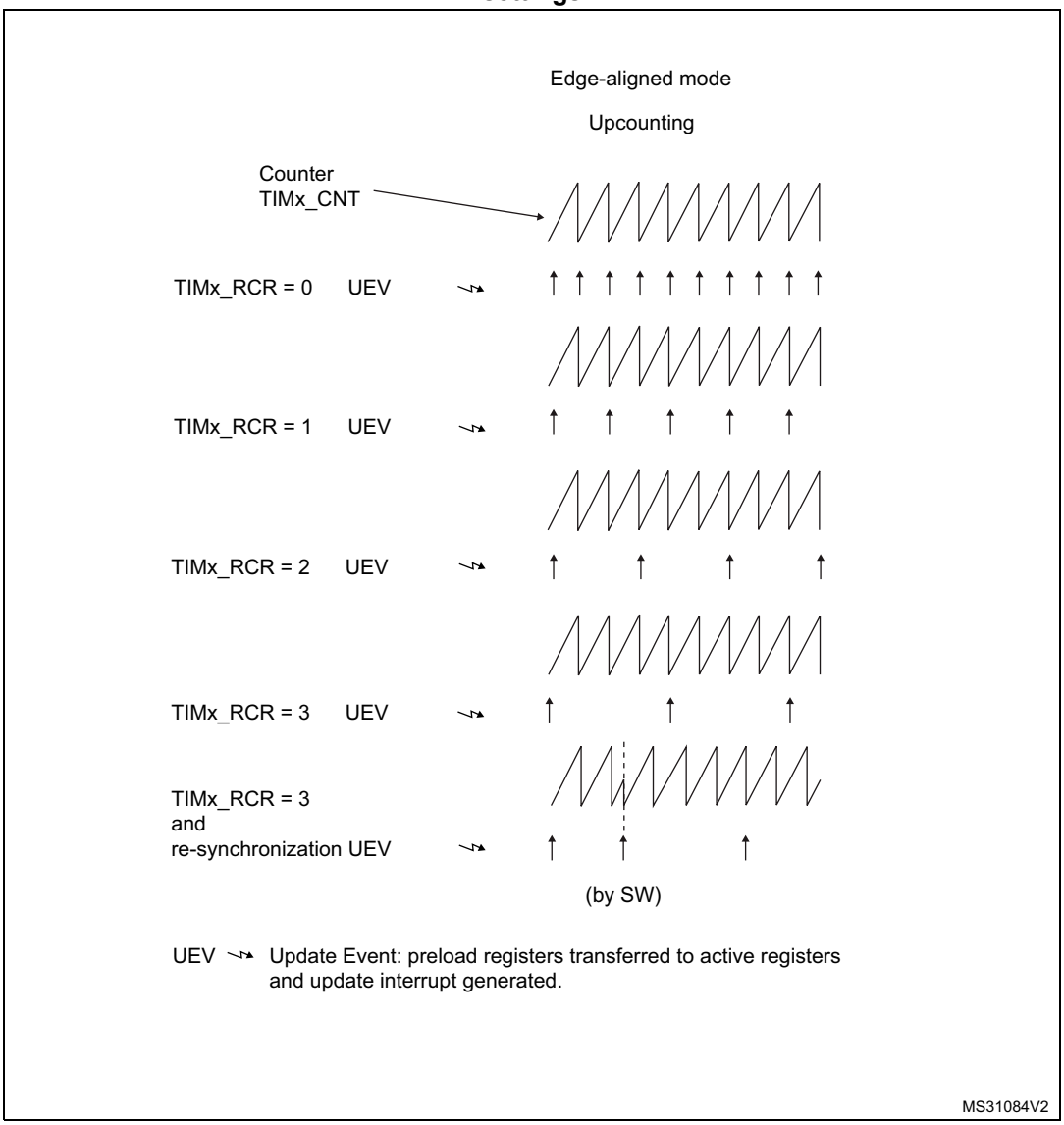
Section 20.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 222*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 222. Update rate examples depending on mode and TIMx_RCR register settings



20.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer on page 518* for more details.

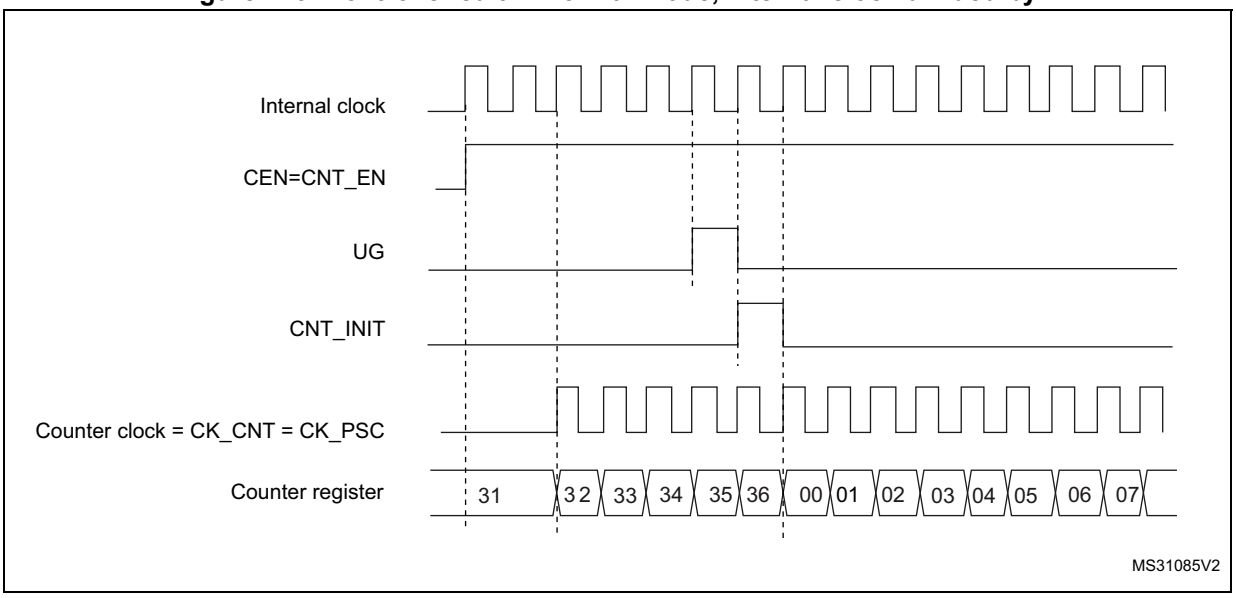
Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 223 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

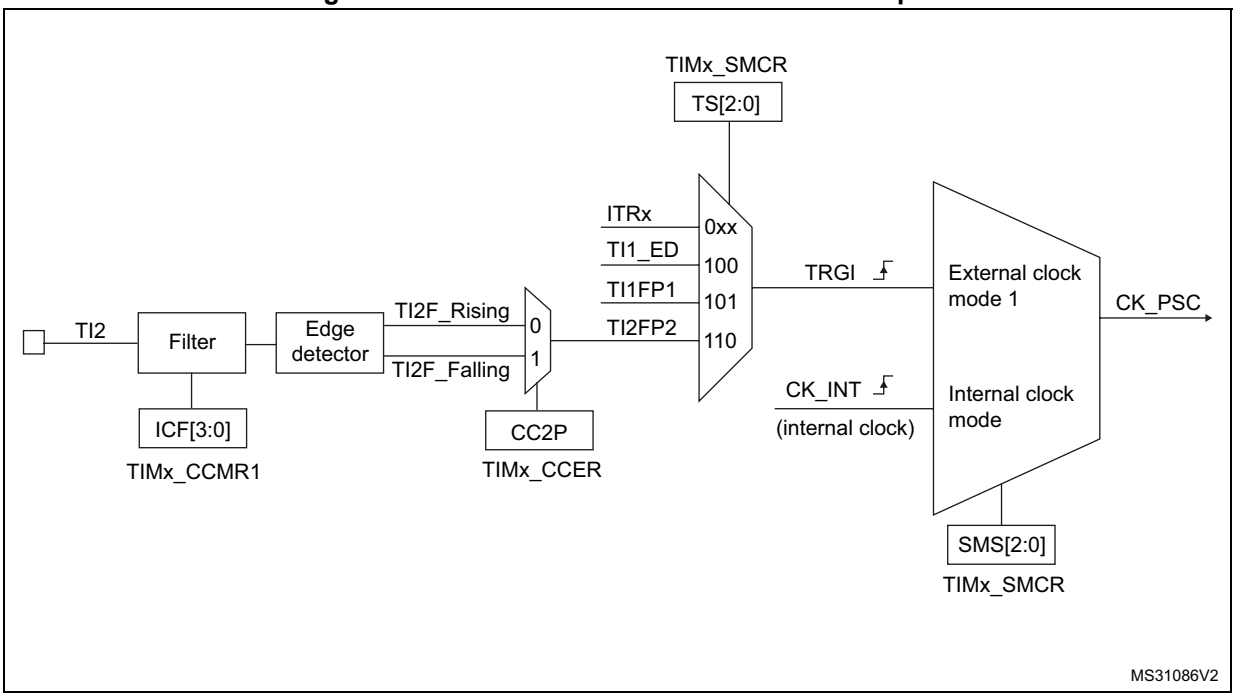
Figure 223. Control circuit in normal mode, internal clock divided by 1



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 224. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

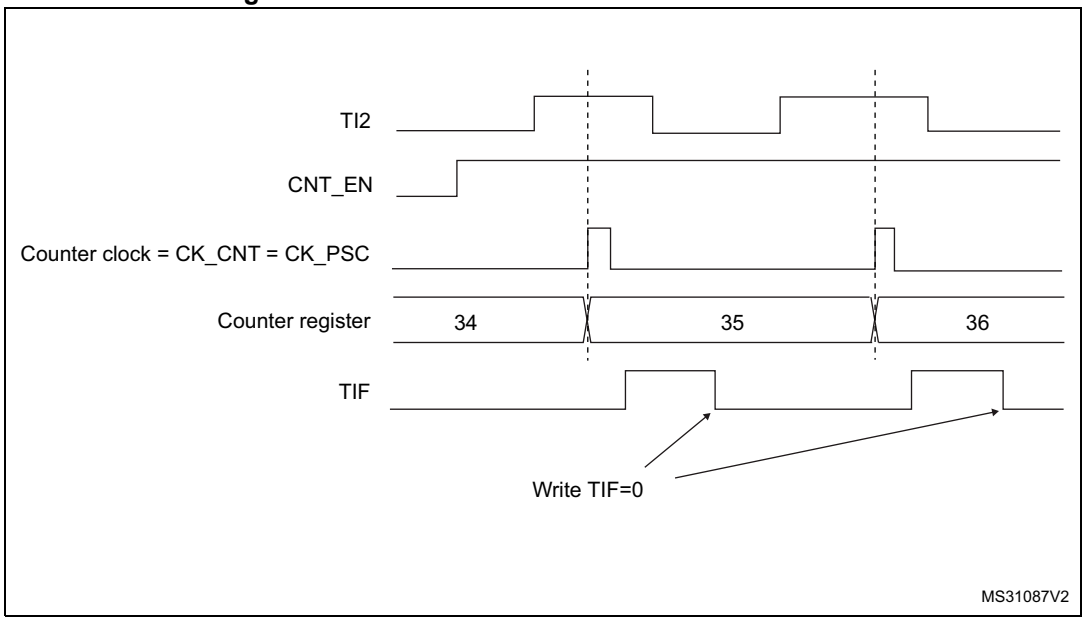
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 225. Control circuit in external clock mode 1



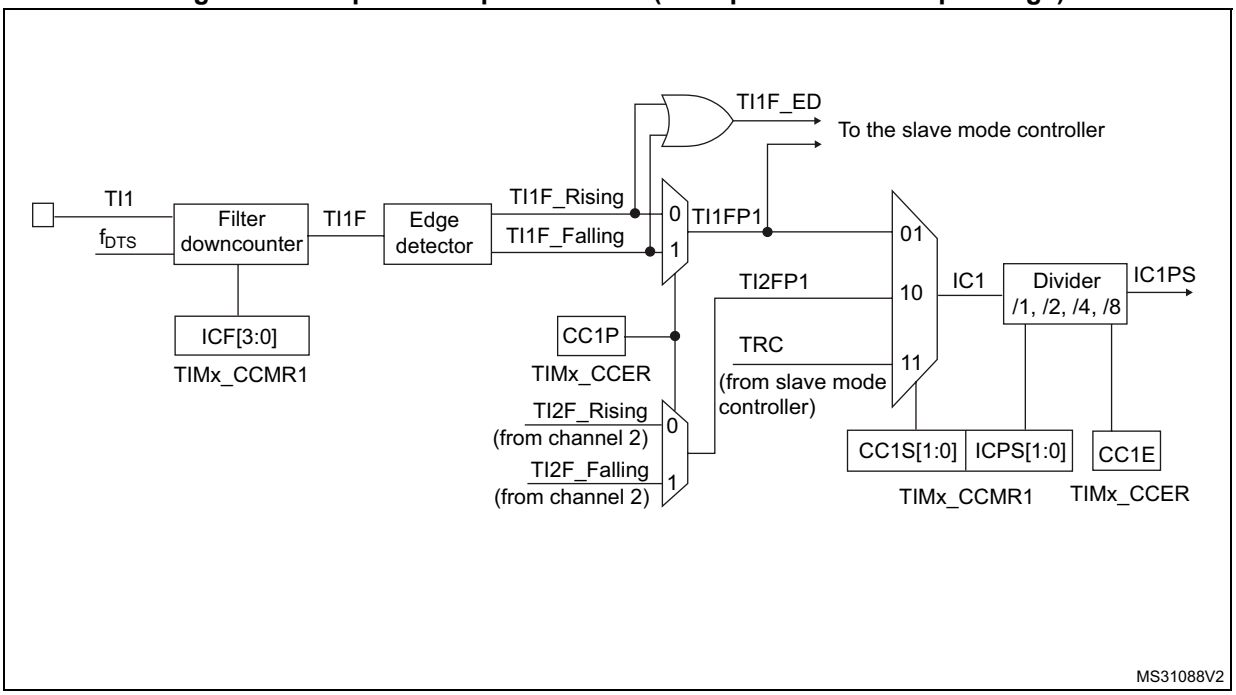
20.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 226 to *Figure 229* 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 226. 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 227. Capture/compare channel 1 main circuit

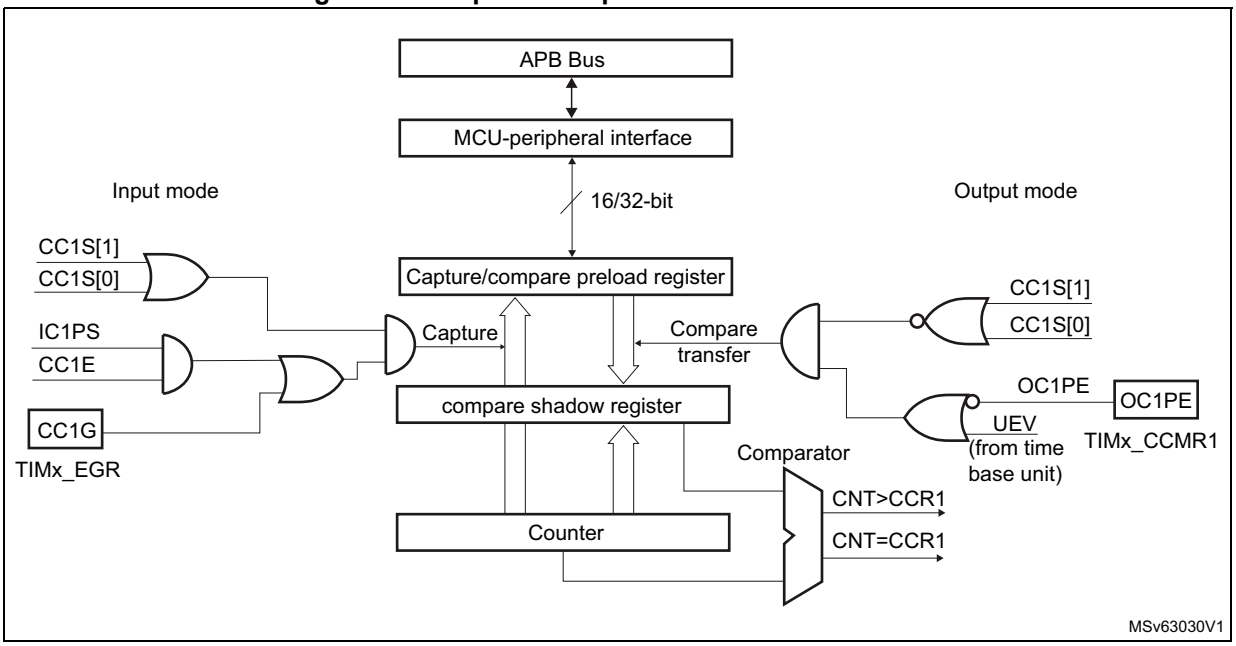


Figure 228. Output stage of capture/compare channel (channel 1)

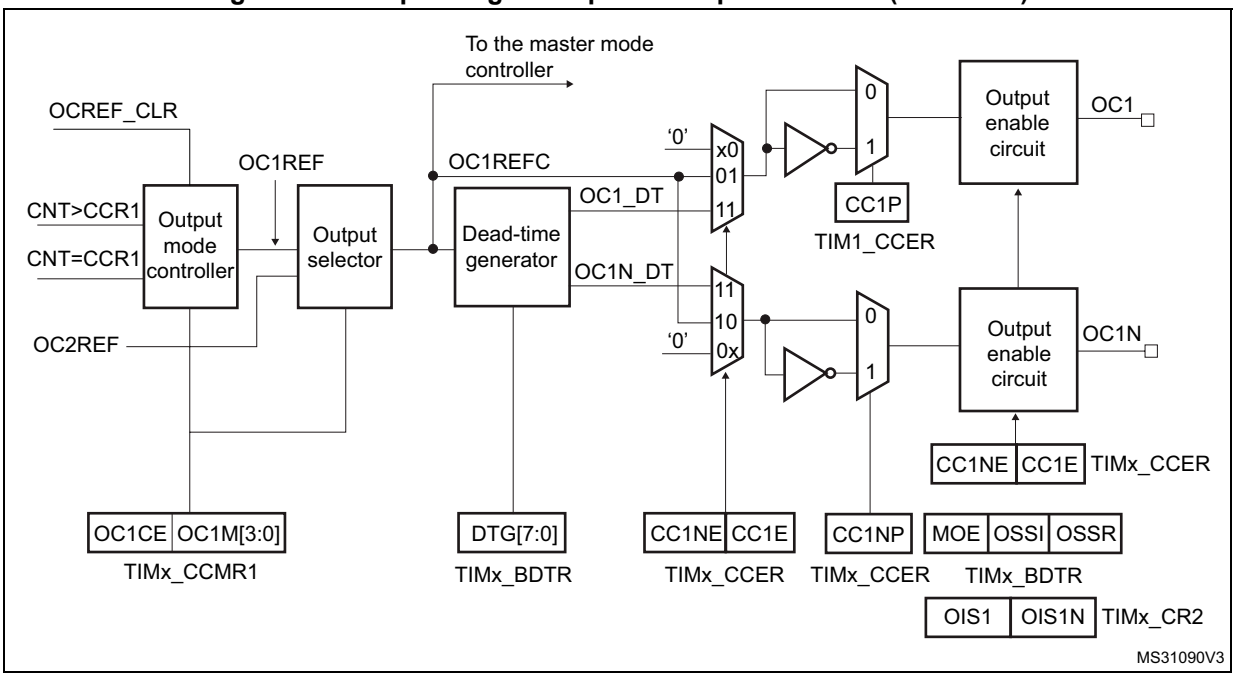
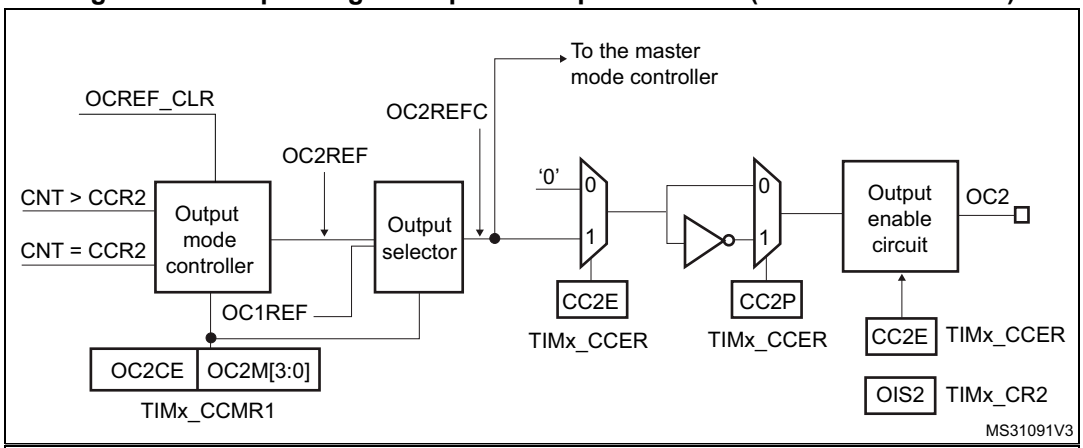
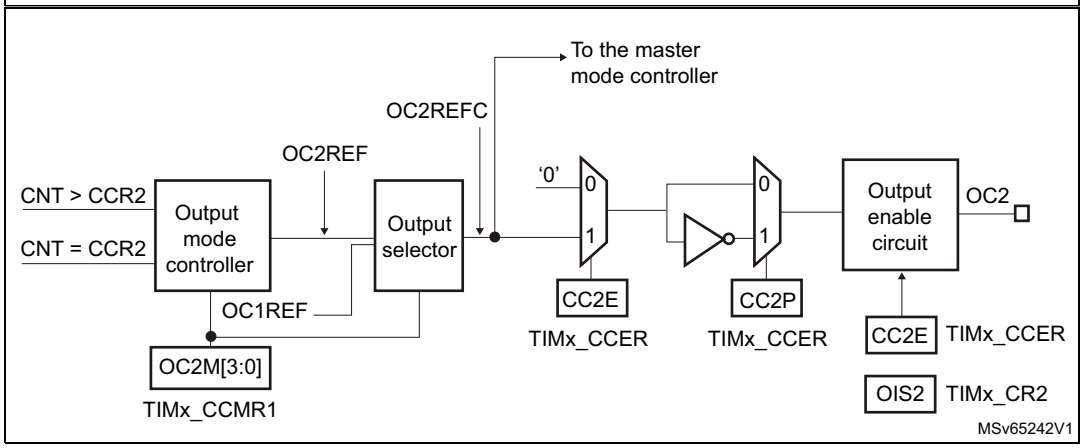


Figure 229. Output stage of capture/compare channel (channel 2 for TIM15)





The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

20.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

20.4.7 PWM input mode (only for TIM15)

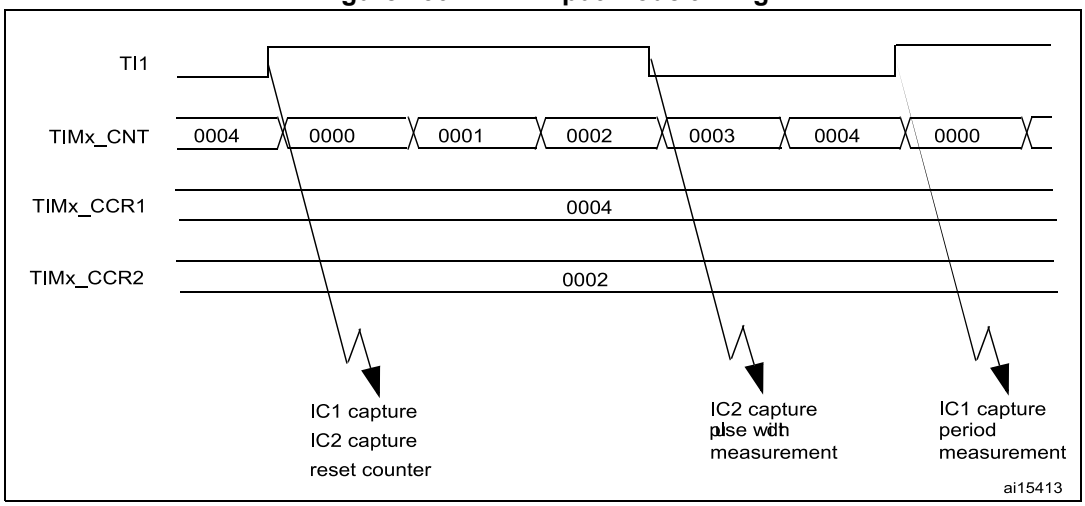
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to '10' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 230. 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.

20.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

20.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

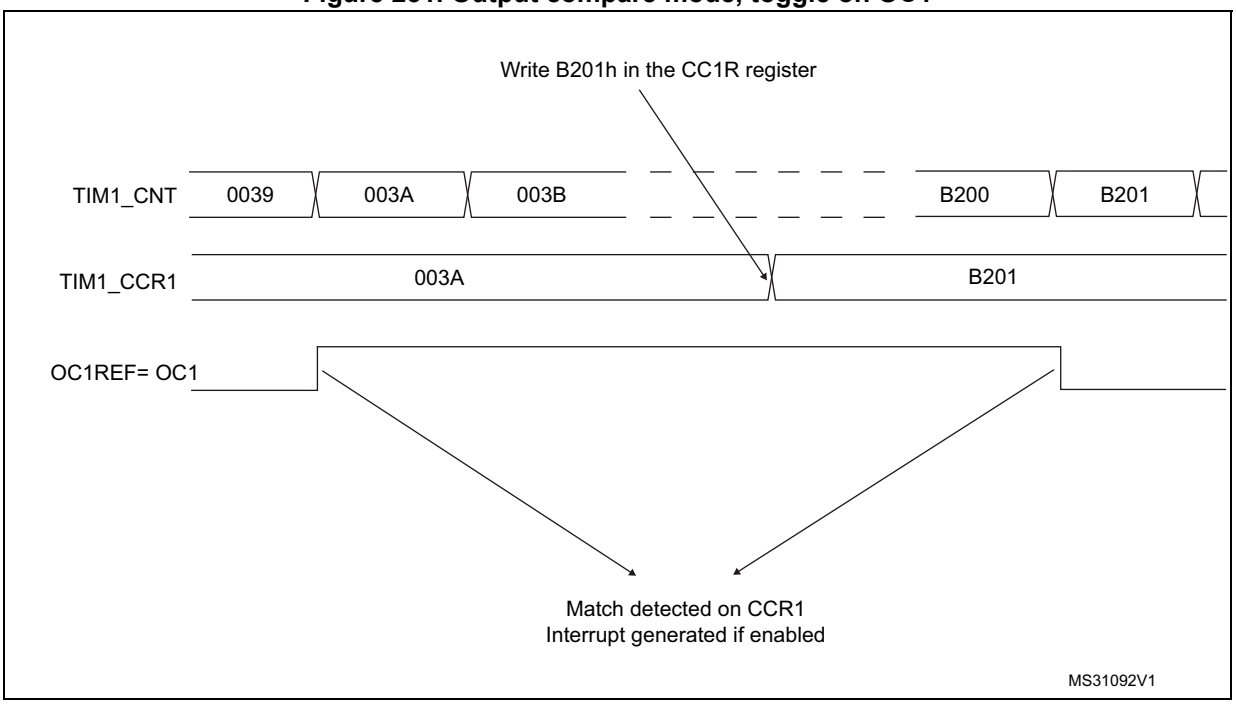
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 231*.

Figure 231. Output compare mode, toggle on OC1



20.4.10 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

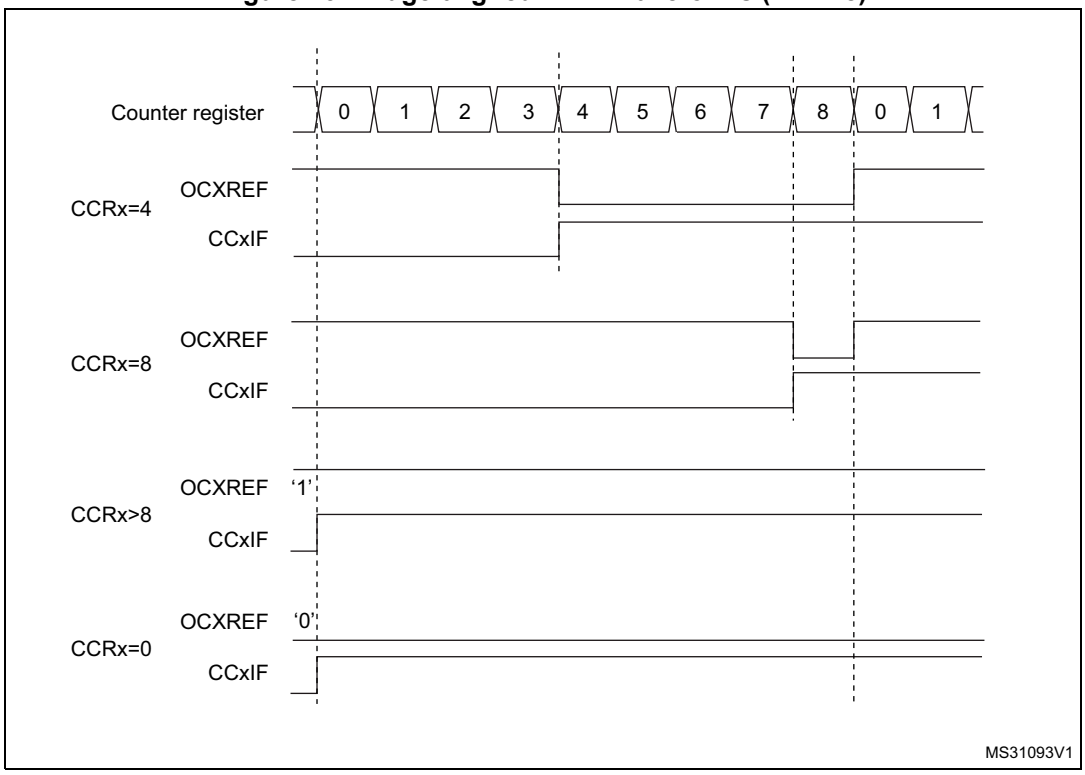
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for 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 TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 554*.

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 232* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 232. Edge-aligned PWM waveforms (ARR=8)



20.4.11 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined

by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

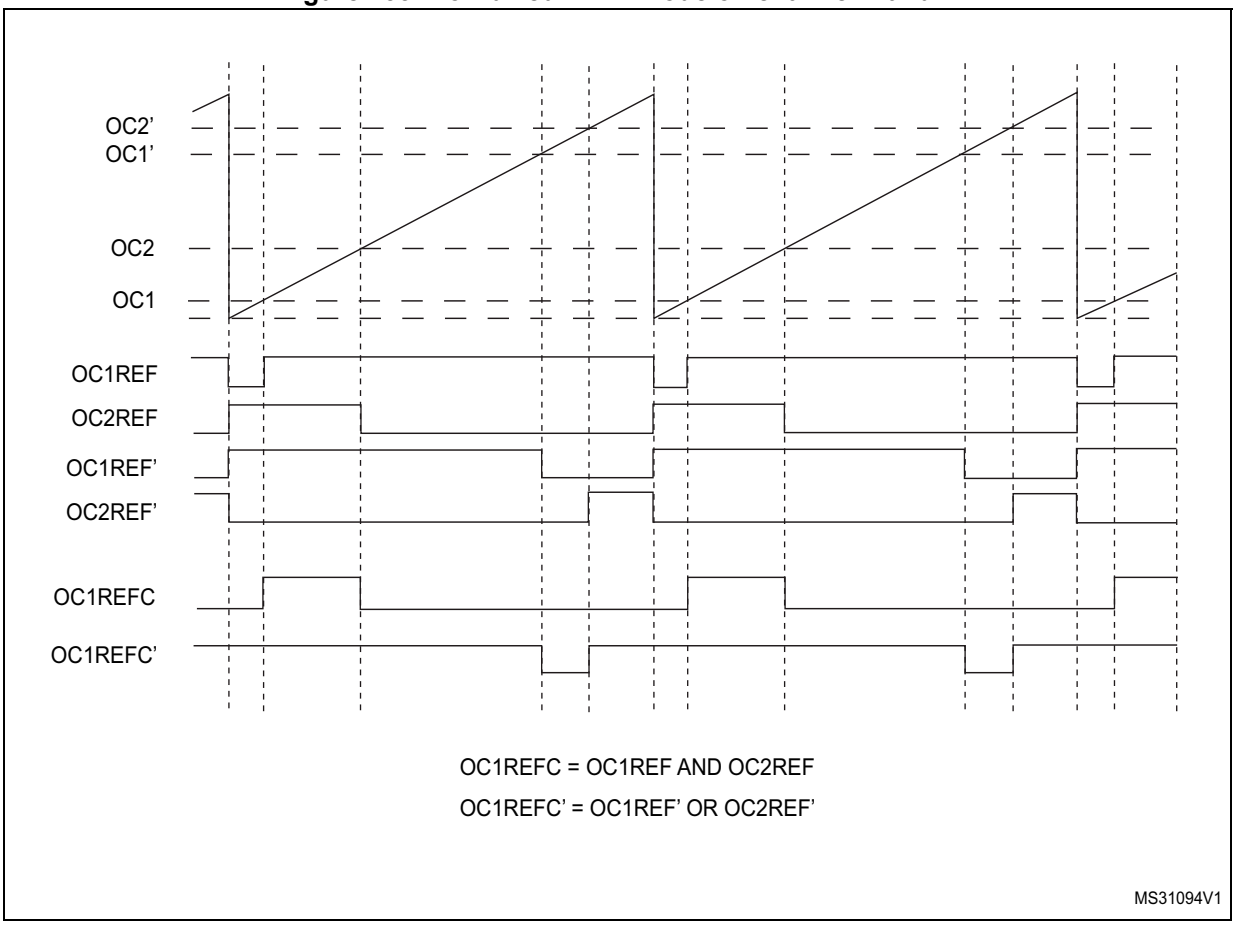
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 233 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 233. Combined PWM mode on channel 1 and 2



20.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected 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 78: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 618* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 234. Complementary output with dead-time insertion.

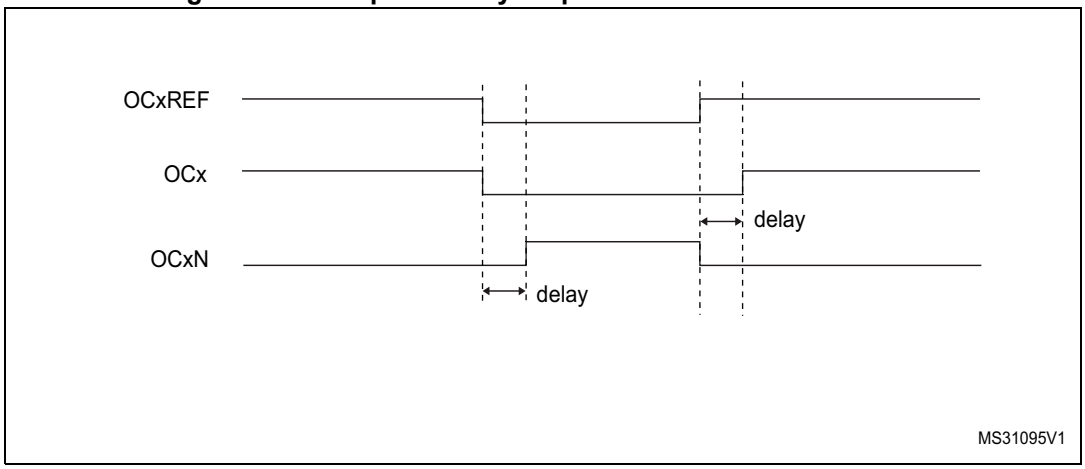


Figure 235. Dead-time waveforms with delay greater than the negative pulse.

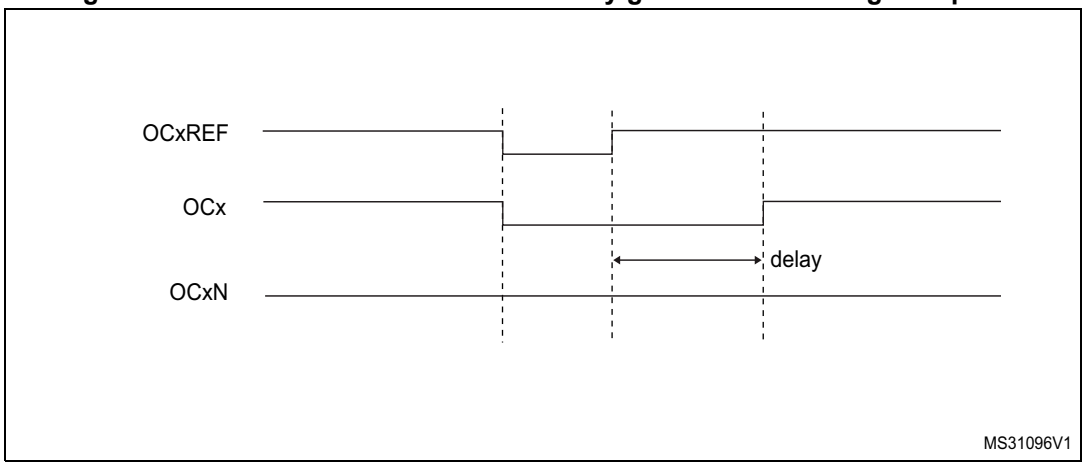
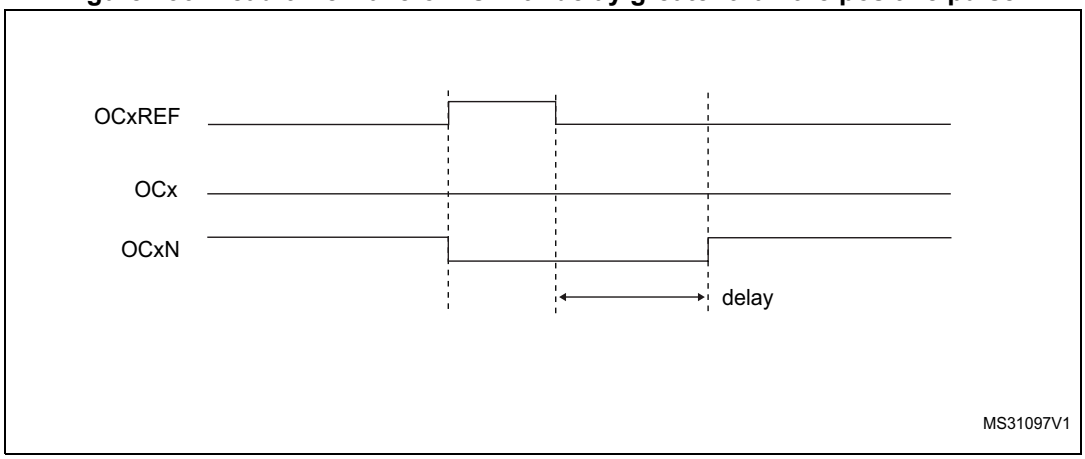


Figure 236. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 20.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 621* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

20.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

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 76: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 599* for more details.

The break source can be:

- An external source connected to BKIN pin
- An internal source:
  - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 8.2.7: Clock security system (CSS)*
  - An output from a comparator
  - A PVD output
  - SRAM parity error signal
  - Cortex®-M4 LOCKUP (Hardfault) output

Warning: The internal sources protection is not available when the timer is automatic output enable mode (AOE bit set in the TIMx_BDTR). The MOE bit is set again on the next update event, regardless of any pending error on the BRK_ACTH input.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the BRK inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)

It is also possible to generate break events by software using BG bit in TIMx_EGR register.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the 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, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

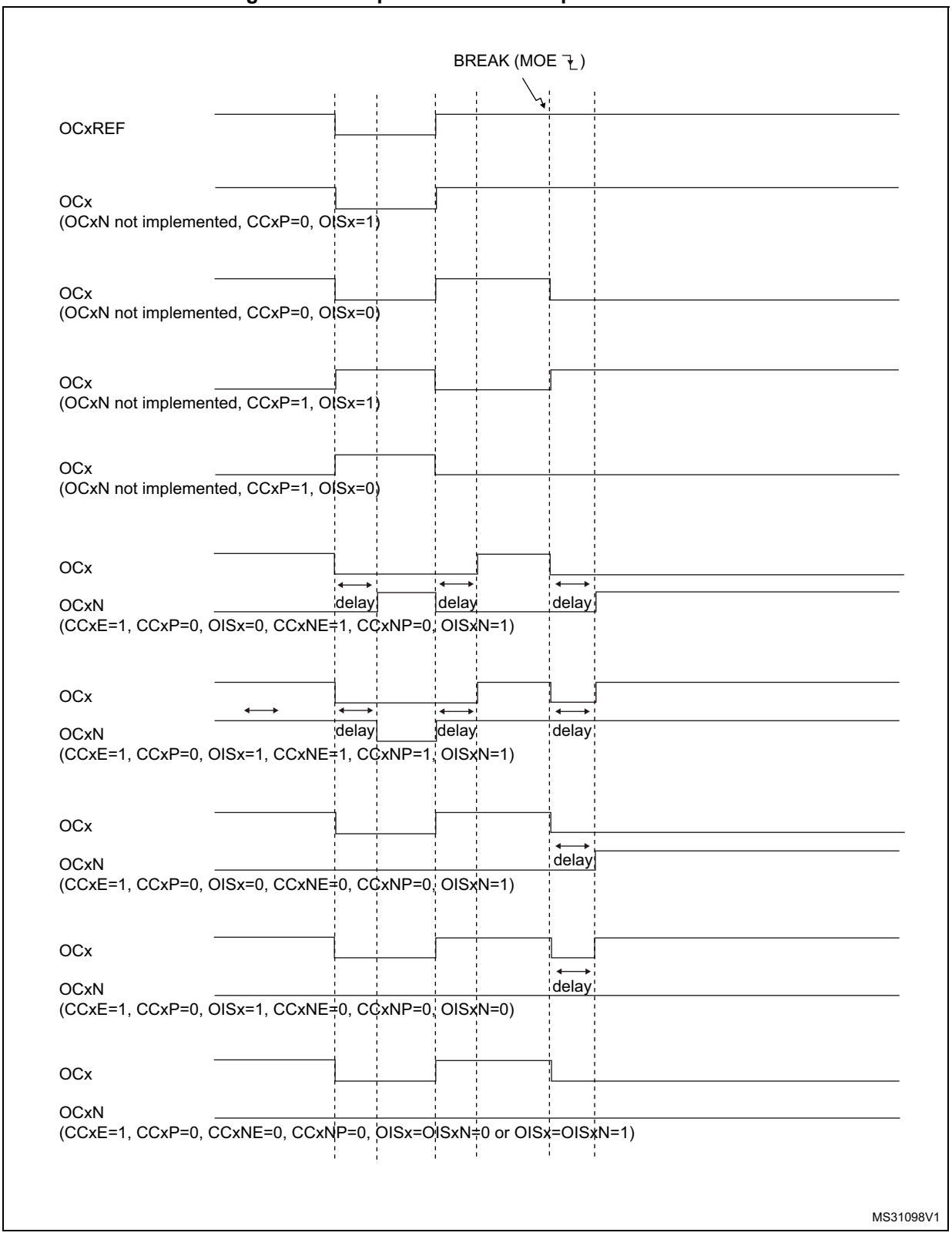
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 20.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 621*. The LOCK bits can be written only once after an MCU reset.

The *Figure 237* shows an example of behavior of the outputs in response to a break.

Figure 237. Output behavior in response to a break



20.4.14 One-pulse mode

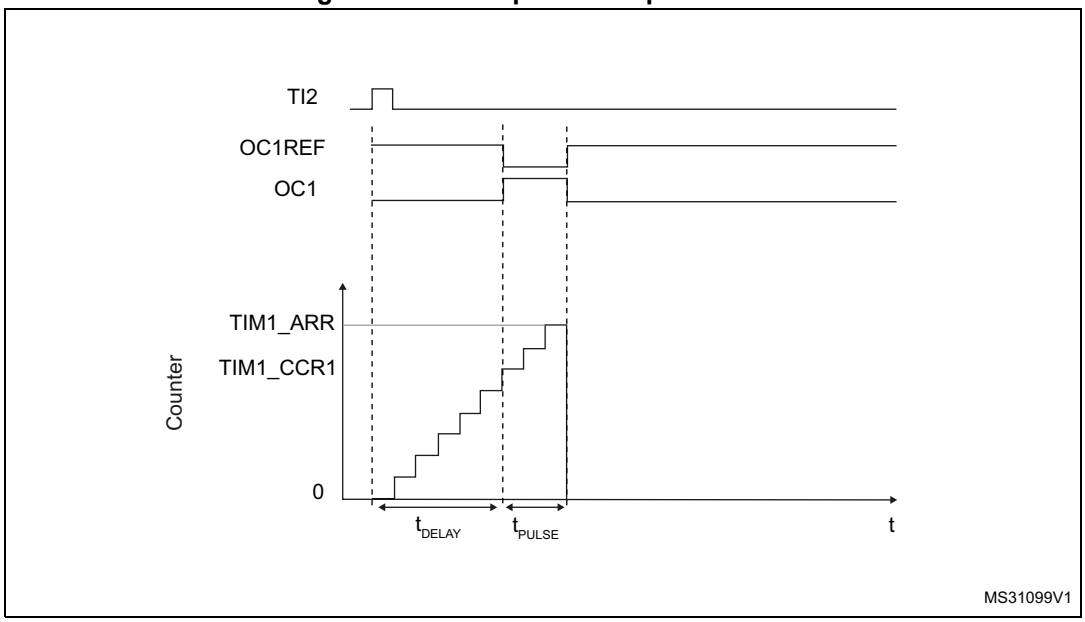
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 238. Example of one pulse mode



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

20.4.15 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 20.4.14*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

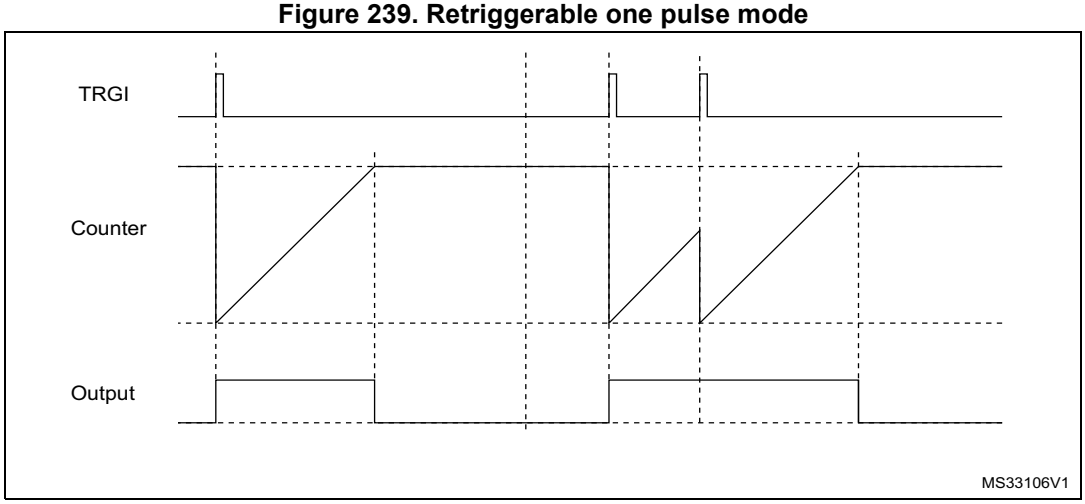
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 239. Retriggerable one pulse mode



20.4.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions

caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

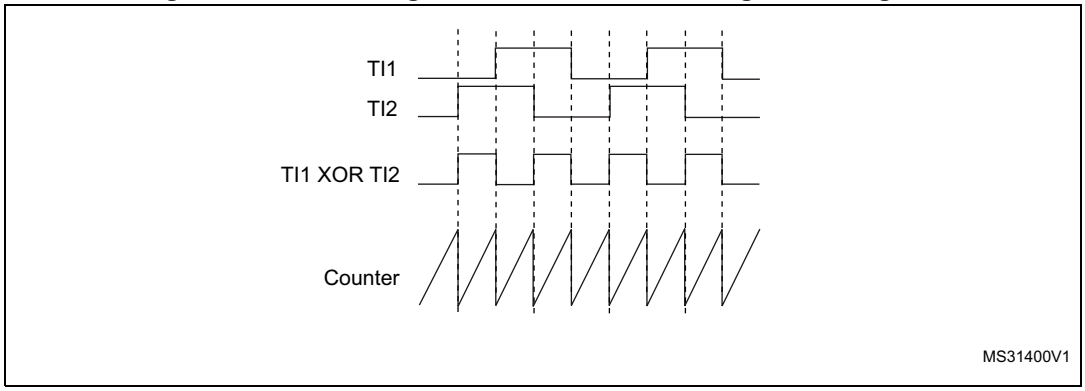
There is no latency between the assertions of the UIF and UIFCPY flags.

20.4.17 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 240*.

Figure 240. Measuring time interval between edges on 2 signals



20.4.18 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

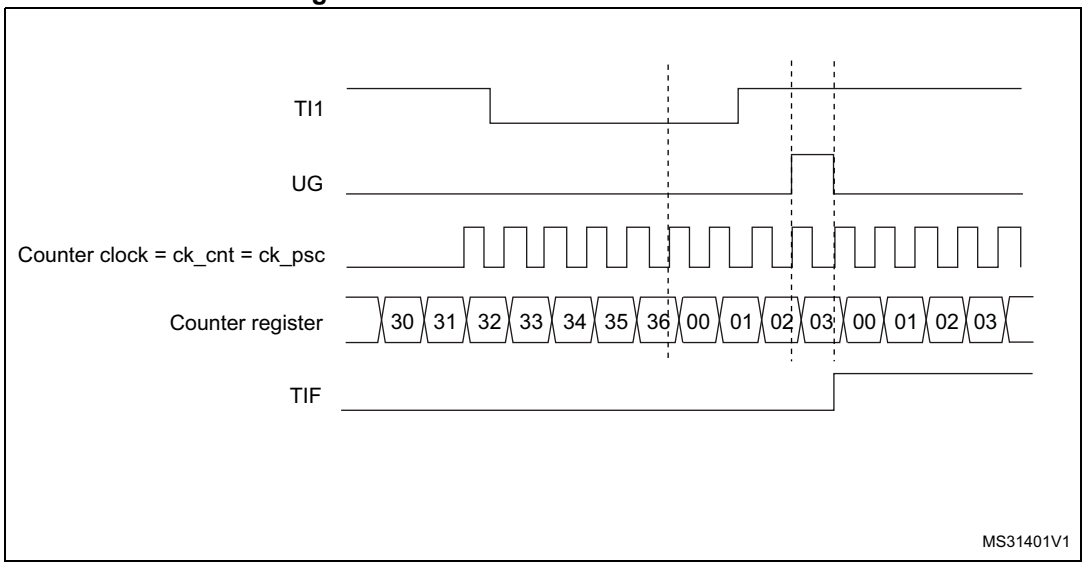
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 241. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

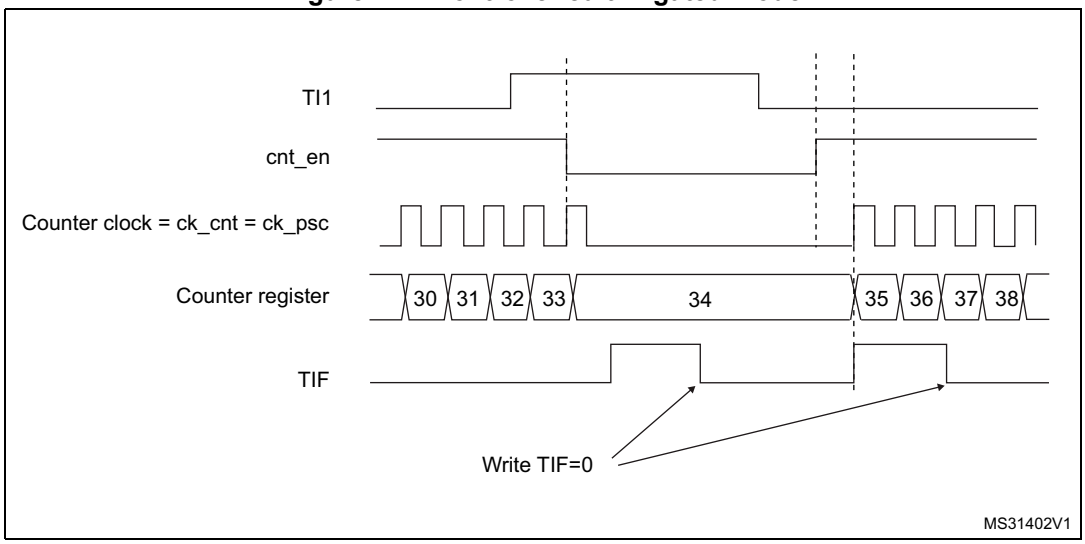
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 242. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

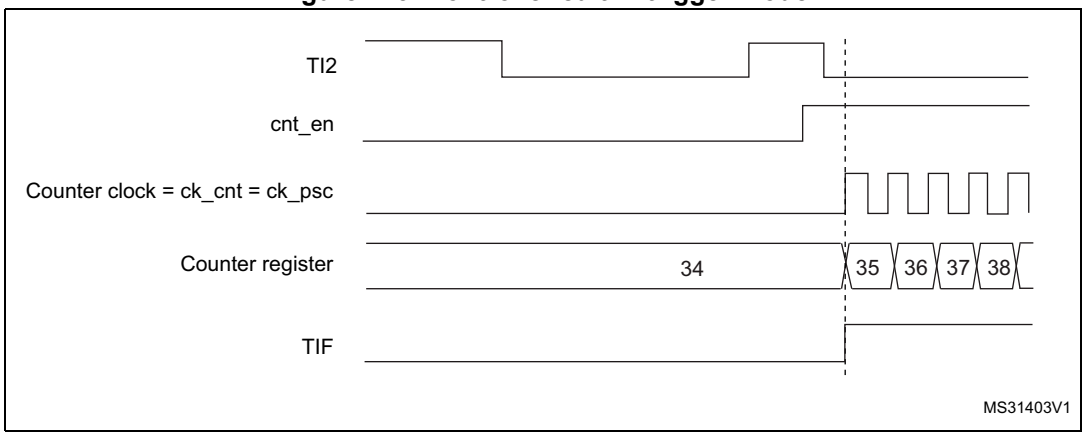
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 243. Control circuit in trigger mode



20.4.19 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

20.4.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

20.4.21 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 19.3.19: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

20.4.22 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

20.4.23 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to *Section 31.15.2: Debug support for timers, watchdog, bxCAN and I^2C*.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

20.5 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

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

20.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (TIx)
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2*t_{CK\_INT}$
10: $t_{DTS} = 4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

20.5.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 10 **OIS2:** Output idle state 2 (OC2 output)
0: OC2=0 when MOE=0
1: OC2=1 when MOE=0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BDTR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_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 TIM15_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO).
101: **Compare** - OC2REFC signal is used as trigger output (TRGO).

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.

20.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

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

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

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)

See *Table 75: TIMx Internal trigger connection on page 589* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 75. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM2	TIM3	TIM16 OC1	TIM17 OC1

20.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

20.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 20.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

20.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

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

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

20.5.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output Compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: OC1Ref is not affected by the OCREF_CLR input.
1: OC1Ref is cleared as soon as a High level is detected on OCREF_CLR input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode, 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.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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 decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.5.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 76* for details.

Table 76. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>OCx output state</th><th>OCxN output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=0
OCxN=0, OCxN_EN=0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCx=0</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCxN=0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
OCx=CCxP</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF xor CCxP,
OCx_EN=1</td><td>Off-State (output enabled with inactive state)
OCxN=CCxNP, OCxN_EN=1</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: OCx=CCxP, OCxN=CCxNP
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</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

20.5.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

20.5.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

20.5.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 20.4.1: Time-base unit on page 552* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

20.5.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

20.5.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

20.5.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

20.5.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

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

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 AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (*Section 20.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 597*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 20.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 597*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 20.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 597*).

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 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

20.5.17 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

20.5.18 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

20.5.19 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 77. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0

Table 77. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	**TIM15_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value																0									0	0	0	0		0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0			0	0	0	0	0	0			0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	**TIM15_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM15_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OCICE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 77. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**TIM15_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM15_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM15_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM15_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM15_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

20.6 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

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

20.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

20.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

20.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

20.6.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

20.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

20.6.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

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

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: OC1Ref is not affected by the OCREF_CLR input.
1: OC1Ref is cleared as soon as a High level is detected on OCREF_CLR input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.
All other values: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

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 decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

20.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high
1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 78* for details.

Table 78. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP 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 (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

20.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

20.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

20.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 20.4.1: Time-base unit on page 552* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

20.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

20.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

20.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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 AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 20.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 616*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low
1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled
1; Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.
See OC/OCN enable description for more details (*Section 20.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 616*).
0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 20.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 616*).
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 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

20.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

20.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

20.6.17 TIM16 option register (TIM16_OR)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1RMP	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits1:0 **TI1_RMP**: Timer 16 input 1 connection.
This bit is set and cleared by software.
00: TIM16 TI1 is connected to GPIO
01: TIM16 TI1 is connected to RTC_clock
10: TIM16 TI1 is connected to HSE/32
11: TIM16 TI1 is connected to MCO

20.6.18 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 79. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 79. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIMx_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM16_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP [1:0]	
	Reset value																															0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

21 High-Resolution Timer (HRTIM)

21.1 Introduction

The high-resolution timer can generate up to 10 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture allows to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedbacks and synchronization signals. This allows to produce a large variety of control signal (PWM, phase-shifted, constant Ton,...) and address most of conversion topologies.

For control and monitoring purposes, the timer has also timing measure capabilities and links to built-in ADC and DAC converters. Last, it features light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

21.2 Main features

- High-resolution timing units
  - 217 ps resolution, compensated against voltage and temperature variations
  - High-resolution available on all outputs, possibility to adjust duty-cycle, frequency and pulse width in triggered one-pulse mode
  - 6 16-bit timing units (each one with an independent counter and 4 compare units)
  - 10 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with 1 or 2 switches or few large multi-switch topologies
- Up to 10 external events, available for any timing unit
  - Programmable polarity and edge sensitivity
  - 5 events with a fast asynchronous mode
  - 5 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Multiple links to built-in analog peripherals
  - 4 triggers to ADC converters
  - 3 triggers to DAC converters
  - 3 comparators for analog signal conditioning
- Versatile protection scheme
  - 5 fault inputs can be combined and associated to any timing unit
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs
- Versatile output stage
  - High-resolution Deadtime insertion (down to 868 ps)
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 7 interrupt vectors, each one with up to 14 sources
- 6 DMA requests with up to 14 sources, with a burst mode for multiple registers update

21.3 Functional description

21.3.1 General description

The HRTIM can be partitioned into several sub entities:

- The master timer
- The timing units (Timer A to Timer E)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 10 outputs via 4 compare units and it provides synchronization signals to the 5 timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The outputs set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers or from external events.

The output stage has several duties

- Addition of deadtime when the 2 outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 10)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

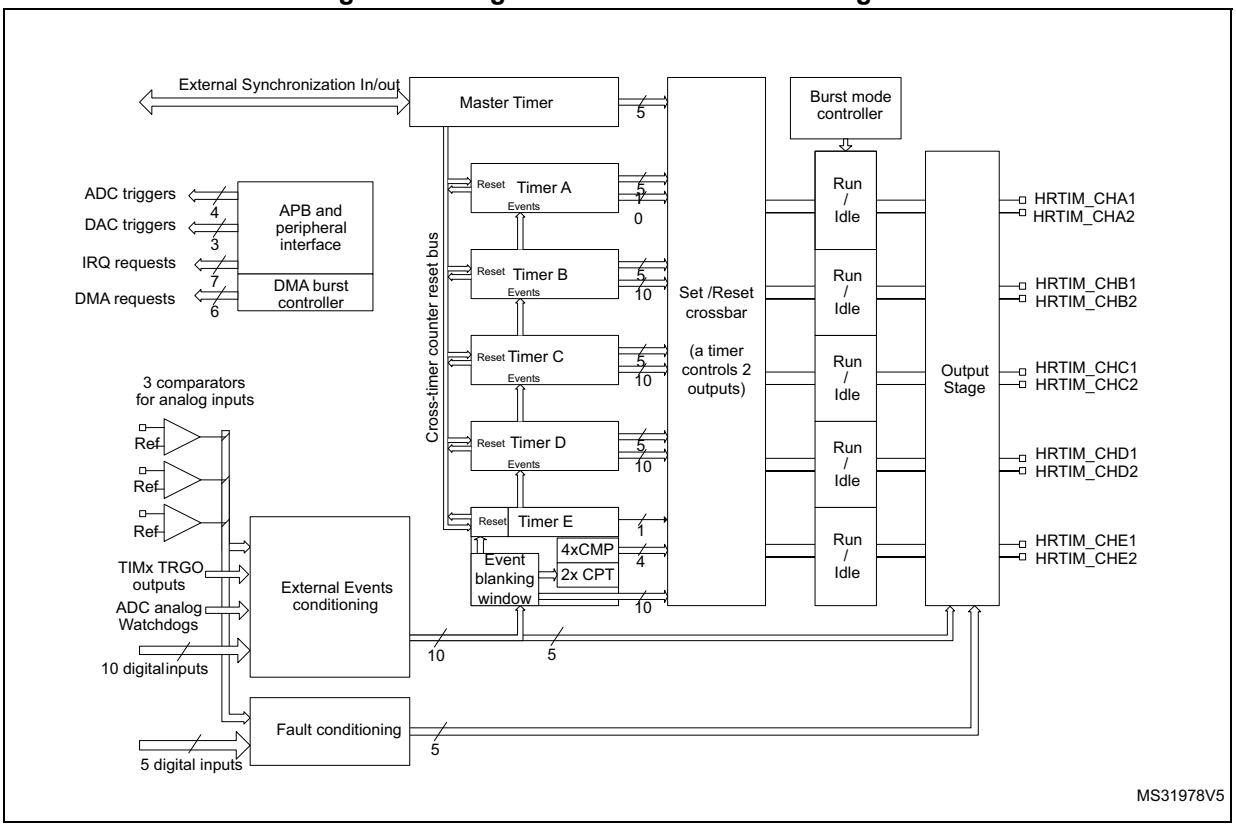
The HRTIM registers are split into 7 groups:

- Master timer registers
- Timer A to Timer E registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the 5 timing units in the text and in registers are generalized using the "x" letter, where x can be any value from A to E.

The block diagram of the timer is shown in *Figure 244*.

Figure 244. High-resolution timer block diagram



21.3.2 HRTIM pins and internal signals

The table here below summarizes the HRTIM inputs and outputs, both on-chip and off-chip.

Table 80. HRTIM Input/output summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
HRTIM_FLT[5:1], HRTIM_FLT_in[5:1]	Digital input	Fault inputs: immediately disable the HRTIM outputs when asserted (5 on-chip inputs and 5 off-chip HRTIM_FLTx inputs).

Table 80. HRTIM Input/output summary (continued)

Signal name	Signal type	Description
SYSFLT	Digital input	System fault gathering MCU internal fault events (Clock security system, SRAM parity error, Cortex®-M4 lockup (HardFault), PVD output).
HRTIM_SCIN[3:1]	Digital Input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: HRTIM_SCIN1: reserved HRTIM_SCIN2: the source is a regular TIMx timer (via on-chip interconnect) HRTIM_SCIN3: the source is an external HRTIM (via the HRTIM_SCIN input pins)
HRTIM_SCOUT[2:1]	Digital output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: HRTIM_SCOUT1: reserved HRTIM_SCOUT2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins)
HRTIM_EEV1[4:1]	Digital input	External events. Each of the 10 events can be selected among 4 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins)
HRTIM_EEV2[4:1]		
HRTIM_EEV3[4:1]		
HRTIM_EEV4[4:1]		
HRTIM_EEV5[4:1]		
HRTIM_EEV6[4:1]		
HRTIM_EEV7[4:1]		
HRTIM_EEV8[4:1]		
HRTIM_EEV9[4:1]		
HRTIM_EEV10[4:1]		
UPD_EN[3:1]	Digital input	HRTIM register update enable inputs (on-chip interconnect) trigger the transfer from shadow to active registers
BMtrig	Digital input	Burst mode trigger event (on-chip interconnect)
BMClk[4:1]	Digital input	Burst mode clock (on-chip interconnect)
ADCtrigOut[4:1]	Digital output	ADC start of conversion triggers
DACtrigOut[3:1]	Digital output	DAC conversion update triggers
IRQ[7:1]	Digital output	Interrupt requests
DMA[6:1]	Digital output	DMA requests

21.3.3 Clocks

The HRTIM must be supplied by the t_{HRTIM} system clock to offer a full resolution. The t_{HRTIM} clock period is evenly divided into up to 32 intermediate steps using an edge positioning logic. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: main HRTIM clock . All subsequent clocks are derived and synchronous with this source.

f_{HRCK}: high-resolution equivalent clock. Considering the f_{HRTIM} clock period division by 32, it is equivalent to a frequency of 144 x 32 = 4.608 GHz.

f_{DTG}: deadtime generator clock. For convenience, only the t_{DTG} period ($t_{DTG} = 1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: chopper stage clock source.

f_{1STPW}: clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period ($t_{1STPW} = 1/f_{1STPW}$) is used in this document.

f_{BRST}: burst mode controller counter clock.

$f_{SAMPLING}$: clock needed to sample the fault or the external events inputs.

f_{FLTS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which allows you to adjust the timer resolution. (See *Table 81*).

Table 81. Timer resolution and min. PWM frequency for f_{HRTIM} = 144 MHz

CKPSC[2:0]	Prescaling ratio	f_{HRCK} equivalent frequency	Resolution	Min PWM frequency
000	1	144 x 32 MHz = 4.608 GHz	217 ps	70.3 kHz
001	2	144 x 16MHz = 2.304 GHz	434 ps	35.1 kHz
010	4	144 x 8MHz = 1.152 GHz	868 ps	17.6 kHz
011	8	144 x 4MHz = 576 MHz	1.73 ns	8.8 kHz
100	16	144 x 2MHz = 288 MHz	3.47 ns	4.4 kHz
101	32	144 MHz	6.95 ns	2.2 kHz
110	64	144/2 MHz = 72 MHz	13.88 ns	1.1 kHz
111	128	144/4 MHz = 36 MHz	27.7 ns	550 Hz

The High-resolution is available for edge positioning, PWM period adjustment and externally triggered pulse duration.

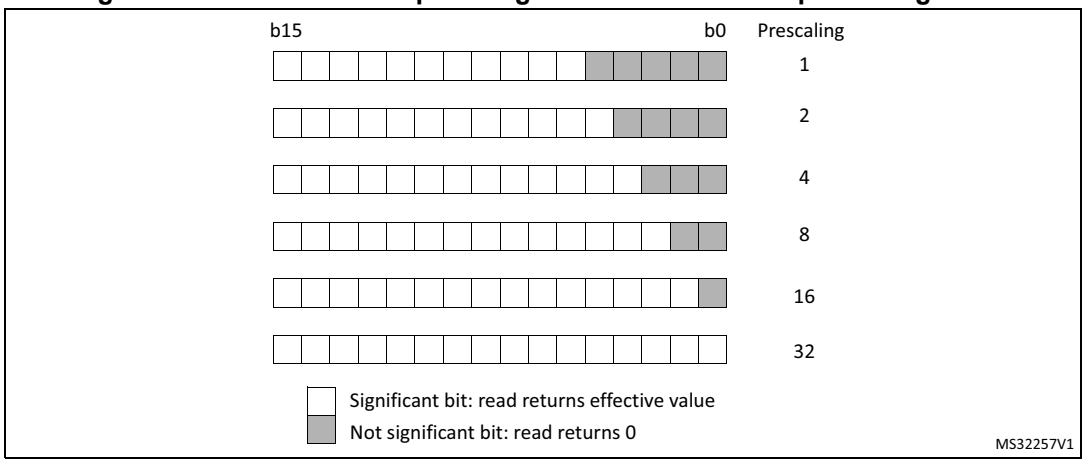
The high-resolution is not available for the following features

- Timer counter read and write accesses
- Capture unit

For clock prescaling ratios below 32 (CKPSC[2:0] <5), the least significant bits of the counter and capture registers are not significant. The least significant bits cannot be written (counter register only) and return 0 when read.

For instance, if CKPSC[2:0] = 2 (prescaling by 4), writing 0xFFFF into the counter register will yield an effective value of 0xFFF0. Conversely, any counter value between 0xFFFF and 0xFFF0 will be read as 0xFFF0.

Figure 245. Counter and capture register format vs clock prescaling factor



Initialization

At start-up, it is mandatory to initialize first the prescaler bitfields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: **It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers' information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter or capture triggers. Prescaler factors not equal will yield to unpredictable results.**

Deadtime generator clock

The deadtime prescaler is supplied by f_{HRTIM} / 8 / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 868 ps to 6.94 ns for f_{HRTIM} = 144 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 31, so that 562.5 kHz <= f_{CHPFRQ} <= 9 MHz for f_{HRTIM} = 144 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (9 MHz for f_{HRTIM}= 144 MHz).

Burst Mode Prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (4.4 kHz for f_{HRTIM} = 144 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from 144 MHz to 18 MHz for f_{HRTIM} = 144 MHz.

External Event input sampling clock

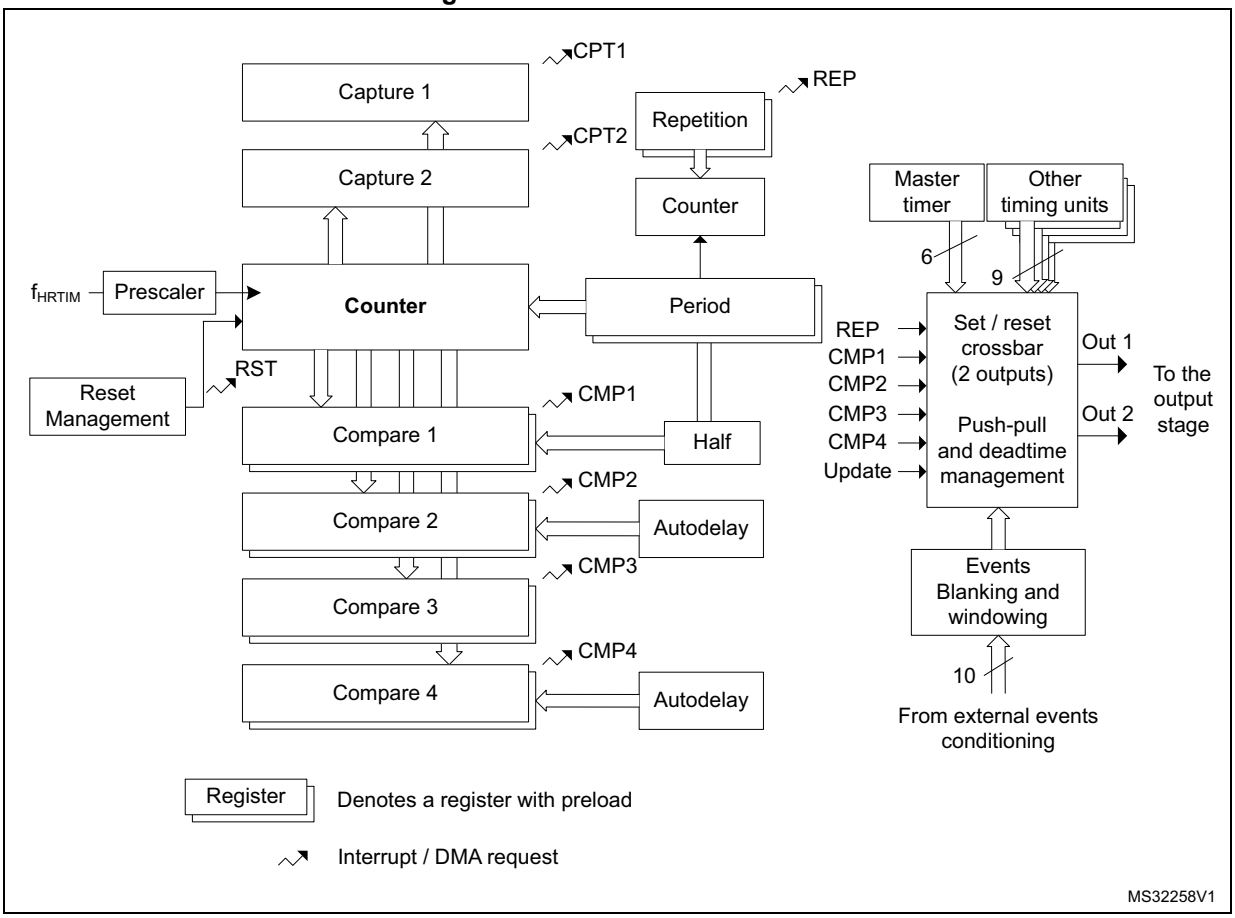
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from 144 MHz to 18 MHz for f_{HRTIM} = 144 MHz.

21.3.4 Timer A..E timing units

The HRTIM embeds 5 identical timing units made of a 16-bit up-counter with an auto-reload mechanism to define the counting period, 4 compare and 2 capture units, as per *Figure 246*. Each unit includes all control features for 2 outputs, so that it can operate as a standalone timer.

Figure 246. Timer A..E overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 82*:

- The minimum value must be greater than or equal to 3 periods of the f_{HRTIM} clock
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock

Table 82. Period and Compare registers min and max values

CKPSC[2:0] value	Min	Max
0	0x0060	0xFFDF
1	0x0030	0xFFEF
2	0x0018	0xFFF7
3	0x000C	0xFFFB

Table 82. Period and Compare registers min and max values (continued)

CKPSC[2:0] value	Min	Max
4	0x0006	0xFFFD
≥ 5	0x0003	0xFFFD

Note: A compare value greater than the period register value will not generate a compare match event.

Counter operating mode

Timer A..E can operate in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or non-retriggerable. Details of operation are summarized on *Table 83* and on *Figure 247* and *Figure 248*.

Table 83. Timer operating modes

CONT	RETRIG	Operating mode	Start / Stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 247. Continuous timer operation

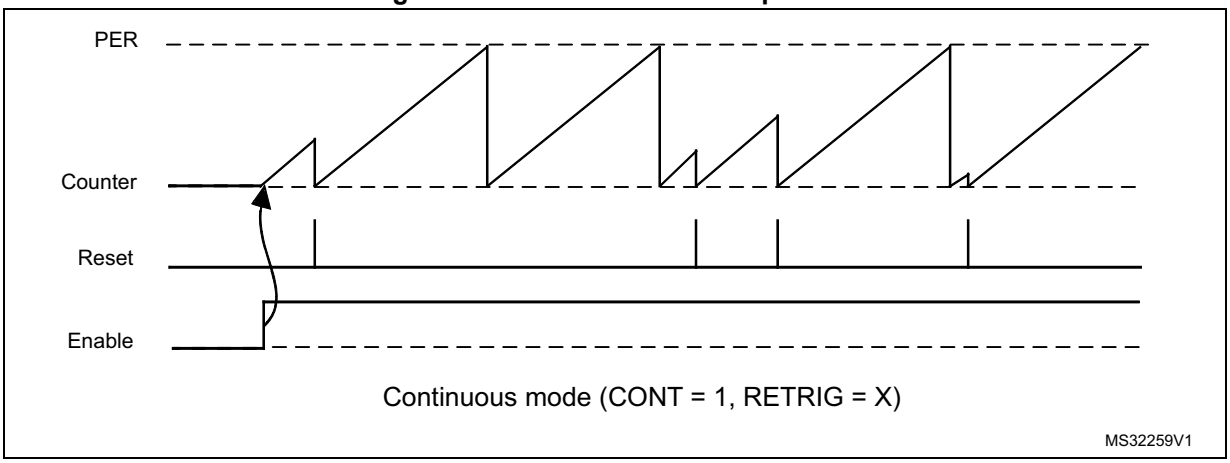
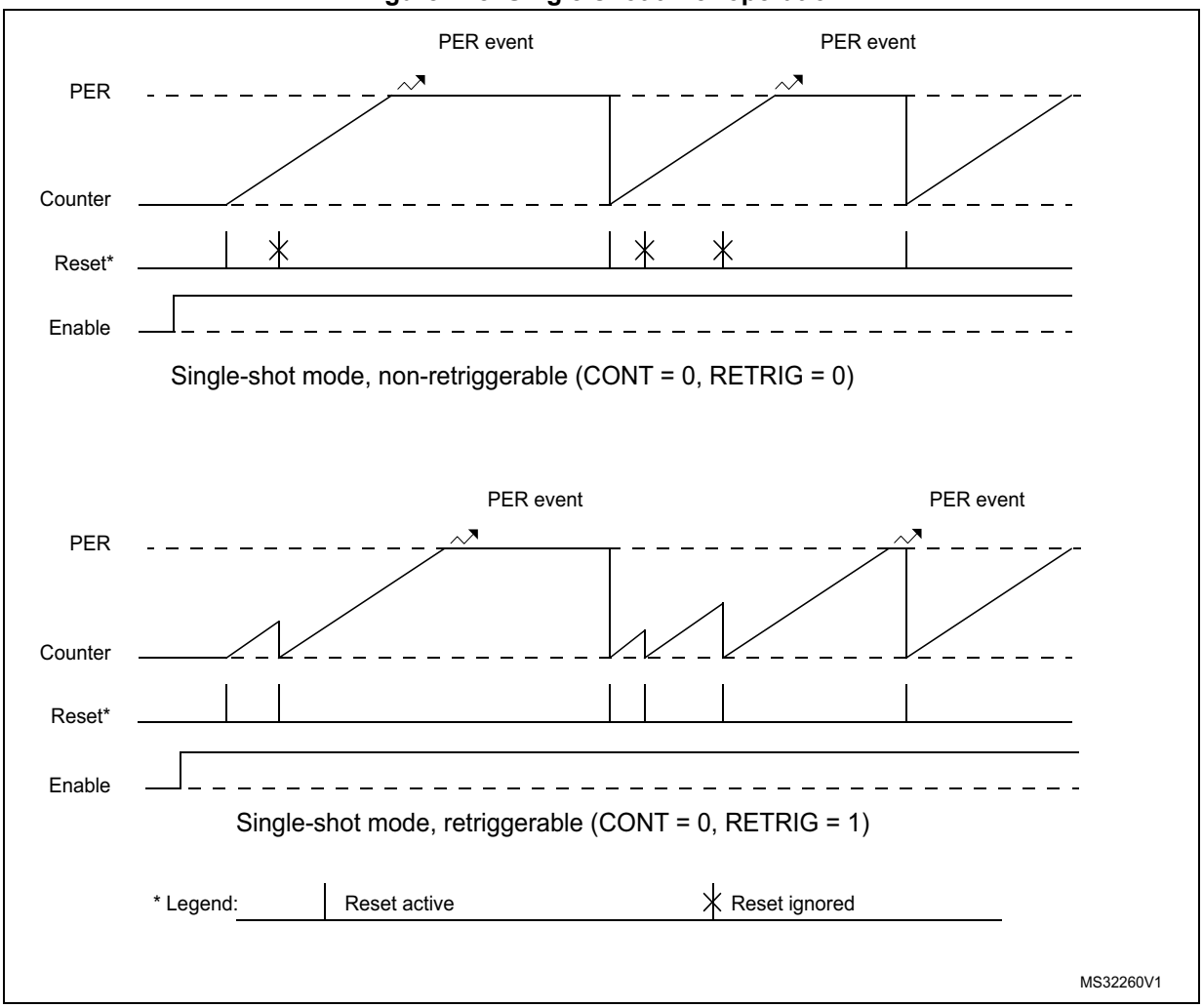


Figure 248. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- as an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it will overflow at the maximum period value and the repetition counter will not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 30 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: Compare 2, Compare 4 and Update (3 events)
- The master timer: Reset and Compare 1..4 (5 events)
- The external events EXTEVNT1..10 (10 events)
- All other timing units (e.g. Timer B..E for timer A): Compare 1, 2 and 4 (12 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When 2 counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

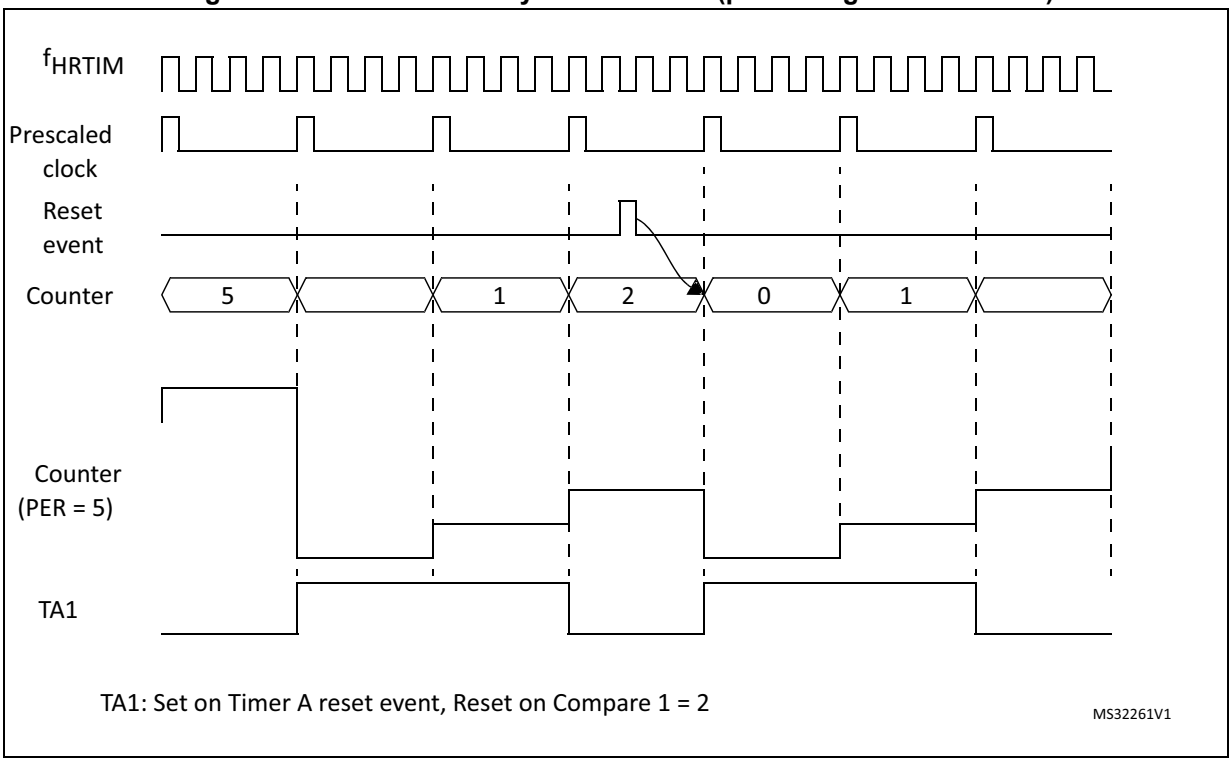
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 32 (counting period above f_{HRTIM}), the counter reset event is delayed to the next active edge of the prescaled clock. This allows to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 249 shows how the reset is handled for a clock prescaling ratio of 128 (f_{HRTIM} divided by 4).

Figure 249. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

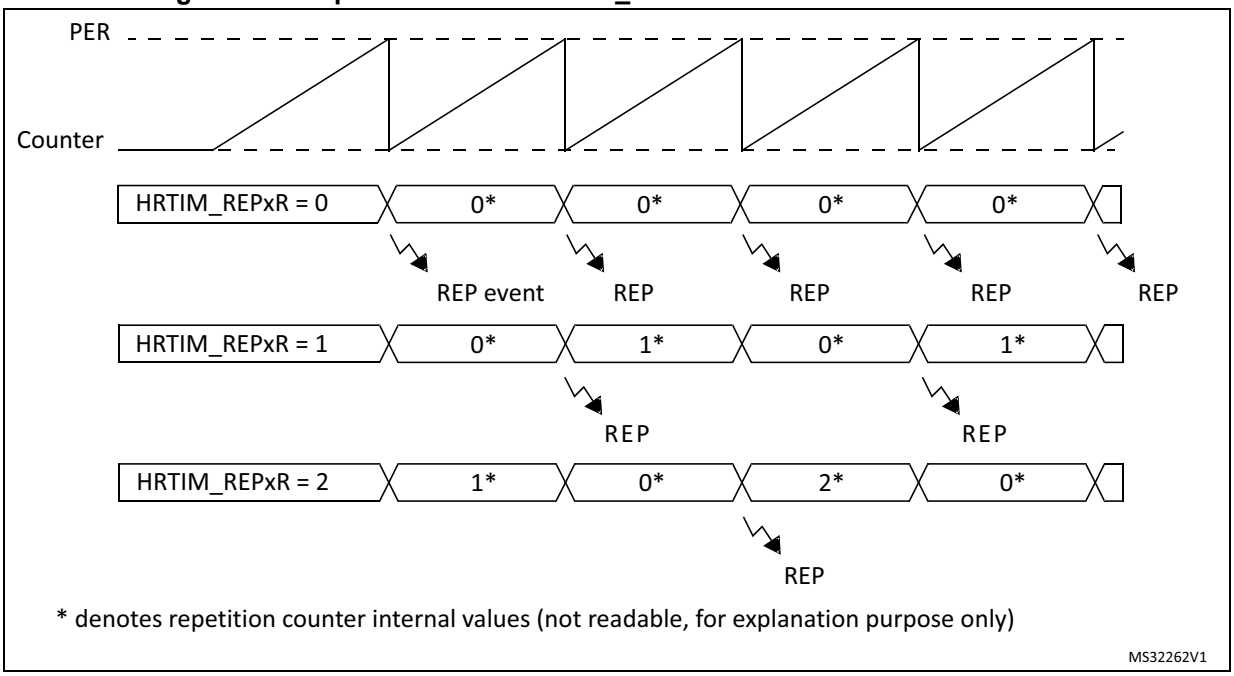
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an auto-reload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, any time the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

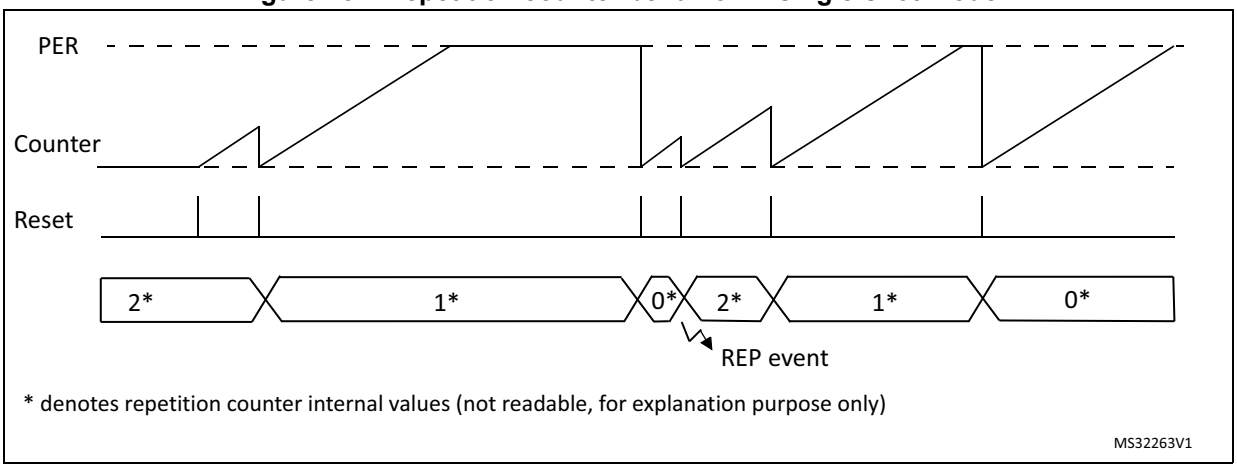
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 250* presents the repetition counter operation for various values, in continuous mode.

Figure 250. Repetition rate vs HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (*Figure 251* here-below). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 251. Repetition counter behavior in single-shot mode



A reset or start event from the HRTIM_SCIN[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter will be decremented only on the 1st reset event following the period. Any subsequent reset will not alter the repetition counter until the counter is re-started by a new request on HRTIM_SCIN[3:1] inputs.

Set / reset crossbar

A “set” event correspond to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.
The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These 2 outputs can be set, reset or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: Period, Compare 1..4, register update (6 events)
- The master timer: Period, Compare 1..4, HRTIM synchronization (6 events)
- All other timing units (e.g. Timer B..E for timer A): TIMEVNT1..9 (9 events described in *Table 84*)
- The external events EXTEVNT1..10 (10 events)
- A software forcing (1 event)

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..E and y stands for the output 1or 2 (e.g. HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it will toggle the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer start-up.

Table 84 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating non-available events.

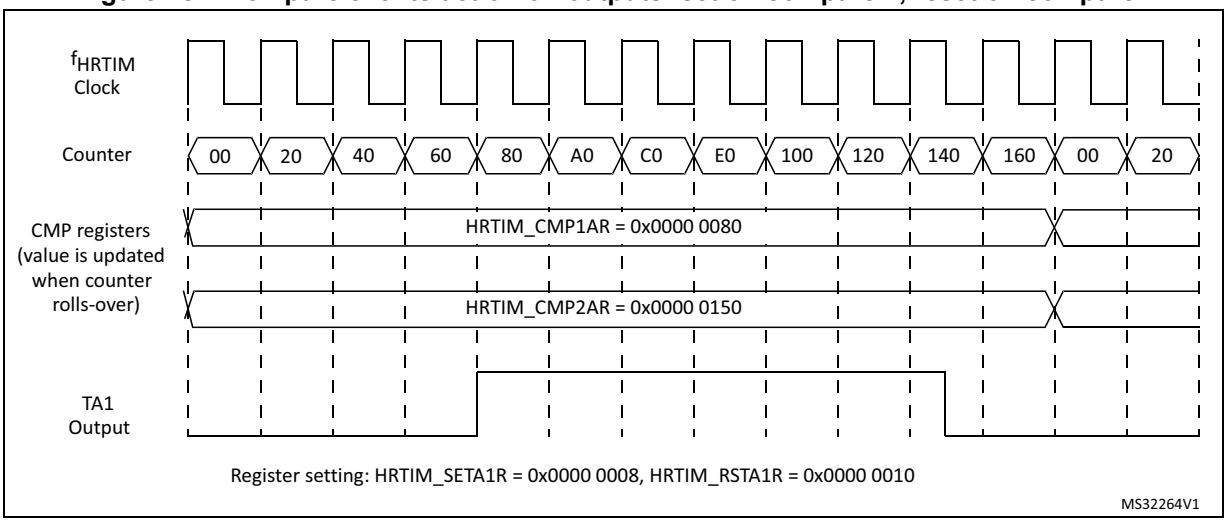
For instance, Timer A outputs can be set or reset by the following events: Timer B Compare1, 2 and 4, Timer C Compare 2 and 3,... and Timer E Compare 3 will be listed as TIMEVNT8 in HRTIM_SETA1R.

Table 84. Events mapping across Timer A to E

Source		Timer A				Timer B				Timer C				Timer D				Timer E			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	Timer A	-	-	-	-	1	2	-	3	-	4	5	-	6	7	-	-	-	-	8	9
	Timer B	1	2	-	3	-	-	-	-	-	-	4	5	-	-	6	7	8	9	-	-
	Timer C	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	7	8	9
	Timer D	1	-	-	2	-	3	-	4	5	-	6	7	-	-	-	-	8	-	-	9
	Timer E	-	-	1	2	-	-	3	4	5	6	-	-	7	8	-	9	-	-	-	-

Figure 252 represents how a PWM signal is generated using two compare events.

Figure 252. Compare events action on outputs: set on compare 1, reset on compare 2



Set/Reset on Update events

A set or reset event on update is done at low resolution. When CKPSC[2:0] < 5, the high-resolution delay is set to its maximum value so that a set/reset event on update will always lag as compared to other compare set/reset events, with a jitter varying between 0 and 31/32 of a f_{HRTIM} clock period.

Half mode

This mode aims at generating square signal with fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It allows to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the Compare 1 value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on CMP1 event, and one transition on the period event, as follows:

– HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or

– HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes Compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR / 2 value.

When the preload is enabled (PREEN = 1, MUDIS, TxUDIS), Compare 1 active register is refreshed on the Update event. If the preload is disabled (PREEN= 0), Compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to 6 periods of the f_{HRTIM} clock (0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2,...) when the HALF mode is enabled.

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- measure events arrival timings or occurrence intervals
- update Compare 2 and Compare 4 values in auto-delayed mode (see *Auto-delayed mode*).

The capture is done with f_{HRTIM} resolution: for a clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the register are not significant (read as 0).

The timer has 2 capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

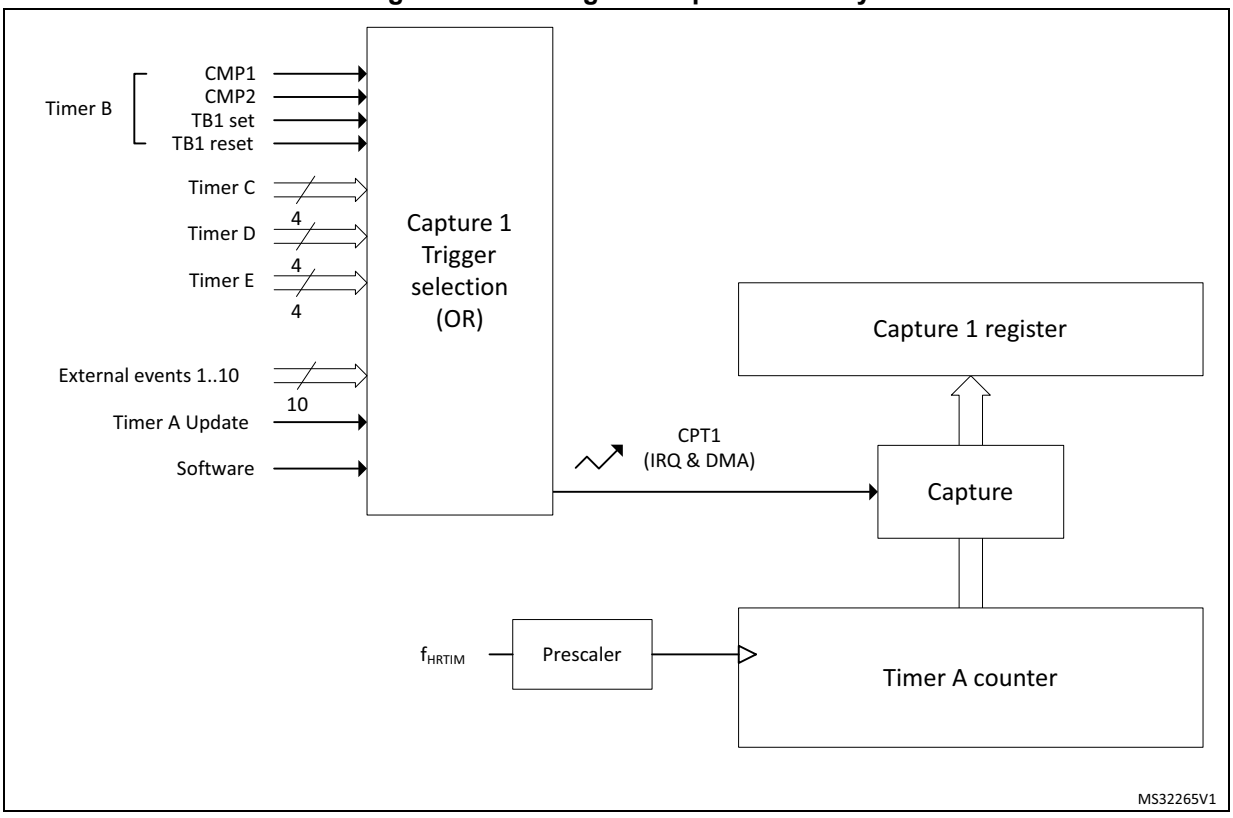
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNT1..10 (10 events)
- All other timing units (e.g. Timer B..E for timer A): Compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: Update (1 event)
- A software capture (1 event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Over-capture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 253. Timing unit capture circuitry



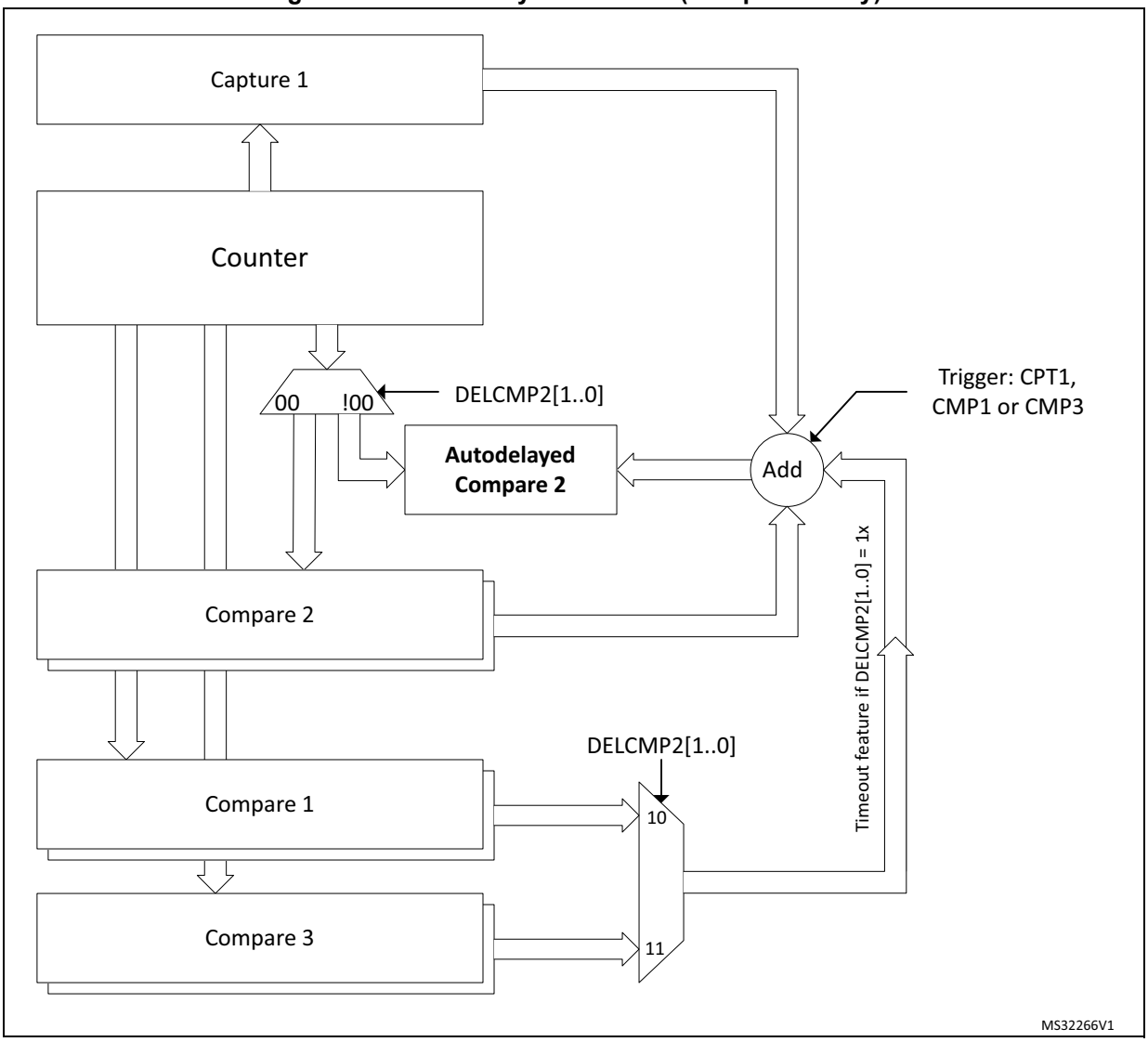
Auto-delayed mode

This mode allows to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the Compare value. Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal auto-delayed compare register, as seen on *Figure 254*. The auto-delayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for Compare 2 and Compare 4 registers. Compare 2 is associated with capture 1, while Compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR Compares cannot be programmed with a value below 3 f_{HRTIM} clock periods, as in the regular mode.

Figure 254. Auto-delayed overview (Compare 2 only)



The auto-delayed Compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is re-armed with Compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register allow to configure the auto-delayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Auto-delayed mode: Compare 2 and Compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.

- 1X
  Auto-delayed mode with timeout: Compare 2 and Compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a Compare 1 match (DELCMPx[1:0]= 10) or a Compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the 2 compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the auto-delayed summation is above 0xFFFF (overflow), the value is ignored and no compare event will be generated until a new period is started.

Note: DELCMPx[1:0] bitfield must be reset when reprogrammed from one value to the other to re-initialize properly the auto-delayed mechanism, for instance:

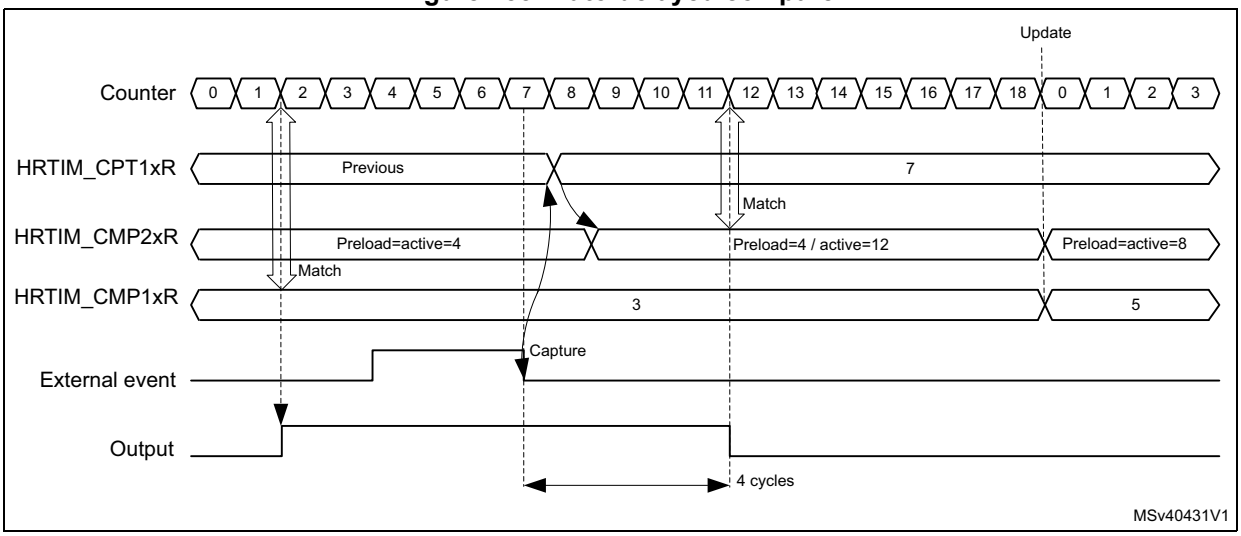
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 255* shows how the following signal can be generated:

- Output set when the counter is equal to Compare 1 value
- Output reset 4 cycles after a falling edge on a given external event

Note: To simplify the figure, the high-resolution is not used in this example (CKPSC[2:0] = 101), thus the counter is incremented at the f_{HRTIM} rate. Similarly, the external event signal is shown without any resynchronization delay: practically, there is a delay of 1 to 2 f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 255. Auto-delayed compare



A regular compare channel (e.g. Compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the auto-delayed value 4 is summed to the capture equal to 7 to give a value of 12 in the auto-delayed compare register. From this time on, the compare event can be generated and will happen when the counter is equal to 12, causing the output to be reset.

Overcapture management in auto-delayed mode

Overcapture is prevented when the auto-delayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the auto-delayed compare value. A new capture is possible only:

- Once the auto-delayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing auto-delayed compare values

When the auto-delayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (see *Figure 255*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, Compare 2 event will occur at = 0x140

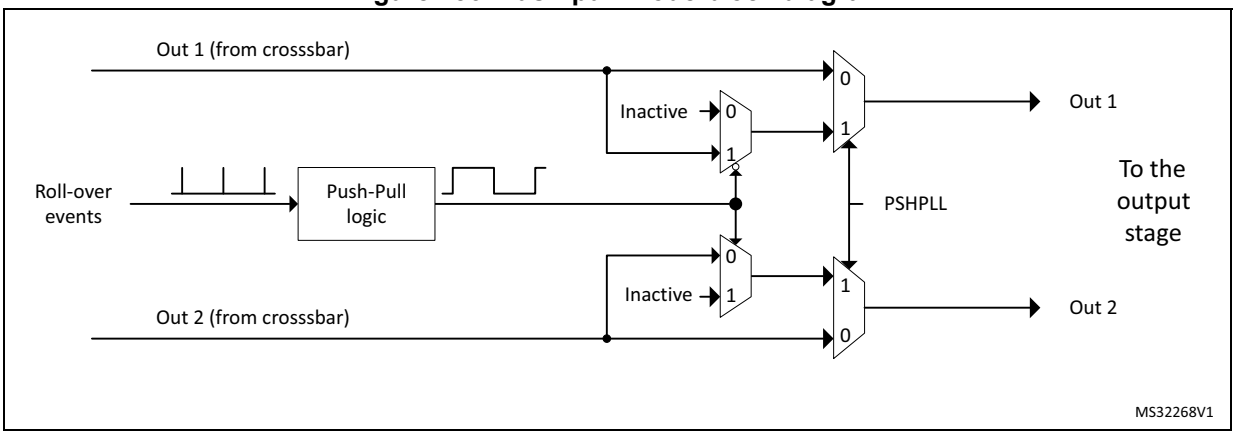
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 21.3.9: Delayed Protection*).

The push-pull mode is enabled by setting PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown on *Figure 256*. The push-pull period is twice the timer counting period.

Figure 256. Push-pull mode block diagram



The push-pull mode is only available when the timer operates in continuous mode: the counter must not be reset once it has been enabled (TxCEN bit set). It is necessary to disable the timer to stop a push-pull operation and to reset the counter before re-enabling it.

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

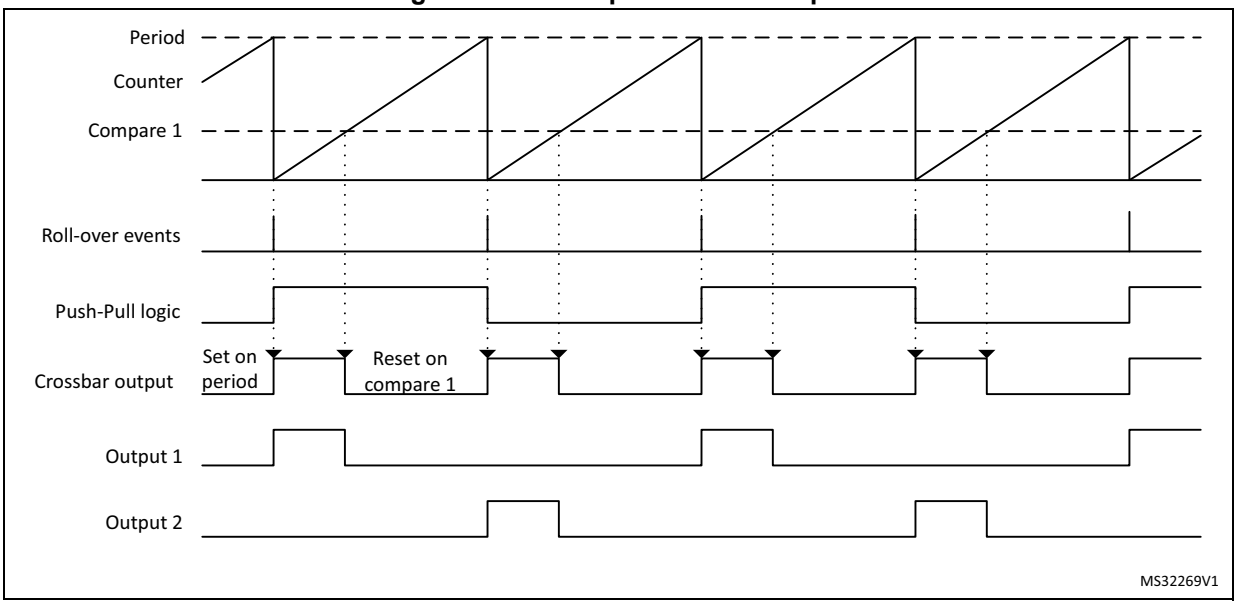
Note: The push-pull operation cannot be used when a deadtime is enabled (mutually exclusive functions).

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given on *Figure 257*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on Compare 1 match event

Figure 257. Push-pull mode example



Deadtime

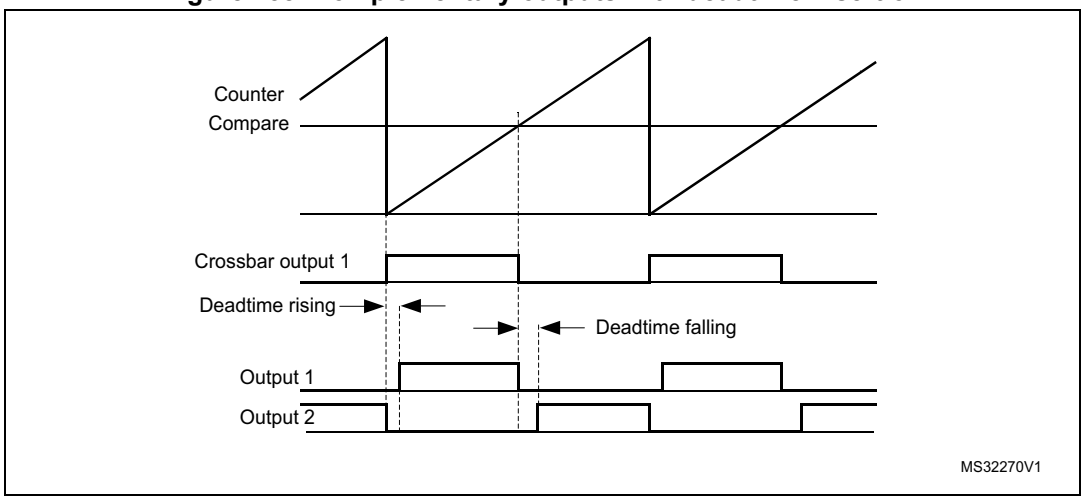
A deadtime insertion unit allows to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only 1 waveform is programmed and controlled to drive two outputs.

The Dead time insertion is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set.

Note: The deadtime cannot be used simultaneously with the push-pull mode.

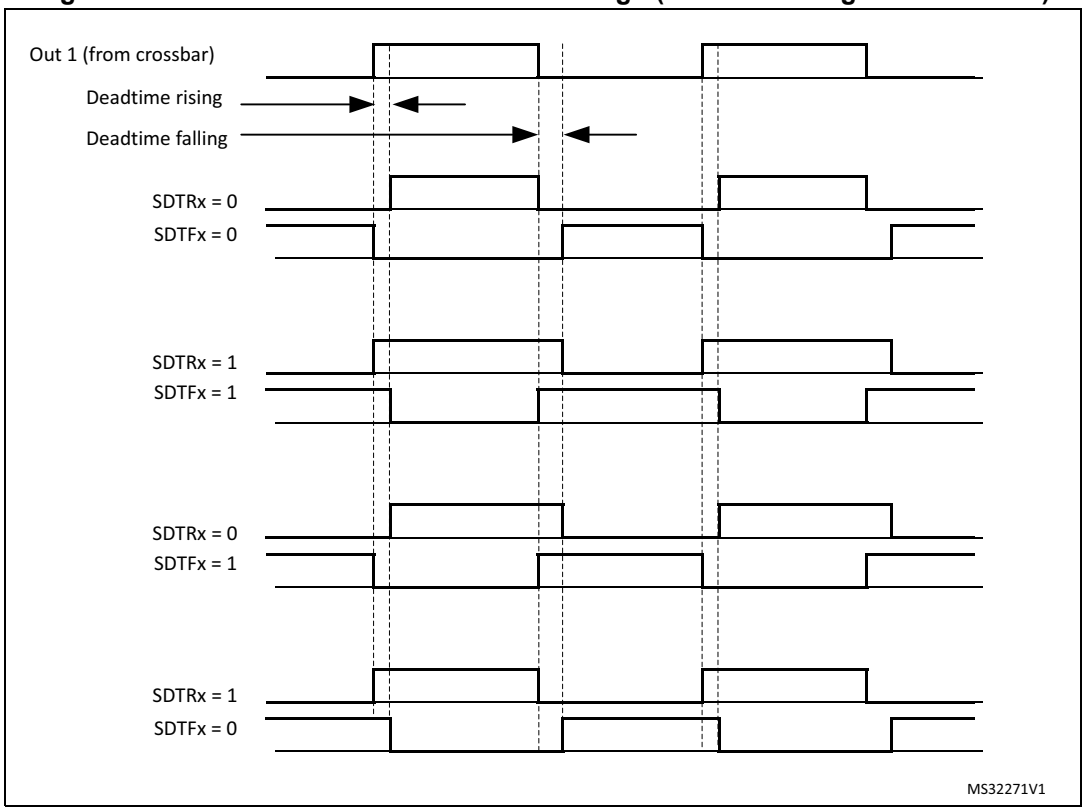
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 258*.

Figure 258. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 259* shows complementary signal waveforms depending on respective signs.

Figure 259. Deadtime insertion vs deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bitfields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

t_{DTx} = +/- DTx[8:0] x t_{DTG}

where x is either R or F and t_{DTG} = ($2^{(DTPRSC[2:0])}$) x (t_{HRTIM} / 8).

Table 85 gives the resolution and maximum absolute values depending on the prescaler value.

Table 85. Deadtime resolution and max absolute values

DTPRSC[2:0]	t_{DTG}	t_{DTx} max	f_{HRTIM}= 144MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
000	t_{HRTIM} / 8	511 * t_{DTG}	0.87	0.44
001	t_{HRTIM} / 4		1.74	0.89
010	t_{HRTIM} / 2		3.47	1.77
011	t_{HRTIM}		6.94	3.54
100	2 * t_{HRTIM}		13.89	7.10
101	4 * t_{HRTIM}		27.78	14.19
110	8 * t_{HRTIM}		55.55	28.39
111	16 * t_{HRTIM}		111.10	56.77

Figure 260 to *Figure 263* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 260. Complementary outputs for low pulse width (SDTRx = SDTFx = 0)

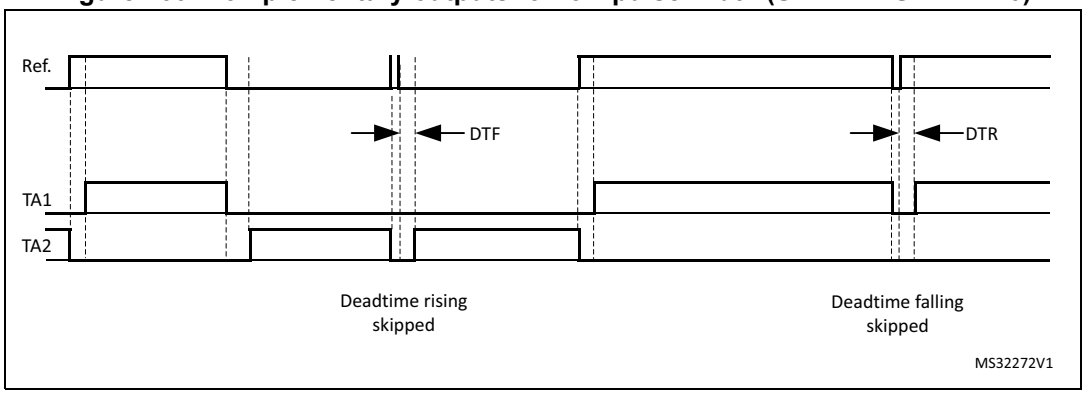


Figure 261. Complementary outputs for low pulse width (SDTRx = SDTFx = 1)

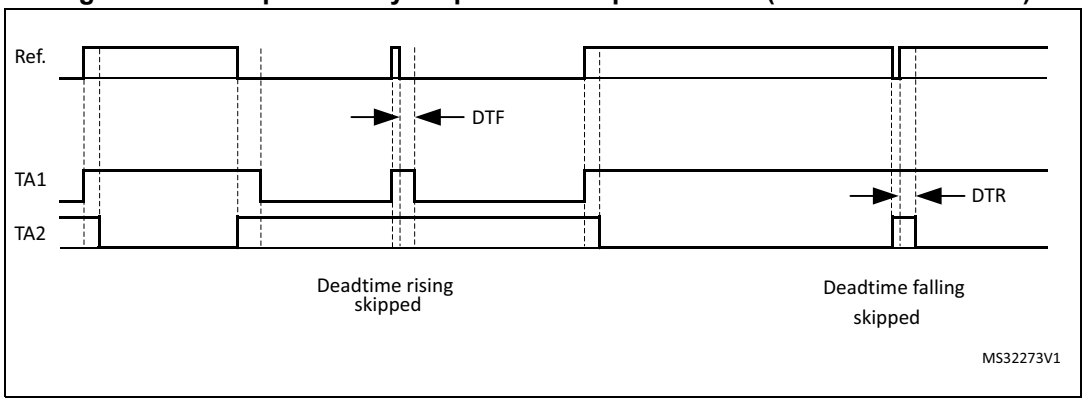


Figure 262. Complementary outputs for low pulse width (SDTRx = 0, SDTFx = 1)

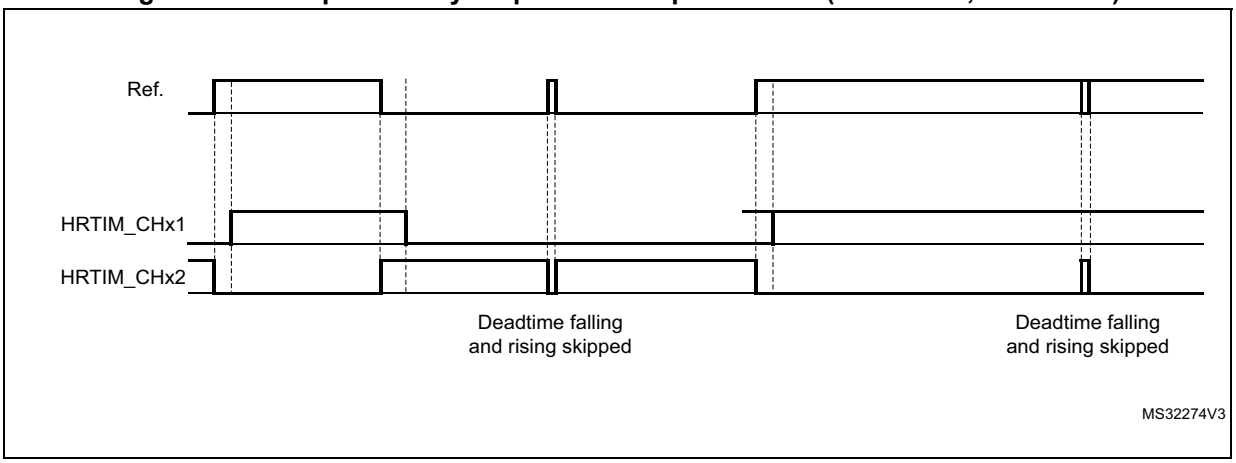
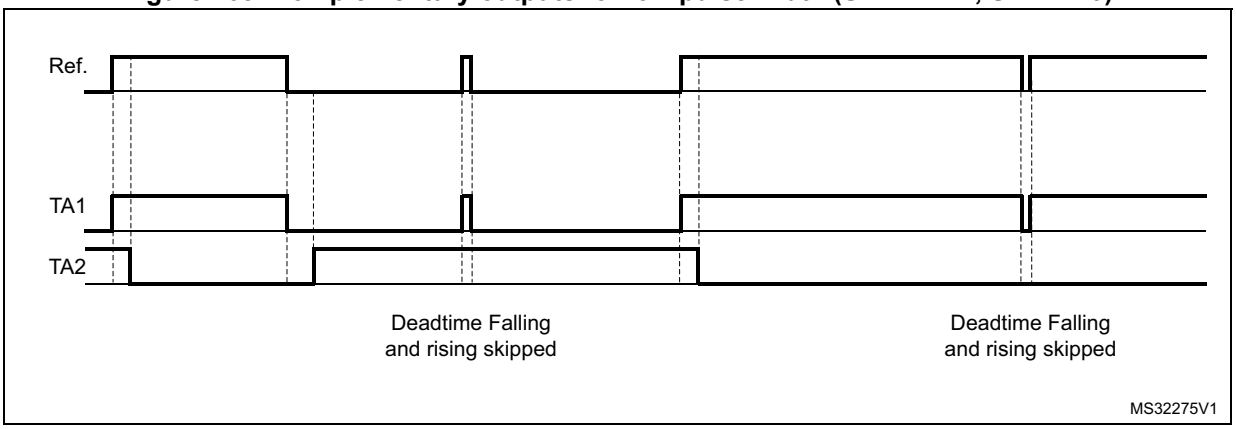


Figure 263. Complementary outputs for low pulse width (SDTRx = 1, SDTFx=0)



For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx and DTRSLKx. Once these bits are set, the related bits and bitfields are becoming read only until the next system reset.

Caution: DTEN bit must not be changed in the following cases:
- When the timer is enabled (TxEN bit set)
- When the timer outputs are set/reset by another timer (while TxEN is reset)

Otherwise, an unpredictable behavior would result.
It is therefore necessary to disable the timer (TxCEN bit reset) and have the corresponding outputs disabled.

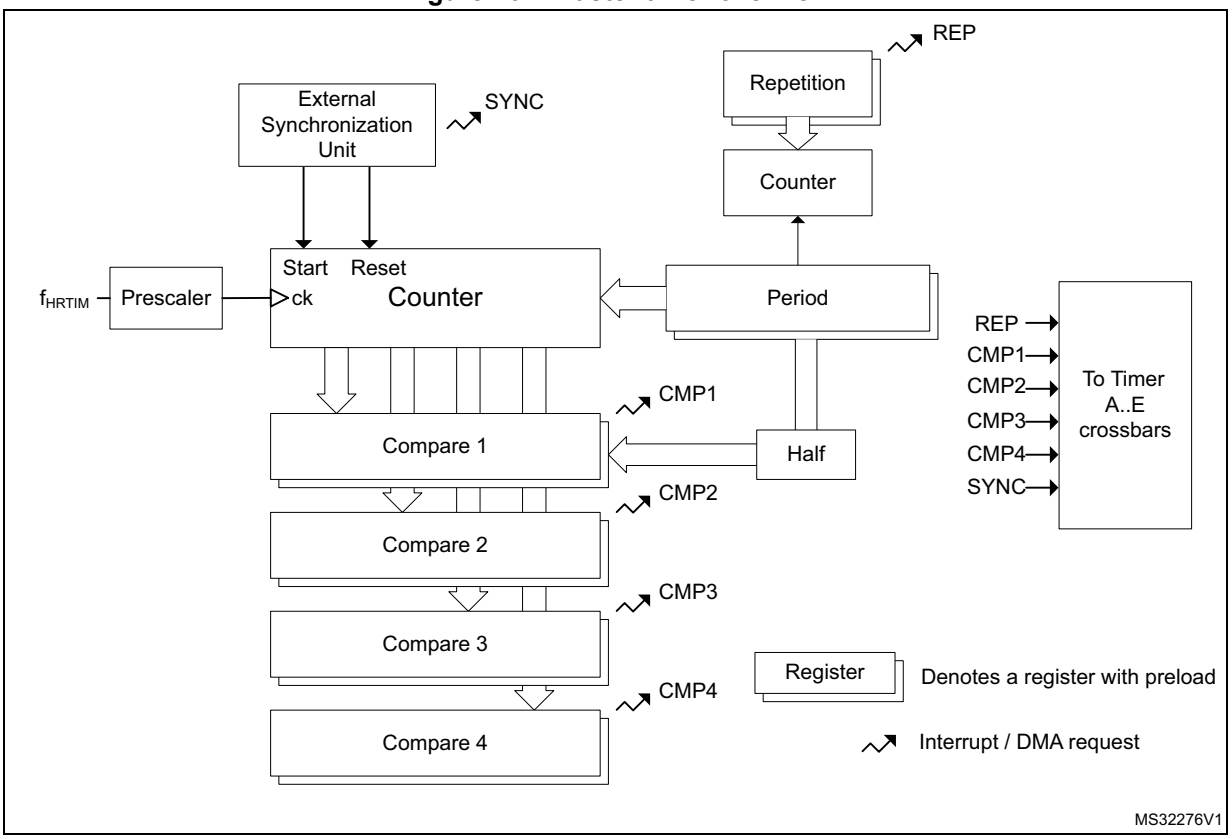
For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

21.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 5 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 264 provides an overview of the master timer.

Figure 264. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control
- It does not have its own crossbar unit, nor push-pull or deadtime mode
- It can only be reset by the external synchronization circuitry
- It does not have a capture unit, nor the auto-delayed mode
- It does not include external event blanking and windowing circuitry
- It has a limited set of interrupt / DMA requests: Compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and Timer A..E timing units. This allows to have all timer synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (see *Section 21.3.17: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units' registers.

21.3.6 Set/reset events priorities and narrow pulses management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

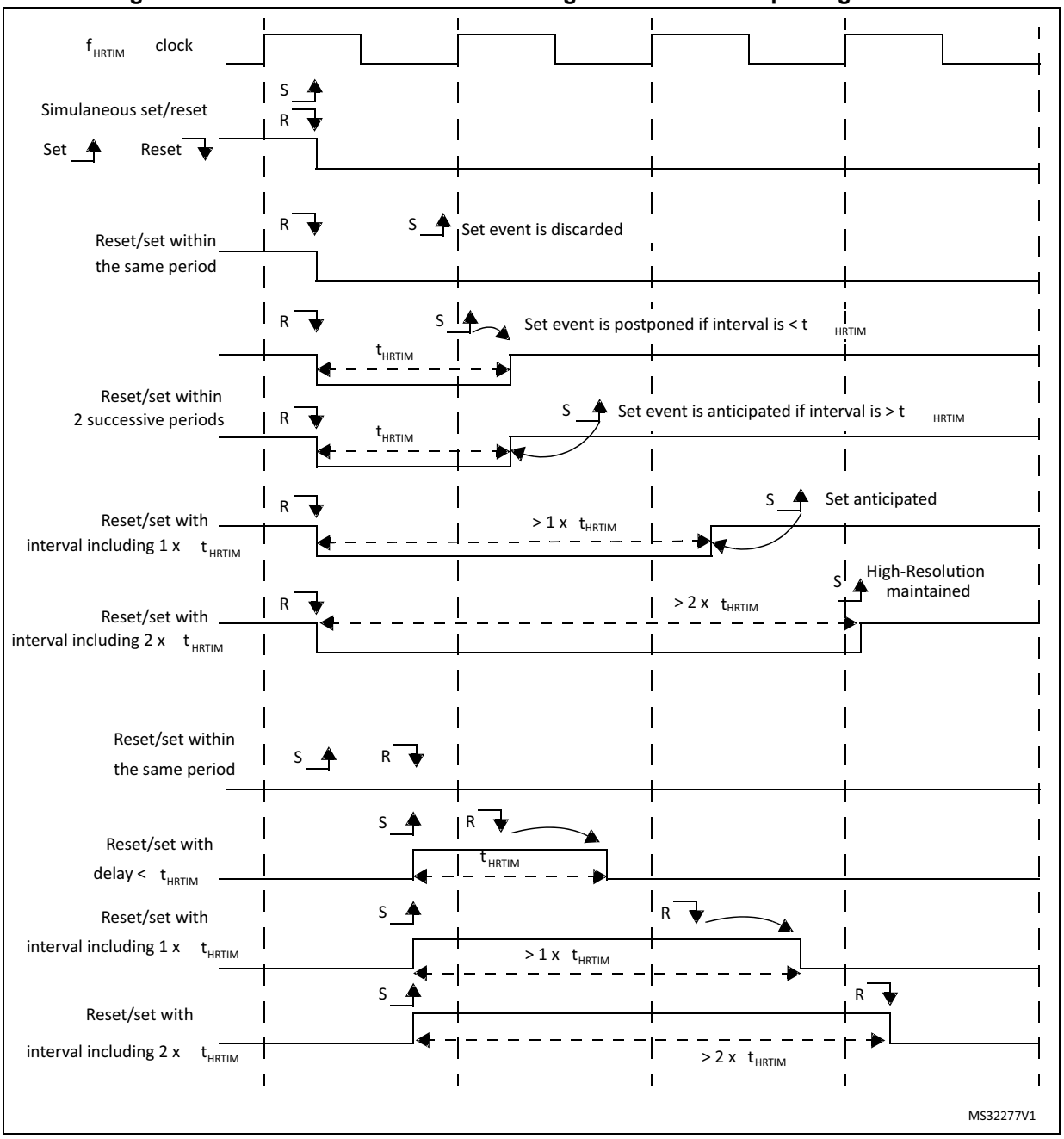
Case 1: clock prescaler CKPSC[2:0] < 5

An arbitration is performed during each t_{HRTIM} period, in 3 steps:

1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events (from highest to lowest priority CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1 ⟶ PER, see *Concurrent set request / Concurrent reset requests*.
3. A high-resolution delay-based arbitration is performed with reset having the highest priority, among the low-resolution events and events having won the predefined arbitration.

When set and reset requests from two different sources are simultaneous, the reset action has the highest priority. If the interval between set and reset requests is below 2 f_{HRTIM} period, the behavior depends on the time interval and on the alignment with the f_{HRTIM} clock, as shown on *Figure 265*.

Figure 265. Short distance set/reset management for narrow pulse generation



If the set and reset events are generated within the same t_{HRTIM} period, the reset event has the highest priority and the set event is ignored.

If the set and reset events are generated with an interval below t_{HRTIM} period, across 2 periods, a pulse of 1 t_{HRTIM} period is generated.

If the set and reset events are generated with an interval below 2 t_{HRTIM} periods, a pulse of 2 t_{HRTIM} periods is generated.

If the set and reset events are generated with an interval between 2 and 3 t_{HRTIM} periods, the high-resolution is available if the interval is over 2 complete t_{HRTIM} periods.

If the set and reset events are generated with an interval above 3 t_{HRTIM} periods, the high-resolution is always available.

Concurrent set request / Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending on:

1. the source (CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1),
2. the delay.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request will be taken into account, whatever the effective high-resolution setting (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

Note: It is advised to avoid generating multiple set (reset) requests from the master timer to a given timer with an interval below 3x t_{HRTIM} to maintain the high-resolution.

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest):

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance only when using auto-delayed Compare 2 and Compare 4 simultaneously (i.e. when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to low-resolution events: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST). The update event is considered as having the largest delay (0x1F if PSC = 0).

As a summary, in case of a close vicinity (events occurring within the same f_{HRTIM} clock period), the effective set (reset) event will be arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The “low-resolution events”.

The same arbitration principle applies for concurrent reset requests. In this case, the reset request has the highest priority.

Case 2: clock prescaler CKPSC[2:0] ≥ 5

The narrow pulse management is simplified when the high-resolution is not effective.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event will be considered.

21.3.7 External events global conditioning

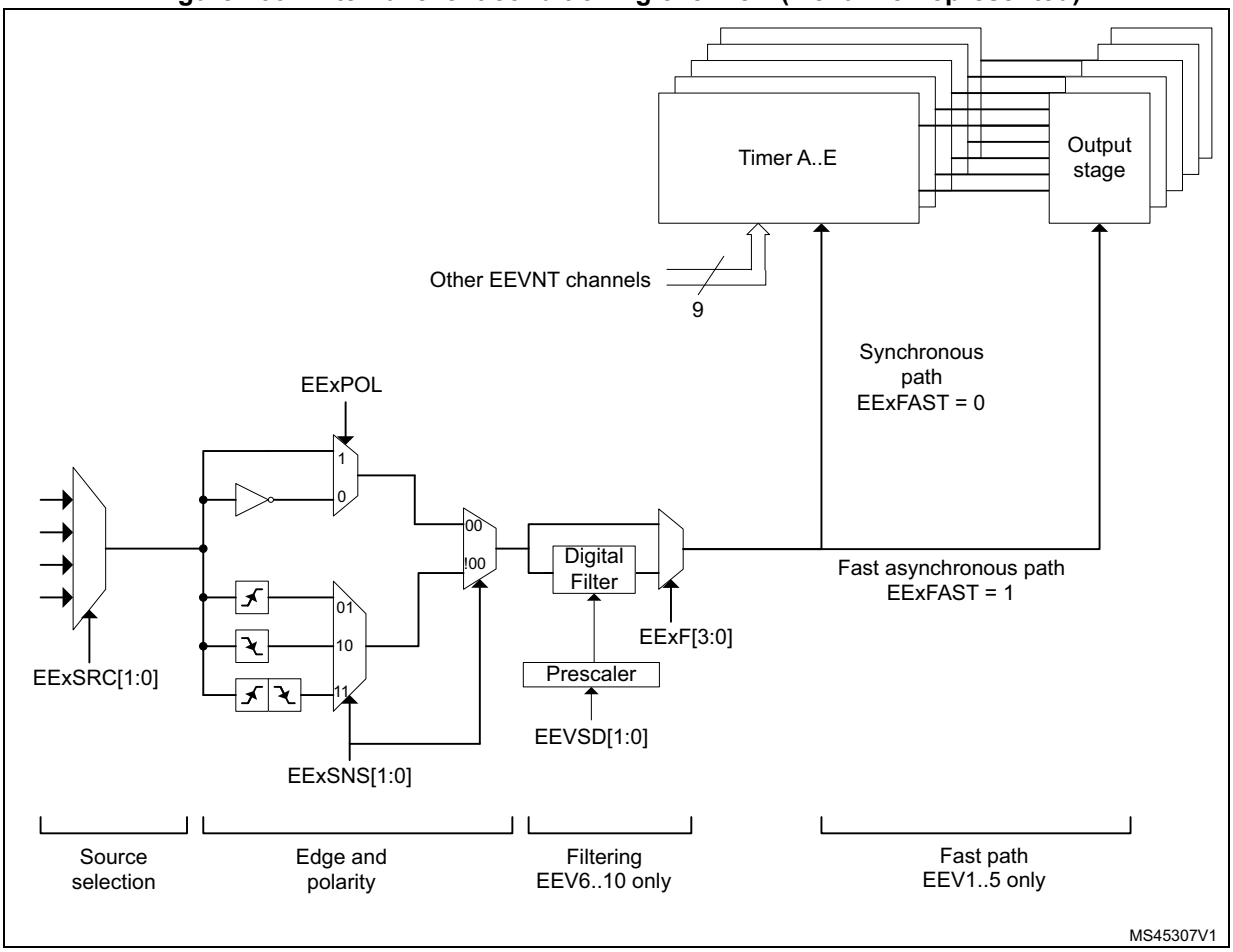
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- built-in comparators,
- digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- on-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 4:1 multiplexer) and to convert it into an information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 10 external event channels can be conditioned and are available simultaneously for any of the 5 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up will be related to the applications noise level and signature). *Figure 266* presents an overview of the conditioning logic for a single channel.

Figure 266. External event conditioning overview (1 channel represented)



The 10 external events are initialized using the HRTIM_EECR1 and HRTIM EECR2 registers:

- to select up to 4 sources with the EExSRC[1:0] bits,
- to select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- to select the polarity, in case of a level sensitivity, with EExPOL bit,
- to have a low latency mode, with EExFAST bits (see *Latency to external events*), for external events 1 to 5.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10, using EExF[3:0] bits in the HRTIM_EECR3 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 86 summarizes the available sources and features associated with each of the 10 external events channels.

Table 86. External events mapping and associated features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E	Src1	Src 2	Src3	Src4	Comparator and input sources available per package		
									32-pin	48-pin	64-pin
1	Yes	-	-	-	PC12	COMP2	TIM1_TRGO	ADC1_AWD1	Comp	Comp	Comp & Input
2	Yes	-	-	-	PC11	COMP4	TIM2_TRGO	ADC1_AWD2	Comp	Comp	Comp & Input
3	Yes	-	-	-	PB7	COMP6	TIM3_TRGO	ADC1_AWD3	Input	Comp & Input	Comp & Input
4	Yes	-	-	-	PB6	OPAMP2 (1)	-	ADC2_AWD1	OPAMP & Input	OPAMP & Input	OPAMP & Input
5	Yes	-	-	-	PB9	-	-	ADC2_AWD2	-	Input	Input
6	-	Yes	Yes	-	PB5	COMP2	TIM6_TRGO	ADC2_AWD3	Comp & Input	Comp & Input	Comp & Input

Table 86. External events mapping and associated features (continued)

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E	Src1	Src 2	Src3	Src4	Comparator and input sources available per package		
									32-pin	48-pin	64-pin
7	-	Yes	Yes	-	PB4	COMP4	TIM7_ TRGO	-	Comp & Input	Comp & Input	Comp & Input
8	-	Yes	-	Yes	PB8	COMP6	-	-	-	Comp & Input	Comp & Input
9	-	Yes	-	Yes	PB3	OPAMP2 (1)	TIM15_ TRGO	-	OPAMP & Input	OPAMP & Input	OPAMP & Input
10	-	Yes	-	-	PC6	-	-	-	-	-	Input

1. OPAMP2_OUT can be used as High-resolution timer internal event source. In this case, the software must set OPAMP2_DIG as of PA6 alternate function (AF13) to redirect OPAMP2_VOUT signal to the HRTIM external events through the Schmitt trigger.

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.
- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 register allow to define the operating for channels 1 to 5. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 87. Output set/reset latency and jitter vs external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to external events (both blanking and windowing with EExFLTR[3:0] != 0000, see *Section 21.3.8*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: if must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 267 and *Figure 268* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 267. Latency to external events falling edge (counter reset and output set)

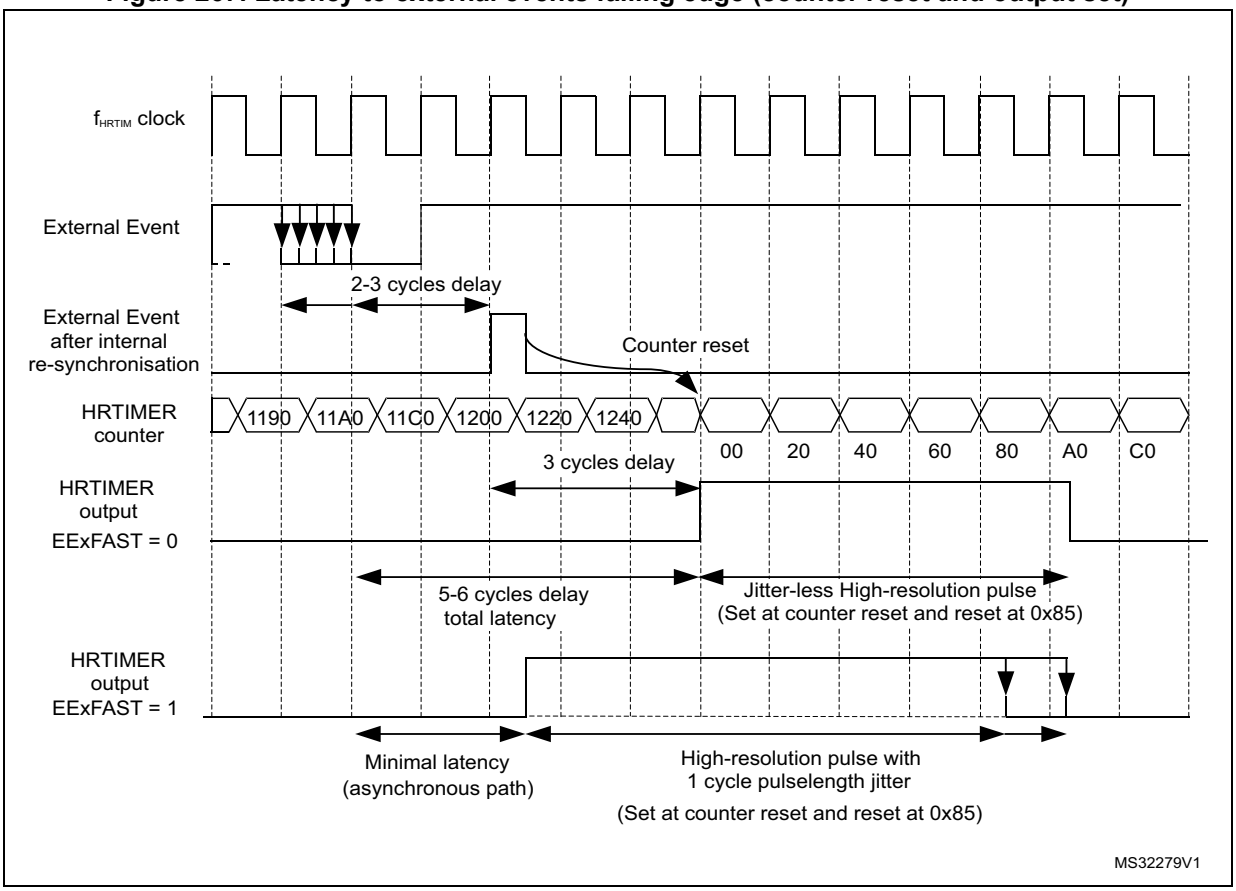
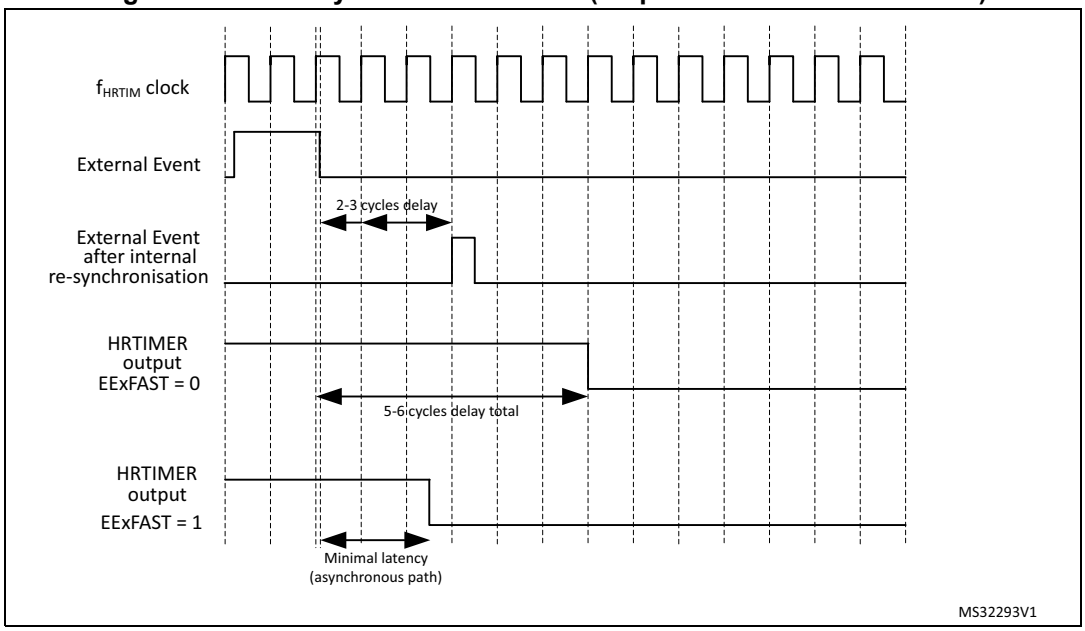


Figure 268. Latency to external events (output reset on external event)



21.3.8 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

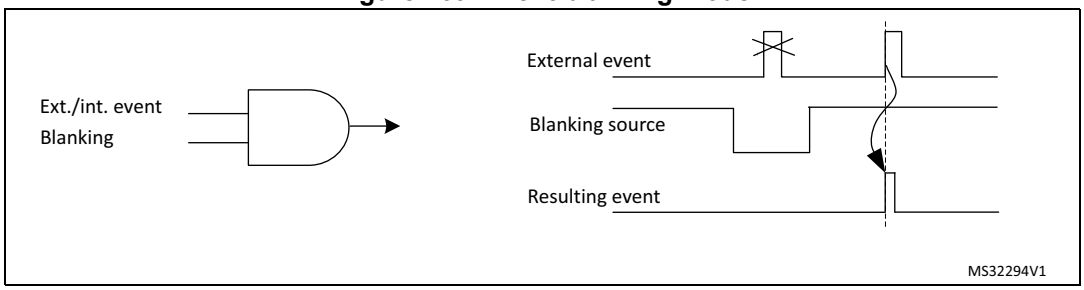
- blanking, to mask external events during a defined time period,
- windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[3:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 5 TimerA..E timing units has its own programmable filter settings for the 10 external events.

Blanking mode

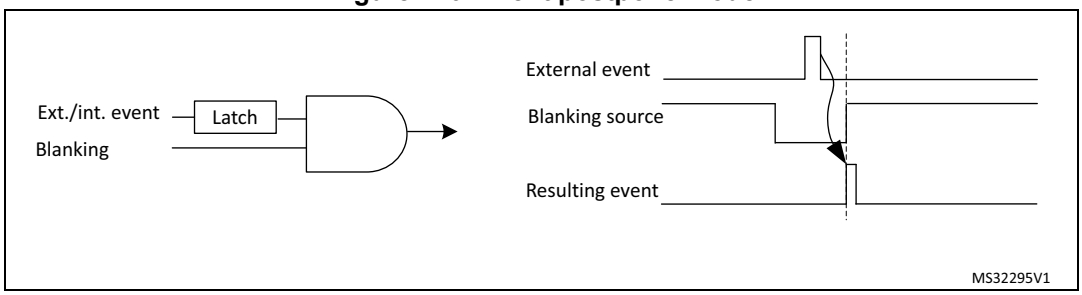
In event blanking mode (see *Figure 269*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[3:0] bitfield values ranging from 0001 to 1100.

Figure 269. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown on *Figure 270*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 270. Event postpone mode



The blanking signal comes from several sources:

- the timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[3:0] = 0001 to 0100 for Compare 1 to Compare 4)
- from other timing units (EExFLTR[3:0] = 0101 to 1100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EEXFLTR[3:0] configurations from 0101 to 1100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 88* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, Timer B (TIMFLTR6) is Timer C output 2 waveform.

Table 88. Filtering signals mapping per time

	Source	Timer A				Timer B				Timer C				Timer D				Timer E			
		CMP 1	CMP 2	CMP 4	TA2	CMP 1	CMP 2	CMP 4	TB2	CMP 1	CMP 2	CMP 4	TC2	CMP 1	CMP 2	CMP 4	TD2	CMP 1	CMP 2	CMP 4	TE2
Destination	Timer A	-	-	-	-	1	-	2	3	4	-	5	6	7	-	-	-	-	8	-	-
	Timer B	1	-	2	3	-	-	-	-	4	5	-	6	-	7	-	-	8	-	-	-
	Timer C	-	1	-	-	2	-	3	4	-	-	-	-	5	-	6	7	-	-	8	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-	-	-	-	6	-	7	8
	Timer E	-	1	-	-	2	-	-	-	3	-	4	5	6	-	7	8	-	-	-	-

Figure 271 and *Figure 272* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 271. External trigger blanking with edge-sensitive trigger

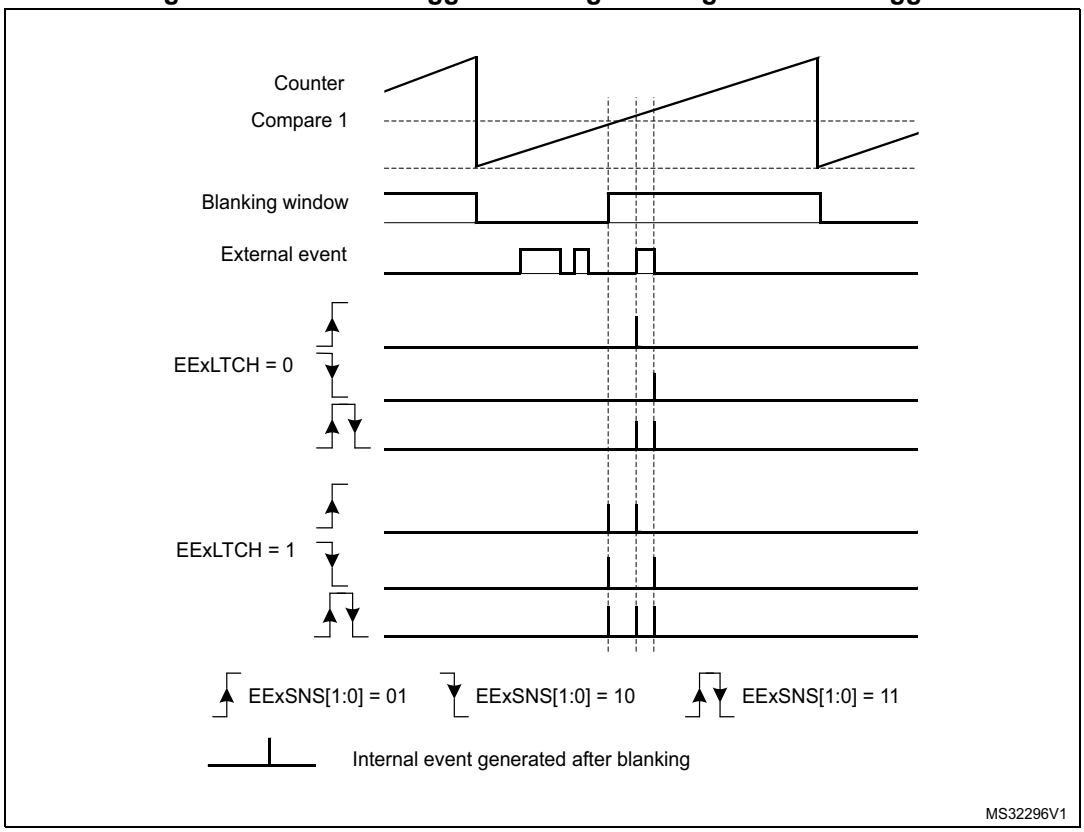
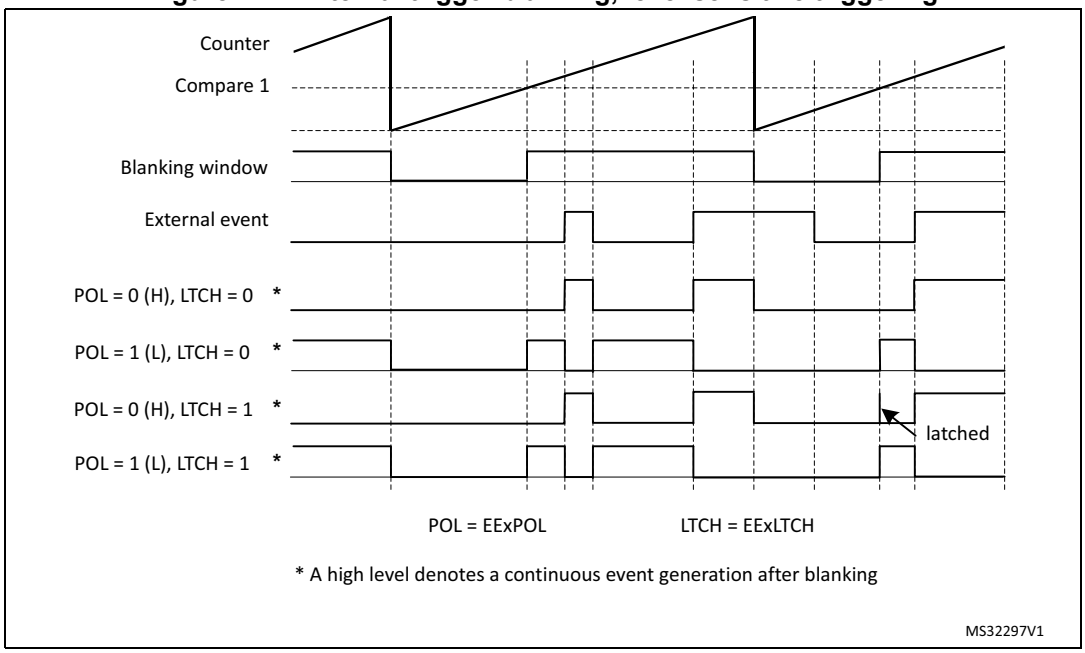


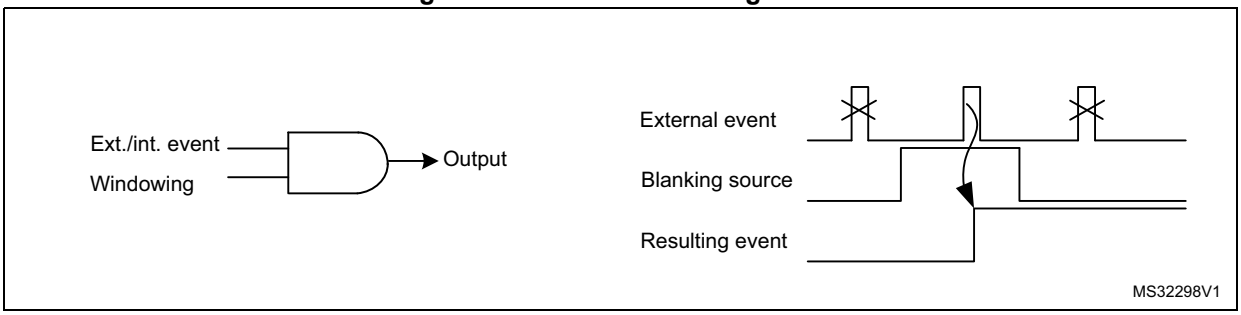
Figure 272. External trigger blanking, level sensitive triggering



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[3:0] ranging from 1101 to 1111.

Figure 273. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 1101 and 1110: the windowing lasts from the counter reset to the compare match (respectively Compare 2 and Compare 3)
- 1111: the windowing is related to another timing unit and lasts from its counter reset to its Compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 89*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A Compare 2.

Table 89. Windowing signals mapping per timer (EEFLTR[3:0] = 1111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer D CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 274 and *Figure 275* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 274. External trigger windowing with edge-sensitive trigger

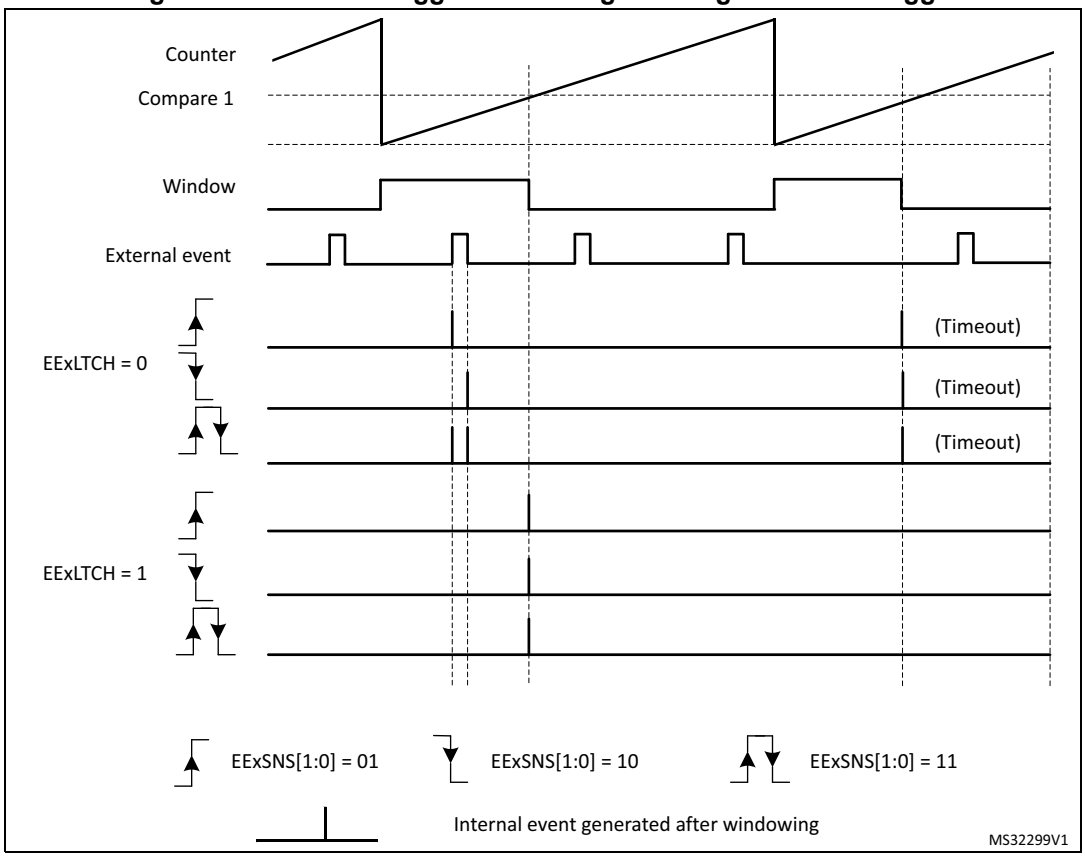
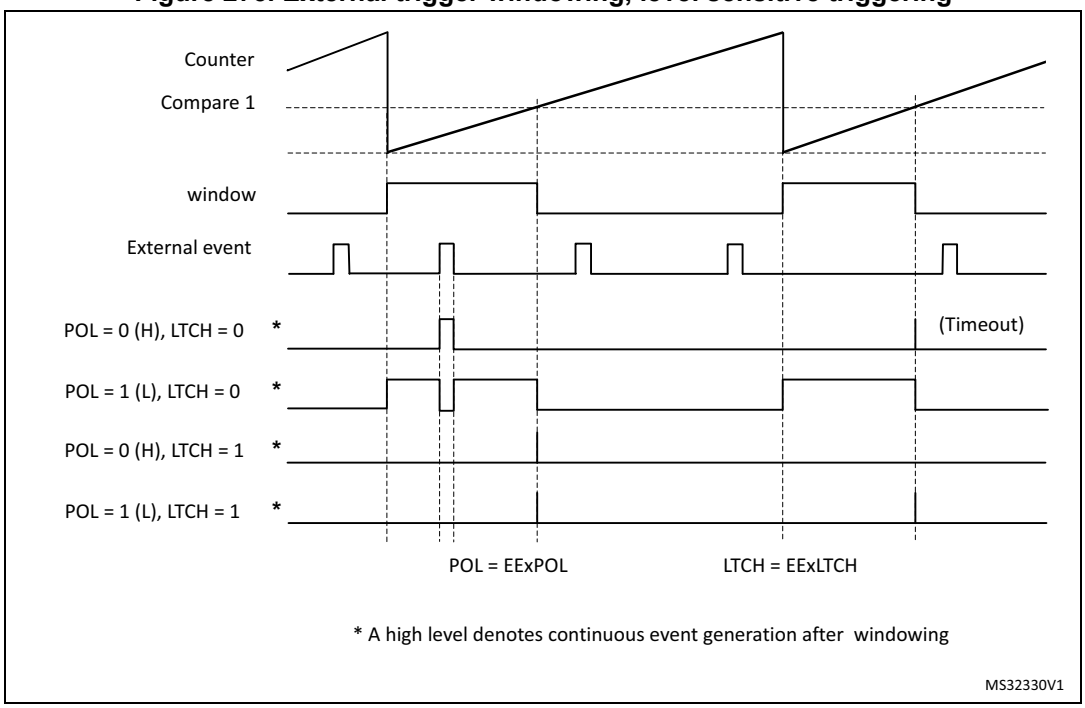


Figure 275. External trigger windowing, level sensitive triggering



21.3.9 Delayed Protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed Idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state will change on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

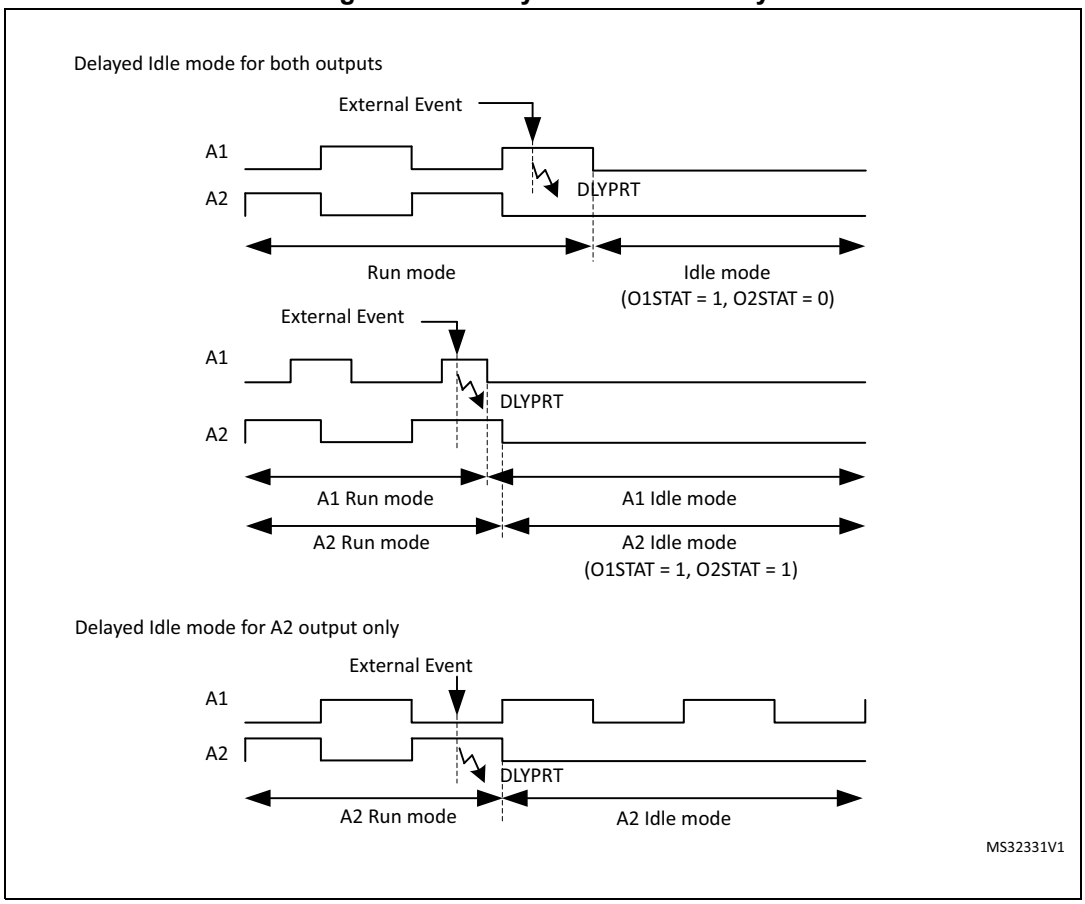
When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with 2 external events only:

- EEV6 and EEV7 for Timer A, B and C
- EEV8 and EEV9 for Timer D and E

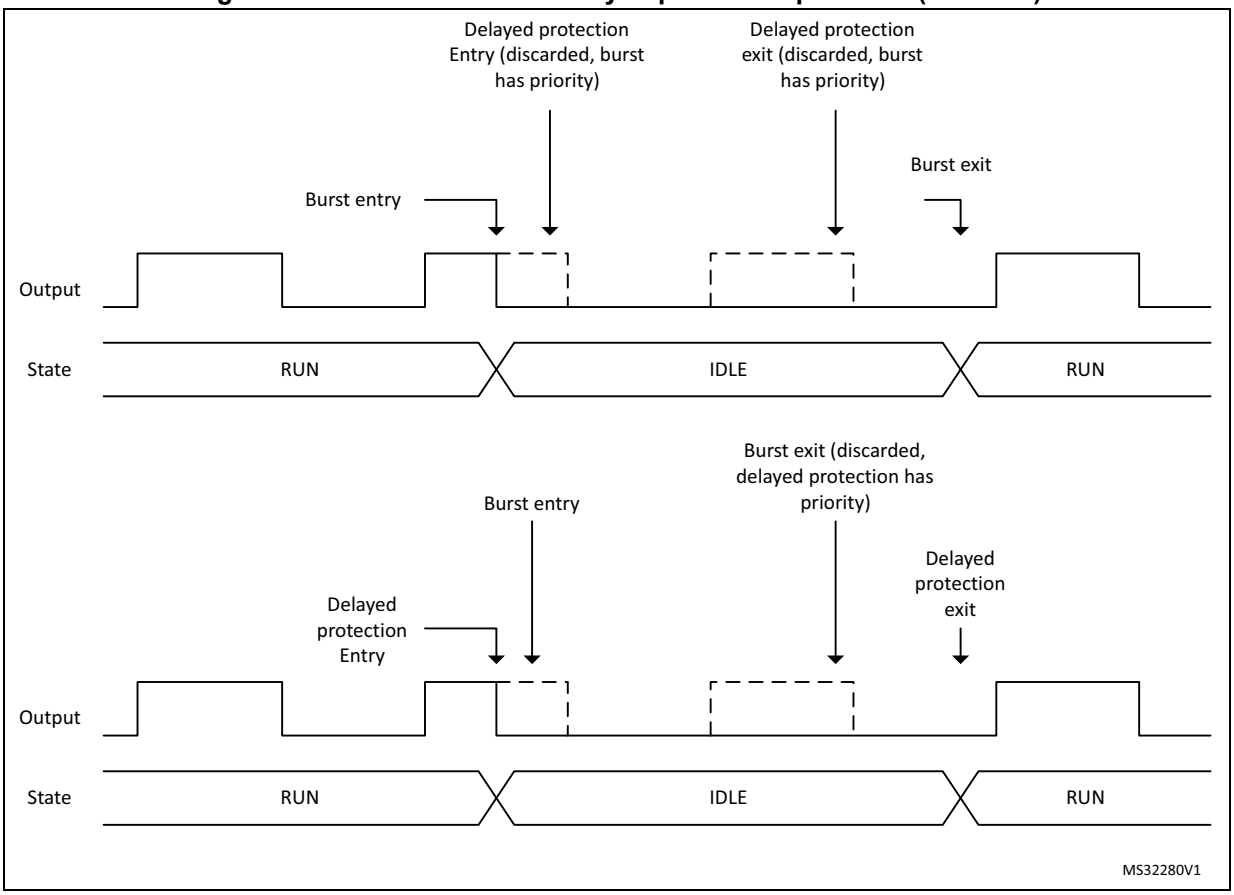
The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

Figure 276. Delayed Idle mode entry



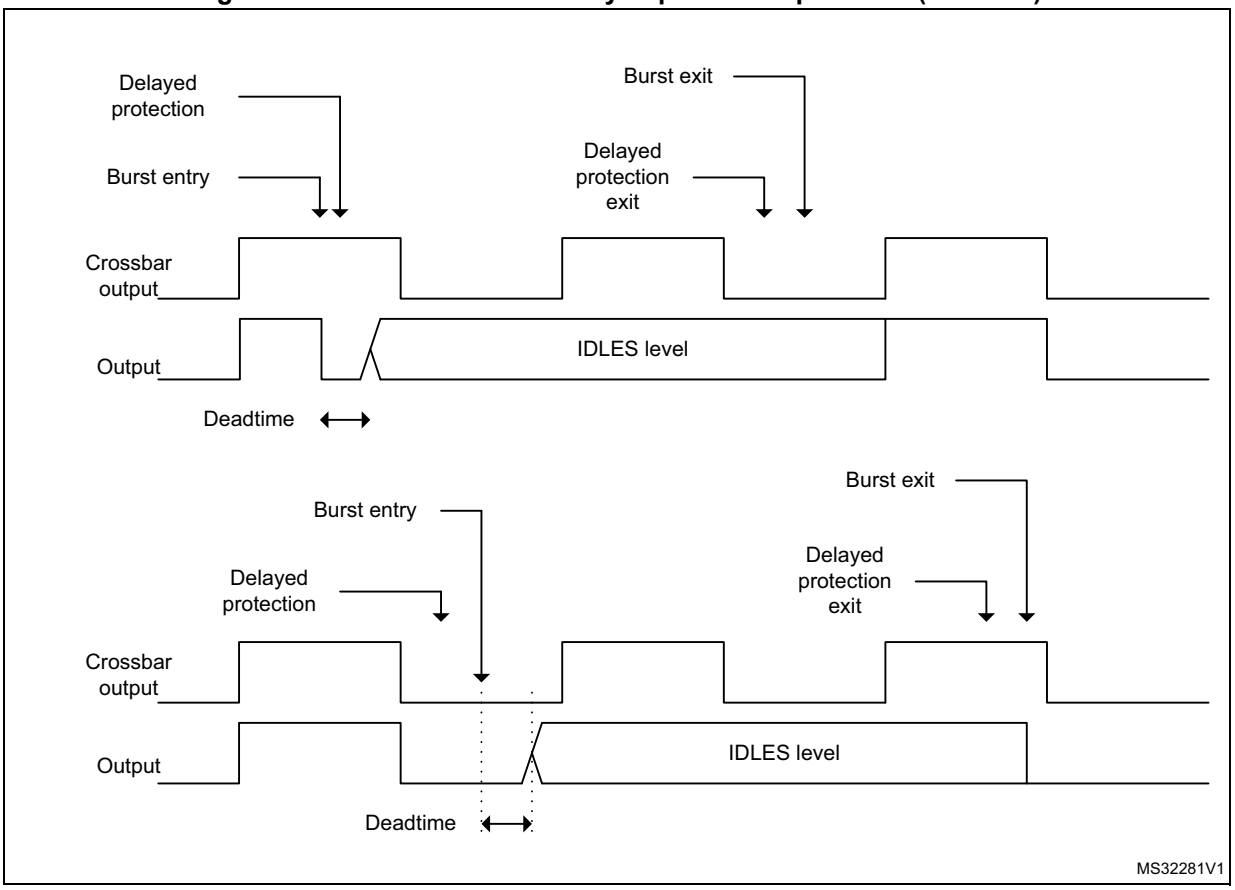
The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode will be resumed normally and the output will be maintained in the idle state until the burst mode exits. *Figure 277* gives an overview of these different scenarios.

Figure 277. Burst mode and delayed protection priorities (DIDL = 0)



The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown on *Figure 278* below.

Figure 278. Burst mode and delayed protection priorities (DIDL = 1)



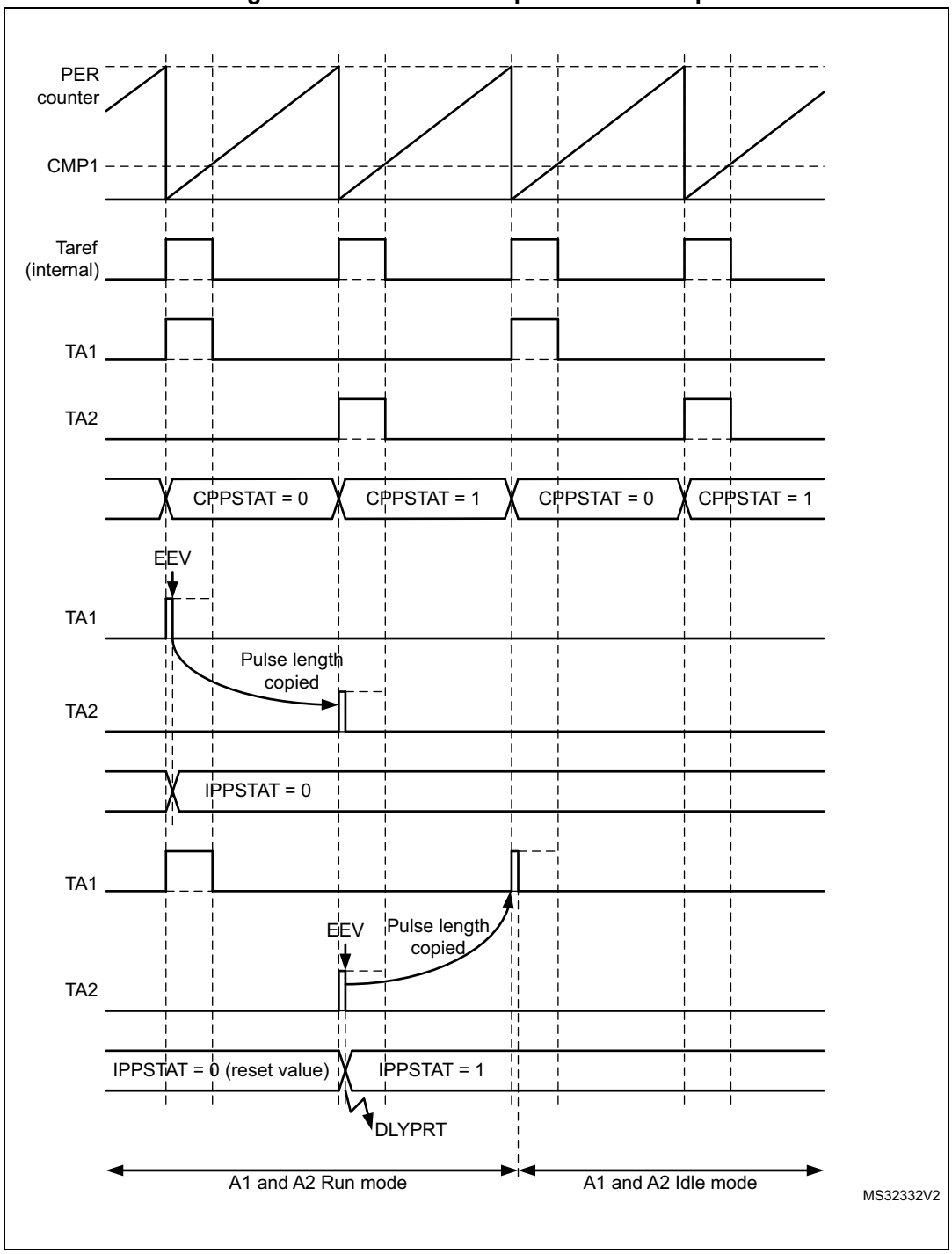
Balanced Idle

Only available in push-pull mode, it allows to have balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, which was terminated earlier than programmed, is copied on the alternate output and the two outputs are then put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bitfield in HRTIM_OUTxR.

This mode is available with 2 external events only:

- EEV6 and EEV7 for Timer A, B and C
- EEV8 and EEV9 for Timer D and E

Figure 279. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the Compare 4 active register (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag, while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (HRTIM_CHA1 then HRTIM_CHA2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- timer operating in continuous mode (CONT = 1)
- Push-pull mode enabled
- HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- DELCMP4[1:0] bit field must be set to 00 (auto-delayed mode disabled)
- DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it will cause the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry).

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods (0x80 when CKPSC[2:0] = 0, 0x40 if CKPSC[2:0] = 1, 0x20 if CKPSC[2:0] = 2,...).

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulsewidths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state will change on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation will be resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output will be active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if EEV arrives during output 1 active, the re-start sequence will be initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, see *Section 21.3.13*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode will be resumed normally.

Note: Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (see Section 21.3.16) during the idle period following the delayed protection:
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

21.3.10 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PRREN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 90 lists the registers which can be preloaded, together with a summary of available update events.

Table 90. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master Timer	HRTIM_DIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..E	HRTIM_TIMxDIER HRTIM_TIMxPER HRTIM_TIMxREP HRTIM_TIMxCMP1R HRTIM_TIMxCMP1CR HRTIM_TIMxCMP2R HRTIM_TIMxCMP3R HRTIM_TIMxCMP4R HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx Repetition event TIMx Reset Event Burst DMA event Update event from other timers (TIMy, Master) Update event following a burst DMA event Update enable input 1..3 Update event following an update enable input 1..3
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or Master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has 4 update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once Burst DMA is completed (see *Section 21.3.21* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.
   Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (i.e. forced update mode). If SWU bit is reset, the update will be done on the next update event following the end of the burst sequence.
4. Update done when the master counter rolls over following a Burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..E) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a Burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a Burst DMA completion (the event can be enabled with TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on the update enable input 1..3. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on the update enable input 1..3 (the event can be enabled with TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs 1..3 allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 91 lists the connections between update enable inputs and the on-chip sources.

Table 91. Update enable inputs and sources

Update enable input	Update source
Update enable input 1	TIM16_OC
Update enable input 2	TIM17_OC
Update enable input 3	TIM6_TRGO

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

An interrupt or a DMA request can be generated by the Timx update event.

MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event. This allows to modify several registers in multiple timers. The regular update event takes place once these bits are reset.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter

In HRTIM_MCR, MREPU bit is set: the update will occur at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers will be done as soon as the master repetition event occurs.

Second converter

In HRTIM_TIMACR, TAREPU bit is set: the update will occur at the end of the Timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with Timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers will be done as soon as the Timer A repetition event occurs.

21.3.11 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the Master timer update

The sources listed in *Table 92* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 92. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a Burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 93* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 93. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable input	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR
Another TIMx update (TIMz>TIMy>TIMx)	TzU=1 in HRTIM_TIMyCR TyU=1 in TIMxCR	No	Must be done with TzU=1 in HRTIM_TIMxCR TzU=1 in HRTIM_TIMyCR
Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	No	Must be done using TIMzCMPn in HRTIM_RSTxR
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-

Table 93. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	No	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx Counter reset causing a TIMx update

Table 94 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 94. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over		Yes	
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External Event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 95 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 95. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable input	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a Burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR MREPU = 1 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-
Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR MSWU = 1 in HRTIM_CR2	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update
TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR TyREPU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR EXTEVNTn or CMP4/2 in HRTIM_RSTyCR	Yes	-
TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-

Table 95. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

21.3.12 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporary disabled during a normal operating mode; refer to *Section 21.3.13* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 96*.

Table 96. Output state programming, x= A..E, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register allows to disable the output by software.

Figure 280. Output management overview

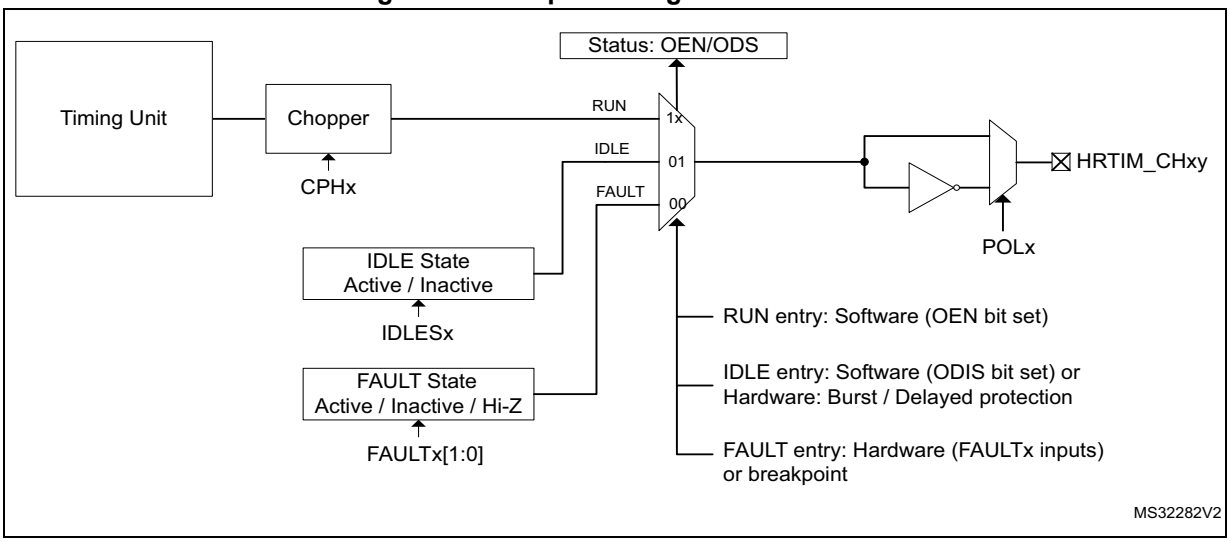
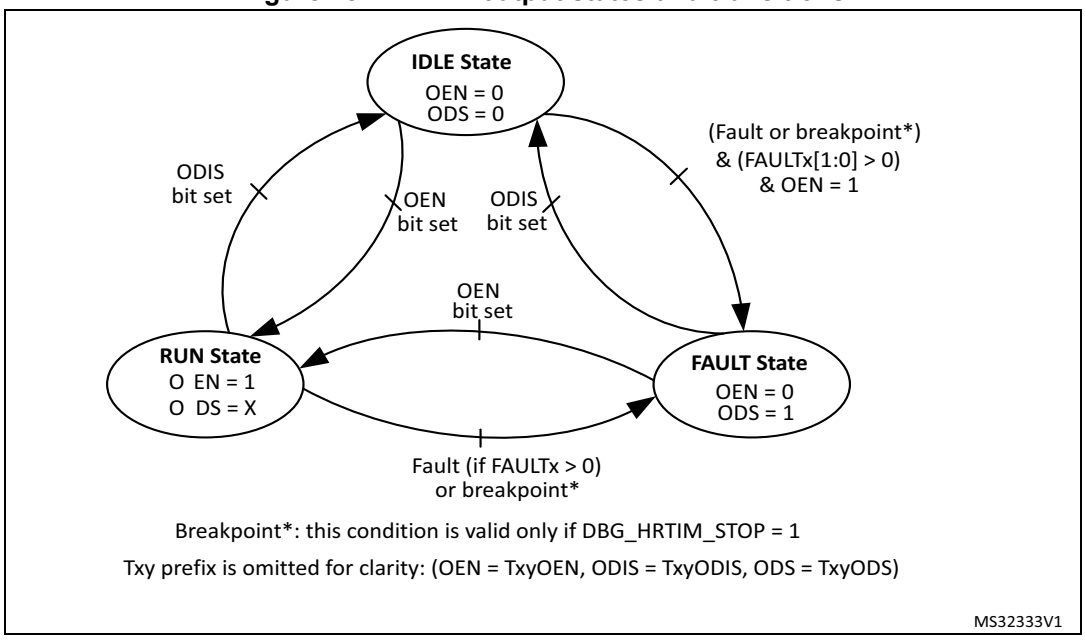


Figure 281 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in *Section 21.3.15*, while the Idle state can be entered when the burst mode or delayed protections are active.

Figure 281. HRTIM output states and transitions



The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a Fault event, the FAULT state will be entered. The condition is given on the transition IDLE →

FAULT, as in *Figure 281*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if DBG_HRTIM_STOP = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS vs. NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state
- 01: output at active level when in FAULT
- 10: output at inactive level when in FAULT
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level will either be inactive (after an HRTIM reset) or correspond to the RUN level (when the timer was stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to have it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

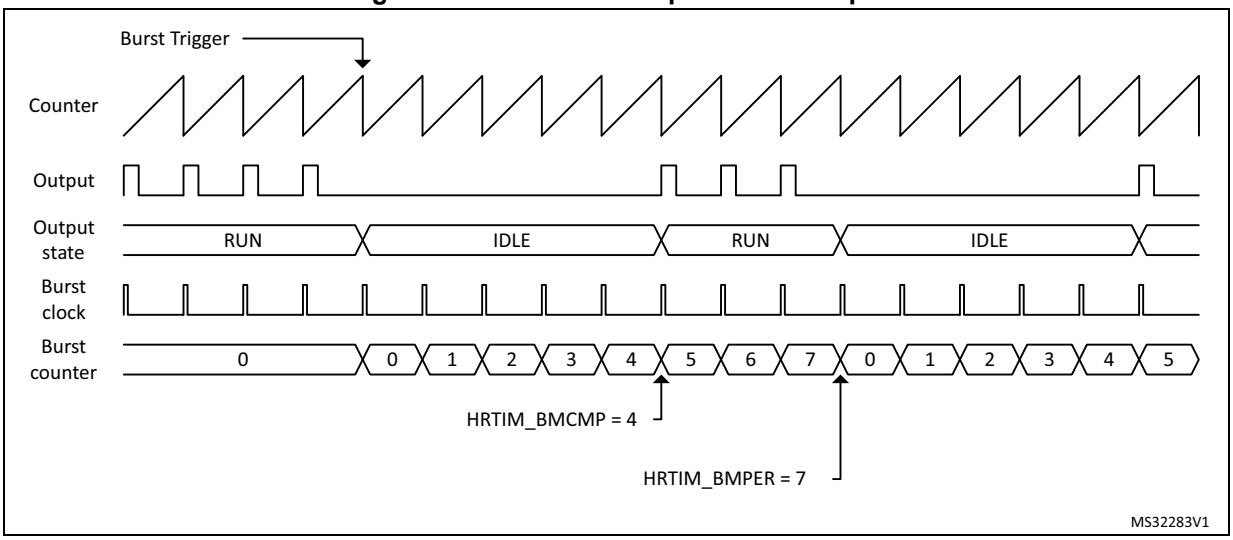
21.3.13 Burst mode controller

The burst mode controller allows to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputs followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 282*.

Figure 282. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 97*.

Table 97. Timer output programming for burst mode

IDLEMx	IDLESx	**Output state during burst mode**
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start not reset events are generated on the HRTIM_SCOUT output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit.The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The Burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and 2 registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bitfields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds 6 control bits for this purpose: MTBM (master) and TABM..TEBM for Timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This allows to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This allows to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: Master timer and TIMA..E reset/roll-over events. This allows to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the general purpose timers, as in *Table 98*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period (a pulse on the output may be interrupted, resulting a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

Table 98. Burst mode clock sources from general purpose timer

BMCLK[3:0]	Clock source
0110	TIM16 OC
0111	TIM17 OC
1000	TIM7 TRGO
1001	Reserved

The pulsewidth on TIM16/17 OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

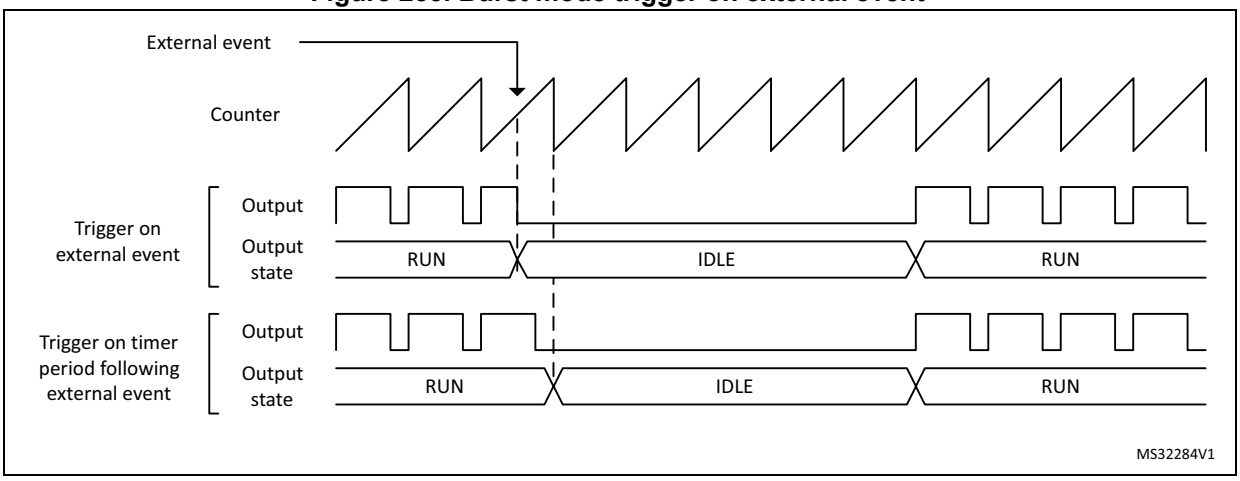
- Software trigger (set by software and reset by hardware)
- 6 Master timer events: repetition, reset/roll-over, Compare 1 to 4
- 5 x 4 events from timers A..E: repetition, reset/roll-over, Compare 1 and 2
- External Event 7 (including TIMA event filtering) and External Event 8 (including TIMD event filtering)
- Timer A period following External Event 7 (including TIMA event filtering)
- Timer D period following External Event 8 (including TIMD event filtering)
- On-chip events coming from other general purpose timer (TIM7_TRGO output)

These sources can be combined to have multiple concurrent triggers.

Burst mode is not re-triggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 283 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 283. Burst mode trigger on external event



For TAEEV7 and TDEEV8 combined triggers (trigger on a Timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the on-going burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEV7 or TDEEV8 bit set in the BMTRG register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is re-triggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

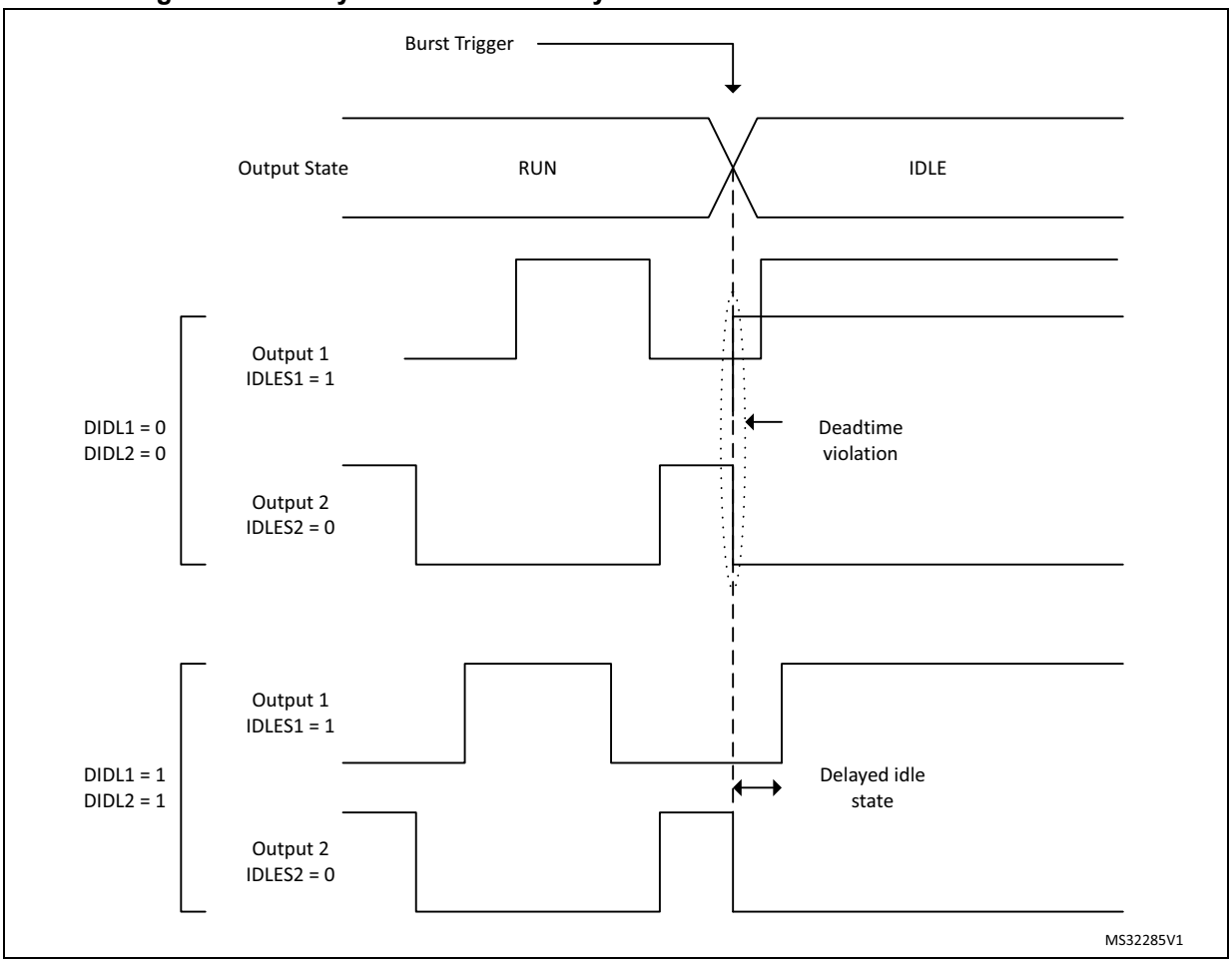
Note: TAEEV7 and TDEEV8 triggers are valid only after a period event. If the counter is reset before the period event, the pending EEV7/8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown on *Figure 284*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 284. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

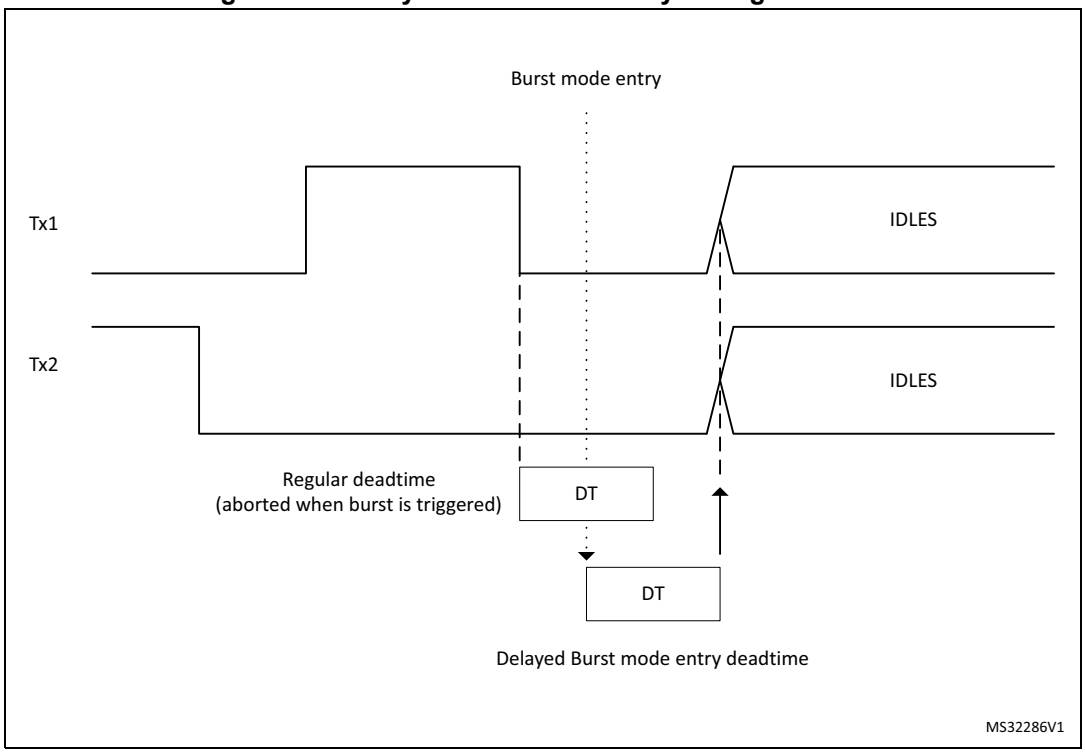
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the 2 DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than 3 f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 21.3.6* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is re-started corresponding to the inactive period, as on *Figure 285*.

Figure 285. Delayed Burst mode entry during deadtime



Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

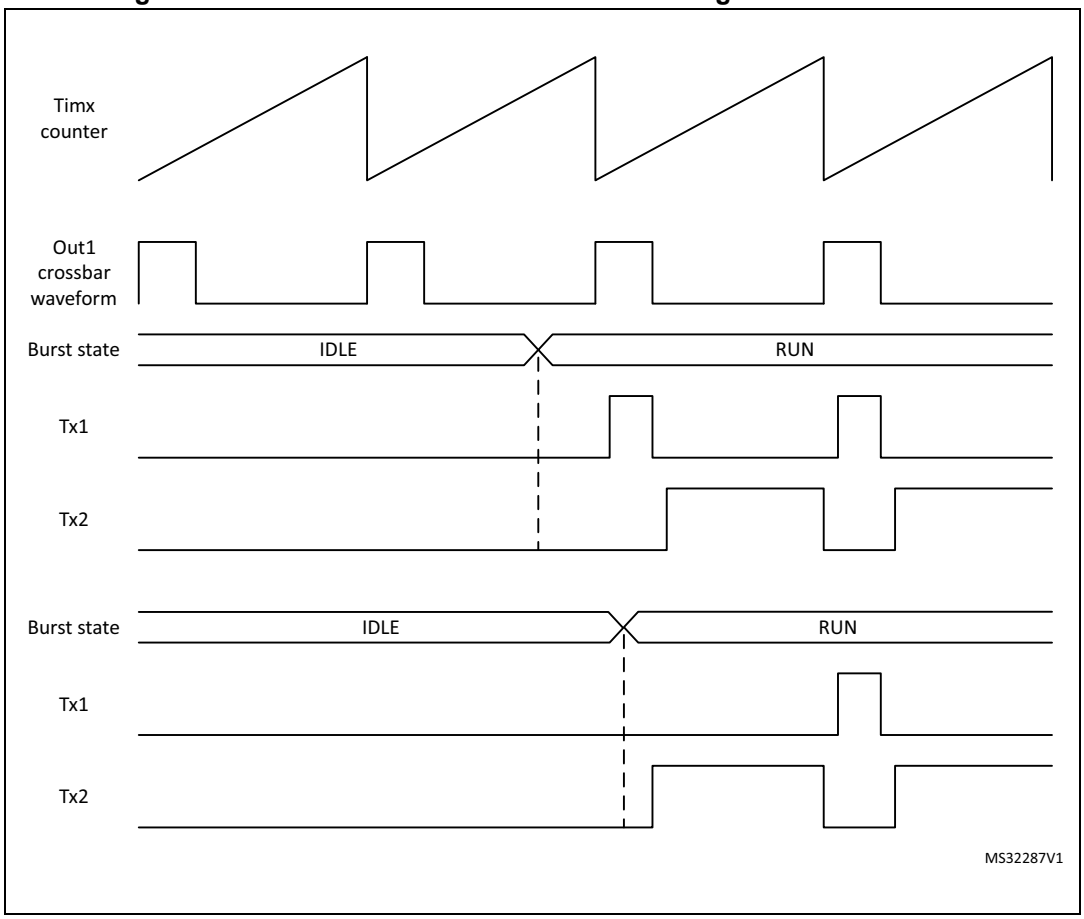
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 286 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 286. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (Burst mode Preload Enable) allows to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- when the burst mode is enabled (BME = 1),
- at the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to re-write the compare to have the new values taken into account.

When BMPREN bits is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the 2 cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match will occur and the idle state will last until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period will not be taken into account, the burst mode counter will overflow (at 0xFFFF) and the change will be effective in the next period. In single-shot mode, the counter will roll over at 0xFFFF and the burst mode will re-start for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the Compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

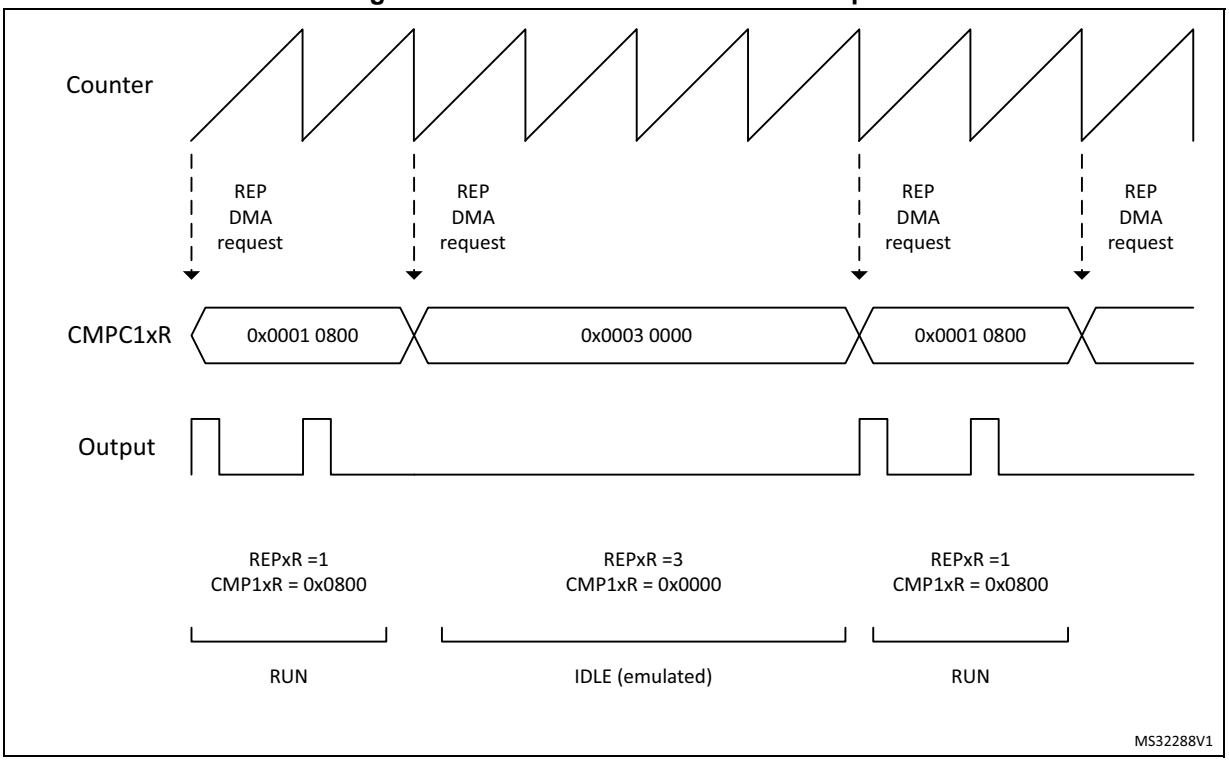
For instance, the values:

{0x0003 0000}: CMP1 = 0 for 3 periods

{0x0001 0800}: CMP1 = 0x0800 for 1 period

will provide a burst mode with 2 periods active every 6 PWM periods, as shown on *Figure 287*.

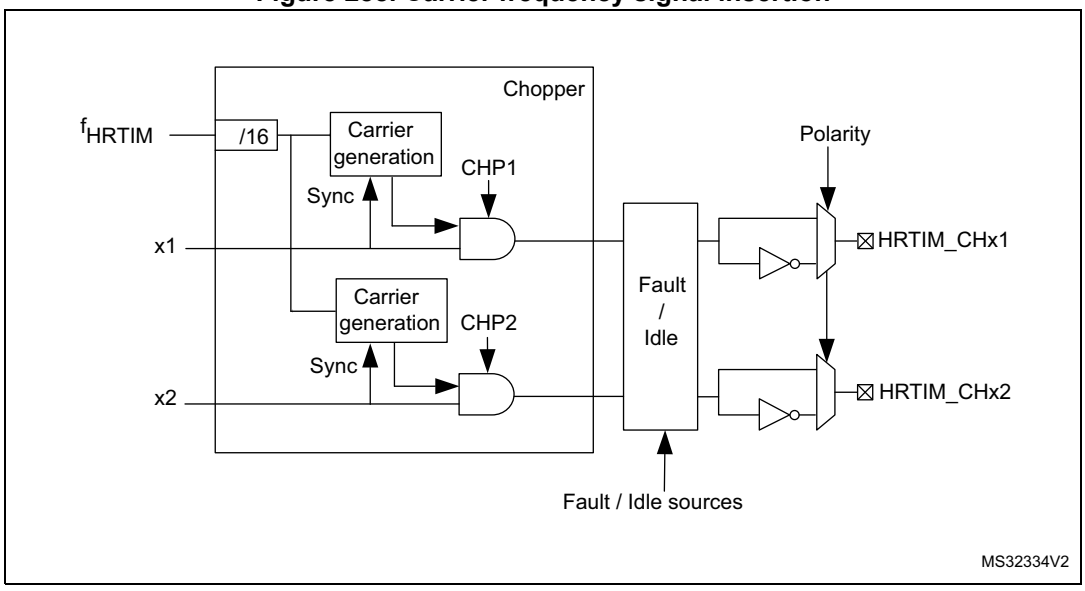
Figure 287. Burst mode emulation example



21.3.14 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown on *Figure 288*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 288. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 289*.

CARFRQ[3:0] bits define the frequency, ranging from 562.5 kHz to 9 MHz (for f_{HRTIM} = 144 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

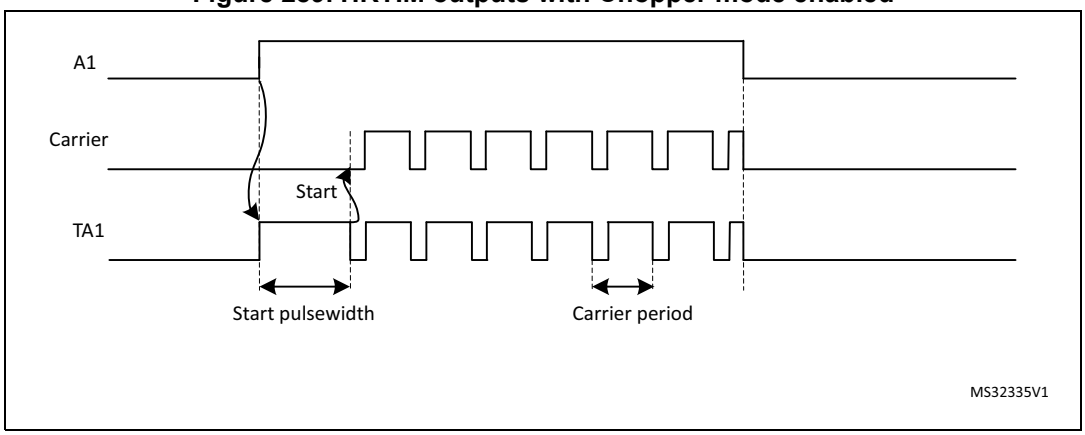
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bitfield as follows: t1STPW = (STRPW[3:0]+1) x 16 x t_{HRTIM} and ranges from 111 ns to 1.77 µs (for f_{HRTIM}=144 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and are not dependent from the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 289. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

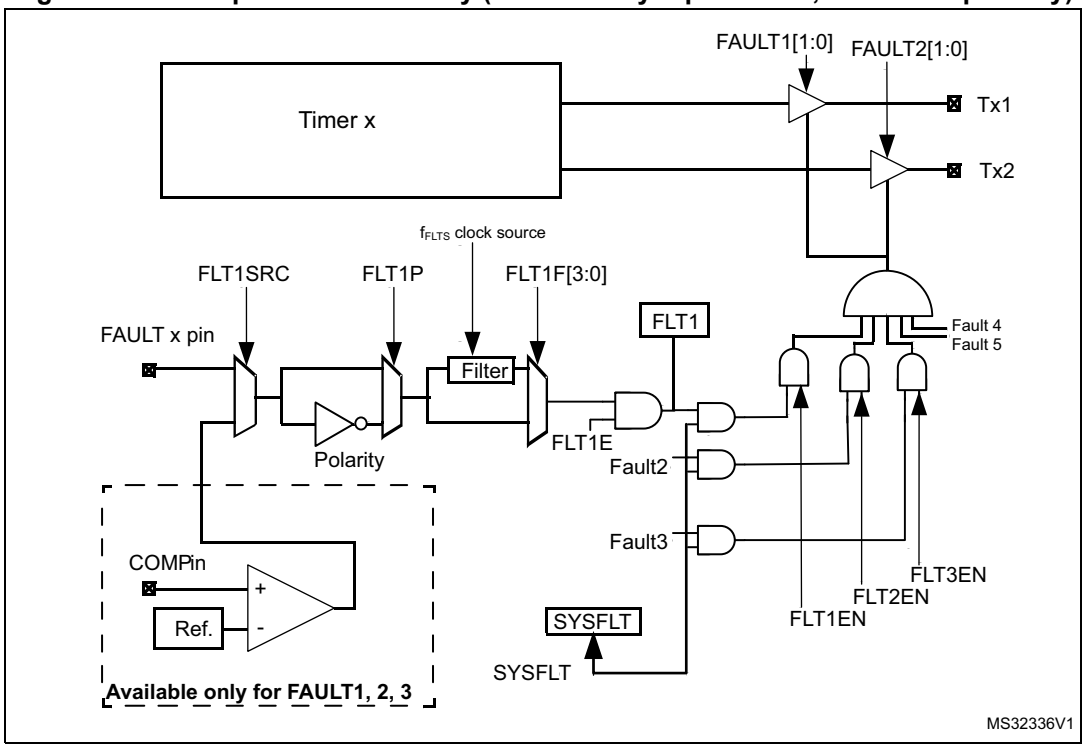
CARFRQ[2:0], CARDTY[2:0] and STRPW[3:0] bitfields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

21.3.15 Fault protection

The HRTIMER has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is re-enabled by software. In case of a permanent fault request, the output will remain in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has 5 FAULT input channels; all of them are available and can be combined for each of the 5 timing units, as shown on *Figure 290*.

Figure 290. Fault protection circuitry (FAULT1 fully represented, FAULT2..5 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1 and HRTIM_FLTINR2 registers before being routed to the timing units. FLTxSRC bit selects the source of the Fault signal, that can be either a digital input or an internal event (built-in comparator output).

Table 99 summarizes the available sources for each of the 10 faults channels:

Table 99. Fault inputs

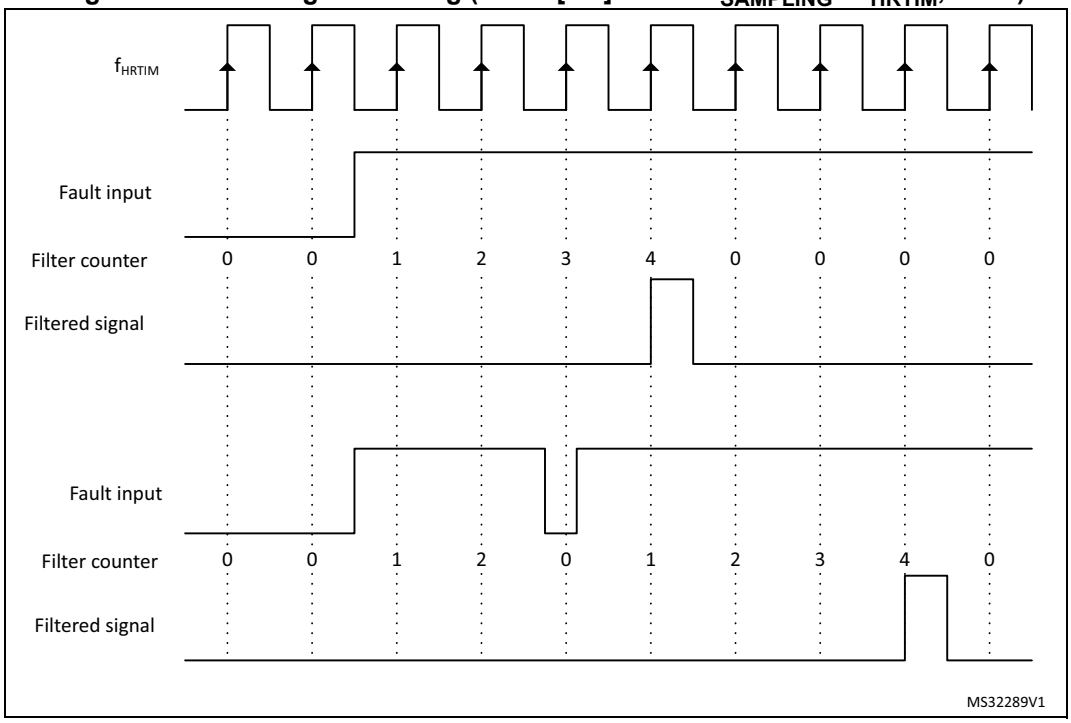
Fault channel	External Input (FLTxSRC = 0)	On-chip source (FLTxSRC = 1)
FAULT 1	PA12	COMP2
FAULT 2	PA15	COMP4
FAULT 3	PB10	COMP6
FAULT 4	PB11	NC
FAULT 5	PC7	NC

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bitfield is set to 0000, the signal is not filtered and will act asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bitfield values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external

event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 291* shows how a spurious fault signal is filtered.

Figure 291. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from 2 cycles of the f_{HRTIM} clock up to 8 cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 100* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 100. Sampling rate and filter length vs FLTFxF[3:0] and clock setting

	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 144 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
0001,0010,0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 13.9 ns	f_{HRTIM}, N =8 55.5 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 83.3 ns	f_{HRTIM} /16, N = 8 888.9 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 166.7 ns	f_{HRTIM} /32, N = 8 1.777 µs
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 333.3 ns	f_{HRTIM} /64, N = 8 3.55 µs

Table 100. Sampling rate and filter length vs FLTFxF[3:0] and clock setting (continued)

	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 144 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 555.5ns	f_{HRTIM} /128, N = 8 7.11 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 1.11 µs	f_{HRTIM} /256, N = 8 14.22 µs

System fault input (SYSFLT)

This fault is provided by the MCU Class B circuitry (see the System configuration controller (SYSCFG) section for details) and corresponds to a system fault coming from:

- the Clock Security System
- the SRAM parity checker
- the Cortex®-M4-lockup signal
- the PVD detector

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register allows to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the 5 fault channels are enabled using bits FLT1EN to FLT5EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This allows to have, for instance:

- One fault channel simultaneously disabling several timing units
- Multiple fault channels being ORed to disable a single timing unit

A write-once FLTLCK bit in the HRTIM_FLTxR register allows to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (see *Section 21.3.12*).

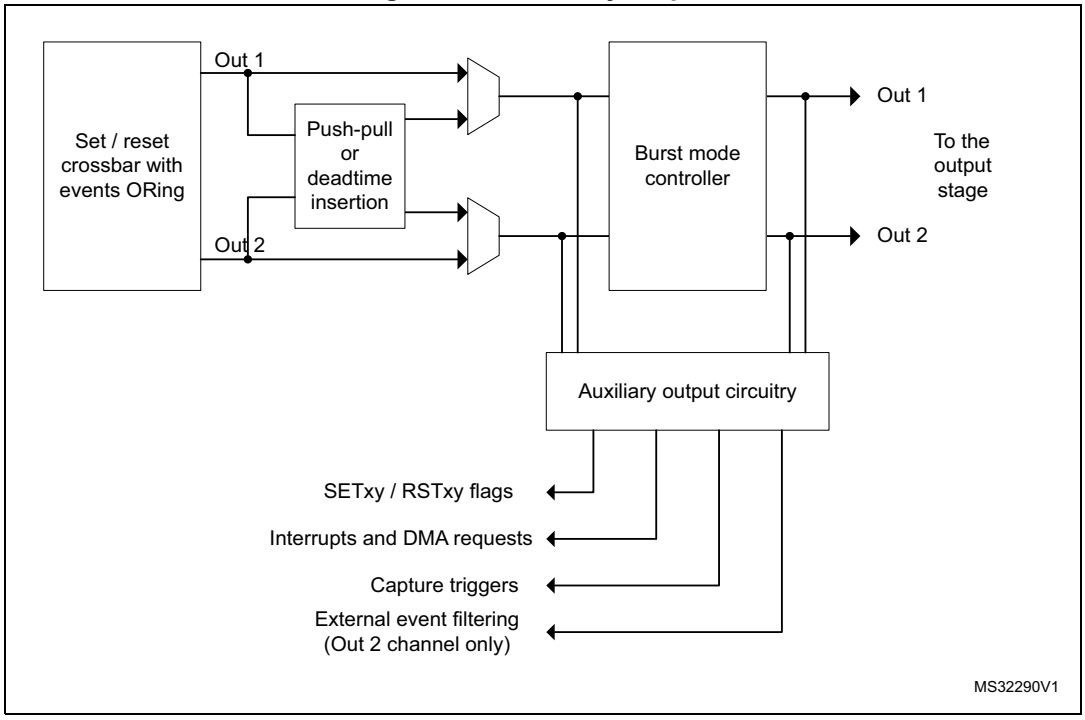
21.3.16 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset
- External event filters following a Tx2 output copy (see details in *Section 21.3.8*)

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given on *Figure 292*.

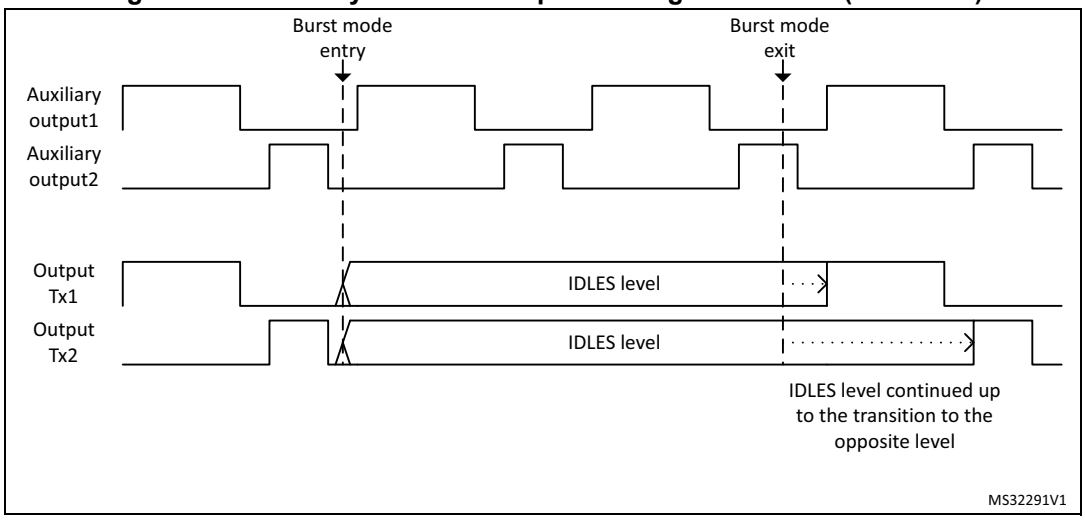
Figure 292. Auxiliary outputs



By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

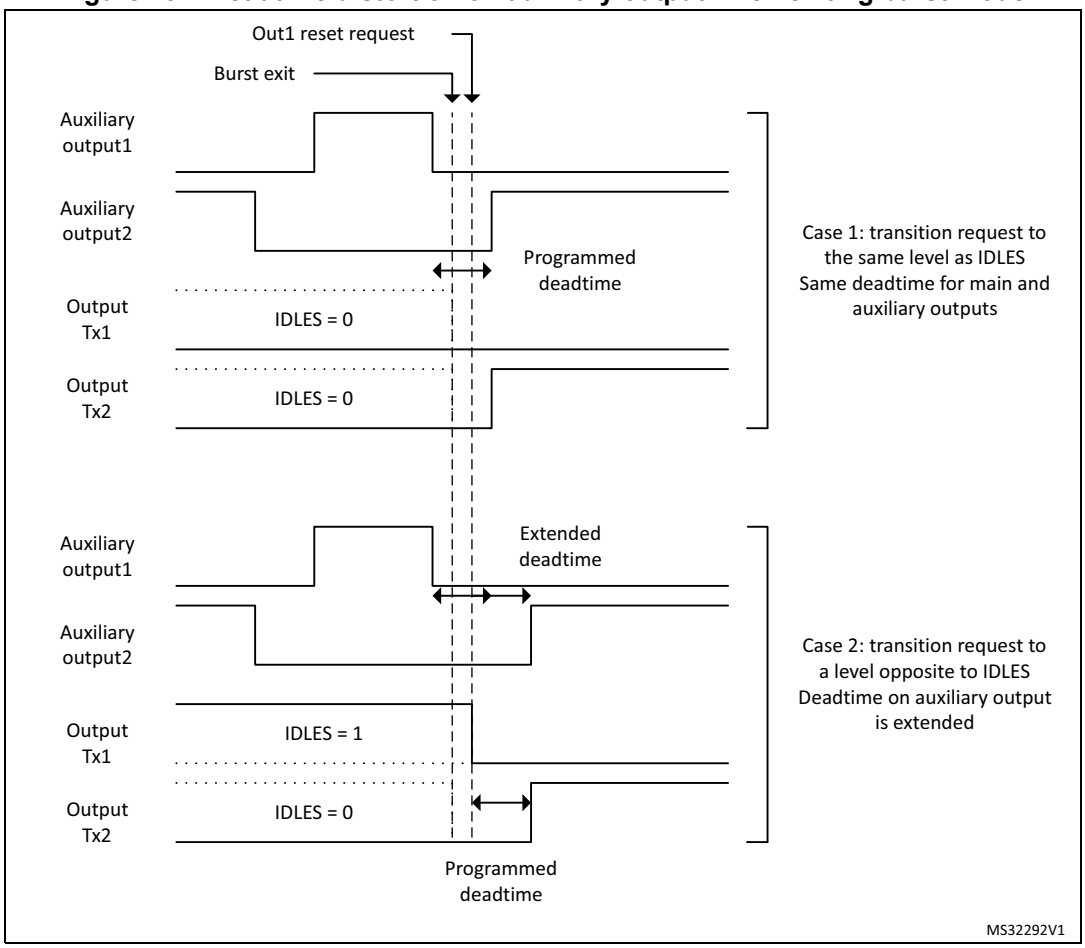
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are 2 cases:
  a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (see *Figure 293*).
  b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 293. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when re-enabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 294* gives some examples.

Figure 294. Deadtime distortion on auxiliary output when exiting burst mode



21.3.17 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: Master timer Start
  This event is generated when MCEN bit is set or when the timer is re-started after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: Master timer Compare 1 event
- 10: Timer A start
  This event is generated when TACEN bit is set or when the counter is reset and re-starts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot non-retriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: Timer A Compare 1 event

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated.

The synchronization pulses are generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is $f_{HRTIM}/32$.

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (i.e. the bitfield value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bitfield configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the General-purpose I/Os section)
3. Master or Timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting or is reset while running. Any reset request clearing the counter without causing it to start will not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled
- 01: reserved configuration
- 10: the on-chip TIM1 general purpose timer (TIM1 TRGO output)
- 11: a positive pulse on the HRTIM_SCIN input pin

This bitfield cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (see *Table 101* for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter will not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in *Table 101*.

Table 101. Effect of sync event vs timer operating modes

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

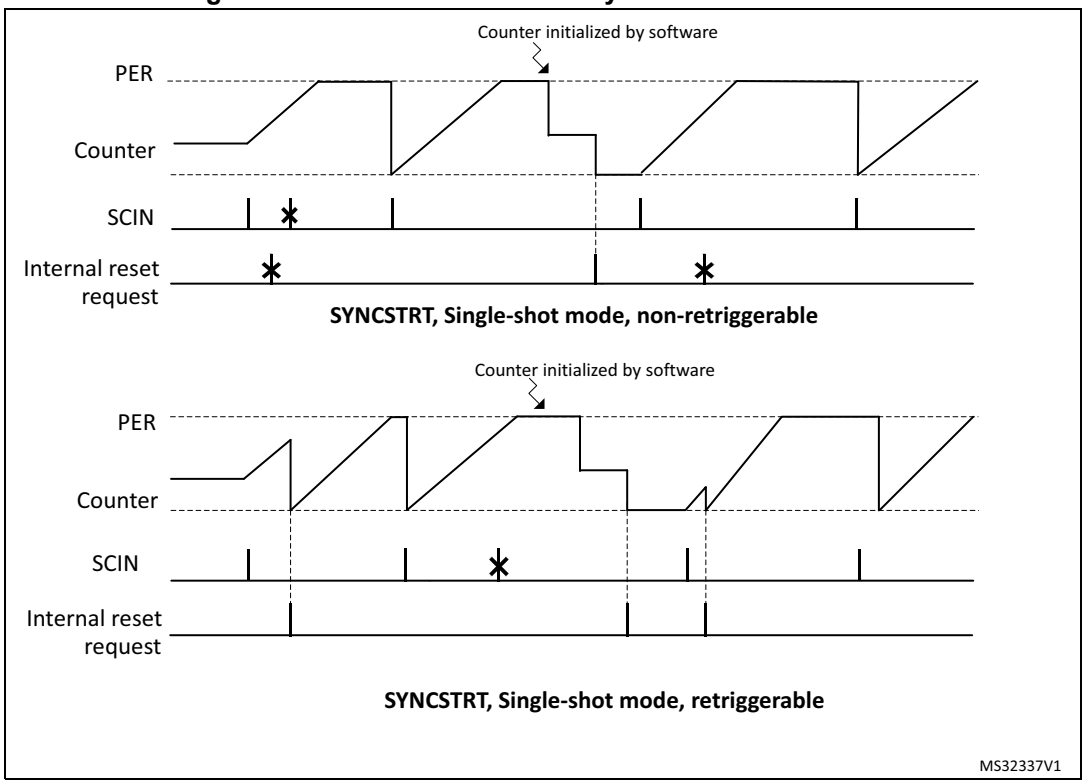
Table 101. Effect of sync event vs timer operating modes (continued)

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

When a synchronization reset event occurs within the same f_{HRTIM} *clock cycle as the period event, this reset is postponed to a programmed period event (since both events are causing a counter roll-over). This applies only when the high-resolution is active (*CKPSC[2:0] < 5).

Figure 295 presents how the synchronized start is done in single-shot mode.

Figure 295. Counter behavior in synchronized start mode



21.3.18 ADC triggers

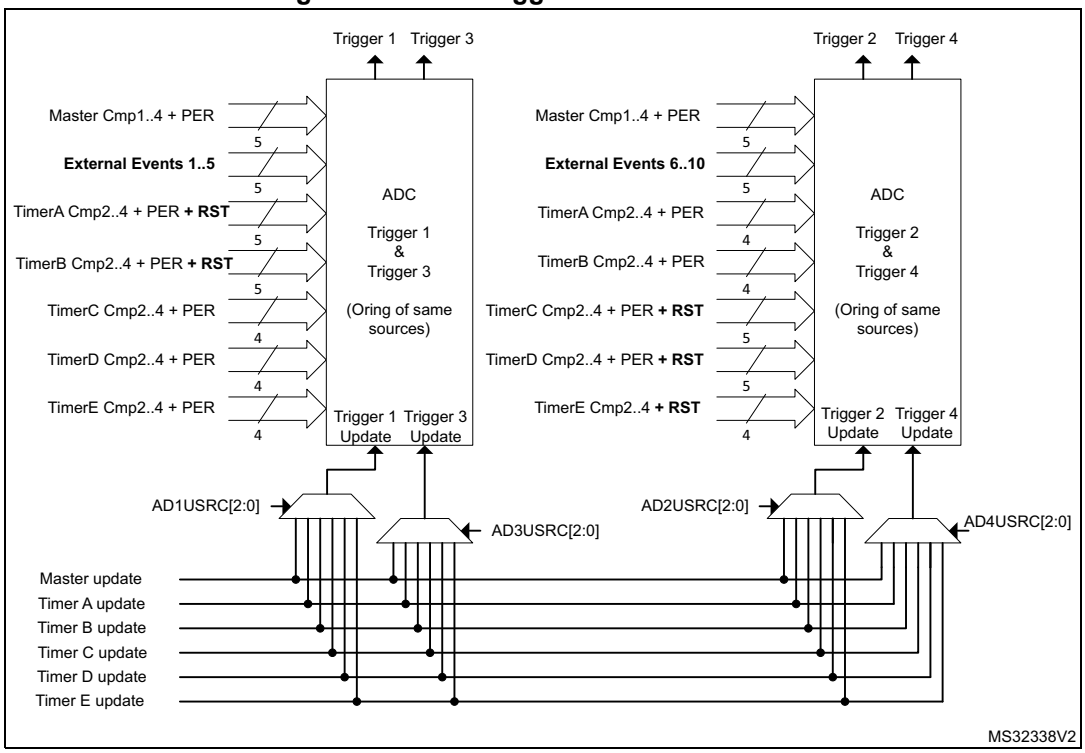
The ADCs can be triggered by the master and the 5 timing units.

4 independent triggers are available to start both the regular and the injected sequencers of the 2 ADCs. Up to 32 events can be combined (ORed) for each trigger output, in registers HRTIM_ADC1R to HRTIM_ADC4R, as shown on *Figure 296*. Triggers 1/3 and 2/4 are using the same source set.

The external events can be used as a trigger. They are taken right after the conditioning defined in HRTIM_EECRx registers, and are not depending on EEFxR1 and EEFxR2 register settings.

Multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a non-overlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

Figure 296. ADC trigger selection overview



HRTIM_ADC1R to HRTIM_ADC4R registers are preloaded and can be updated synchronously with the timer they are related to. The update source is defined with ADxUSRC[2:0] bits in the HRTIM_CR1 register.

For instance, if ADC trigger 1 outputs Timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R will be typically updated simultaneously with Timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access will result in an immediate update of the trigger source.

21.3.19 DAC triggers

The HRTIMER allows to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the 3 DACtrigOutx outputs.

Note: Each timer has its own DAC-related control register.

DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

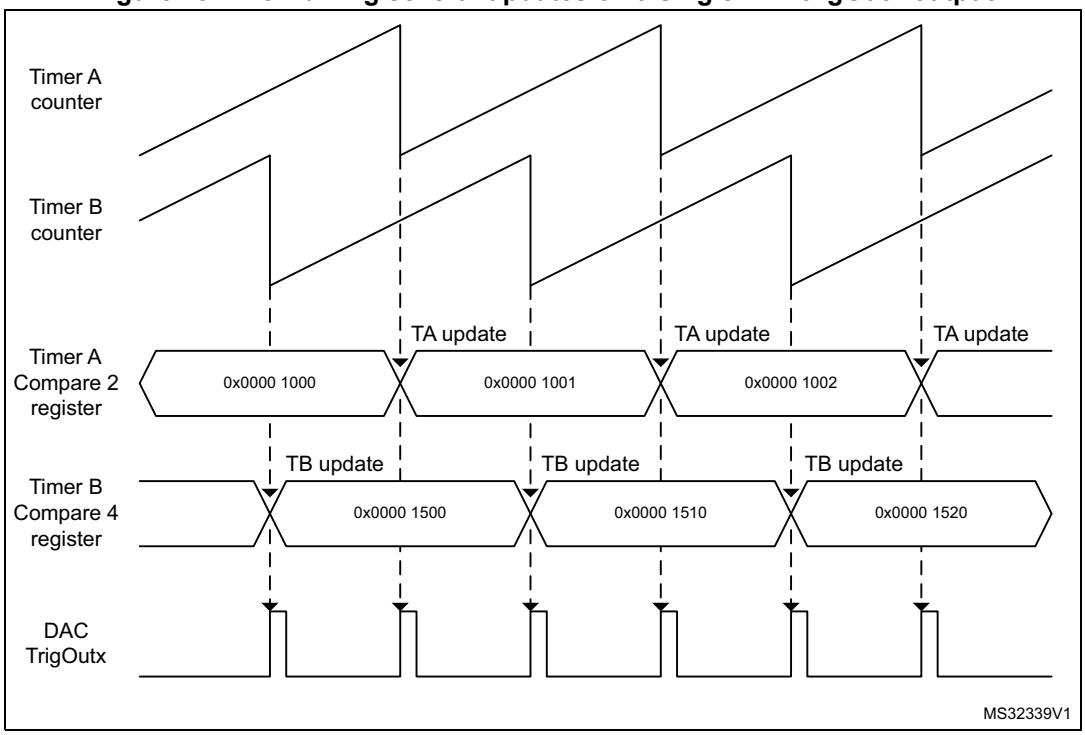
- 00: No update generated
- 01: Update generated on DACtrigOut1
- 10: Update generated on DACtrigOut2
- 11: Update generated on DACtrigOut3

An output pulse of 32 f_{HRTIM} clock periods is generated on the DACtrigOutx output.

Note: The synchronization pulse is followed by an idle level of 32 APB clock cycles during which any new DAC update request is ignored. Consequently, the maximum synchronization frequency is $f_{apb}/64$.

When DACSYNC[1:0] bits are enabled in multiple timers, the DACtrigOutx output will consist of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A will be ORed with the update event in timer B to generate a DAC update trigger on the corresponding DACtrigOutx output, as shown on *Figure 297*.

Figure 297. Combining several updates on a single DACtrigOutx output



DACtrigOutx pins are connected to the DACs as follows:

- DACtrigOut1: DAC1_CH1 trigger input 3 (TSEL1[2:0] = 011 in DAC_CR of DAC1 peripheral)
- DACtrigOut2: DAC1_CH2 trigger input 5 (TSEL1[2:0] = 101 in DAC_CR of DAC1 peripheral and DAC1_TRIG3_RMP bit set in SYSCFG_CFGR2)
- DACTrigOut3: DAC2_CH1 trigger input 5 (TSEL1[2:0] = 101 in DAC_CR of DAC2 peripheral)

21.3.20 HRTIM Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master Compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and Fault 1 to 5 (regardless of the timing unit attribution)
- DLL calibration done
- Burst mode period completed

The interrupt requests are grouped in 7 vectors as follows:

- IRQ1: Master timer interrupts (Master Update, Sync Input, Repetition, MCMP1..4) and global interrupt except faults (Burst mode period and DLL ready interrupts)
- IRQ2: TIMA interrupts
- IRQ3: TIMB interrupts
- IRQ4: TIMC interrupts
- IRQ5: TIMD interrupts
- IRQ6: TIME interrupts
- IRQ7: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 102 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 102. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
IRQ1	Burst mode period completed	BMPER	BMPERIE	BMPERC
	DLL calibration done	DLLRDY	DLLRDYIE	DLLRDYC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master Compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C
IRQ2 IRQ3 IRQ4 IRQ5 IRQ6	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
IRQ7	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 5	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C

21.3.21 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...E) has its own DMA enable register.

The individual DMA requests are ORed into 6 channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 103 is a summary of the events with their associated DMA enable bits.

Table 103. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
Master timer: Channel 2	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master Compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
Timer A: Channel 3 Timer B: Channel 4 Timer C: Channel 5 Timer D: Channel 6 Timer E: Channel 7	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE

Table 103. HRTIM DMA request summary (continued)

DMA Channel	Event	DMA capable	DMA enable bit
N/A	System fault	No	N/A
	Fault 1 to 5	No	N/A
	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A

Burst DMA transfers

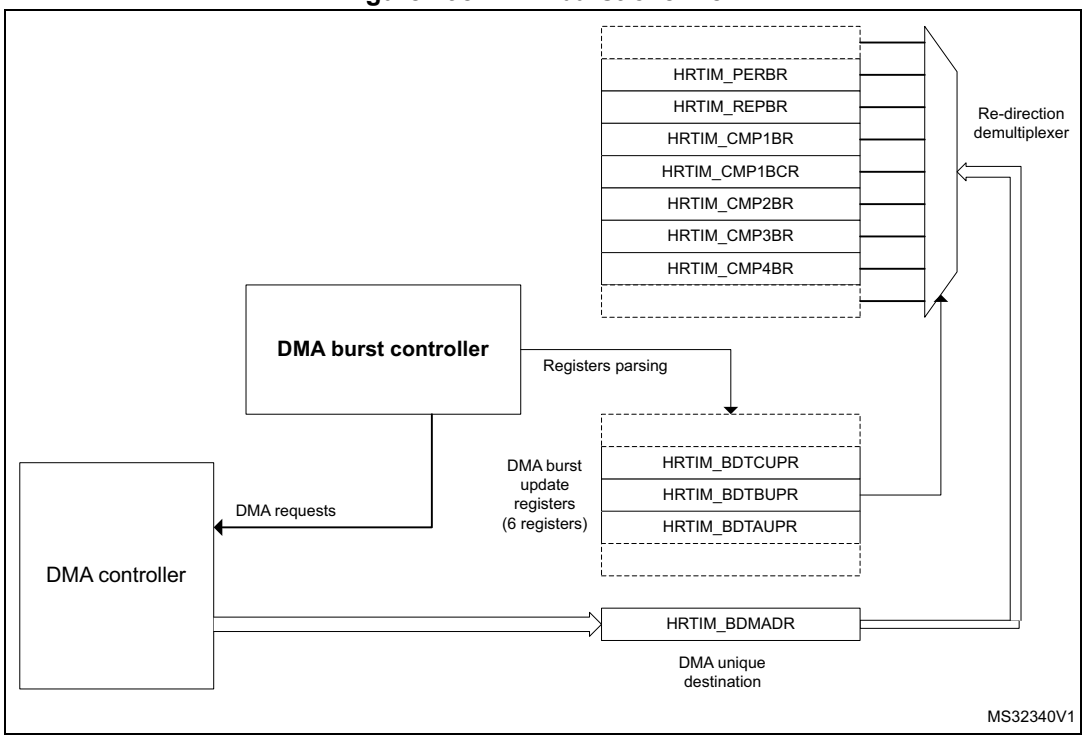
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- update multiple data registers with one DMA channel only,
- reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the 6 channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism allows to forward the DMA write accesses to the HRTIM registers automatically, as shown on *Figure 298*.

Figure 298. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the 6 update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (see the flowchart on *Figure 299*).

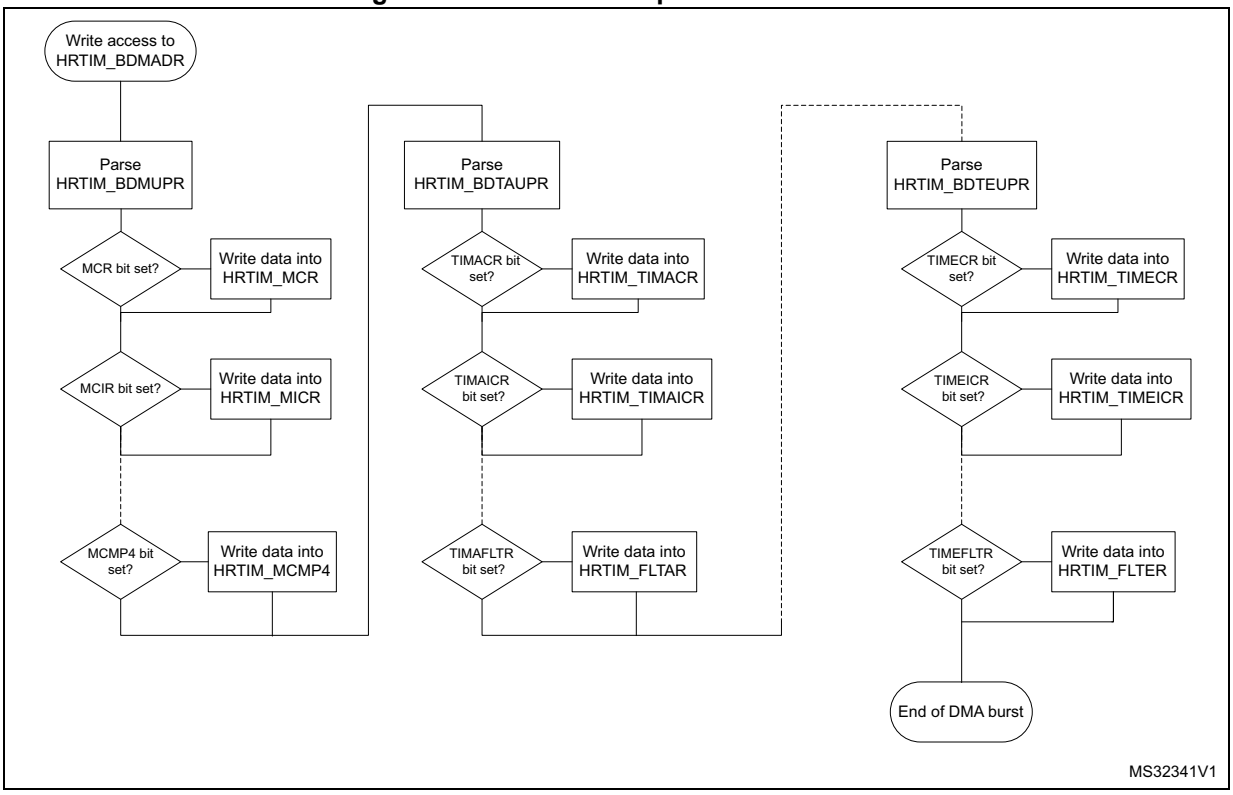
Note: Any trigger occurring while the burst is on-going is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enable bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data re-routing to the final destination register).

To re-initialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the 6 update registers.

Figure 299. Burst DMA operation flowchart



Several options are available once the DMA burst is completed, depending on the register update strategy.

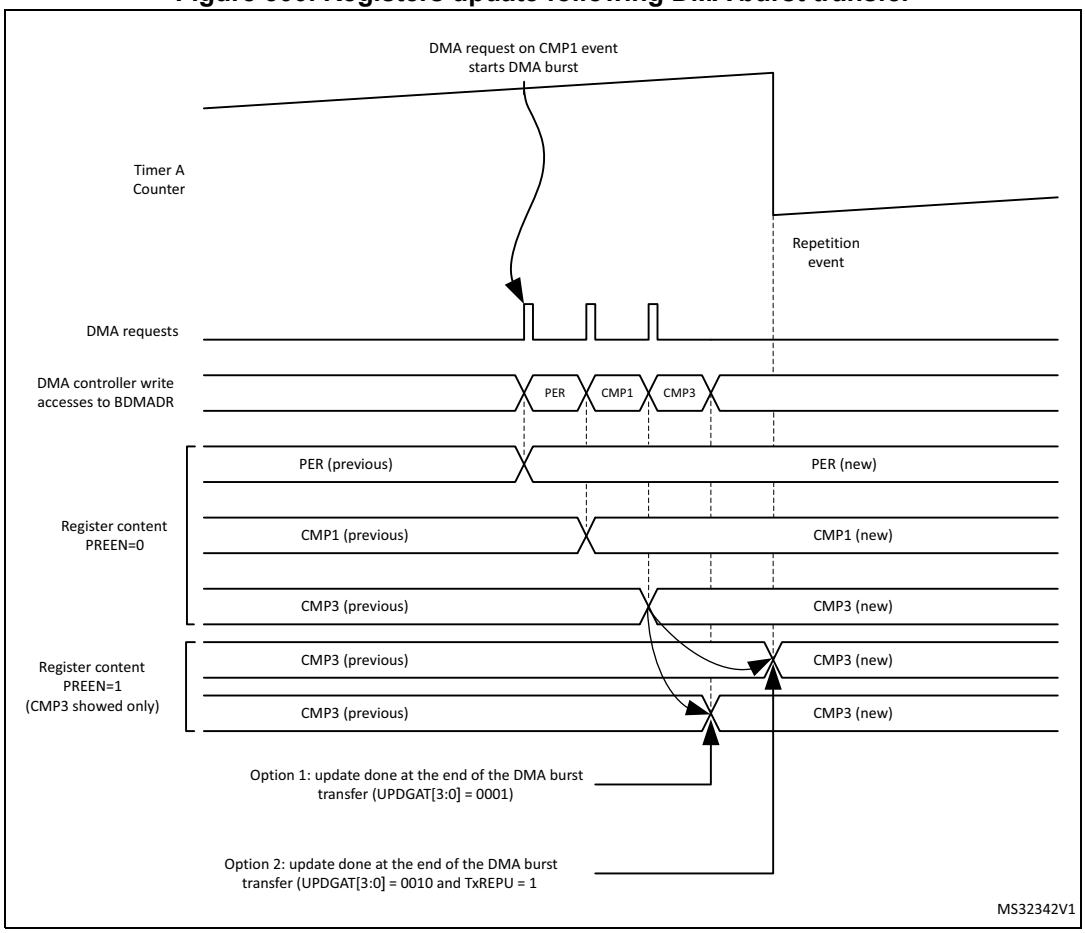
If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are 3 use cases:

1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is on-going, only part of the registers will be loaded and the complete register update will require 2 consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).
3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is on-going, it will be discarded and the effective update will happen on the next coming update request.

The chronogram on *Figure 300* presents the active register content for 3 cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 300. Registers update following DMA burst transfer



21.3.22 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the Reset and Clock control unit (RCC), while respecting the $_{fHRTIM}$ range for the DLL lock.

The DLL calibration must be started by setting CAL bit in HRTIM_DLLCR register.

The HRTIM master and timing units can be started only once the high-resolution unit is ready. This is indicated by the DLLRDY flag set. The DLLRDY flag can be polled before resuming the initialization or the calibration can run in background while other registers of the HRTIM or other MCU peripherals are initialized. In this case, the DLLRDY flag must be checked before starting the counters (an end-of-calibration interrupt can be issued if necessary, enabled with DLLRDYIE flag in HRTIM_IER). Once the DLL calibration is done, CALEN bit must be set to have it done periodically and compensate for potential voltage and temperature drifts. The calibration periodicity is defined using the CALRTE[1:0] bitfield in the HRTIM_DLLCR register.

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- the polarity must be defined using POLx bits in HRTIM_OUTxR
- the FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the product datasheet.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The 2 outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) are ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the Reset and Clock Controller, the output control is released to the GPIO controller and the outputs are tri stated.

21.3.23 Debug

When a microcontroller enters the debug mode (Cortex®-M4 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_HRTIM_STOP configuration bit in DBG module:

- DBG_HRTIM_STOP = 0: no behavior change, the HRTIM continues to operate.
- DBG_HRTIM_STOP = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This allows to maintain a safe state during the execution stepping. The outputs can be enabled again by settings TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when DBG_HRTIM_STOP = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output will keep on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN will be set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by DBG_HRTIM_STOP bit state).

Synchronization, counter reset, start and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event will be emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but will be active again as soon as the debug will be exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on UPD_EN[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

DLL calibration is not blocked while the MCU is halted (the DLLRDY flag can be set).

21.4 Application use cases

21.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 10 buck converters with 6 independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:.

$$V_{out} = D \times V_{in}$$

The topology is given on *Figure 301* with the connection to the ADC for voltage reading.

Figure 301. Buck converter topology

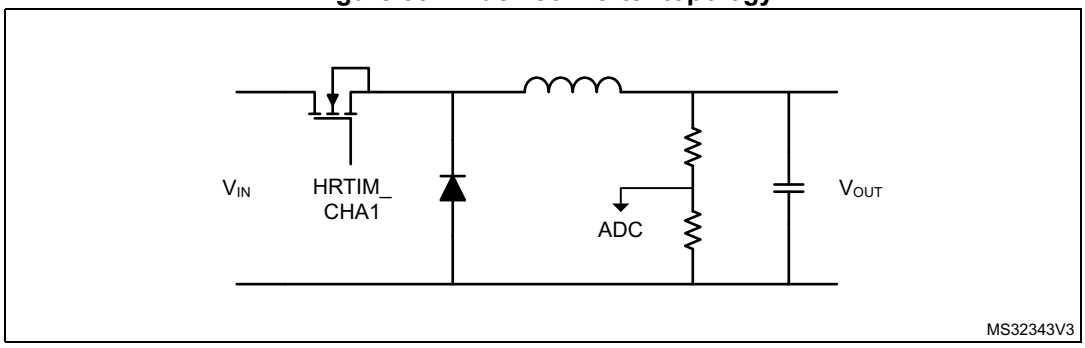
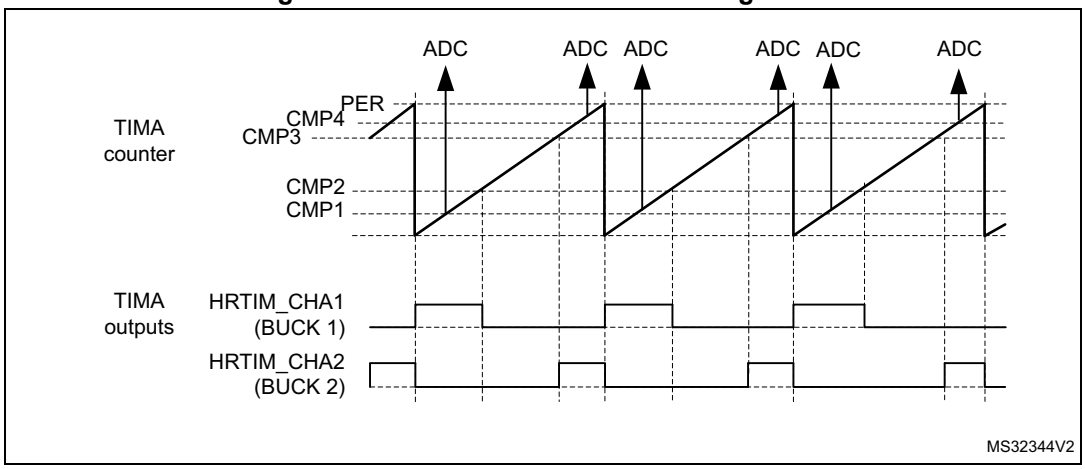


Figure 302 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1
- HRTIM_CHA2 set on CMP3, reset on PER

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 302. Dual Buck converter management

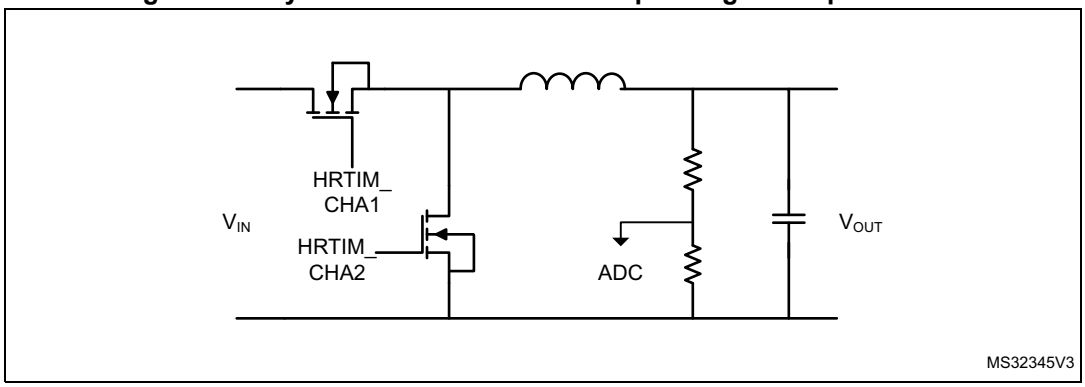


Timers A..E provide either 10 buck converters coupled by pairs (both with identical switching frequencies) or 6 completely independent converters (each of them having a different switching frequency), using the master timer as the 6th time base.

21.4.2 Buck converter with synchronous rectification

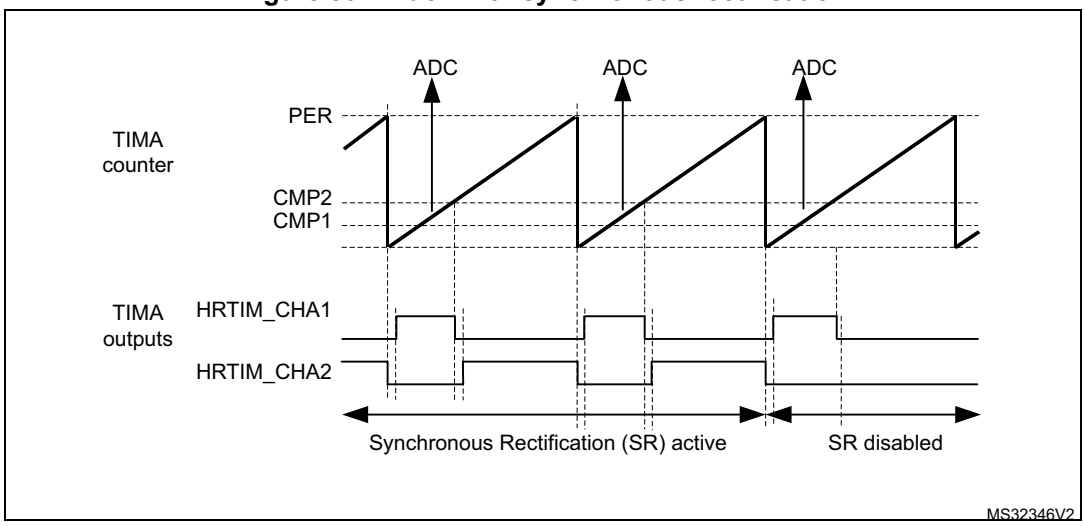
Synchronous rectification allows to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown on *Figure 303*.

Figure 303. Synchronous rectification depending on output current



The main difference vs. a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (see *Figure 304*).

Figure 304. Buck with synchronous rectification



21.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

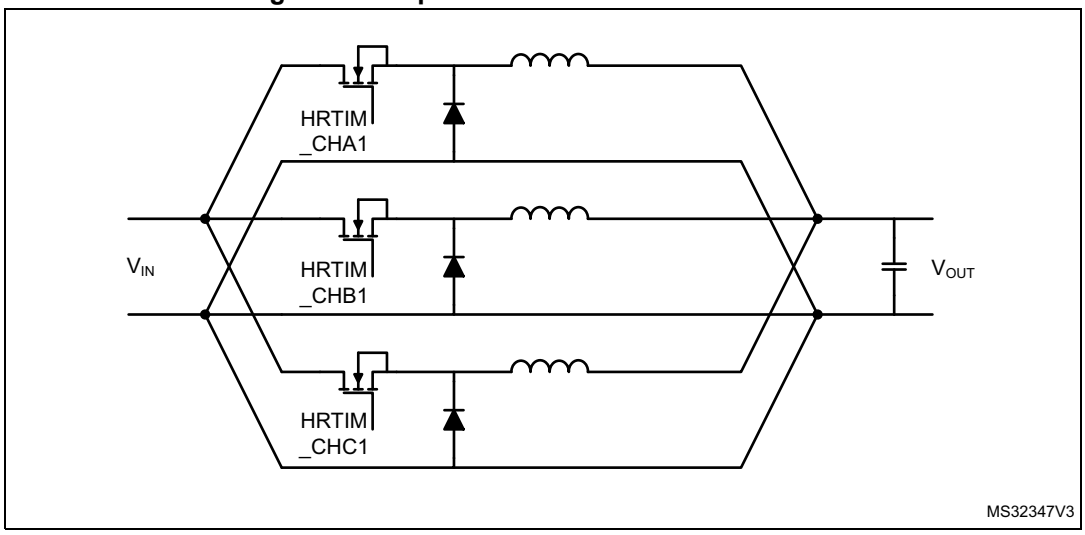
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM is able to manage multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- 5 buck converters with synchronous rectification (SR), using the master timer and the 5 timers
- 4 buck converters (without SR), using the master timer and 2 timers
- ...

Figure 306 presents the topology of a 3-phase interleaved buck converter.

Figure 305. 3-phase interleaved buck converter



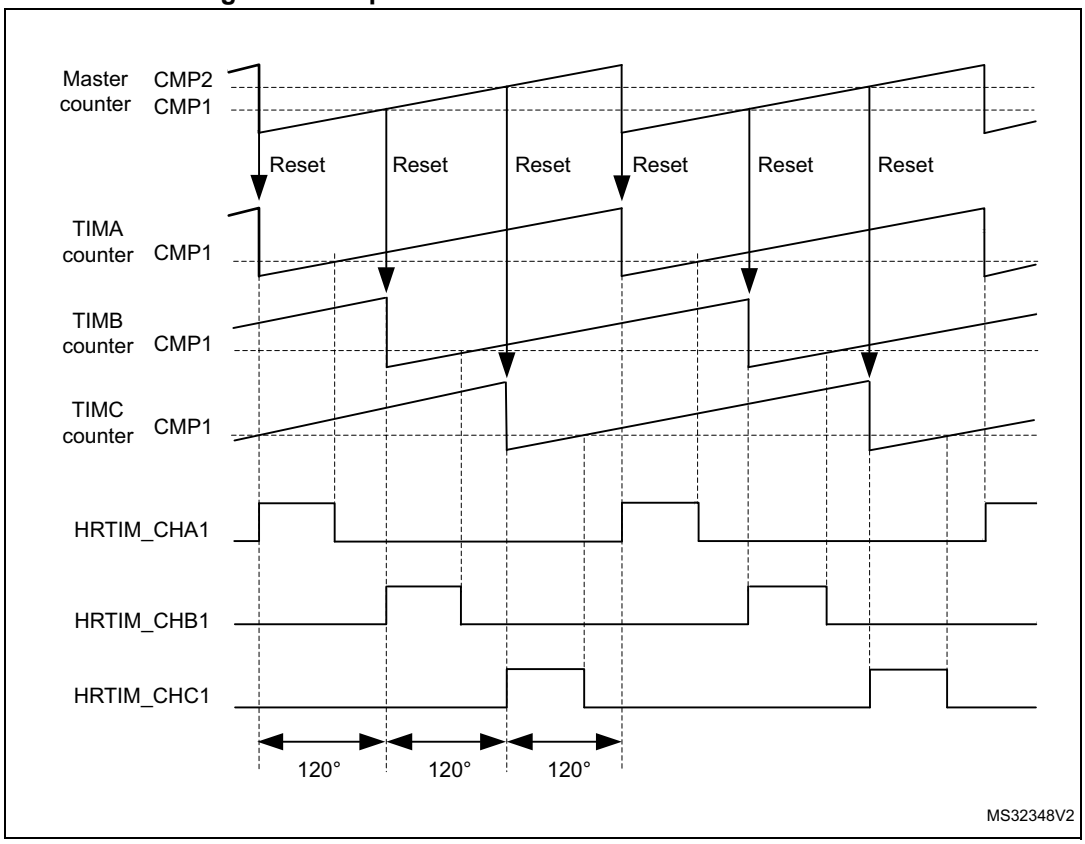
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multi-phase converter).

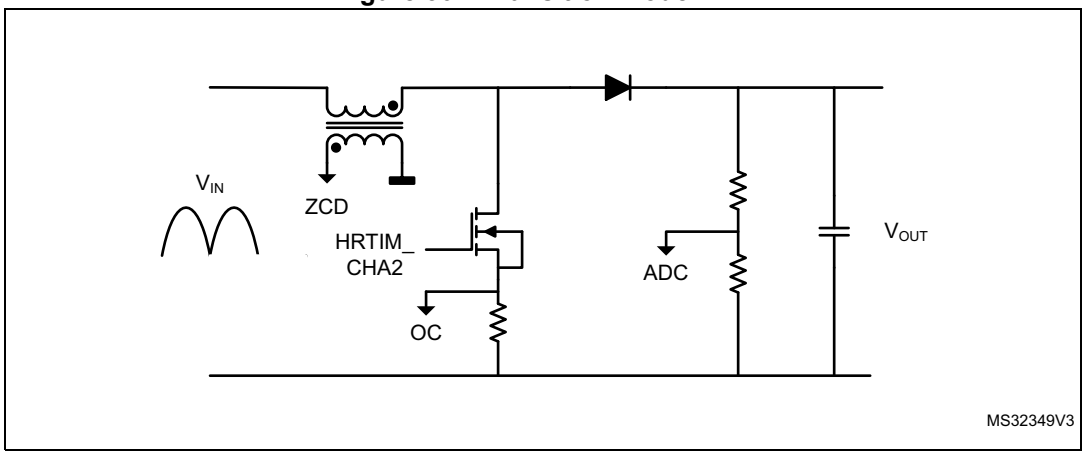
Figure 306. 3-phase interleaved buck converter control



21.4.4 Transition mode Power Factor Correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current will then decay during the Toff time, and the period will be re-started when it becomes null. This is detected using a Zero Crossing Detection circuitry (ZCD), as shown on *Figure 307*. With a constant Ton time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 307. Transition mode PFC



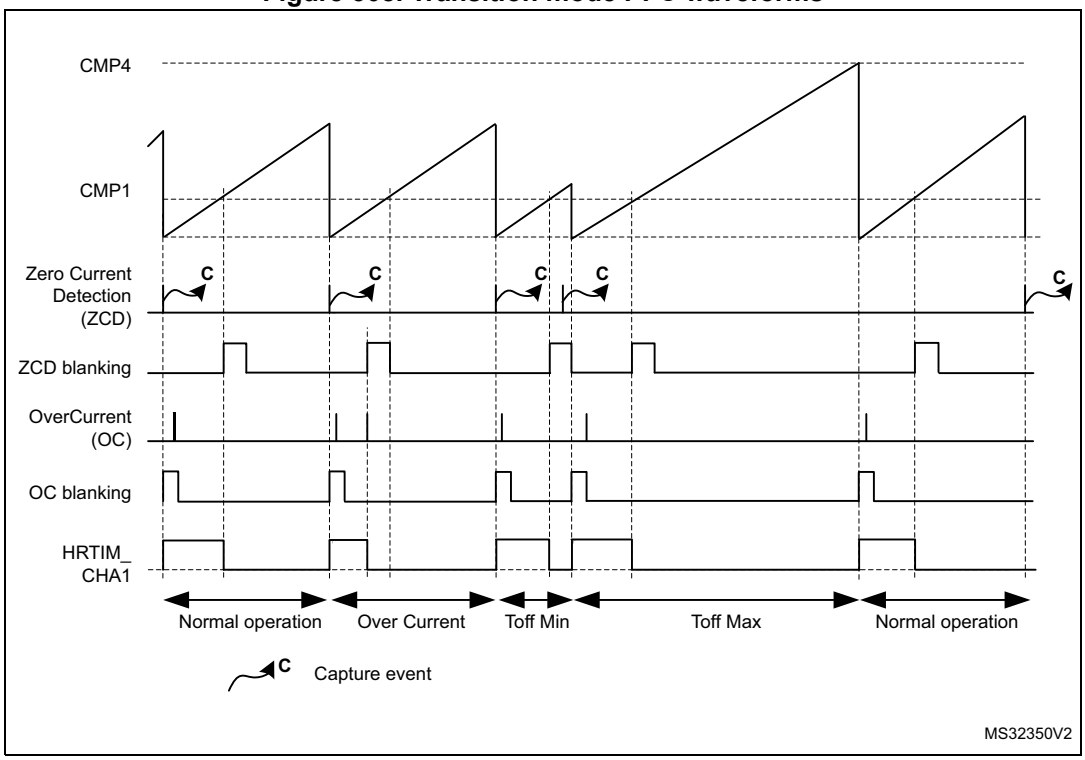
This converter is operating with a constant Ton time and a variable frequency due the Toff time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of over-current (OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 308 presents the waveform during the various operating modes, with the following parameters defined:

- Ton Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking
- Ton Max: practically, the converter set-point. It is defined by CMP1
- Toff Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- Toff Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in auto-delayed mode.

Both Toff values are auto-delayed since the value must be relative to the output falling edge.

Figure 308. Transition mode PFC waveforms



21.5 HRTIM registers

21.5.1 HRTIM Master Timer Control Register (HRTIM_MCR)

Address offset: 0x0000h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	Res.	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
rw	rw	rw		rw	rw	rw				rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		Res.	Res.	HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: *Burst DMA Update*

These bits define how the update occurs relatively to a burst DMA transaction.
00: Update done independently from the DMA burst transfer completion
01: Update done when the DMA burst transfer is completed
10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.
11: reserved

Bit 29 **MREPU**: Master *Timer Repetition update*

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.
0: Update on repetition disabled
1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: *Preload enable*

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM active or preload registers.
0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** *DAC Synchronization*

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 21.3.19: DAC triggers* for connections details).
00: No DAC trigger generated
01: Trigger generated on DACtrigOut1
10: Trigger generated on DACtrigOut2
11: Trigger generated on DACtrigOut3

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **TECEN**: *Timer E counter enable*

This bit starts the Timer E counter.
0: Timer E counter disabled
1: Timer E counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: *Timer D counter enable*
This bit starts the Timer D counter.
0: Timer D counter disabled
1: Timer D counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: *Timer C counter enable*
This bit starts the Timer C counter.
0: Timer C counter disabled
1: Timer C counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: *Timer B counter enable*
This bit starts the Timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: *Timer A counter enable*
This bit starts the Timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: *Master timer counter enable*
This bit starts the Master timer counter.
0: Master counter disabled
1: Master counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: *Synchronization source*
These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer Start
01: Master timer Compare 1 event
10: Timer A start/reset
11: Timer A Compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: *Synchronization output*
These bits define the routing and conditioning of the synchronization output event.
00: disabled
01: Reserved.
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: *Synchronization Starts Master*
This bit enables the Master timer start when receiving a synchronization input event:
0: No effect on the Master timer
1: A synchronization input event starts the Master timer

Bit 10 **SYNCRSTM**: *Synchronization Resets Master*
This bit enables the Master timer reset when receiving a synchronization input event:
0: No effect on the Master timer
1: A synchronization input event resets the Master timer

Bits 9:8 **SYNCIN[1:0]** *Synchronization input*

These bits are defining the synchronization input source.
00: disabled. HRTIM is not synchronized and runs in standalone mode.
01: Reserved.
10: Internal event: the HRTIM is synchronized with the on-chip timer (see *Synchronization input*).
11: External event (input pin). A positive pulse on HRTIM_SCIN input triggers the HRTIM.

Note: This parameter cannot be changed once the impacted timers are enabled.

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

Bit 5 **HALF**: *Half mode*

This bit enables the half duty-cycle mode: the HRTIM_MCMP1xR active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.
0: Half mode disabled
1: Half mode enabled

Bit 4 **RETRIG**: *Re-triggerable mode*

This bit defines the behavior of the master timer counter in single-shot mode.
0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)
1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: *Continuous mode*

0: The timer operates in single-shot mode and stops when it reaches the MPER value
1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: *Clock prescaler*

These bits define the master timer high-resolution clock prescaler ratio.
The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.
The prescaling ratio cannot be modified once the timer is enabled.

21.5.2 HRTIM Master Timer Interrupt Status Register (HRTIM_MISR)

Address offset: 0x0004h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
									r	r	r	r	r	r	r

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

Bit 6 **MUPD**: Master Update Interrupt Flag
This bit is set by hardware when the Master timer registers are updated.
0: No Master Update interrupt occurred
1: Master Update interrupt occurred

Bit 5 **SYNC**: Sync Input Interrupt Flag
This bit is set by hardware when a synchronization input event is received.
0: No Sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master Repetition Interrupt Flag
This bit is set by hardware when the Master timer repetition period has elapsed.
0: No Master Repetition interrupt occurred
1: Master Repetition interrupt occurred

Bit 3 **MCMP4**: Master Compare 4 Interrupt Flag
Refer to MCMP1 description

Bit 2 **MCMP3**: Master Compare 3 Interrupt Flag
Refer to MCMP1 description

Bit 1 **MCMP2**: Master Compare 2 Interrupt Flag
Refer to MCMP1 description

Bit 0 **MCMP1**: Master Compare 1 Interrupt Flag
This bit is set by hardware when the Master timer counter matches the value programmed in the master Compare 1 register.
0: No Master Compare 1 interrupt occurred
1: Master Compare 1 interrupt occurred

21.5.3 HRTIM Master Timer Interrupt Clear Register (HRTIM_MICR)

Address offset: 0x0008h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

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

Bit 6 **MUPDC**: Master update Interrupt flag clear
Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register

Bit 5 **SYNCC**: Sync Input Interrupt flag clear
Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register

Bit 4 **MREPC**: Repetition Interrupt flag clear
Writing 1 to this bit clears the MREP flag in HRTIM_MISR register

Bit 3 **MCMP4C**: Master Compare 4 Interrupt flag clear
Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register

Bit 2 **MCMP3C**: Master Compare 3 Interrupt flag clear
Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register

Bit 1 **MCMP2C**: Master Compare 2 Interrupt flag clear
Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register

Bit 0 **MCMP1C**: Master Compare 1 Interrupt flag clear
Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register

21.5.4 HRTIM Master Timer DMA / Interrupt Enable Register (HRTIM_MDIER)

Address offset: 0x000Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **MUPDDE**: Master Update DMA request Enable
This bit is set and cleared by software to enable/disable the Master update DMA requests.
0: Master update DMA request disabled
1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync Input DMA request Enable
This bit is set and cleared by software to enable/disable the Sync input DMA requests.
0: Sync input DMA request disabled
1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master Repetition DMA request Enable
This bit is set and cleared by software to enable/disable the Master timer repetition DMA requests.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master Compare 4 DMA request Enable
Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master Compare 3 DMA request Enable
Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master Compare 2 DMA request Enable
Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master Compare 1 DMA request Enable
This bit is set and cleared by software to enable/disable the Master timer Compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 6 **MUPDIE**: Master Update Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer registers update interrupts
0: Master update interrupts disabled
1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync Input Interrupt Enable
This bit is set and cleared by software to enable/disable the Sync input interrupts
0: Sync input interrupts disabled
1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master Repetition Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer repetition interrupts
0: Master repetition interrupt disabled
1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master Compare 4 Interrupt Enable
Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master Compare 3 Interrupt Enable
Refer to MCMP1IE description

Bit 1 **MCMP2IE**: MAster Compare 2 Interrupt Enable
Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master Compare 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer Compare 1 interrupt
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

21.5.5 HRTIM Master Timer Counter Register (HRTIM_MCNTR)

Address offset: 0x0010h

Reset value: 0x0000 0000

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

Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: For HR clock prescaling ratio below 32 (CKPSCCKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

21.5.6 HRTIM Master Timer Period Register (HRTIM_MPER)

Address offset: 0x0014h

Reset value: 0x0000 FFDF

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

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

21.5.7 HRTIM Master Timer Repetition Register (HRTIM_MREP)

Address offset: 0x0018h

Reset value: 0x0000 0000

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

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

Bits 7:0 **MREP[7:0]**: *Master Timer Repetition period value*

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

21.5.8 HRTIM Master Timer Compare 1 Register (HRTIM_MCMP1R)

Address offset: 0x001Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP1[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 **MCMP1[15:0]**: *Master Timer Compare 1 value*

This register holds the master timer Compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.9 HRTIM Master Timer Compare 2 Register (HRTIM_MCMP2R)

Address offset: 0x0024h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP2[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 **MCMP2[15:0]**: *Master Timer Compare 2 value*

This register holds the master timer Compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.10 HRTIM Master Timer Compare 3 Register (HRTIM_MCMP3R)

Address offset: 0x0028h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP3[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 **MCMP3[15:0]**: *Master Timer Compare 3 value*

This register holds the master timer Compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.11 HRTIM Master Timer Compare 4 Register (HRTIM_MCMP4R)

Address offset: 0x002Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP4[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 **MCMP4[15:0]**: *Master Timer Compare 4 value*

This register holds the master timer Compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.12 HRTIM Timerx Control Register (HRTIM_TIMxCR)

Address offset: 0x0000h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	Res.	TxRSTU	TxREPU	Res.
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
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	Res.	Res.	Res.	PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: *Update Gating*

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs 1 to 3 (see *Table 91: Update enable inputs and sources*)
The update events, as mentioned below, can be: MSTU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.
0000: the update occurs independently from the DMA burst transfer
0001: the update occurs when the DMA burst transfer is completed
0010: the update occurs on the update event following the DMA burst transfer completion
0011: the update occurs on a rising edge of HRTIM update enable input 1
0100: the update occurs on a rising edge of HRTIM update enable input 2
0101: the update occurs on a rising edge of HRTIM update enable input 3
0110: the update occurs on the update event following a rising edge of HRTIM update enable input 1
0111: the update occurs on the update event following a rising edge of HRTIM update enable input 2
1000: the update occurs on the update event following a rising edge of HRTIM update enable input 3
Other codes: reserved

Note: This bitfield must be reset before programming a new value.
For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: *Preload enable*

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register is done into the active or the preload register.
0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** *DAC Synchronization*

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 21.3.19: DAC triggers* for connections details).
00: No DAC trigger generated
01: Trigger generated on DACtrigOut1
10: Trigger generated on DACtrigOut2
11: Trigger generated on DACtrigOut3

Bit 24 **MSTU**: *Master Timer update*

Register update is triggered by the master timer update.
0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR:

TEU: *Timer E update*

Register update is triggered by the timer E update
0: Update by timer E disabled
1: Update by timer E enabled

In HRTIM_TIMECR:

Reserved, must be kept at reset value

Bit 22 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR:

TDU: *Timer D update*

Register update is triggered by the timer D update
0: Update by timer D disabled
1: Update by timer D enabled

In HRTIM_TIMDCR:

Reserved, must be kept at reset value

Bit 21 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR:

TCU: *Timer C update*

Register update is triggered by the timer C update
0: Update by timer C disabled
1: Update by timer C enabled

In HRTIM_TIMCCR:

Reserved, must be kept at reset value

Bit 20 In HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR:

TBU: *Timer B update*

Register update is triggered by the timer B update
0: Update by timer B disabled
1: Update by timer B enabled

In HRTIM_TIMBCR:

Reserved, must be kept at reset value

Bit 19 In HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR:

TAU: *Timer A update*

Register update is triggered by the timer A update
0: Update by timer A disabled
1: Update by timer A enabled

In HRTIM_TIMACR:

Reserved, must be kept at reset value

Bit 18 **TxRSTU**: *Timerx reset update*

Register update is triggered by Timerx counter reset or roll-over to 0 after reaching the period value in continuous mode.
0: Update by timer x reset / roll-over disabled
1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: *Timer x Repetition update*
Register update is triggered when the counter rolls over and HRTIM_REPx = 0
0: Update on repetition disabled
1: Update on repetition enabled

Bit 16 Reserved, must be kept at reset value.

Bits 15:14 **DELCMP4[1:0]**: *CMP4 auto-delayed mode*
This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Auto-delayed mode*).
00: CMP4 register is always active (standard compare mode)
01: CMP4 value is recomputed and is active following a capture 2 event
10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)
11: CMP4 value is recomputed and is active following a capture event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bits 13:12 **DELCMP2[1:0]**: *CMP2 auto-delayed mode*
This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Auto-delayed mode*).
00: CMP2 register is always active (standard compare mode)
01: CMP2 value is recomputed and is active following a capture 1 event
10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after Compare 1 match (timeout function if capture event is missing)
11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTx**: *Synchronization Starts Timer x*
This bit defines the Timer x behavior following the synchronization event:
0: No effect on Timer x
1: A synchronization input event starts the Timer x

Bit 10 **SYNCRSTx**: *Synchronization Resets Timer x*
This bit defines the Timer x behavior following the synchronization event:
0: No effect on Timer x
1: A synchronization input event resets the Timer x

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PSHPLL**: *Push-Pull mode enable*
This bit enables the push-pull mode.
0: Push-Pull mode disabled
1: Push-Pull mode enabled
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 5 **HALF**: *Half mode enable*
This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.
0: Half mode disabled
1: Half mode enabled

Bit 4 **RETRIG**: *Re-triggerable mode*

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: *Continuous mode*

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: *HRTIM Timer x Clock prescaler*

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

21.5.13 HRTIM Timerx Interrupt Status Register (HRTIM_TIMxISR)

Address offset: 0x0004h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	r	r	r	r	r	r	r	r	r		r	r	r	r	r

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

Bit 21 **O2CPY**: Output 2 Copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 Copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 Status

This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 2 was inactive

1: Output 2 was active

Bit 18 **O1STAT**: Output 1 Status

This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 1 was inactive

1: Output 1 was active

Bit 17 **IPPSTAT**: Idle Push Pull Status

This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).

0: Protection occurred when the output 1 was active and output 2 forced inactive

1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current Push Pull Status

This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.

0: Signal applied on output 1 and output 2 forced inactive

1: Signal applied on output 2 and output 1 forced inactive

Bit 15 Reserved

Bit 14 **DLYPRT**: Delayed Protection Flag

This bit indicates delayed idle or the balanced idle mode entry.

Bit 13 **RST**: Reset and/or roll-over Interrupt Flag
This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.
0: No TIMx counter reset/roll-over interrupt occurred
1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 Reset Interrupt Flag
Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 Set Interrupt Flag
Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 Reset Interrupt Flag
This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).
0: No Tx1 output reset interrupt occurred
1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 Set Interrupt Flag
This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).
0: No Tx1 output set interrupt occurred
1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture2 Interrupt Flag
Refer to CPT1 description

Bit 7 **CPT1**: Capture1 Interrupt Flag
This bit is set by hardware when the timer x capture 1 event occurs.
0: No timer x Capture 1 reset interrupt occurred
1: Timer x output 1 reset interrupt occurred

Bit 6 **UPD**: Update Interrupt Flag
This bit is set by hardware when the timer x update event occurs.
0: No timer x update interrupt occurred
1: Timer x update interrupt occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REP**: Repetition Interrupt Flag
This bit is set by hardware when the timer x repetition period has elapsed.
0: No timer x repetition interrupt occurred
1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 Interrupt Flag
Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 Interrupt Flag
Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 Interrupt Flag
Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 Interrupt Flag
This bit is set by hardware when the timer x counter matches the value programmed in the Compare 1 register.
0: No Compare 1 interrupt occurred
1: Compare 1 interrupt occurred

21.5.14 HRTIM Timerx Interrupt Clear Register (HRTIM_TIMxICR)

Address offset: 0x0008h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 14 **DLYPRTC**: Delayed Protection Flag Clear
Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset Interrupt flag Clear
Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 Reset flag Clear
Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 Set flag Clear
Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 Reset flag Clear
Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 Set flag Clear
Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture2 Interrupt flag Clear
Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture1 Interrupt flag Clear
Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update Interrupt flag Clear
Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPC**: Repetition Interrupt flag Clear
Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 Interrupt flag Clear
Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 Interrupt flag Clear
Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 Interrupt flag Clear
Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 Interrupt flag Clear
Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

21.5.15 HRTIM Timerx DMA / Interrupt Enable Register (HRTIM_TIMxDIER)

Address offset: 0x000Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved

Bit 30 **DLYPRTDE**: Delayed Protection DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on delayed protection.
0: Delayed protection DMA request disabled
1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over DMA request disabled
1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 Reset DMA request Enable
Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 Set DMA request Enable
Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 Reset DMA request Enable
This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.
0: Tx1 output reset DMA request disabled
1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 Set DMA request Enable
This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.
0: Tx1 output set DMA request disabled
1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request Enable
Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request Enable
This bit is set and cleared by software to enable/disable Capture 1 DMA requests.
0: Capture 1 DMA request disabled
1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on update event.
0: Update DMA request disabled
1: Update DMA request enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **REPDE**: Repetition DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on repetition event.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request Enable
Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request Enable
Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request Enable
Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request Enable
This bit is set and cleared by software to enable/disable the Compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bit 15 Reserved

Bit 14 **DLYPRTIE**: Delayed Protection Interrupt Enable
This bit is set and cleared by software to enable/disable interrupts on delayed protection.
0: Delayed protection interrupts disabled
1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over Interrupt Enable
This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over interrupt disabled
1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 Reset Interrupt Enable
Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 Set Interrupt Enable
Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 Reset Interrupt Enable
This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.
0: Tx1 output reset interrupts disabled
1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 Set Interrupt Enable
This bit is set and cleared by software to enable/disable Tx1 output set interrupts.
0: Tx1 output set interrupts disabled
1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture Interrupt Enable
Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture Interrupt Enable
This bit is set and cleared by software to enable/disable Capture 1 interrupts.
0: Capture 1 interrupts disabled
1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update Interrupt Enable
This bit is set and cleared by software to enable/disable update event interrupts.
0: Update interrupts disabled
1: Update interrupts enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPIE**: Repetition Interrupt Enable
This bit is set and cleared by software to enable/disable repetition event interrupts.
0: Repetition interrupts disabled
1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 Interrupt Enable
Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 Interrupt Enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 Interrupt Enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

21.5.16 HRTIM Timerx Counter Register (HRTIM_CNTxR)

Address offset: 0x0010h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

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

This register holds the Timerx counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: For HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

21.5.17 HRTIM Timerx Period Register (HRTIM_PERxR)

Address offset: 0x14h (this offset address is relative to timer x base address)

Reset value: 0x0000 FFDF

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

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

21.5.18 HRTIM Timerx Repetition Register (HRTIM_REPxR)

Address offset: 0x18h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

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

Bits31:8 Reserved, must be kept at reset value.

Bits 7:0 **REPx[7:0]**: *Timerx Repetition period value*

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

21.5.19 HRTIM Timerx Compare 1 Register (HRTIM_CMP1xR)

Address offset: 0x1Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[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 **CMP1x[15:0]**: *Timerx Compare 1 value*

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.20 HRTIM Timerx Compare 1 Compound Register (HRTIM_CMP1CxR)

Address offset: 0x20h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[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
CMP1x[15:0]															
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 **REPx[7:0]**: *Timerx Repetition value (aliased from HRTIM_REPx register)*
This bitfield is an alias from the REPx[7:0] bitfield in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: *Timerx Compare 1 value*
This bitfield is an alias from the CMP1x[15:0] bitfield in the HRTIMx_CMP1xR register.

21.5.21 HRTIM Timerx Compare 2 Register (HRTIM_CMP2xR)

Address offset: 0x24h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2x[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 **CMP2x[15:0]**: *Timerx Compare 2 value*
This register holds the Compare 2 value.
This register holds either the content of the preload register or the content of the active register if preload is disabled.
The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...
This register can behave as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR.

21.5.22 HRTIM Timerx Compare 3 Register (HRTIM_CMP3xR)

Address offset: 0x28h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3x[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 **CMP3x[15:0]**: *Timerx Compare 3 value*

This register holds the Compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

21.5.23 HRTIM Timerx Compare 4 Register (HRTIM_CMP4xR)

Address offset: 0x2Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP4x[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 **CMP4x[15:0]**: *Timerx Compare 4 value*

This register holds the Compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR.

21.5.24 HRTIM Timerx Capture 1 Register (HRTIM_CPT1xR)

Address offset: 0x30h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[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 **CPT1x[15:0]**: *Timerx Capture 1 value*

This register holds the counter value when the capture 1 event occurred.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

21.5.25 HRTIM Timerx Capture 2 Register (HRTIM_CPT2xR)

Address offset: 0x34h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT2x[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 **CPT2x[15:0]**: *Timerx Capture 2 value*

This register holds the counter value when the capture 2 event occurred.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

21.5.26 HRTIM Timerx Deadtime Register (HRTIM_DTxR)

Address offset: 0x38h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]								
rwo	rwo					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTRLKx	DTRSLKx	Res.	DTPRSC[1:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: *Deadtime Falling Lock*

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: *Deadtime Falling Sign Lock*

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **SDTFx**: *Sign Deadtime Falling value*

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: *Deadtime Falling value*

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: *Deadtime Rising Lock*

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled

0: Deadtime rising value and sign is writable

1: Deadtime rising value and sign is read-only

Note: This bit is not preloaded

Bit 14 **DTRSLKx**: *Deadtime Rising Sign Lock*

This write-once bit prevents the sign of deadtime to be modified, if enabled

0: Deadtime rising sign is writable

1: Deadtime rising sign is read-only

Note: This bit is not preloaded

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: *Deadtime Prescaler*

This register holds the value of the deadtime clock prescaler.

$t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$

(i.e. 000: 868 ps, 001= 1.736ns,...)

This bitfield is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: *Sign Deadtime Rising value*

This register determines whether the deadtime is positive or negative (overlapping signals)

0: Positive deadtime on rising edge

1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: *Deadtime Rising value*

This register holds the value of the deadtime following a rising edge of reference PWM signal.

$t_{DTR} = DTRx[8:0] \times t_{DTG}$

21.5.27 HRTIM Timerx Output1 Set Register (HRTIM_SETx1R)

Address offset: 0x3Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
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
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: *Registers update (transfer preload to active)*
Register update event forces the output to its active state.

Bit 30 **EXTEVNT10**: *External Event 10*
Refer to EXTEVNT1 description

Bit 29 **EXTEVNT9**: *External Event 9*
Refer to EXTEVNT1 description

Bit 28 **EXTEVNT8**: *External Event 8*
Refer to EXTEVNT1 description

Bit 27 **EXTEVNT7**: *External Event 7*
Refer to EXTEVNT1 description

Bit 26 **EXTEVNT6**: *External Event 6*
Refer to EXTEVNT1 description

Bit 25 **EXTEVNT5**: *External Event 5*
Refer to EXTEVNT1 description

Bit 24 **EXTEVNT4**: *External Event 4*
Refer to EXTEVNT1 description

Bit 23 **EXTEVNT3**: *External Event 3*
Refer to EXTEVNT1 description

Bit 22 **EXTEVNT2**: *External Event 2*
Refer to EXTEVNT1 description

Bit 21 **EXTEVNT1**: *External Event 1*
External event 1 forces the output to its active state.

Bit 20 **TIMEVNT9**: *Timer Event 9*
Refer to TIMEVNT1 description

Bit 19 **TIMEVNT8**: *Timer Event 8*
Refer to TIMEVNT1 description

Bit 18 **TIMEVNT7**: *Timer Event 7*
Refer to TIMEVNT1 description

Bit 17 **TIMEVNT6**: *Timer Event 6*
Refer to TIMEVNT1 description

Bit 16 **TIMEVNT5**: *Timer Event 5*
Refer to TIMEVNT1 description

Bit 15 **TIMEVNT4**: *Timer Event 4*
Refer to TIMEVNT1 description

Bit 14 **TIMEVNT3**: *Timer Event 3*
Refer to TIMEVNT1 description

Bit 13 **TIMEVNT2**: *Timer Event 2*
Refer to TIMEVNT1 description

Bit 12 **TIMEVNT1**:*Timer Event 1*
Timers event 1 forces the output to its active state (refer to *Table 84* for Timer Events assignments)

Bit 11 **MSTCMP4**: *Master Compare 4*
Master Timer Compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: *Master Compare 3*
Master Timer Compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: *Master Compare 2*
Master Timer Compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: *Master Compare 1*
Master Timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: *Master Period*
The master timer counter roll-over in continuous mode, or to the master timer reset in single-shot mode forces the output to its active state.

Bit 6 **CMP4**: *Timer x Compare 4*
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: *Timer x Compare 3*
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: *Timer x Compare 2*
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: *Timer x Compare 1*
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: *Timer x Period*
Timer A Period event forces the output to its active state.

Bit 1 **RESYNC:** *Timer A resynchronization*
Timer A reset event coming solely from software or SYNC input forces the output to its active state.
Note: Other timer reset are not affecting the output when RESYNC=1

Bit 0 **SST**: *Software Set trigger*
This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.
Note: This bit is not preloaded

21.5.28 HRTIM Timerx Output1 Reset Register (HRTIM_RSTx1R)

Address offset: 0x40h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
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
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx1 output to its inactive state.

21.5.29 HRTIM Timerx Output2 Set Register (HRTIM_SETx2R)

Address offset: 0x44h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
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
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx2 output to its active state.

21.5.30 HRTIM Timerx Output2 Reset Register (HRTIM_RSTx2R)

Address offset: 0x48h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
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
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx2 output to its inactive state.

21.5.31 HRTIM Timerx External Event Filtering Register 1 (HRTIM_EEFxR1)

Address offset: 0x4Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3FLTR[2:0]			EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
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 28:25 **EE5FLTR[3:0]**: *External Event 5 filter*
Refer to EE1FLTR[3:0] description

Bit 24 **EE5LTCH**: *External Event 5 latch*
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE4FLTR[3:0]**: *External Event 4 filter*
Refer to EE1FLTR[3:0] description

Bit 18 **EE4LTCH**: *External Event 4 latch*
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE3FLTR[3:0]**: *External Event 3 filter*
Refer to EE1FLTR[3:0] description

Bit 12 **EE3LTCH**: *External Event 3 latch*
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE2FLTR[3:0]**: *External Event 2 filter*
Refer to EE1FLTR[3:0] description

Bit 6 **EE2LTCH**: *External Event 2 latch*
Refer to EE1LTCH description

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE1FLTR[3:0]**: *External Event 1 filter*

0000: No filtering
0001: Blanking from counter reset/roll-over to Compare 1
0010: Blanking from counter reset/roll-over to Compare 2
0011: Blanking from counter reset/roll-over to Compare 3
0100: Blanking from counter reset/roll-over to Compare 4
0101: Blanking from another timing unit: TIMFLTR1 source (see *Table 88* for details)
0110: Blanking from another timing unit: TIMFLTR2 source (see *Table 88* for details)
0111: Blanking from another timing unit: TIMFLTR3 source (see *Table 88* for details)
1000: Blanking from another timing unit: TIMFLTR4 source (see *Table 88* for details)
1001: Blanking from another timing unit: TIMFLTR5 source (see *Table 88* for details)
1010: Blanking from another timing unit: TIMFLTR6 source (see *Table 88* for details)
1011: Blanking from another timing unit: TIMFLTR7 source (see *Table 88* for details)
1100: Blanking from another timing unit: TIMFLTR8 source (see *Table 88* for details)
1101: Windowing from counter reset/roll-over to Compare 2
1110: Windowing from counter reset/roll-over to Compare 3
1111: Windowing from another timing unit: TIMWIN source (see *Table 89* for details)

Note: Whenever a compare register is used for filtering, the value must be strictly above 0. This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EE1LTCH**: *External Event 1 latch*

0: Event 1 is ignored if it happens during a blank, or passed through during a window.
1: Event 1 is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EE1FLTR[3:0]=1101, 1110, 1111) if EE1LTCH = 0, except if the External event is programmed in fast mode (EExFAST = 1). This bitfield must not be modified once the counter is enabled (TxCEN bit set)

21.5.32 HRTIM Timerx External Event Filtering Register 2 (HRTIM_EEFxR2)

Address offset: 0x50h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8FLTR[2:0]			EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
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 28:25 **EE10FLTR[3:0]**: *External Event 10 filter*
Refer to EE1FLTR[3:0] description

Bit 24 **EE10LTCH**: *External Event 10 latch*
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE9FLTR[3:0]**: *External Event 9 filter*
Refer to EE1FLTR[3:0] description

Bit 18 **EE9LTCH**: *External Event 9 latch*
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE8FLTR[3:0]**: *External Event 8 filter*
Refer to EE1FLTR[3:0] description

Bit 12 **EE8LTCH**: *External Event 8 latch*
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE7FLTR[3:0]**: *External Event 7 filter*
Refer to EE1FLTR[3:0] description

Bit 6 **EE7LTCH**: *External Event 7 latch*
Refer to EE1LTCH description

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE6FLTR[3:0]**: *External Event 6 filter*
Refer to EE1FLTR[3:0] description

Bit 0 **EE6LTCH**: *External Event 6 latch*
Refer to EE1LTCH description

21.5.33 HRTIM Timerx Reset Register (HRTIM_RSTxR)

HRTIM TimerA Reset Register (HRTIM_RSTAR)

Address offset: 0xD4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	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
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TECPM4**: *Timer E Compare 4*
The timer A counter is reset upon timer E Compare 4 event.

Bit 29 **TECMP2**: *Timer E Compare 2*
The timer A counter is reset upon timer E Compare 2 event.

Bit 28 **TECMP1**: *Timer E Compare 1*
The timer A counter is reset upon timer E Compare 1 event.

Bit 27 **TDCMP4**: *Timer D Compare 4*
The timer A counter is reset upon timer D Compare 4 event.

Bit 26 **TDCMP2**: *Timer D Compare 2*
The timer A counter is reset upon timer D Compare 2 event.

Bit 25 **TDCMP1**: *Timer D Compare 1*
The timer A counter is reset upon timer D Compare 1 event.

Bit 24 **TCCMP4**: *Timer C Compare 4*
The timer A counter is reset upon timer C Compare 4 event.

Bit 23 **TCCMP2**: *Timer C Compare 2*
The timer A counter is reset upon timer C Compare 2 event.

Bit 22 **TCCMP1**: *Timer C Compare 1*
The timer A counter is reset upon timer C Compare 1 event.

Bit 21 **TBCMP4**: *Timer B Compare 4*
The timer A counter is reset upon timer B Compare 4 event.

Bit 20 **TBCMP2**: *Timer B Compare 2*
The timer A counter is reset upon timer B Compare 2 event.

Bit 19 **TBCMP1**: *Timer B Compare 1*
The timer A counter is reset upon timer B Compare 1 event.

Bit 18 **EXTEVNT10**: *External Event*
The timer A counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: *External Event 9*
The timer A counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: *External Event 8*
The timer A counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: *External Event 7*
The timer A counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: *External Event 6*
The timer A counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: *External Event 5*
The timer A counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: *External Event 4*
The timer A counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: *External Event 3*
The timer A counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: *External Event 2*
The timer A counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: *External Event 1*
The timer A counter is reset upon external event 1.

Bit 8 **MSTCMP4**: *Master compare 4*
The timer A counter is reset upon master timer Compare 4 event.

Bit 7 **MSTCMP3**: *Master compare 3*
The timer A counter is reset upon master timer Compare 3 event.

Bit 6 **MSTCMP2**: *Master compare 2*
The timer A counter is reset upon master timer Compare 2 event.

Bit 5 **MSTCMP1**: *Master compare 1*
The timer A counter is reset upon master timer Compare 1 event.

Bit 4 **MSTPER** *Master timer Period*
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: *Timer A compare 4 reset*
The timer A counter is reset upon Timer A Compare 4 event.

Bit 2 **CMP2**: *Timer A compare 2 reset*
The timer A counter is reset upon Timer A Compare 2 event.

Bit 1 **UPDT**: *Timer A Update reset*
The timer A counter is reset upon update event.

Bit 0 Reserved, must be kept at reset value.

HRTIM TimerB Reset Register (HRTIM_RSTBR)

Address offset: 0x154h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	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
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMC, TIMD and TIME)

HRTIM TimerC Reset Register (HRTIM_RSTCR)

Address offset: 0x1D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	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
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMD and TIME)

HRTIM TimerD Reset Register (HRTIM_RSTDR)

Address offset: 0x254h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	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
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMC and TIME)

HRTIM Timerx Reset Register (HRTIM_RSTER)

Address offset: 0x2D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXTEV NT10	EXTEV NT9	EXTEV NT8
	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
EXTEV NT7	EXTEV NT6	EXTEV NT5	EXTEV NT4	EXTEV NT3	EXTEV NT2	EXTEV NT1	MSTC MP4	MSTC MP3	MSTC MP2	MSTC MP1	MSTPE R	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMC and TIMD)

21.5.34 HRTIM Timerx Chopper Register (HRTIM_CHPxR)

Address offset: 0x58h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

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

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

Bits 10:7 **STRPW[3:0]**: *Timerx start pulsewidth*

This register defines the initial pulsewidth following a rising edge on output signal.
This bitfield cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

0000: 111 ns (1/9 MHz)
...
1111: 1.77 µs (16/9 MHz)

Bits 6:4 **CARDTY[2:0]**: *Timerx chopper duty cycle value*

This register defines the duty cycle of the carrier signal. This bitfield cannot be modified when one of the CHPx bits is set.
000: 0/8 (i.e. only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: *Timerx carrier frequency value*

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bitfield cannot be modified when one of the CHPx bits is set.
0000: 9 MHz (f_{HRTIM}/ 16)
...
1111: 562.5 kHz (f_{HRTIM} / 256)

21.5.35 HRTIM Timerx Capture 1 Control Register (HRTIM_CPT1xCR)

Address offset: 0x5Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved (for TIME only)				Reserved (for TIMD only)				Reserved (for TIMC only)				Reserved (for TIMB only)			
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
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
Reserved (for TIMA only)				EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
TACMP2	TACMP1	TA1RST	TA1SET												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

21.5.36 HRTIM Timerx Capture 2 Control Register (HRTIM_CPT2xCR)

Address offset: 0x60h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved (for TIME only)				Reserved (for TIMD only)				Reserved (for TIMC only)				Reserved (for TIMB only)			
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
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
Reserved (for TIMA only)				EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
TACMP2	TACMP1	TA1RST	TA1SET												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer E

Bit 30 **TECMP1**: Timer E Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer E

Bit 29 **TE1RST**: Timer E output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer E

Bit 28 **TE1SET**: Timer E output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer E

Bit 27 **TDCMP2**: Timer D Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer D

Bit 26 **TDCMP1**:Timer D Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer D

Bit 25 **TD1RST**: Timer D output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer D

Bit 24 **TD1SET**: Timer D output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer D

Bit 23 **TCCMP2**: Timer C Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer C

Bit 22 **TCCMP1**:Timer C Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer C

Bit 21 **TC1RST**: Timer C output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer C

Bit 20 **TC1SET**: Timer C output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer C

Bit 19 **TBCMP2**: Timer B Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer B

Bit 18 **TBCMP1**: Timer B Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer B

Bit 17 **TB1RST**: Timer B output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer B

Bit 16 **TB1SET**: Timer B output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer B

Bit 15 **TACMP2**: Timer A Compare 2
Timer A Compare 2 triggers Capture 2.
Note: This bit is reserved for Timer A

Bit 14 **TACMP1**: Timer A Compare 1
Timer A Compare 1 triggers Capture 2.
Note: This bit is reserved for Timer A

Bit 13 **TA1RST**: Timer B output 1 Reset
Capture 2 is triggered by HRTIM_CHA1 output active to inactive transition.
Note: This bit is reserved for Timer A

Bit 12 **TA1SET**: Timer B output 1 Set
Capture 2 is triggered by HRTIM_CHA1 output inactive to active transition.
Note: This bit is reserved for Timer A

Bit 11 **EXEV10CPT**: *External Event 10 Capture*
Refer to EXEV1CPT description

Bit 10 **EXEV9CPT**: *External Event 9 Capture*
Refer to EXEV1CPT description

Bit 9 **EXEV8CPT**: *External Event 8 Capture*
Refer to EXEV1CPT description

Bit 8 **EXEV7CPT**: *External Event 7 Capture*
Refer to EXEV1CPT description

Bit 7 **EXEV6CPT**: *External Event 6 Capture*
Refer to EXEV1CPT description

Bit 6 **EXEV5CPT**: *External Event 5 Capture*
Refer to EXEV1CPT description

Bit 5 **EXEV4CPT**: *External Event 4 Capture*
Refer to EXEV1CPT description

Bit 4 **EXEV3CPT**: *External Event 3 Capture*
Refer to EXEV1CPT description

Bit 3 **EXEV2CPT**: *External Event 2 Capture*
Refer to EXEV1CPT description

Bit 2 **EXEV1CPT**: *External Event 1 Capture*
The External event 1 triggers the Capture 2.

Bit 1 **UPDCPT**: *Update Capture*
The update event triggers the Capture 2.

Bit 0 **SWCPT**: *Software Capture*
This bit forces the Capture 2 by software. This bit is set only, reset by hardware

21.5.37 HRTIM Timerx Output Register (HRTIM_OUTxR)

Address offset: 0x64h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bit 23 **DIDL2**: *Output 2 Deadtime upon burst mode Idle entry*

This bit can delay the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the Output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 can be set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: *Output 2 Chopper enable*

This bit enables the chopper on output 2

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: *Output 2 Fault state*

These bits select the output 2 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: *Output 2 Idle State*

This bit selects the output 2 idle state

0: Inactive

1: Active

Note: This parameter must be set prior to have the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: *Output 2 Idle mode*

This bit selects the output 2 idle mode

0: No action: the output is not affected by the burst mode operation

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and can be changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: *Output 2 polarity*

This bit selects the output 2 polarity
0: positive polarity (output active high)
1: negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:12 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: *Delayed Protection*

These bits define the source and outputs on which the delayed protection schemes are applied.
In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:
000: Output 1 delayed Idle on external Event 6
001: Output 2 delayed Idle on external Event 6
010: Output 1 and output 2 delayed Idle on external Event 6
011: Balanced Idle on external Event 6
100: Output 1 delayed Idle on external Event 7
101: Output 2 delayed Idle on external Event 7
110: Output 1 and output 2 delayed Idle on external Event 7
111: Balanced Idle on external Event 7
In HRTIM_OUTDR, HRTIM_OUTER:
000: Output 1 delayed Idle on external Event 8
001: Output 2 delayed Idle on external Event 8
010: Output 1 and output 2 delayed Idle on external Event 8
011: Balanced Idle on external Event 8
100: Output 1 delayed Idle on external Event 9
101: Output 2 delayed Idle on external Event 9
110: Output 1 and output 2 delayed Idle on external Event 9
111: Balanced Idle on external Event 9

Note: This bitfield must not be modified once the delayed protection is enabled (DLYPRTEN bit set)

Bit 9 **DLYPRTEN**: *Delayed Protection Enable*

This bit enables the delayed protection scheme
0: No action
1: Delayed protection is enabled, as per DLYPRT[2:0] bits

Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: *Deadtime enable*

This bit enables the deadtime insertion on output 1 and output 2
0: Output 1 and output 2 signals are independent.
1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed once the timer is operating (TxEN bit set) or if its outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: *Output 1 Deadtime upon burst mode Idle entry*

This bit can delay the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.
0: The programmed Idle state is applied immediately to the Output 1
1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.
DIDL=1 can be set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: *Output 1 Chopper enable*

This bit enables the chopper on output 1

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: *Output 1 Fault state*

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: *Output 1 Idle State*

This bit selects the output 1 idle state

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: *Output 1 Idle mode*

This bit selects the output 1 idle mode

0: No action: the output is not affected by the burst mode operation

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and can be changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: *Output 1 polarity*

This bit selects the output 1 polarity

0: positive polarity (output active high)

1: negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved

21.5.38 HRTIM Timerx Fault Register (HRTIM_FLTxR)

Address offset: 0x68h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rwo															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
											rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: *Fault sources Lock*
0: FLT1EN..FLT5EN bits are read/write
1: FLT1EN..FLT5EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system reset.

Bits 30:5 Reserved, must be kept at reset value.

Bit 4 **FLT5EN**: *Fault 5 enable*
0: Fault 5 input ignored
1: Fault 5 input is active and can disable HRTIM outputs.

Bit 3 **FLT4EN**: *Fault 4 enable*
0: Fault 4 input ignored
1: Fault 4 input is active and can disable HRTIM outputs.

Bit 2 **FLT3EN**: *Fault 3 enable*
0: Fault 3 input ignored
1: Fault 3 input is active and can disable HRTIM outputs.

Bit 1 **FLT2EN**: *Fault 2 enable*
0: Fault 2 input ignored
1: Fault 2 input is active and can disable HRTIM outputs.

Bit 0 **FLT1EN**: *Fault 1 enable*
0: Fault 1 input ignored
1: Fault 1 input is active and can disable HRTIM outputs.

21.5.39 HRTIM Control Register 1 (HRTIM_CR1)

Address offset: 0x380h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
										rw	rw	rw	rw	rw	rw

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

Bits 27:25 **AD4USRC[2:0]**: *ADC Trigger 4* Update Source
Refer to AD1USRC[2:0] description

Bits 24:22 **AD3USRC[2:0]**: *ADC Trigger 3 Update Source*
Refer to AD1USRC[2:0] description

Bits 21:19 **AD2USRC[2:0]**: *ADC Trigger 2 Update Source*
Refer to AD1USRC[2:0] description

Bits 18:16 **AD1USRC[2:0]**: *ADC Trigger 1 Update Source*
These bits define the source which will trigger the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.
000: Master Timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110, 111: Reserved

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TEUDIS**: *Timer E Update Disable*
Refer to TAUDIS description

Bit 4 **TDUDIS**: *Timer D Update Disable*
Refer to TAUDIS description

Bit 3 **TCUDIS**: *Timer C Update Disable*
Refer to TAUDIS description

Bit 2 **TBUDIS**: *Timer B Update Disable*

Refer to TAUDIS description

Bit 1 **TAUDIS**: *Timer A Update Disable*

This bit is set and cleared by software to enable/disable an update event generation temporarily on Timer A.

0: update enabled. The update occurs upon generation of the selected source.

1: update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master *Update Disable*

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: update enabled.

1: update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

21.5.40 HRTIM Control Register 2 (HRTIM_CR2)

Address offset: 0x384h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	Res.	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

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

Bit 13 **TERST**: *Timer E counter software reset*
Refer to TARST description

Bit 12 **TDRST**: *Timer D counter software reset*
Refer to TARST description

Bit 11 **TCRST**: *Timer C counter software reset*
Refer to TARST description

Bit 10 **TBRST**: *Timer B counter software reset*
Refer to TARST description

Bit 9 **TARST**: *Timer A counter software reset*
Setting this bit resets the TimerA counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master *Counter software reset*
Setting this bit resets the Master timer counter.
The bit is automatically reset by hardware.

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

Bit 5 **TESWU**: *Timer E Software Update*
Refer to TASWU description

Bit 4 **TDSWU**: *Timer D Software Update*
Refer to TASWU description

Bit 3 **TCSWU**: *Timer C Software Update*
Refer to TASWU description

Bit 2 **TBSWU**: *Timer B Software Update*
Refer to TASWU description

Bit 1 **TASWU**: *Timer A Software update*
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is cancelled.

Bit 0 **MSWU**: *Master Timer Software update*
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is cancelled.

21.5.41 HRTIM Interrupt Status Register (HRTIM_ISR)

Address offset: 0x388h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
											r	r	r	r	r

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

Bit 17 **BMPER**: Burst mode Period Interrupt Flag

This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.

0: No Burst mode period interrupt occurred

1: Burst mode period interrupt occurred

Bit 16 **DLLRDY**: DLL Ready Interrupt Flag

This bit is set by hardware when the DLL calibration is completed. It is cleared by software writing it at 1.

0: No DLL calibration ready interrupt occurred

1: DLL calibration ready interrupt occurred

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLT**: System Fault Interrupt Flag

Refer to FLT1 description

Bit 4 **FLT5**: Fault 5 Interrupt Flag

Refer to FLT1 description

Bit 3 **FLT4**: Fault 4 Interrupt Flag

Refer to FLT1 description

Bit 2 **FLT3**: Fault 3 Interrupt Flag

Refer to FLT1 description

Bit 1 **FLT2**: Fault 2 Interrupt Flag

Refer to FLT1 description

Bit 0 **FLT1**: Fault 1 Interrupt Flag

This bit is set by hardware when Fault 1 event occurs. It is cleared by software writing it at 1.

0: No Fault 1 interrupt occurred

1: Fault 1 interrupt occurred

21.5.42 HRTIM Interrupt Clear Register (HRTIM_ICR)

Address offset: 0x38Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
											w	w	w	w	w

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

Bit 17 **BMPERC**: Burst mode period flag Clear
Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bit 16 **DLLRDYC**: DLL Ready Interrupt flag Clear
Writing 1 to this bit clears the DLLRDY flag in HRTIM_ISR register.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLTC**: System Fault Interrupt Flag Clear
Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 Interrupt Flag Clear
Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 Interrupt Flag Clear
Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 Interrupt Flag Clear
Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 Interrupt Flag Clear
Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 Interrupt Flag Clear
Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

21.5.43 HRTIM Interrupt Enable Register (HRTIM_IER)

Address offset: 0x390h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
											rw	rw	rw	rw	rw

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

Bit 17 **BMPERIE**: Burst mode period Interrupt Enable
This bit is set and cleared by software to enable/disable the Burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bit 16 **DLLRDYIE**: DLL Ready Interrupt Enable
This bit is set and cleared by software to enable/disable the DLL ready interrupt.
0: DLL ready interrupt disabled
1: DLL ready interrupt enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLTIE**: System Fault Interrupt Enable
Refer to FLT1IE description

Bit 4 **FLT5IE**: Fault 5 Interrupt Enable
Refer to FLT1IE description

Bit 3 **FLT4IE**: Fault 4 Interrupt Enable
Refer to FLT1IE description

Bit 2 **FLT3IE**: Fault 3 Interrupt Enable
Refer to FLT1IE description

Bit 1 **FLT2IE**: Fault 2 Interrupt Enable
Refer to FLT1IE description

Bit 0 **FLT1IE**: Fault 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

21.5.44 HRTIM Output Enable Register (HRTIM_OENR)

Address offset: 0x394h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
						rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2OEN**: Timer E Output 2 Enable
Refer to TA1OEN description

Bit 8 **TE1OEN**: Timer E Output 1 Enable
Refer to TA1OEN description

Bit 7 **TD2OEN**: Timer D Output 2 Enable
Refer to TA1OEN description

Bit 6 **TD1OEN**: Timer D Output 1 Enable
Refer to TA1OEN description

Bit 5 **TC2OEN**: Timer C Output 2 Enable
Refer to TA1OEN description

Bit 4 **TC1OEN**: Timer C Output 1 Enable
Refer to TA1OEN description

Bit 3 **TB2OEN**: Timer B Output 2 Enable
Refer to TA1OEN description

Bit 2 **TB1OEN**: Timer B Output 1 Enable
Refer to TA1OEN description

Bit 1 **TA2OEN**: Timer A Output 2 Enable
Refer to TA1OEN description

Bit 0 **TA1OEN**: Timer A Output 1 (HRTIM_CHA1) Enable
Setting this bit enables the Timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output HRTIM_CHA1 disabled. The output is either in Fault or Idle state.
1: output HRTIM_CHA1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

21.5.45 HRTIM Output Disable Register (HRTIM_ODISR)

Address offset: 0x398h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
						w	w	w	w	w	w	w	w	w	w

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2ODIS**: Timer E Output 2 disable
Refer to TA1ODIS description

Bit 8 **TE1ODIS**: Timer E Output 1 disable
Refer to TA1ODIS description

Bit 7 **TD2ODIS**: Timer D Output 2 disable
Refer to TA1ODIS description

Bit 6 **TD1ODIS**: Timer D Output 1 disable
Refer to TA1ODIS description

Bit 5 **TC2ODIS**: Timer C Output 2 disable
Refer to TA1ODIS description

Bit 4 **TC1ODIS**: Timer C Output 1 disable
Refer to TA1ODIS description

Bit 3 **TB2ODIS**: Timer B Output 2 disable
Refer to TA1ODIS description

Bit 2 **TB1ODIS**: Timer B Output 1 disable
Refer to TA1ODIS description

Bit 1 **TA2ODIS**: Timer A Output 2 disable
Refer to TA1ODIS description

Bit 0 **TA1ODIS**: Timer A Output 1 (HRTIM_CHA1) disable
Setting this bit disables the Timer A output 1. The output enters the idle state, either from the run state or from the fault state.
Writing “0” has no effect.

21.5.46 HRTIM Output Disable Status Register (HRTIM_ODSR)

Address offset: 0x39Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2ODS**: Timer E Output 2 disable status
Refer to TA1ODS description

Bit 8 **TE1ODS**: Timer E Output 1 disable status
Refer to TA1ODS description

Bit 7 **TD2ODS**: Timer D Output 2 disable status
Refer to TA1ODS description

Bit 6 **TD1ODS**: Timer D Output 1 disable status
Refer to TA1ODS description

Bit 5 **TC2ODS**: Timer C Output 2 disable status
Refer to TA1ODS description

Bit 4 **TC1ODS**: Timer C Output 1 disable status
Refer to TA1ODS description

Bit 3 **TB2ODS**: Timer B Output 2 disable status
Refer to TA1ODS description

Bit 2 **TB1ODS**: Timer B Output 1 disable status
Refer to TA1ODS description

Bit 1 **TA2ODS**: Timer A Output 2 disable status
Refer to TA1ODS description

Bit 0 **TA1ODS**: Timer A Output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: output HRTIM_CHA1 disabled, in Idle state.
1: output HRTIM_CHA1 disabled, in Fault state.

21.5.47 HRTIM Burst Mode Control Register (HRTIM_BMCR)

Address offset: 0x3A0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: *Burst Mode Status*

This bit gives the current operating state.

0: Normal operation

1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **TEBM**: *Timer E Burst Mode*

Refer to TABM description

Bit 20 **TDBM**: *Timer D Burst Mode*

Refer to TABM description

Bit 19 **TCBM**: *Timer C Burst Mode*

Refer to TABM description

Bit 18 **TBBM**: *Timer B Burst Mode*

Refer to TABM description

Bit 17 **TABM**: *Timer A Burst Mode*

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Timer A counter clock is maintained and the timer operates normally

1: Timer A counter clock is stopped and the counter is reset

Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11)

Bit 16 **MTBM**: *Master Timer Burst Mode*

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Master Timer counter clock is maintained and the timer operates normally

1: Master Timer counter clock is stopped and the counter is reset

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

Bit 10 **BMPREN**: *Burst Mode Preload Enable*

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.

0: Preload disabled: the write access is directly done into active registers

1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: *Burst Mode Prescaler*

Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bitfield cannot be changed while the burst mode is enabled.
0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101:Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: *Burst Mode Clock source*

This bitfield defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 98* for on-chip events 1..4 connections details).
0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip Event 1 (BMClk[1]), acting as a burst mode counter clock
0111: On-chip Event 2 (BMClk[2]) acting as a burst mode counter clock
1000: On-chip Event 3 (BMClk[3]) acting as a burst mode counter clock
1001: On-chip Event 4 (BMClk[4]) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
Other codes reserved

Bit 1 **BMOM**: *Burst Mode operating mode*

This bit defines if the burst mode is entered once or if it is continuously operating.
0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: *Burst Mode enable*

This bit starts the burst mode controller which becomes ready to receive the start trigger. Writing this bit to 0 causes a burst mode early termination.
0: Burst mode disabled
1: Burst mode enabled

21.5.48 HRTIM Burst Mode Trigger Register (HRTIM_BMTRGR)

Address offset: 0x3A4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TERST	TDCMP2	TDCMP1	TDREP	TDRST	TCCMP2	TCCMP1	TCREP
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
TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: *On-chip Event*

A rising edge on an on-chip Event (see *Section : Burst mode triggers*) triggers a burst mode entry.

Bit 30 **EEV8**: *External Event 8 (TIMD filters applied)*

The external event 8 conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEV7**: *External Event 7 (TIMA filters applied)*

The external event 7 conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEV8**: *Timer D period following External Event 8*

The timer D period following an external event 8 (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEV7**: *Timer A period following External Event 7*

The timer A period following an external event 7 (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: *Timer E Compare 2 event*

Refer to TACMP1 description

Bit 25 **TECMP1**: *Timer E Compare 1 event*

Refer to TACMP1 description

Bit 24 **TEREP**: *Timer E repetition*

Refer to TAREP description

Bit 23 **TERST**: *Timer E counter reset or roll-over*

Refer to TARST description

Bit 22 **TDCMP2**: *Timer D Compare 2 event*

Refer to TACMP1 description

Bit 21 **TDCMP1**: *Timer D Compare 1 event*

Refer to TACMP1 description

Bit 20 **TDREP**: *Timer D repetition*

Refer to TAREP description

Bit 19 **TDRST**: *Timer D reset or roll-over*

Refer to TARST description

Bit 18 **TCCMP2**: *Timer C Compare 2 event*

Refer to TACMP1 description

Bit 17 **TCCMP1**: *Timer C Compare 1 event*
Refer to TACMP1 description

Bit 16 **TCREP**: *Timer C repetition*
Refer to TAREP description

Bit 15 **TCRST**: *Timer C reset or roll-over*
Refer to TARST description

Bit 14 **TBCMP2**: *Timer B Compare 2 event*
Refer to TACMP1 description

Bit 13 **TBCMP1**: *Timer B Compare 1 event*
Refer to TACMP1 description

Bit 12 **TBREP**: *Timer B repetition*
Refer to TAREP description

Bit 11 **TBRST**: *Timer B reset or roll-over*
Refer to TARST description

Bit 10 **TACMP2**: *Timer A Compare 2 event*
Refer to TACMP1 description

Bit 9 **TACMP1**: *Timer A Compare 1 event*
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: *Timer A repetition*
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: *Timer A reset or roll-over*
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: *Master Compare 4*
Refer to MSTCMP1 description

Bit 5 **MSTCMP3**: *Master Compare 3*
Refer to MSTCMP1 description

Bit 4 **MSTCMP2**: *Master Compare 2*
Refer to MSTCMP1 description

Bit 3 **MSTCMP1**: *Master Compare 1*
The master timer Compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: *Master repetition*
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: *Master reset or roll-over*
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: *Software start*
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

21.5.49 HRTIM Burst Mode Compare Register (HRTIM_BMCMPR)

Address offset: 0x3A8h

Reset value: 0x0000 0000

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

Defines the number of periods during which the selected timers are in idle state.

This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).

21.5.50 HRTIM Burst Mode Period Register (HRTIM_BMPER)

Address offset: 0x3ACh

Reset value: 0x0000 0000

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

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

21.5.51 HRTIM Timer External Event Control Register 1 (HRTIM_EECR1)

Address offset: 0x3B0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[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
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
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.

Bit 29 **EE5FAST**: *External Event 5 Fast mode*
Refer to EE1FAST description

Bits 28:27 **EE5SNS[1:0]**: *External Event 5 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 26 **EE5POL**: *External Event 5 Polarity*
Refer to EE1POL description

Bits 25:24 **EE5SRC[1:0]**: *External Event 5 Source*
Refer to EE1SRC[1:0] description

Bit 23 **EE4FAST**: *External Event 4 Fast mode*
Refer to EE1FAST description

Bits 22:21 **EE4SNS[1:0]**: *External Event 4 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 20 **EE4POL**: *External Event 4 Polarity*
Refer to EE1POL description

Bits 19:18 **EE4SRC[1:0]**: *External Event 4 Source*
Refer to EE1SRC[1:0] description

Bit 17 **EE3FAST**: *External Event 3 Fast mode*
Refer to EE1FAST description

Bits 16:15 **EE3SNS[1:0]**: *External Event 3 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 14 **EE3POL**: *External Event 3 Polarity*
Refer to EE1POL description

Bits 13:12 **EE3SRC[1:0]**: *External Event 3 Source*
Refer to EE1SRC[1:0] description

Bit 11 **EE2FAST**: *External Event 2 Fast mode*
Refer to EE1FAST description

Bits 10:9 **EE2SNS[1:0]**: *External Event 2 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 8 **EE2POL**: *External Event 2 Polarity*
Refer to EE1POL description

Bits 7:6 **EE2SRC[1:0]**: *External Event 2 Source*
Refer to EE1SRC[1:0] description

Bit 5 **EE1FAST**: *External Event 1 Fast mode*
0: External Event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External Event 1 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set)

Bits 4:3 **EE1SNS[1:0]**: *External Event 1 Sensitivity*
00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: *External Event 1 Polarity*
This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: *External Event 1 Source*
00: EE1Src1
01: EE1Src2
10: EE1Src3
11: EE1Src4
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

21.5.52 HRTIM Timer External Event Control Register 2 (HRTIM_EECR2)

Address offset: 0x3B4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[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
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: *External Event 10 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 26 **EE10POL**: *External Event 10 Polarity*
Refer to EE1POL description

Bits 25:24 **EE10SRC[1:0]**: *External Event 10 Source*
Refer to EE1SRC[1:0] description

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: *External Event 9 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 20 **EE9POL**: *External Event 9 Polarity*
Refer to EE1POL description

Bits 19:18 **EE9SRC[1:0]**: *External Event 9 Source*
Refer to EE1SRC[1:0] description

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: *External Event 8 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 14 **EE8POL**: *External Event 8 Polarity*
Refer to EE1POL description

Bits 13:12 **EE8SRC[1:0]**: *External Event 8 Source*
Refer to EE1SRC[1:0] description

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: *External Event 7 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 8 **EE7POL**: *External Event 7 Polarity*
Refer to EE1POL description

Bits 7:6 **EE7SRC[1:0]**: *External Event 7 Source*
Refer to EE1SRC[1:0] description

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: *External Event 6 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 2 **EE6POL**: *External Event 6 Polarity*
Refer to EE1POL description

Bits 1:0 **EE6SRC[1:0]**: *External Event 6 Source*
Refer to EE1SRC[1:0] description

21.5.53 HRTIM Timer External Event Control Register 3 (HRTIM_EECR3)

Address offset: 0x3B8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
rw	rw			rw	rw	rw	rw			rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External Event Sampling clock division
This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the External Event signal sampling clock (f_{EEVS}) used by the digital filters.
00: $f_{EEVS}=f_{HRTIM}$
01: $f_{EEVS}=f_{HRTIM}$ / 2
10: $f_{EEVS}=f_{HRTIM}$ / 4
11: $f_{EEVS}=f_{HRTIM}$ / 8

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

Bits 27:24 **EE10F[3:0]**: External Event 10 filter
Refer to EE6F[3:0] description

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External Event 9 filter
Refer to EE6F[3:0] description

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External Event 8 filter
Refer to EE6F[3:0] description

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External Event 7 filter
Refer to EE6F[3:0] description

Bits 4:5 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External Event 6 filter
This bitfield defines the frequency used to sample External Event 6 input and the length of the digital filter applied to EEV6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING}= f_{HRTIM}$, N=2
0010: $f_{SAMPLING}= f_{HRTIM}$, N=4
0011: $f_{SAMPLING}= f_{HRTIM}$, N=8
0100: $f_{SAMPLING}= f_{EEVS}/2$, N=6
0101: $f_{SAMPLING}= f_{EEVS}/2$, N=8
0110: $f_{SAMPLING}= f_{EEVS}/4$, N=6
0111: $f_{SAMPLING}= f_{EEVS}/4$, N=8
1000: $f_{SAMPLING}= f_{EEVS}/8$, N=6
1001: $f_{SAMPLING}= f_{EEVS}/8$, N=8
1010: $f_{SAMPLING}= f_{EEVS}/16$, N=5
1011: $f_{SAMPLING}= f_{EEVS}/16$, N=6
1100: $f_{SAMPLING}= f_{EEVS}/16$, N=8
1101: $f_{SAMPLING}= f_{EEVS}/32$, N=5
1110: $f_{SAMPLING}= f_{EEVS}/32$, N=6
1111: $f_{SAMPLING}= f_{EEVS}/32$, N=8

21.5.54 HRTIM ADC Trigger 1 Register (HRTIM_ADC1R)

Address offset: 0x3BCh

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD1TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3
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
AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 These bits select the trigger source for th ADC Trigger 1 output . Refer to HRTIM_ADC3R bits description for details

21.5.55 HRTIM ADC Trigger 2 Register (HRTIM_ADC2R)

Address offset: 0x3C0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4
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
AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 These bits select the trigger source for th ADC Trigger 2 output . Refer to HRTIM_ADC4R bits description for details

21.5.56 HRTIM ADC Trigger 3 Register (HRTIM_ADC3R)

Address offset: 0x3C4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TEC2	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TDC2	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TCC2	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3
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
ADC3TBC2	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TAC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: *ADC trigger 3 on Timer E Period*
Refer to ADC3TAPER description

Bit 30 **ADC3TEC4**: *ADC trigger 3 on Timer E Compare 4*
Refer to ADC3TAC2 description

Bit 29 **ADC3TEC3**: *ADC trigger 3 on Timer E Compare 3*
Refer to ADC3TAC2 description

Bit 28 **ADC3TEC2**: *ADC trigger 3 on Timer E Compare 2*
Refer to ADC3TAC2 description

Bit 27 **ADC3TDPER**: *ADC trigger 3 on Timer D Period*
Refer to ADC3TAPER description

Bit 26 **ADC3TDC4**: *ADC trigger 3 on Timer D Compare 4*
Refer to ADC3TAC2 description

Bit 25 **ADC3TDC3**: *ADC trigger 3 on Timer D Compare 3*
Refer to ADC3TAC2 description

Bit 24 **ADC3TDC2**: *ADC trigger 3 on Timer D Compare 2*
Refer to ADC3TAC2 description

Bit 23 **ADC3TCPER**: *ADC trigger 3 on Timer C Period*
Refer to ADC3TAPER description

Bit 22 **ADC3TCC4**: *ADC trigger 3 on Timer C Compare 4*
Refer to ADC3TAC2 description

Bit 21 **ADC3TCC3**: *ADC trigger 3 on Timer C Compare 3*
Refer to ADC3TAC2 description

Bit 20 **ADC3TCC2**: *ADC trigger 3 on Timer C Compare 2*
Refer to ADC3TAC2 description

Bit 19 **ADC3TBRST**: *ADC trigger 3 on Timer B Reset and counter roll-over*
Refer to ADC3TBRST description

Bit 18 **ADC3TBPER**: *ADC trigger 3 on Timer B Period*
Refer to ADC3TAPER description

Bit 17 **ADC3TBC4**: *ADC trigger 3 on Timer B Compare 4*
Refer to ADC3TAC2 description

Bit 16 **ADC3TBC3**: *ADC trigger 3 on Timer B Compare 3*
Refer to ADC3TAC2 description

Bit 15 **ADC3TBC2**: *ADC trigger 3 on Timer B Compare 2*
Refer to ADC3TAC2 description

Bit 14 **ADC3TARST**: *ADC trigger 3 on Timer A Reset and counter roll-over*
This bit enables the generation of an ADC Trigger upon Timer A reset and roll-over event, on ADC Trigger 1 output.

Bit 13 **ADC3TAPER**: *ADC trigger 3 on Timer A Period*
This bit enables the generation of an ADC Trigger upon Timer A period event, on ADC Trigger 3 output.

Bit 12 **ADC3TAC4**: *ADC trigger 3 on Timer A Compare 4*
Refer to ADC3TAC2 description

Bit 11 **ADC3TAC3**: *ADC trigger 3 on Timer A Compare 3*
Refer to ADC3TAC2 description

Bit 10 **ADC3TAC2**: *ADC trigger 3 on Timer A Compare 2*
This bit enables the generation of an ADC Trigger upon Timer A Compare 2 event, on ADC Trigger 3 output.

Bit 9 **ADC3EEV5**: *ADC trigger 3 on External Event 5*
Refer to ADC3EEV1 description

Bit 8 **ADC3EEV4**: *ADC trigger 3 on External Event 4*
Refer to ADC3EEV1 description

Bit 7 **ADC3EEV3**: *ADC trigger 3 on External Event 3*
Refer to ADC3EEV1 description

Bit 6 **ADC3EEV2**: *ADC trigger 3 on External Event 2*
Refer to ADC3EEV1 description

Bit 5 **ADC3EEV1**: *ADC trigger 3 on External Event 1*
This bit enables the generation of an ADC Trigger upon External event 1, on ADC Trigger 3 output.

Bit 4 **ADC3MPER**: *ADC trigger 3 on Master Period*
This bit enables the generation of an ADC Trigger upon Master timer period event, on ADC Trigger 3 output.

Bit 3 **ADC3MC4**: *ADC trigger 3 on Master Compare 4*
Refer to ADC3MC1 description

Bit 2 **ADC3MC3**: *ADC trigger 3 on Master Compare 3*
Refer to ADC3MC1 description

Bit 1 **ADC3MC2**: *ADC trigger 3 on Master Compare 2*
Refer to ADC3MC1 description

Bit 0 **ADC3MC1**: *ADC trigger 3 on Master Compare 1*
This bit enables the generation of an ADC Trigger upon Master Compare 1 event, on ADC Trigger 3 output.

21.5.57 HRTIM ADC Trigger 4 Register (HRTIM_ADC4R)

Address offset: 0x3C8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TDC3	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TCC3	ADC4TCC2	ADC4TBPER	ADC4TBC4
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
ADC4TBC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TAC3	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: *ADC trigger 4 on Timer E Reset and counter roll-over* (1)
Refer to ADC4TCRST description

Bit 30 **ADC4TEC4**: *ADC trigger 4 on Timer E Compare 4*
Refer to ADC4TAC2 description

Bit 29 **ADC4TEC3**: *ADC trigger 4 on Timer E Compare 3*
Refer to ADC4TAC2 description

Bit 28 **ADC4TEC2**: *ADC trigger 4 on Timer E Compare 2*
Refer to ADC4TAC2 description

Bit 27 **ADC4TDRST**: *ADC trigger 4 on Timer D Reset and counter roll-over* (1)
Refer to ADC4TCRST description

Bit 26 **ADC4TDPER**: *ADC trigger 4 on Timer D Period*
Refer to ADC4TAPER description

Bit 25 **ADC4TDC4**: *ADC trigger 4 on Timer D Compare 4*
Refer to ADC4TAC2 description

Bit 24 **ADC4TDC3**: *ADC trigger 4 on Timer D Compare 3*
Refer to ADC4TAC2 description

Bit 23 **ADC4TDC2**: *ADC trigger 2 on Timer D Compare 2*
Refer to ADC4TAC2 description

Bit 22 **ADC4TCRST**: *ADC trigger 4 on Timer C Reset and counter roll-over* (1)
This bit enables the generation of an ADC Trigger upon Timer C reset and roll-over event, on ADC Trigger 4 output.

Bit 21 **ADC4TCPER**: *ADC trigger 4 on Timer C Period*
Refer to ADC4TAPER description

Bit 20 **ADC4TCC4**: *ADC trigger 4 on Timer C Compare 4*
Refer to ADC4TAC2 description

Bit 19 **ADC4TCC3**: *ADC trigger 4 on Timer C Compare 3*
Refer to ADC4TAC2 description

Bit 18 **ADC4TCC2**: *ADC trigger 4 on Timer C Compare 2*
Refer to ADC4TAC2 description

Bit 17 **ADC4TBPER**: *ADC trigger 4 on Timer B Period*
Refer to ADC4TAPER description

Bit 16 **ADC4TBC4**: *ADC trigger 4 on Timer B Compare 4*
Refer to ADC4TAC2 description

Bit 15 **ADC4TBC3**: *ADC trigger 4 on Timer B Compare 3*
Refer to ADC4TAC2 description

Bit 14 **ADC4TBC2**: *ADC trigger 4 on Timer B Compare 2*
Refer to ADC4TAC2 description

Bit 13 **ADC4TAPER**: *ADC trigger 4 on Timer A Period*
This bit enables the generation of an ADC Trigger upon Timer A event, on ADC Trigger 4 output.

Bit 12 **ADC4TAC4**: *ADC trigger 4 on Timer A Compare 4*
Refer to ADC4TAC2 description

Bit 11 **ADC4TAC3**: *ADC trigger 4 on Timer A Compare 3*
Refer to ADC4TAC2 description

Bit 10 **ADC4TAC2**: *ADC trigger 4 on Timer A Compare 2*
This bit enables the generation of an ADC Trigger upon Timer A Compare 2, on ADC Trigger 4 output.

Bit 9 **ADC4EEV10**: *ADC trigger 4 on External Event 10* (1)
Refer to ADC4EEV6 description

Bit 8 **ADC4EEV9**: *ADC trigger 4 on External Event 9* (1)
Refer to ADC4EEV6 description

Bit 7 **ADC4EEV8**: *ADC trigger 4 on External Event 8* (1)
Refer to ADC4EEV6 description

Bit 6 **ADC4EEV7**: *ADC trigger 4 on External Event 7* (1)
Refer to ADC4EEV6 description

Bit 5 **ADC4EEV6**: *ADC trigger 4 on External Event 6* (1)
This bit enables the generation of an ADC Trigger upon external event 6, on ADC Trigger 4 output.

Bit 4 **ADC4MPER**: *ADC trigger 4 on Master Period*
This bit enables the generation of an ADC Trigger upon Master period event, on ADC Trigger 4 output.

Bit 3 **ADC4MC4**: *ADC trigger 4 on Master Compare 4*
Refer to ADC4MC1 description

Bit 2 **ADC4MC3**: *ADC trigger 4 on Master Compare 3*
Refer to ADC4MC1 description

Bit 1 **ADC4MC2**: *ADC trigger 4 on Master Compare 2*
Refer to ADC4MC1 description

Bit 0 **ADC4MC1**: *ADC trigger 4 on Master Compare 1*
This bit enables the generation of an ADC Trigger upon Master Compare 1 event, on ADC Trigger 4 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

21.5.58 HRTIM DLL Control Register (HRTIM_DLLCR)

Address offset: 0x3CCh

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value

Bits 3:2 **CALRTE[1:0]**: DLL Calibration rate

This defines the DLL calibration periodicity.

00: 1048576 * t_{HRTIM} (7.3 ms)

01: 131072 * t_{HRTIM} (910 µs)

10: 16384 * t_{HRTIM} (114 µs)

11: 2048 * t_{HRTIM} (14 µs)

Bit 1 **CALEN**: DLL Calibration Enable

This bit enables the periodic DLL calibration, as per CALRTE[1:0] bit setting. When CALEN bit is reset, the calibration can be started in single-shot mode with CAL bit.

0: Periodic calibration disabled

1: Calibration is performed periodically, as per CALRTE[1:0] setting

Note: CALEN must not be set simultaneously with CAL bit

Bit 0 **CAL**: DLL Calibration Start

This bit starts the DLL calibration process. It is write-only.

0: No calibration request

1: Calibration start

Note: CAL must not be set simultaneously with CALEN bit

21.5.59 HRTIM Fault Input Register 1 (HRTIM_FLTINR1)

Address offset: 0x3D0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4SRC	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC	FLT3P	FLT3E
rwo	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
FLT2LCK	FLT2F[3:0]				FLT2SRC	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 26 **FLT4SRC**: Fault 4 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 23 **FLT3LCK**: Fault 3 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 18 **FLT3SRC**: Fault 3 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 15 **FLT2LCK**: Fault 2 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 10 **FLT2SRC**: Fault 2 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 7 **FLT1LCK**: Fault 1 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 2 **FLT1SRC**: Fault 1 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 1 **FLT1P**: Fault 1 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 0 **FLT1E**: Fault 1 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

21.5.60 HRTIM Fault Input Register 2 (HRTIM_FLTINR2)

Address offset: 0x3D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5LCK	FLT5F[3:0]				FLT5SRC	FLT5P	FLT5E
								rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FLTSD[1:0]**: Fault Sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.

00: $f_{FLTS}=f_{HRTIM}$

01: $f_{FLTS}=f_{HRTIM} / 2$

10: $f_{FLTS}=f_{HRTIM} / 4$

11: $f_{FLTS}=f_{HRTIM} / 8$

Note: This bitfield must be written prior to any of the FLTxE enable bits.

Bits 23:8 Reserved, must be kept at reset value.

Bit 7 **FLT5LCK**: Fault 5 Lock

The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they can be protected against spurious write accesses.

This bit is write-once. Once it has been set, it cannot be modified till the next system reset.

0: FLT5E, FLT5P, FLT5SRC, FLT5F[3:0] bits are read/write.

1: FLT5E, FLT5P, FLT5SRC, FLT5F[3:0] bits can no longer be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter

This bitfield defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, FLT5 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bitfield can be written only when FLT5E enable bit is reset.
This bitfield cannot be modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC**: Fault 5 source

This bit selects the FAULT5 input source (refer to *Table 99* for connection details).

0: Fault 1 input is HRTIM_FLT5 input pin

1: Fault 1 input is FLT5_Int signal

Note: This bitfield can be written only when FLT5E enable bit is reset

Bit 1 **FLT5P**: Fault 5 polarity

This bit selects the FAULT5 input polarity.

0: Fault 5 input is active low

1: Fault 5 input is active high

Note: This bitfield can be written only when FLT5E enable bit is reset

Bit 0 **FLT5E**: Fault 5 enable

This bit enables the global FAULT5 input circuitry.

0: Fault 5 input disabled
1: Fault 5 input enabled

21.5.61 HRTIM Burst DMA Master timer update Register (HRTIM_BDMUPR)

Address offset: 0x3D8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MCMP4**: *MCMP4R register update enable*
Refer to MCR description

Bit 8 **MCMP3**: *MCMP3R register update enable*
Refer to MCR description

Bit 7 **MCMP2**: *MCMP2R register update enable*
Refer to MCR description

Bit 6 **MCMP1**: *MCMP1R register update enable*
Refer to MCR description

Bit 5 **MREP**: *MREP register update enable*
Refer to MCR description

Bit 4 **MPER**: *MPER register update enable*
Refer to MCR description

Bit 3 **MCNT**: *MCNTR register update enable*
Refer to MCR description

Bit 2 **MDIER**: *MDIER register update enable*
Refer to MCR description

Bit 1 **MICR**: *MICR register update enable*
Refer to MCR description

Bit 0 **MCR**: *MCR register update enable*
This bit defines if the master timer MCR register is part of the list of registers to be updated by the Burst DMA.
0: MCR register is not updated by Burst DMA accesses
1: MCR register is updated by Burst DMA accesses

21.5.62 HRTIM Burst DMA Timerx update Register (HRTIM_BDTxUPR)

Address offset: 0x3DCh-0x3ECh

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMxFLTR	TIMxOUTR	TIMxCHPR	TIMxRSTR	TIMxEEFR2
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxEEFR1	TIMxRST2R	TIMxSET2R	TIMxRST1R	TIMxSET1R	TIMxDTxR	TIMxCMP4	TIMxCMP3	TIMxCMP2	TIMxCMP1	TIMxREP	TIMxPER	TIMxCNT	TIMxDIER	TIMxICR	TIMxCR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable
Refer to TIMxCR description

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable
Refer to TIMxCR description

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable
Refer to TIMxCR description

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable
Refer to TIMxCR description

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable
Refer to TIMxCR description

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable
Refer to TIMxCR description

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable
Refer to TIMxCR description

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable
Refer to TIMxCR description

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable
Refer to TIMxCR description

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable
Refer to TIMxCR description

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable
Refer to TIMxCR description

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable
Refer to TIMxCR description

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable
Refer to TIMxCR description

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable
Refer to TIMxCR description

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable
Refer to TIMxCR description

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable
Refer to TIMxCR description

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable
Refer to TIMxCR description

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable
Refer to TIMxCR description

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable
Refer to TIMxCR description

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable
Refer to TIMxCR description

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the Burst DMA.
0: HRTIM_TIMxCR register is not updated by Burst DMA accesses
1: HRTIM_TIMxCR register is updated by Burst DMA accesses

21.5.63 HRTIM Burst DMA Data Register (HRTIM_BDMADR)

Address offset: 0x3F0h

Reset value: 0x0000 0000

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

Bits 31:0 **BDMADR[31:0]**: Burst DMA Data register
Write accesses to this register triggers:
– the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits
– the increment of the register pointer to the next location to be filled

21.5.64 HRTIM register map

The tables below summarize the HRTIM registers mapping. The address offsets in *Table 105* and *Table 106* are referred to in the base address offsets given in *Table 104*.

Table 104. RTIM global register map

Base address offset	Register
0x000 - 0x07F	Master timer
0x080 - 0x0FF	Timer A
0x100 - 0x17F	Timer B
0x180 - 0x1FF	Timer C
0x200 - 0x27F	Timer D
0x280 - 0x2FF	Timer E
0x300 - 0x37F	Reserved
0x380 - 0x3FF	Common registers

Table 105. HRTIM Register map and reset values: Master timer

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	HRTIM_MCR	BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	Res.	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN	SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		Res.	Res.	HALF	RETRIG	CONT	CKPSC[2:0]		
	Reset value	0	0	0		0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x0004	HRTIM_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
	Reset value																										0	0	0	0	0	0	0
0x0008	HRTIM_MICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
	Reset value																										0	0	0	0	0	0	0
0x000C	HRTIM_MDIER(T)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x0010	HRTIM_MCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 105. HRTIM Register map and reset values: Master timer (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	**HRTIM_MPER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPER[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x0018	**HRTIM_MREP**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x001C	**HRTIM_MCMP1R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0024	**HRTIM_MCMP2R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**HRTIM_MCMP3R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**HRTIM_MCMP4R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 90 on page 673*).

Table 106. HRTIM Register map and reset values: TIMx (x= A..E)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**HRTIM_TIMxCR**	UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	Res.	TxRSTU	TxREPU	Res.	DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	Res.	Res.	Res.	PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0		0	0	0	0	0	0				0	0	0	0	0	0	0
0x0004	**HRTIM_ TIMxISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT	Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	Reset value											0	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x0008	**HRTIM_ TIMxICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	Reset value																	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x000C	**HRTIM_ TIMxDIER**(1)	Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SET1xDE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE	Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	Reset value		0	0	0	0	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	**HRTIM_CNTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTx[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**HRTIM_ PERxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERx[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x0018	**HRTIM_ REPxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x001C	**HRTIM_ CMP1xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	**HRTIM_ CMP1CxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]								CMP1x[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
0x0024	**HRTIM_ CMP2xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**HRTIM_ CMP3xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP3x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**HRTIM_ CMP4xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP4x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	**HRTIM_CPT1xR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 106. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0034	HRTIM_CPT2xR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	HRTIM_DTxR(1)	DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]									DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
	Reset value	0	0					0	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	HRTIM_SETx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	HRTIM_RSTx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT	0
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_SETx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST	0
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_RSTx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT	0
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_EEFxR1	Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3:0]				EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
	Reset value				0	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	HRTIM_EEFxR2	Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3:0]				EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
	Reset value				0	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	HRTIM_RSTAR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 106. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0054	HRTIM_RSTBR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTCR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTDR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTER(1)	Res.	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0058	HRTIM_CHPxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRTPW [3:0]				CARDTY [2:0]			CARFRQ [3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_CPT1ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	Res.	Res.	Res.	Res.	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	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	HRTIM_CPT1BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	Res.	Res.	Res.	Res.	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	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	HRTIM_CPT1CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	Res.	Res.	Res.	Res.	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	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	HRTIM_CPT1DCR	TECMP2	TECMP1	TE1RST	TE1SET	Res.	Res.	Res.	Res.	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0					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 106. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x005C	HRTIM_ CPT1ECR	Res.	Res.	Res.	Res.	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	Res.	Res.	Res.	Res.	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	Res.	Res.	Res.	Res.	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	Res.	Res.	Res.	Res.	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2DCR	TECMP2	TECMP1	TE1RST	TE1SET	Res.	Res.	Res.	Res.	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2ECR	Res.	Res.	Res.	Res.	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value					0	0	0	0	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	HRTIM_OUTxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM2	POL2	Res.	Res.	Res.	Res.	DLYPRT[2:0]		DLYPRTEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM1	POL1	Res.
	Reset value									0	0	0	0	0	0	0						0		0	0	0	0		0	0	0	0	
0x0068	HRTIM_FLTxR	FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
	Reset value	0																											0	0	0	0	0

1. This register can be preloaded (see *Table 90 on page 673*).

Table 107. HRTIM Register map and reset values: Common functions

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**HRTIM_CR1**	Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x0004	**HRTIM_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	Res.	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x008	**HRTIM_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
	Reset value															0	0											0	0	0	0	0	0
0x000C	**HRTIM_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
	Reset value															0	0											0	0	0	0	0	0
0x0010	**HRTIM_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
	Reset value															0	0											0	0	0	0	0	0
0x0014	**HRTIM_OENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x0018	**HRTIM_DISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x001C	**HRTIM_ODSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x0020	**HRTIM_BMCR**	BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEBM	TDBM	TCBM	TBBM	TABM	MTBM	Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
	Reset value	0										0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0024	**HRTIM_BMTRG**	OCHPEV	Res.	Res.	Res.	Res.	TECMP2	TECMP1	TEREP	TERST	TDCMP2	TDCMP1	TDREP	TDRST	TCCMP2	TCCMP1	TCREP	TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW	
	Reset value	0					0	0	0	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	**HRTIM_BMCMPR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMCMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 107. HRTIM Register map and reset values: Common functions (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x002C	HRTIM_BMPER(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	HRTIM_EECR1	Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1:0]		EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[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
0x0034	HRTIM_EECR2	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1:0]		EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[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
0x0038	HRTIM_EECR3	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE6SRC[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
0x003C	HRTIM_ADC1R(1)	AD1TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3	AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	HRTIM_ADC2R(1)	AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4	AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_ADC3R(1)	ADC3TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3	AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_ADC4R(1)	AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4	AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
	Reset value	0	0	0	0	0	0	0	0	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	HRTIM_DLLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALRTE [1:0]		CALEN	CAL
	Reset value																													0	0	0	0
0x0050	HRTIM_FLTINxR1	FLT4LCK	FLT4F[3:0]				FLT4SRC	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]			FLT3SRC	FLT3P	FLT3E	FLT2LCK	FLT2F[3:0]				FLT2SRC	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC	FLT1P	FLT1E
	Reset value	0	0	0	0	0	0	0	0	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 107. HRTIM Register map and reset values: Common functions (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0054	HRTIM_FLTINxR2	Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5LCK	FLT5F[3:0]				FLT5SRC	FLT5P	FLT5E
	Reset value							0	0																	0	0	0	0	0	0	0	0
0x0058	HRTIM_ BDMUPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_BDTAUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMAFLTR	TIMAOUTR	TIMACHPR	TIMARSTR	TIMAEEFR2	TIMAEEFR1	TIMARST2R	TIMASET2R	TIMARST1R	TIMASET1R	TIMADTxR	TIMACMP4	TIMACMP3	TIMACMP2	TIMACMP1	TIMAREP	TIMAPER	TIMACNT	TIMADIER	TIMAICR	TIMACR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ BDTBUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMBFLTR	TIMBOUTR	TIMBCHPR	TIMBRSTR	TIMBEEFR2	TIMBEEFR1	TIMBRST2R	TIMBSET2R	TIMBRST1R	TIMBSET1R	TIMBDTxR	TIMBCMP4	TIMBCMP3	TIMBCMP2	TIMBCMP1	TIMBREP	TIMBPER	TIMBCNT	TIMBDIER	TIMBICR	TIMBCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	HRTIM_ BDTCUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMCFLTR	TIMCOUTR	TIMCCHPR	TIMCRSTR	TIMCEEFR2	TIMCEEFR1	TIMCRST2R	TIMCSET2R	TIMCRST1R	TIMCSET1R	TIMCDTxR	TIMCCMP4	TIMCCMP3	TIMCCMP2	TIMCCMP1	TIMCREP	TIMCPER	TIMCCNT	TIMCDIER	TIMCICR	TIMCCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068	HRTIM_ BDTDUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMDFLTR	TIMDOUTR	TIMDCHPR	TIMDRSTR	TIMDEEFR2	TIMDEEFR1	TIMDRST2R	TIMDSET2R	TIMDRST1R	TIMDSET1R	TIMDDTxR	TIMDCMP4	TIMDCMP3	TIMDCMP2	TIMDCMP1	TIMDREP	TIMDPER	TIMDCNT	TIMDDIER	TIMDICR	TIMDCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x006C	HRTIM_ BDTEUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEFLTR	TIMEOUTR	TIMECHPR	TIMERSTR	TIMEEEFR2	TIMEEEFR1	TIMERST2R	TIMESET2R	TIMERST1R	TIMESET1R	TIMEDTxR	TIMECMP4	TIMECMP3	TIMECMP2	TIMECMP1	TIMEREP	TIMEPER	TIMECNT	TIMEDIER	TIMEICR	TIMECR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0070	HRTIM_BDMADR	BDMADR[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

1. This register can be preloaded (see *Table 90 on page 673*).

Refer to *Section 2.2 on page 47* for the register boundary addresses.

22 Infrared interface (IRTIM)

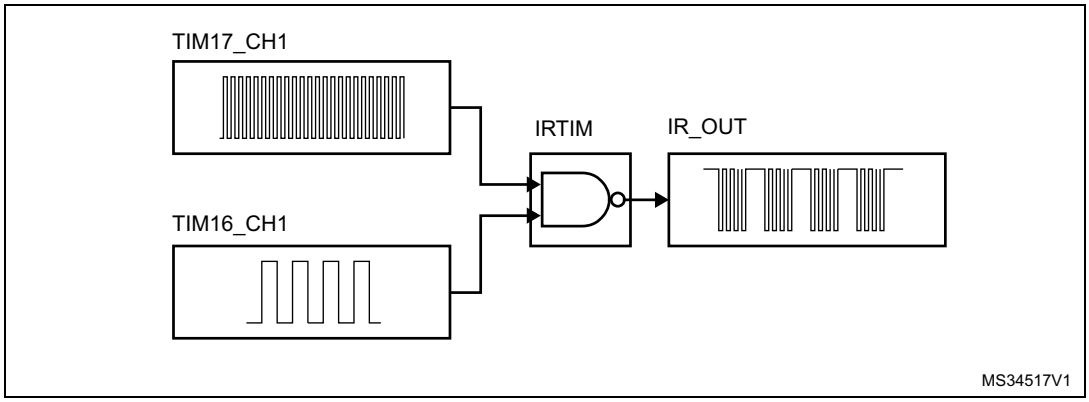
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in *Figure 309*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 309. IRTIM internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

23 Basic timers (TIM6/TIM7)

23.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

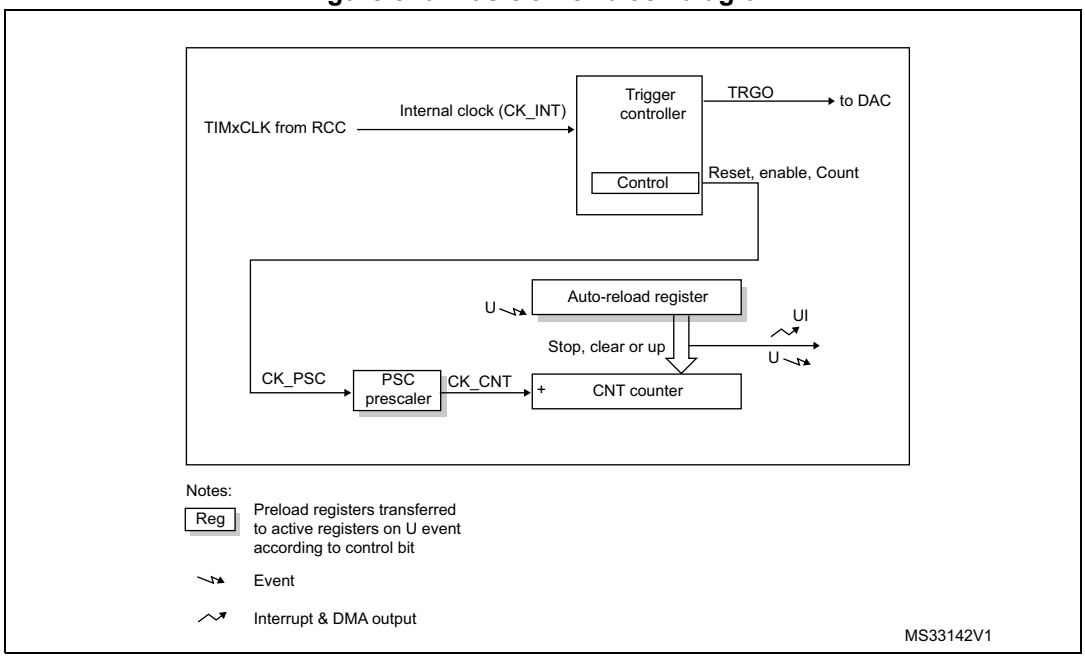
The timers are completely independent, and do not share any resources.

23.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 310. Basic timer block diagram



23.3 TIM6/TIM7 functional description

23.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 311 and *Figure 312* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 311. Counter timing diagram with prescaler division change from 1 to 2

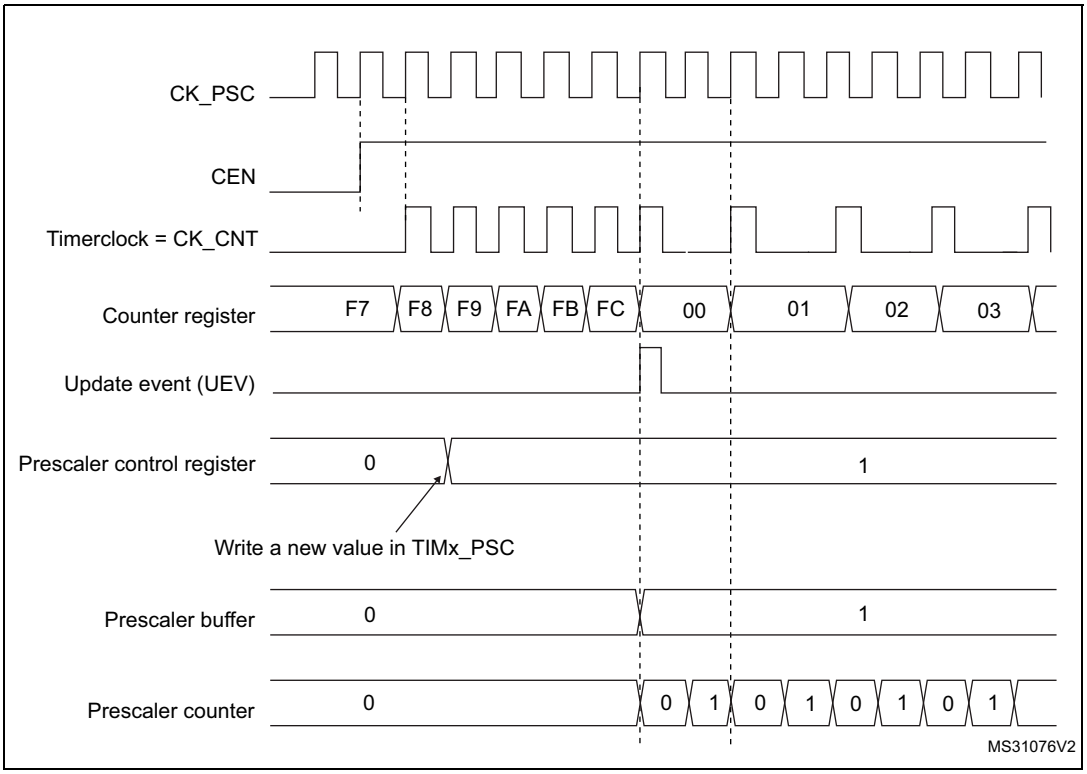
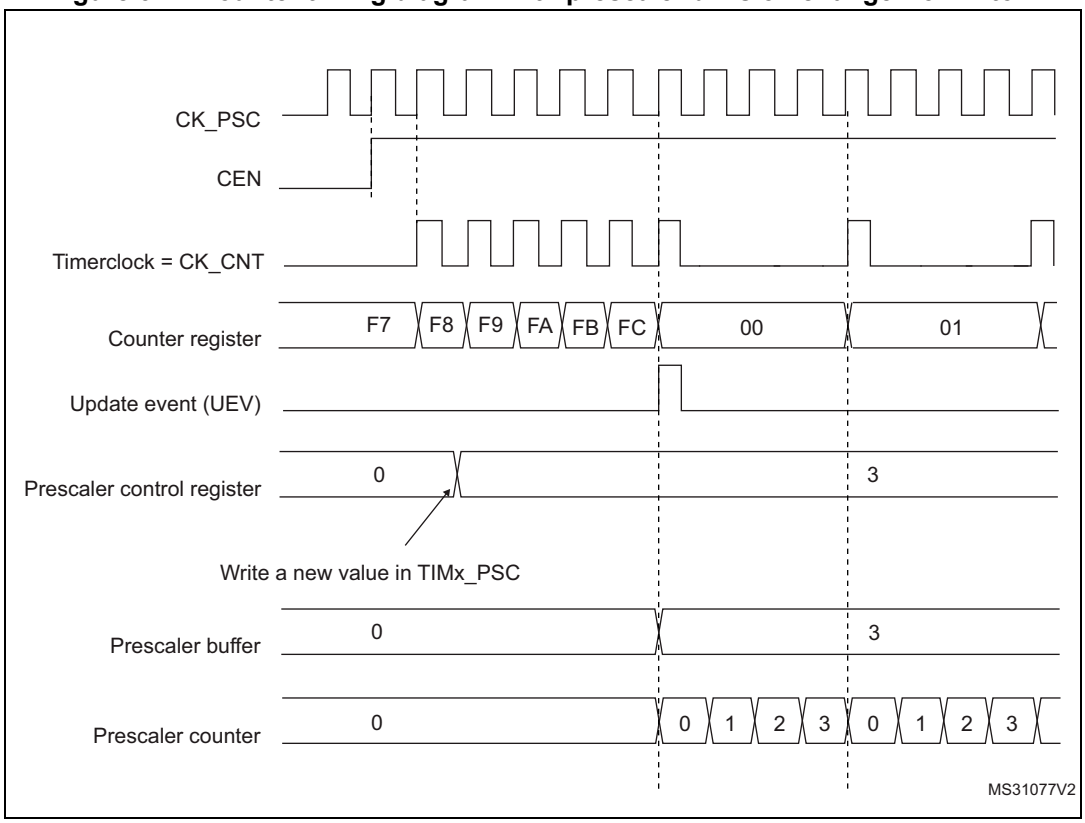


Figure 312. Counter timing diagram with prescaler division change from 1 to 4



23.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be 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 313. Counter timing diagram, internal clock divided by 1

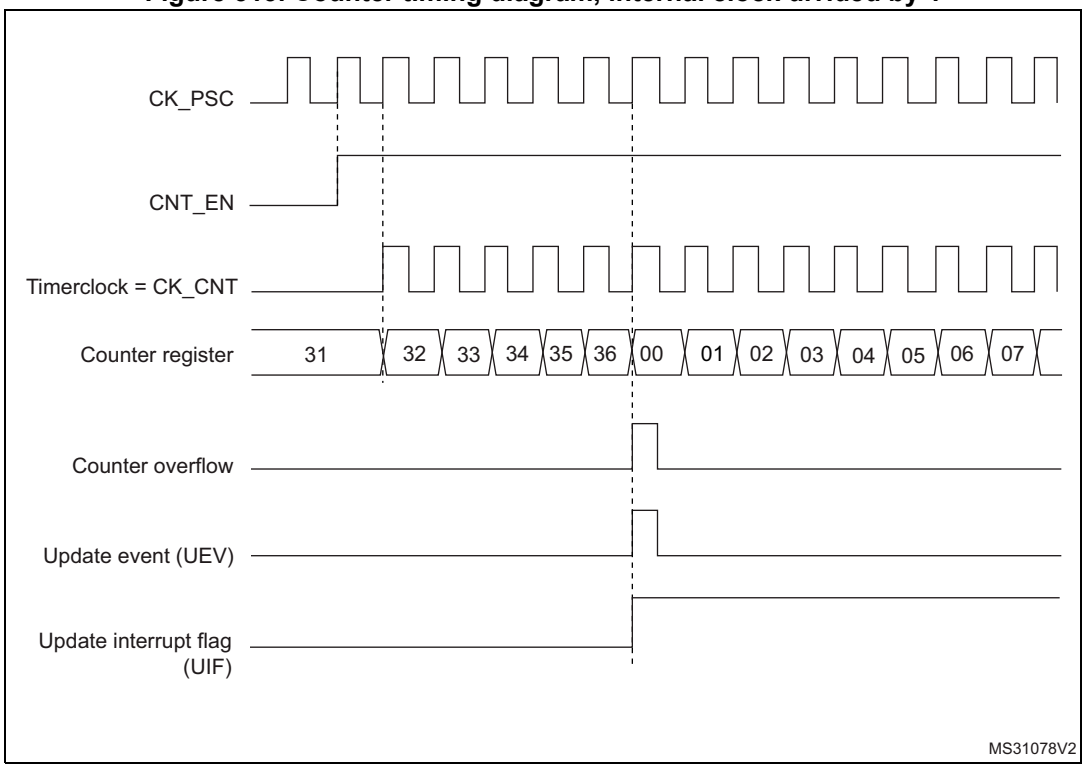


Figure 314. Counter timing diagram, internal clock divided by 2

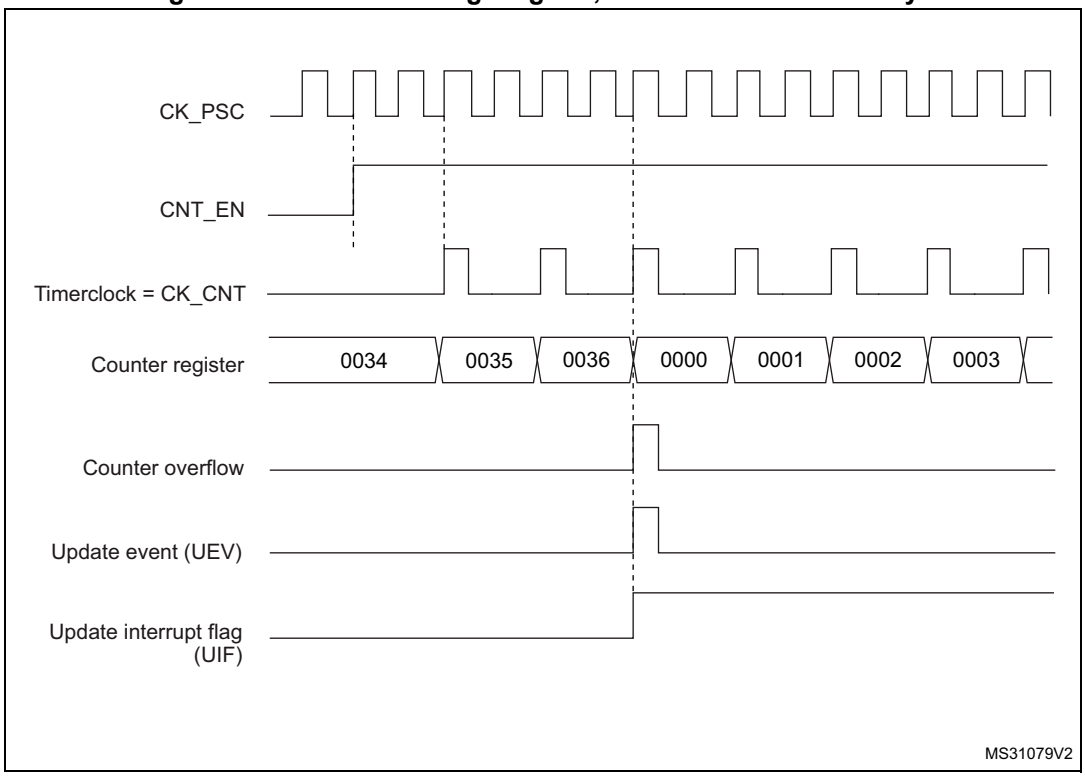


Figure 315. Counter timing diagram, internal clock divided by 4

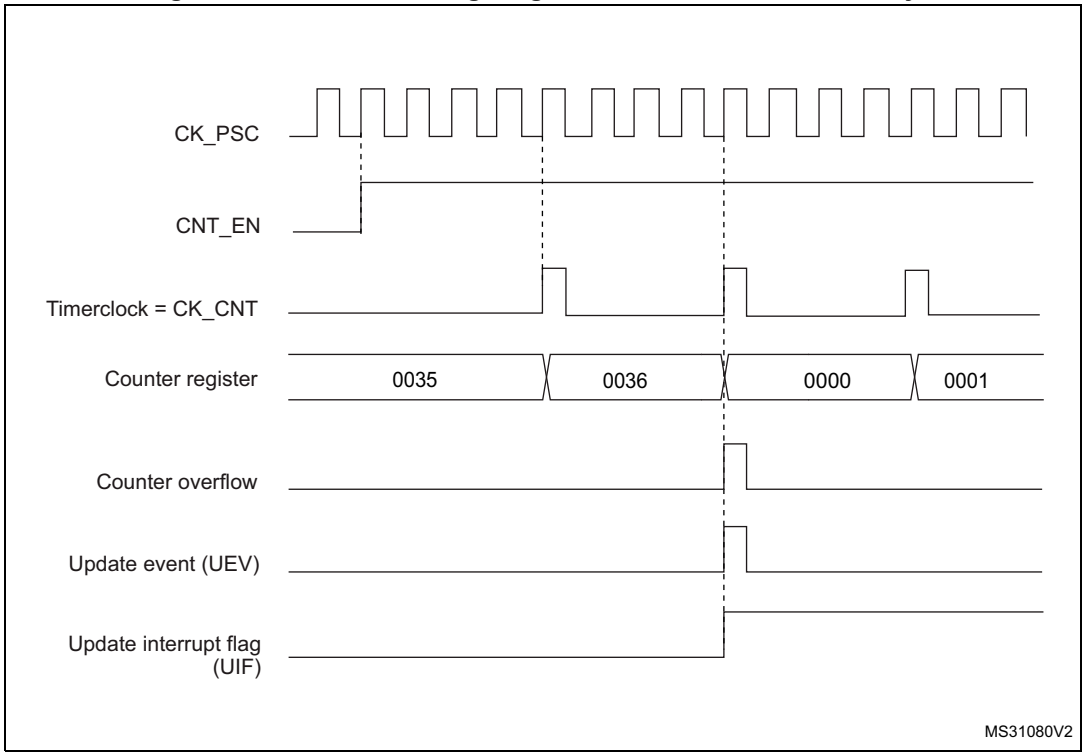


Figure 316. Counter timing diagram, internal clock divided by N

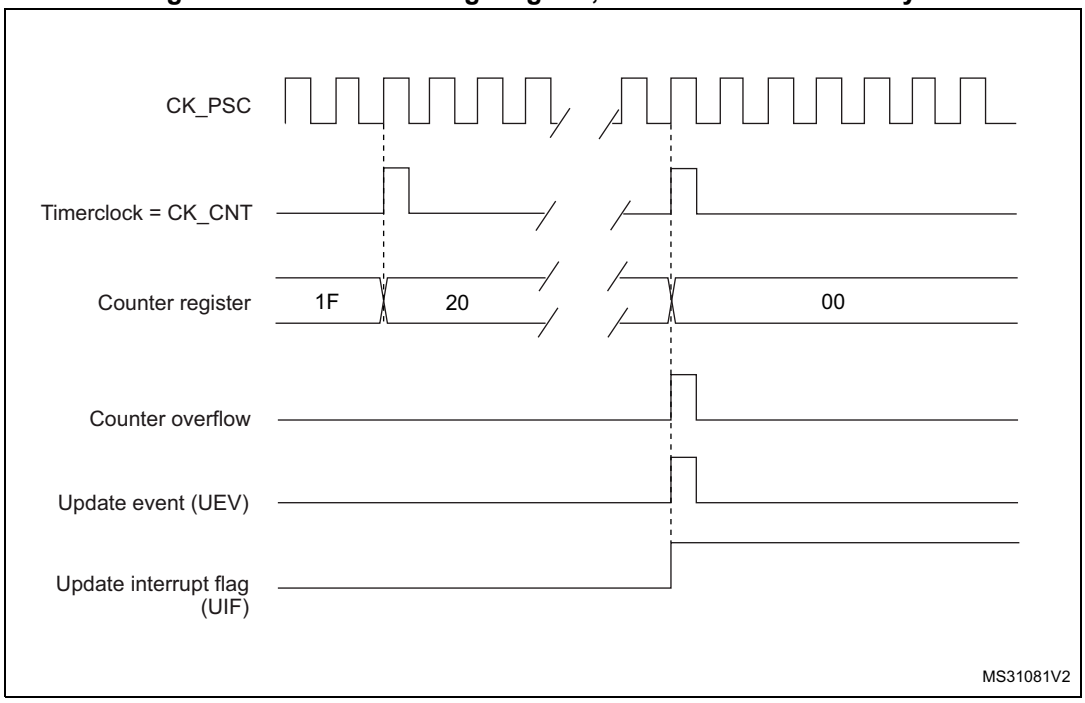


Figure 317. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

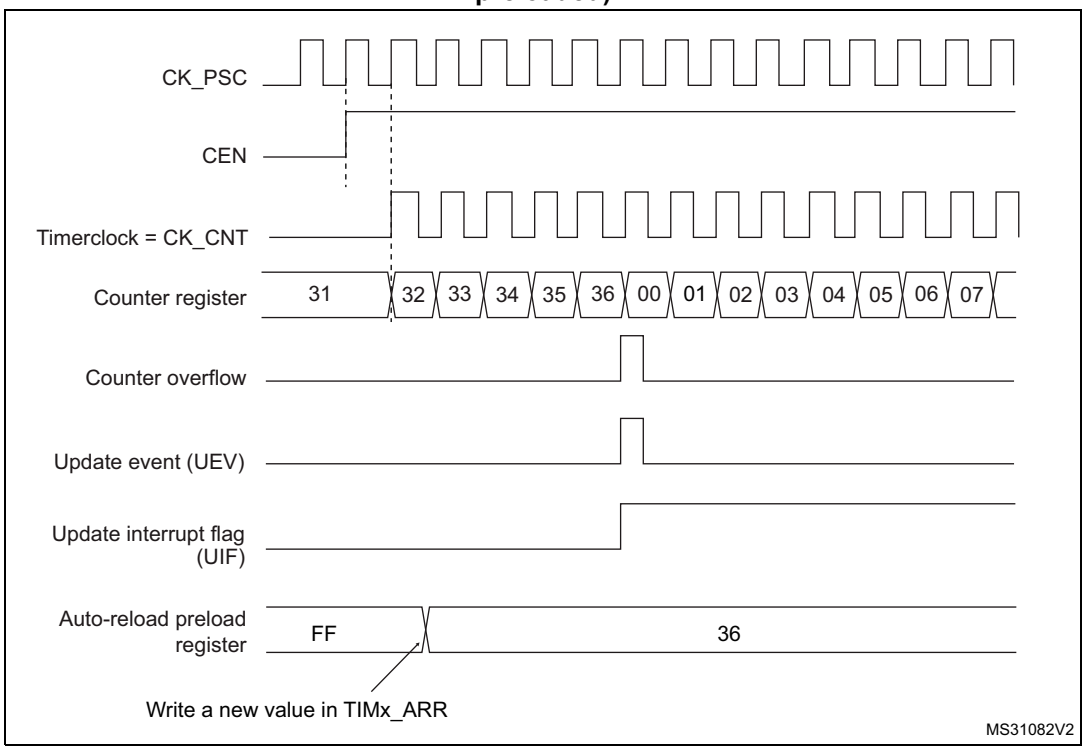
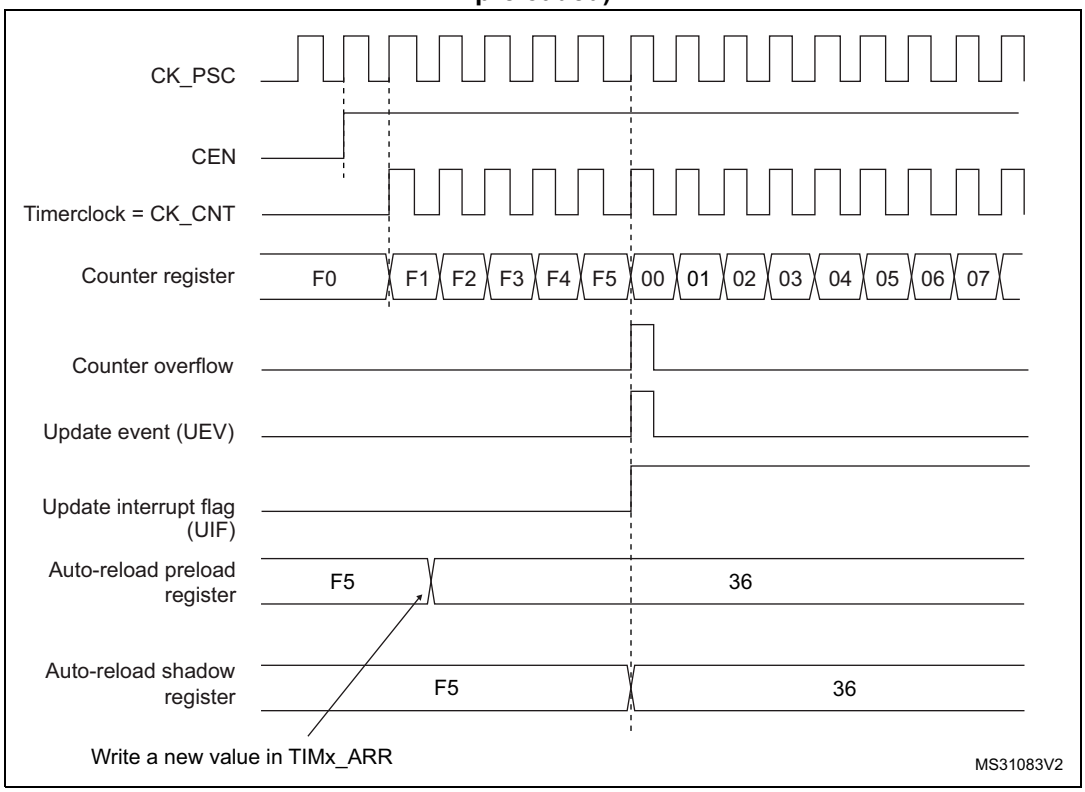


Figure 318. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



23.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

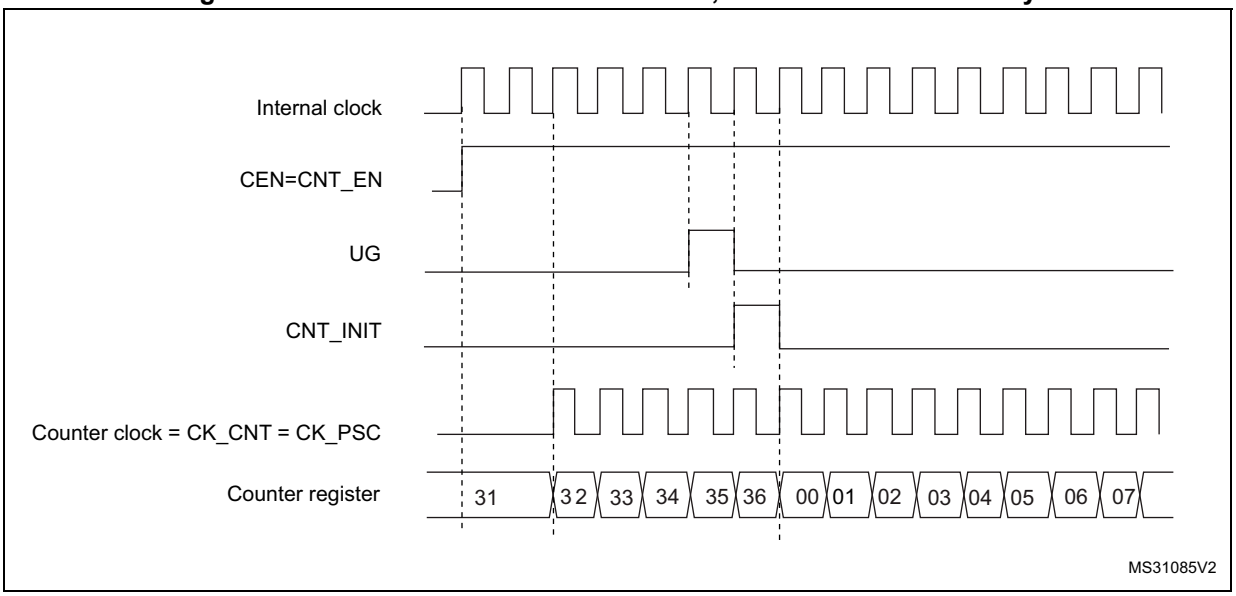
23.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 319 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 319. Control circuit in normal mode, internal clock divided by 1



23.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M4 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module. For more details, refer to *Section 31.15.2: Debug support for timers, watchdog, bxCAN and I^2C*.

23.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

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

23.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

23.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

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

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[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 or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

23.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

23.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

23.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

23.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

23.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

23.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

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]**: Prescaler value
ARR is the value to be loaded into the actual auto-reload register.
Refer to *Section 23.3.1: Time-base unit on page 807* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

23.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 108. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0				0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2 on page 47* for the register boundary addresses.

24 Independent watchdog (IWDG)

24.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to *Section 25 on page 828*.

24.2 IWDG main features

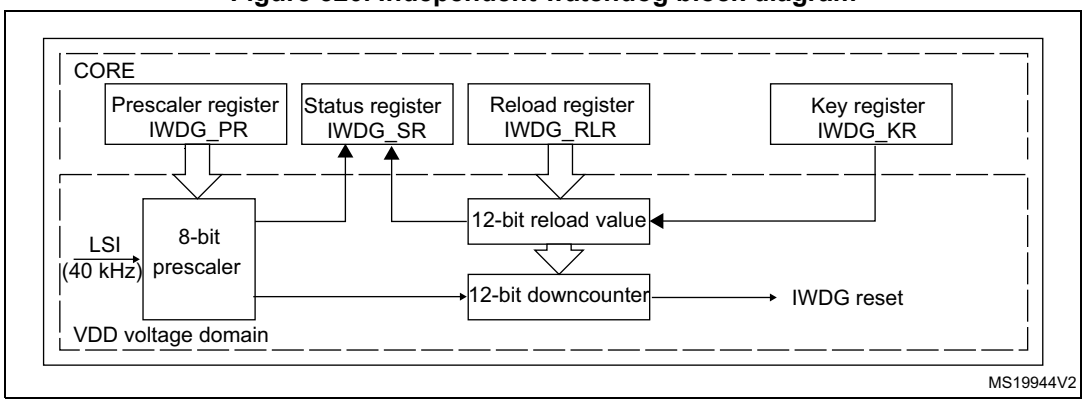
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

24.3 IWDG functional description

24.3.1 IWDG block diagram

Figure 320 shows the functional blocks of the independent watchdog module.

Figure 320. Independent watchdog block diagram



1. The register interface is located in the CORE voltage domain. The watchdog function is located in the V_{DD} voltage domain, still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG key register (IWDG_KR)*, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

24.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG_WINR)*, then a reset is provided.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *IWDG reload register (IWDG_RLR)* value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG_RLR)*.

Note: Writing the window value allows the counter value to be refreshed by the RLR when IWDG status register (IWDG_SR) is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

24.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

24.3.4 Register access protection

Write access to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)* and *IWDG window register (IWDG_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.

24.3.5 Debug mode

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.

24.4 IWDG registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

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

24.4.1 IWDG key register (IWDG_KR)

Address offset: 0x00

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

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

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

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 24.3.4: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

24.4.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 24.3.4: Register access protection*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

24.4.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

24.4.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

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

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

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

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five 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 five RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

24.4.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 24.3.4*, they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

24.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 109. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	**IWDG_WINR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2 on page 47* for the register boundary addresses.

25 System window watchdog (WWDG)

25.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

25.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 322*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

25.3 WWDG functional description

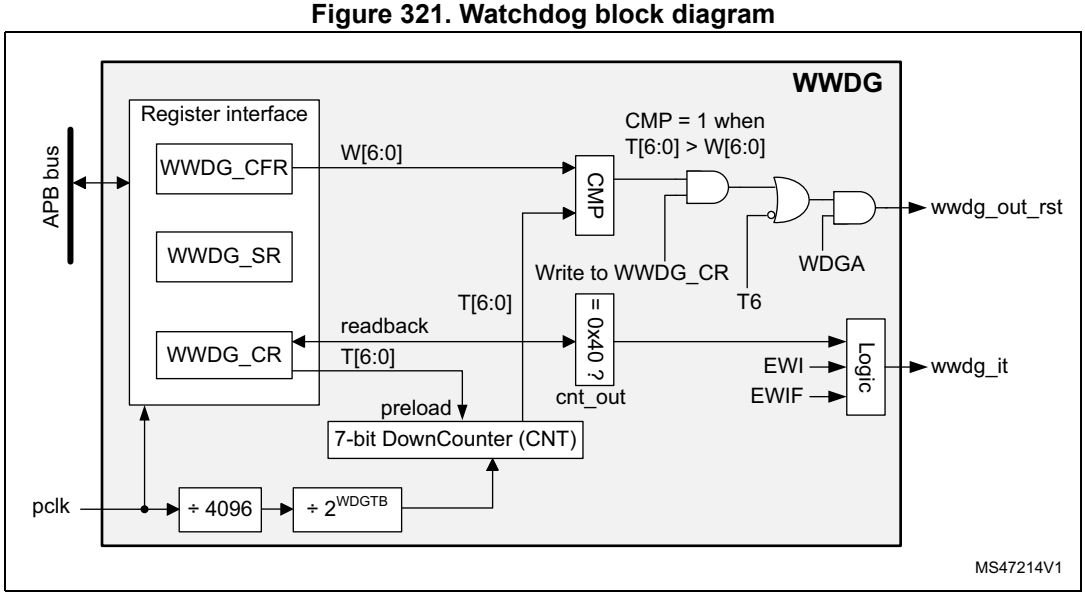
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 321* for the WWDG block diagram.

25.3.1 WWDG block diagram

Figure 321. Watchdog block diagram



25.3.2 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

25.3.3 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 322*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 322* describes the window watchdog process.

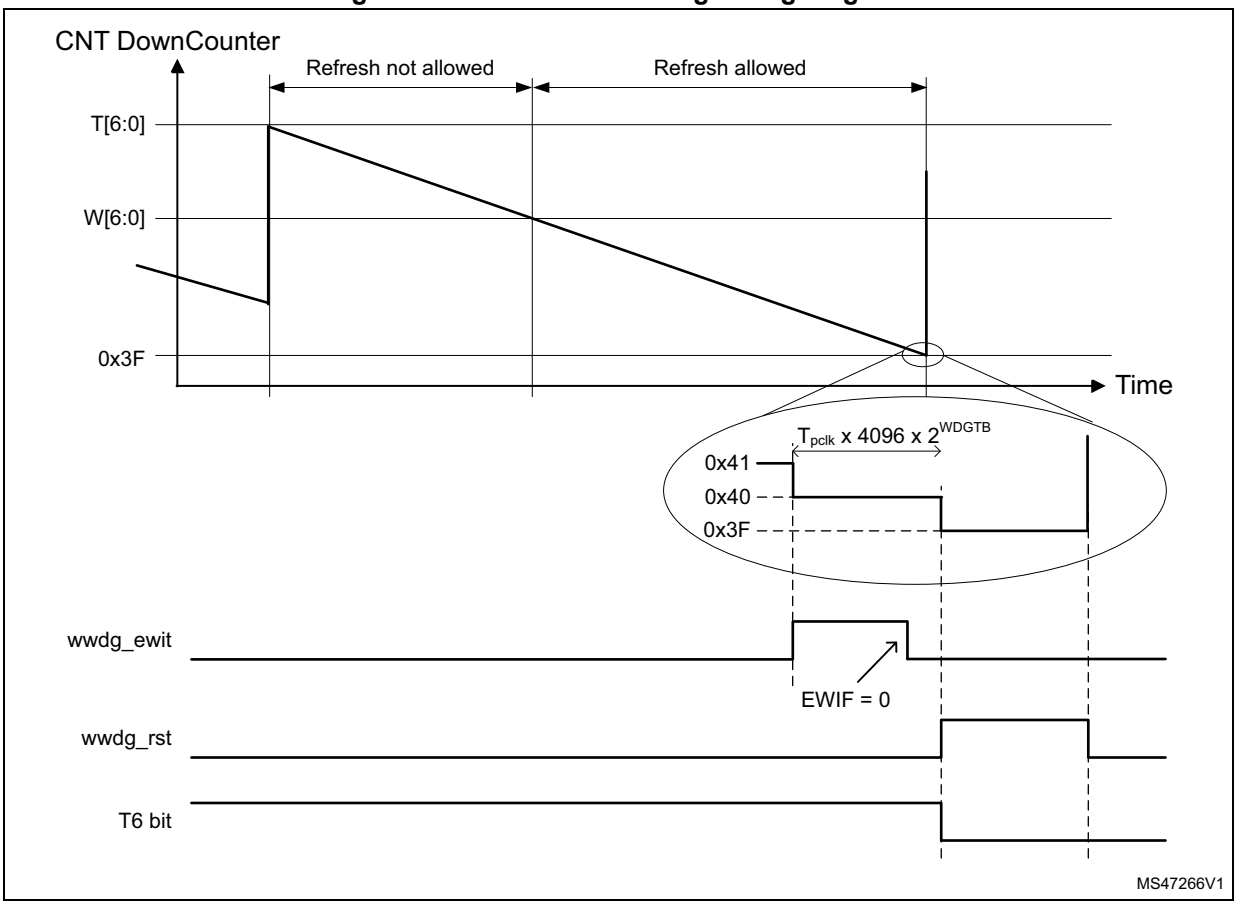
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

25.3.4 How to program the watchdog timeout

Use the formula in *Figure 322* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 322. Window watchdog timing diagram



The formula to calculate the 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_{PCLK}: APB1 clock period measured in ms

4096: value corresponding to internal divider

As an example, lets assume APB1 frequency is equal to 48 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of the t_{WWDG}.

25.3.5 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to .

25.4 WWDG interrupts

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 down-counter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) has to reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset is eventually generated.

25.5 WWDG registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

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

25.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[1:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

25.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		W[6:0]						
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:

00: CK Counter Clock (PCLK div 4096) div 1

01: CK Counter Clock (PCLK div 4096) div 2

10: CK Counter Clock (PCLK div 4096) div 4

11: CK Counter Clock (PCLK div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

25.5.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

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

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing '0'. Writing '1' has no effect. This bit is also set if the interrupt is not enabled.

25.5.4 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
0x000	**WWDG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x008	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

26 Real-time clock (RTC)

26.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After RTC 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

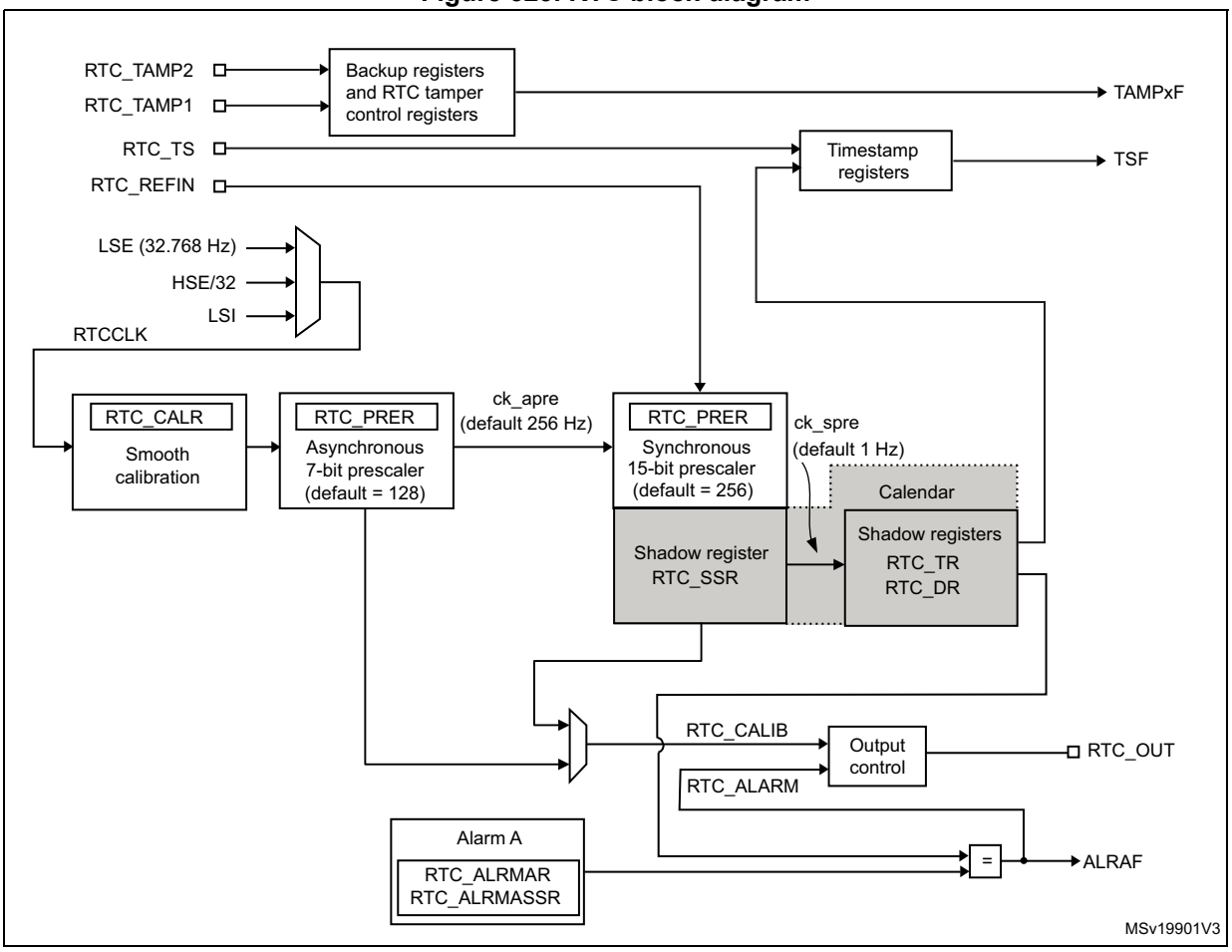
The RTC unit main features are the following (see *Figure 323: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- 16 backup registers.

26.3 RTC functional description

26.3.1 RTC block diagram

Figure 323. RTC block diagram



The RTC includes:

- Two alarms
- Two tamper events from I/Os
  - Tamper detection erases the backup registers.
- One timestamp event from I/O
- Tamper event detection can generate a timestamp event
- 5 x 32-bit backup registers
  - The backup registers (RTC_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.
- Output functions: RTC_OUT which selects one of the following two outputs:
  - RTC_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
  - RTC_ALARM: This output is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the Alarm A, Alarm B or Wakeup outputs.
- Input functions:
  - RTC_TS: timestamp event
  - RTC_TAMP1: tamper1 event detection
  - RTC_TAMP2: tamper2 event detection
  - RTC_REFIN: 50 or 60 Hz reference clock input

26.3.2 GPIOs controlled by the RTC

RTC_OUT, RTC_TS and RTC_TAMP1 are mapped on the same pin (PC13). PC13 pin configuration is controlled by the RTC, whatever the PC13 GPIO configuration, except for the RTC_ALARM output open-drain mode. In this particular case, the GPIO must be configured as input. The RTC functions mapped on PC13 are available in all low-power modes and in VBAT mode.

The selection of the RTC_ALARM output is performed through the RTC_TAFCR register as follows: the PC13VALUE bit is used to select whether the RTC_ALARM output is configured in push-pull or open drain mode.

When PC13 is not used as RTC function, it can be forced in output push-pull mode by setting the PC13MODE bit in the RTC_TAFCR. The output data value is then given by the PC13VALUE bit. In this case, PC13 output push-pull state and data are preserved in Standby mode.

The output mechanism follows the priority order shown in *Table 111*.

When PC14 and PC15 are not used as LSE oscillator, they can be forced in output push-pull mode by setting the PC14MODE and PC15MODE bits in the RTC_TAFCR register respectively. The output data values are then given by PC14VALUE and PC15VALUE. In this case, the PC14 and PC15 output push-pull states and data values are preserved in Standby mode.

The output mechanism follows the priority order shown in *Table 112* and *Table 113*.

Table 111. RTC pin PC13 configuration(1)

Pin configuration and function	RTC_ALARM output enabled	RTC_CALIB output enabled	RTC_TAMP1 input enabled	RTC_TS input enabled	PC13MODE bit	PC13VALUE bit
RTC_ALARM output OD	1	Don't care	Don't care	Don't care	Don't care	0
RTC_ALARM output PP	1	Don't care	Don't care	Don't care	Don't care	1
RTC_CALIB output PP	0	1	Don't care	Don't care	Don't care	Don't care
RTC_TAMP1 input floating	0	0	1	0	Don't care	Don't care
RTC_TS and RTC_TAMP1 input floating	0	0	1	1	Don't care	Don't care
RTC_TS input floating	0	0	0	1	Don't care	Don't care
Output PP forced	0	0	0	0	1	PC13 output data value
Wakeup pin or Standard GPIO	0	0	0	0	0	Don't care

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

Table 112. LSE pin PC14 configuration (1)

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC14MODE bit	PC14VALUE bit
LSE oscillator	1	0	Don't care	Don't care
LSE bypass	1	1	Don't care	Don't care
Output PP forced	0	Don't care	1	PC14 output data value
Standard GPIO	0	Don't care	0	Don't care

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

Table 113. LSE pin PC15 configuration (1)

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC15MODE bit	PC15VALUE bit
LSE oscillator	1	0	Don't care	Don't care
Output PP forced	1	1	1	PC15 output data value
	0	Don't care		
Standard GPIO	0	Don't care	0	Don't care

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

26.3.3 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 8: Reset and clock control (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 323: 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 26.3.6: Periodic auto-wakeup* for details).

26.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK period, 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: RTC initialization and status*

register (RTC_ISR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 1 RTCCLK period.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

26.3.5 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

26.3.6 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer on page 842*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in

the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. RTC_ALARM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

26.3.7 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After RTC domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_TAFCR, RTC_BKPxR and RTC_ISR[13:8].

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: After a system reset, the application can read the INITS flag in the RTC_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its RTC domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

1. Clear ALRAE in RTC_CR to disable Alarm A.
2. Program the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable Alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

26.3.8 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third

read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every 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 1 RTCCLK period: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 841*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 26.3.10: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

26.3.9 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers

(RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper and alternate function configuration register (RTC_TAFCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR).

In addition, when it is clocked by the LSE, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one (refer to the RTC clock section of the Reset and clock controller for details on the list of RTC clock sources not affected by system reset). When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

26.3.10 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (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 output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (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.

26.3.11 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the synchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

26.3.12 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2]=1

causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the

calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

26.3.13 Time-stamp function

Time-stamp is enabled by setting the TSE bit of RTC_CR register to 1.

The calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a time-stamp event is detected on the RTC_TS pin.

When a time-stamp event occurs, the time-stamp flag bit (TSF) in RTC_ISR register is set.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the time-stamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set.To avoid masking a time-stamp event occurring at the same moment, the application must not write ‘0’ into TSF bit unless it has already read it to ‘1’.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in *Section 26.6.16: RTC tamper and alternate function configuration register (RTC_TAFCR)*.

26.3.14 Tamper detection

The RTC_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers
- generate an interrupt, capable to wakeup from Stop and Standby modes

RTC backup registers

The backup registers (RTC_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see *Section 26.6.19: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization on page 848*).

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC_TAFCR register.

Each RTC_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC_ISR register.

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

By setting the TAMPIE bit in the RTC_TAFCR register, an interrupt is generated when a tamper detection event occurs. .

Timestamp on tamper event

With TAMPTS set to ‘1’, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the RTC_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC_TAMPx inputs are deactivated when edge detection is selected.

Caution: To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with the corresponding TAMPxE bit in order to detect a tamper detection event in case it occurs before the RTC_TAMPx pin is enabled.

- When TAMPxTRG = 0: if the RTC_TAMPx is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as the RTC_TAMPx input is enabled, even if there was no rising edge on the RTC_TAMPx input after TAMPxE was set.
- When TAMPxTRG = 1: if the RTC_TAMPx is already low before tamper detection is enabled, a tamper event is detected as soon as the RTC_TAMPx input is enabled (even if there was no falling edge on the RTC_TAMPx input after TAMPxE was set.

After a tamper event has been detected and cleared, the RTC_TAMPx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the RTC_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC_TAMPx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC_TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on RTC_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC_TAMPx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1,The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC_TAMPx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheets for the electrical characteristics of the pull-up resistors.

26.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

Note: When the RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT pin is automatically configured as output.

When COSEL bit is cleared, the RTC_CALIB output is the output of the 6th stage of the asynchronous prescaler.

When COSEL bit is set, the RTC_CALIB output is the output of the 8th stage of the synchronous prescaler.

26.3.16 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm output

The RTC_ALARM pin can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAFCR register.

Note: Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don't care and must be kept cleared).

When the RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT *pin is automatically configured as output.*

26.4 RTC low-power modes

Table 114. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

26.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to *Section 12.3: EXTI registers*.

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Alarm event in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Tamper event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC TimeStamp event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the Wakeup timer even in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_WKUP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC Wakeup timer event.

Table 115. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop mode	Exit from Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes(1)	yes(1)
Alarm B	ALRBF	ALRBIE	yes	yes(1)	yes(1)
RTC_TS input (timestamp)	TSF	TSIE	yes	yes(1)	yes(1)
RTC_TAMP1 input detection	TAMP1F	TAMPIE	yes	yes(1)	yes(1)
RTC_TAMP2 input detection	TAMP2F	TAMPIE	yes	yes(1)	yes(1)
Wakeup timer interrupt	WUTF	WUTIE	yes	yes(1)	yes(1)

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

26.6 RTC registers

Refer to *Section 1.2 on page 43* of the reference manual for a list of abbreviations used in register descriptions.

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

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 841* and *Reading the calendar on page 842*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

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 841* and *Reading the calendar on page 842*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

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

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

26.6.3 RTC control register (RTC_CR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:24 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 (with default prescaler setting)
1: Calibration output is 1 Hz (with default prescaler setting)

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to *Section 26.3.15: Calibration clock output*

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change outside initialization mode.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change outside initialization mode.

Bit 15 **TSIE**: Time-stamp interrupt enable
0: Time-stamp Interrupt disable
1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*
0: Alarm B Interrupt disable
1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled
Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

Bit 9 **ALRBE**: *Alarm B enable*
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE:** Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **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**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Time-stamp event active edge
0: RTC_TS input rising edge generates a time-stamp event
1: RTC_TS input falling edge generates a time-stamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

This register is write protected. The write access procedure is described in RTC register write protection on page 841.

Caution: TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.

26.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to '0'

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to '1' when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to *Re-calibration on-the-fly*.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TAMP2F**: RTC_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP2 input.

It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.

It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).

This flag is cleared by software by writing 0.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).

This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware up to 2 RTCCLK cycles after the WUTE bit has been set to 0 in RTC_CR, and is cleared up to 2 RTCCLK cycles after the WUTE bit has been set to 1. The wakeup timer values can be changed when WUTE bit is cleared and WUTWF is set.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.

26.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 841*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

26.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x14

RTC domain reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

26.6.7 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x1C

RTC 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.

26.6.8 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x20

RTC 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

26.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

26.6.10 RTC sub second register (RTC_SSR)

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

26.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

26.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

26.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

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

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

26.6.14 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

26.6.15 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*.

Address offset: 0x3C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 * CALP) - CALM.

Refer to *Section 26.3.12: RTC smooth digital calibration*.

Bit 14 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to ‘1’, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at “00” when CALW8=’1’. Refer to Section 26.3.12: RTC smooth digital calibration.

Bit 13 **CALW16:** Use a 16-second calibration cycle period

When CALW16 is set to ‘1’, the 16-second calibration cycle period is selected.This bit must not be set to ‘1’ if CALW8=1.

Note: CALM[0] is stuck at ‘0’ when CALW16=’1’. Refer to Section 26.3.12: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 26.3.12: RTC smooth digital calibration on page 845*.

26.6.16 RTC tamper and alternate function configuration register (RTC_TAFCR)

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15 MODE	PC15 VALUE	PC14 MODE	PC14 VALUE	PC13 MODE	PC13 VALUE	Res.	Res.
								rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMPP UDIS	TAMPPRCH [1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPT S	Res.	Res.	TAMP2 TRG	TAMP2 E	TAMPIE	TAMP1 TRG	TAMP1 E
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 **PC15MODE**: PC15 mode

0: PC15 is controlled by the GPIO configuration registers. Consequently PC15 is floating in Standby mode.

1: PC15 is forced to push-pull output if LSE is disabled.

Bit 22 **PC15VALUE**: PC15 value

If the LSE is disabled and PC15MODE = 1, PC15VALUE configures the PC15 output data.

Bit 21 **PC14MODE**: PC14 mode

0: PC14 is controlled by the GPIO configuration registers. Consequently PC14 is floating in Standby mode.

1: PC14 is forced to push-pull output if LSE is disabled.

Bit 20 **PC14VALUE**: PC14 value

If the LSE is disabled and PC14MODE = 1, PC14VALUE configures the PC14 output data.

Bit 19 **PC13MODE**: PC13 mode

0: PC13 is controlled by the GPIO configuration registers. Consequently PC13 is floating in Standby mode.

1: PC13 is forced to push-pull output if all RTC functions are disabled.

Bit 18 **PC13VALUE**: RTC_ALARM output type/PC13 value

If PC13 is used to output RTC_ALARM, PC13VALUE configures the output configuration:

0: RTC_ALARM is an open-drain output

1: RTC_ALARM is a push-pull output

If all RTC functions are disabled and PC13MODE = 1, PC13VALUE configures the PC13 output data.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TAMPPUDIS**: RTC_TAMPx pull-up disable

This bit determines if each of the RTC_TAMPx pins are pre-charged before each sample.

0: Precharge RTC_TAMPx pins before sampling (enable internal pull-up)

1: Disable precharge of RTC_TAMPx pins.

Bits 14:13 **TAMPPRCH[1:0]**: RTC_TAMPx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the RTC_TAMPx inputs.

0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: RTC_TAMPx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC_TAMPx inputs.

0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the RTC_TAMPx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a timestamp to be saved
1: Save timestamp on tamper detection event

TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **TAMP2TRG**: Active level for RTC_TAMP2 input

if TAMPFLT != 00:
0: RTC_TAMP2 input staying low triggers a tamper detection event.
1: RTC_TAMP2 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP2 input rising edge triggers a tamper detection event.
1: RTC_TAMP2 input falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: RTC_TAMP2 input detection enable

0: RTC_TAMP2 detection disabled
1: RTC_TAMP2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable
0: Tamper interrupt disabled
1: Tamper interrupt enabled.

Bit 1 **TAMP1TRG**: Active level for RTC_TAMP1 input
If TAMPFLT != 00
0: RTC_TAMP1 input staying low triggers a tamper detection event.
1: RTC_TAMP1 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP1 input rising edge triggers a tamper detection event.
1: RTC_TAMP1 input falling edge triggers a tamper detection event.

Bit 0 **TAMP1E**: RTC_TAMP1 input detection enable
0: RTC_TAMP1 detection disabled
1: RTC_TAMP1 detection enabled

Caution: When TAMPFLT = 0, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

26.6.17 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 841*

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.
...
12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.
13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.
14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.
15: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

26.6.18 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected.The write access procedure is described in *Section : RTC register write protection*.

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.
0xF: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

26.6.19 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x8C

RTC 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.

26.6.20 RTC register map

Table 116. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
	Reset value									0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C	**RTC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRBWF	ALRAWF
	Reset value																0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 116. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRMBR**	MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0
0x40	**RTC_TAFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15MODE	PC15MODE	PC14VALUE	PC14MODE	PC13VALUE	PC13VALUE	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]			TAMPFREQ[2:0]		TAMPTS	Res.	Res.	TAMP2TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
	Reset value									0	0	0	0	0	0			0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x44	**RTC_ ALRMASSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ ALRMBSSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 116. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4C	**RTC_ OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_ALARM_TYPE	TSINSEL[1:0]		Res.
	Reset value																													0	0	0	
0x50 to 0x8C	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	to **RTC_BKP15R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

27 Inter-integrated circuit (I2C) interface

27.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

27.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 27.3: I2C implementation*):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming
- Wakeup from Stop mode on address match.

27.3 I2C implementation

This manual describes the full set of features implemented in I2C1

Table 117. STM32F334xx I2C implementation

I2C features(1)	I2C1
7-bit addressing mode	X
10-bit addressing mode	X
Standard-mode (up to 100 kbit/s)	X
Fast-mode (up to 400 kbit/s)	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X
Independent clock	X
Wakeup from Stop mode	X
SMBus/PMBus	X

1. X = supported.

27.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

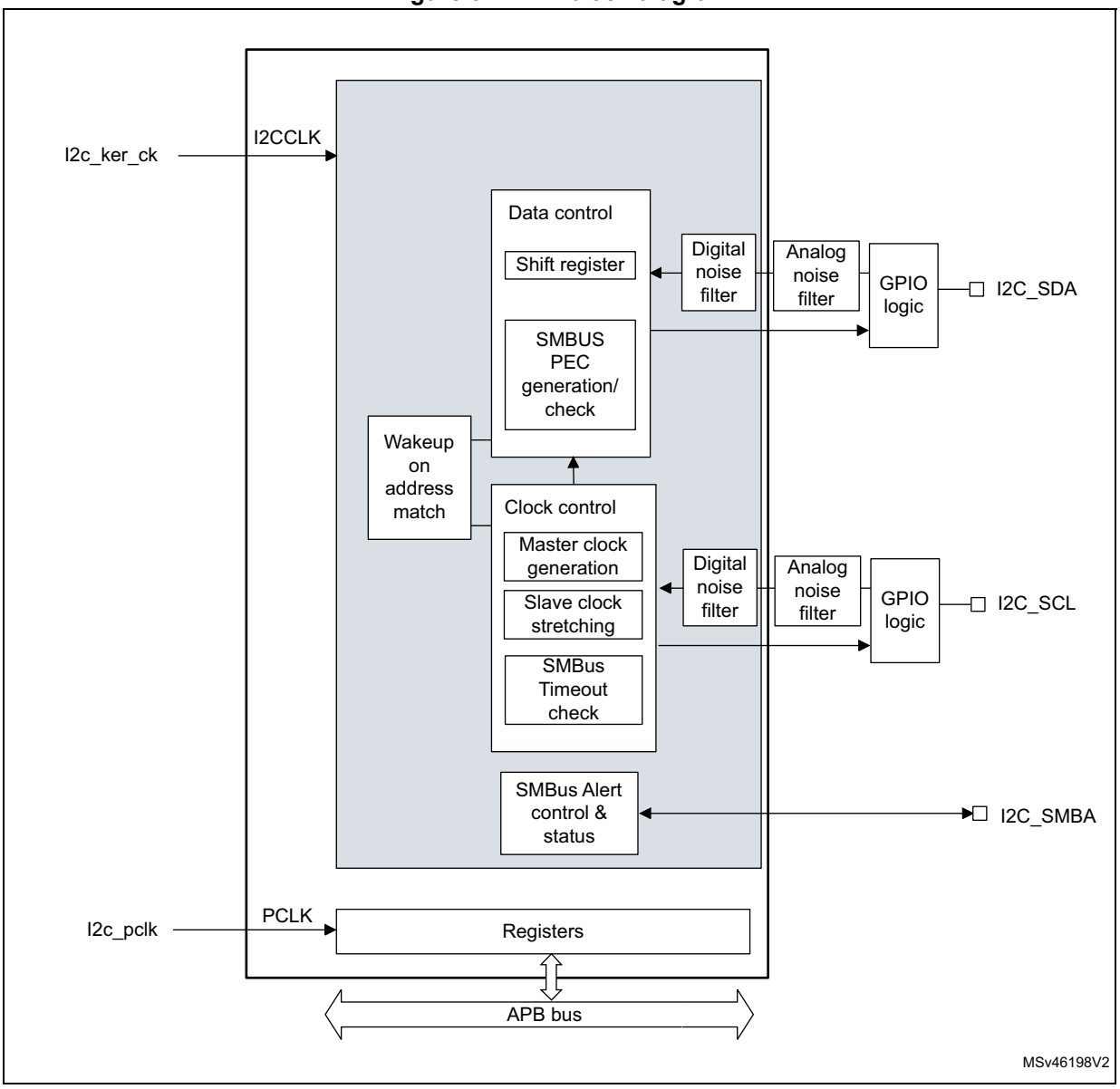
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

27.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 324*.

Figure 324. I2C block diagram



The I2C is clocked by an independent clock source which allows the I2C to operate independently from the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). *Refer to Section 27.3: I2C implementation.*

27.4.2 I2C pins and internal signals

Table 118. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus Alert

Table 119. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 133: I2C Interrupt requests* for the full list of interrupt sources
i2c_rx_dma	Output	I2C Receive Data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C Transmit Data DMA request (I2C_TX)

27.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$ and $t_{I2CCLK} < t_{HIGH}$

with:

t_{LOW}: SCL low time and t_{HIGH}: SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The PCLK clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4/3\ t_{SCL}$

with t_{SCL}: SCL period

Caution: When the I2C kernel is clocked by PCLK, this clock must respect the conditions for t_{I2CCLK}.

27.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

Communication flow

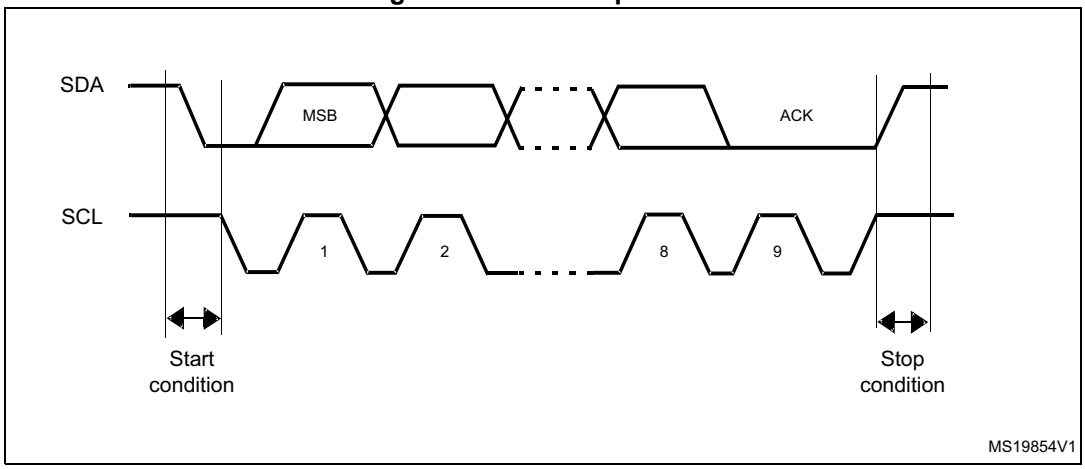
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 325. I^2C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

27.4.5 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller.

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I^2C performs a software reset. Refer to *Section 27.4.6: Software reset* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification which requires the

suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

Table 120. Comparison of analog vs. digital filters

-	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	– Programmable length: extra filtering capability vs. standard requirements – Stable length
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

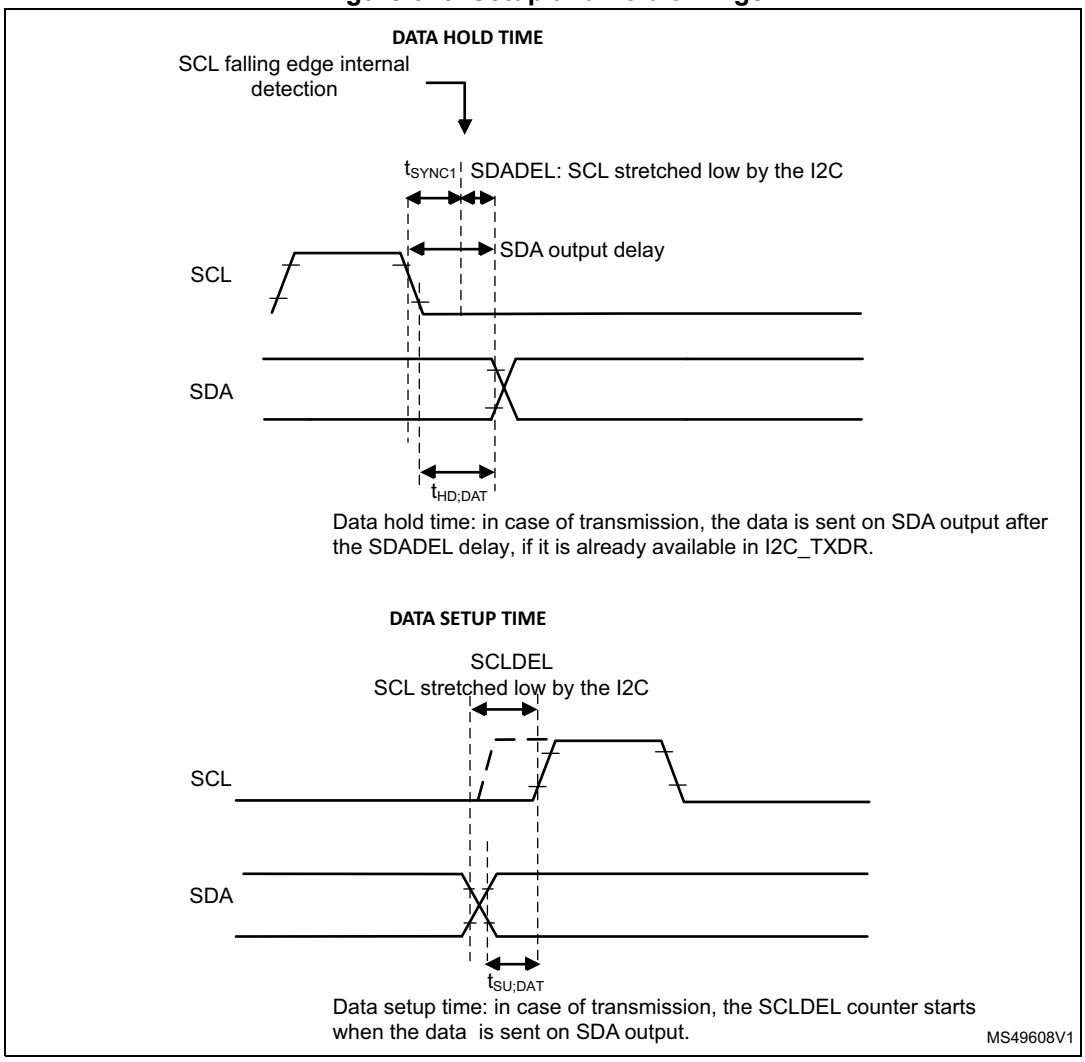
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window

Figure 326. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to *Table 121: I2C-SMBus specification data setup and hold times* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

Refer to *Table 121: I2C-SMBus specification data setup and hold times* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Note: At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH=1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 121. I^2C-SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL} = (SCLL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH} .

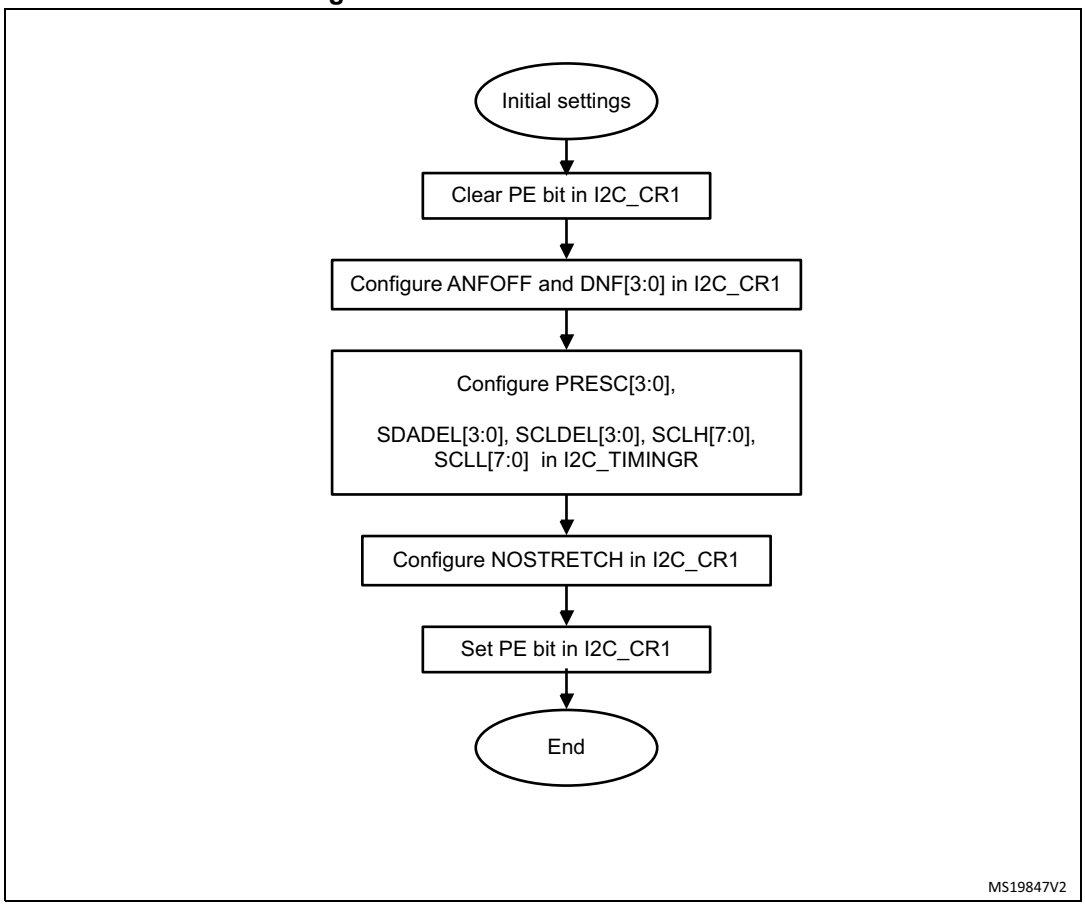
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 327. I2C initialization flowchart



27.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

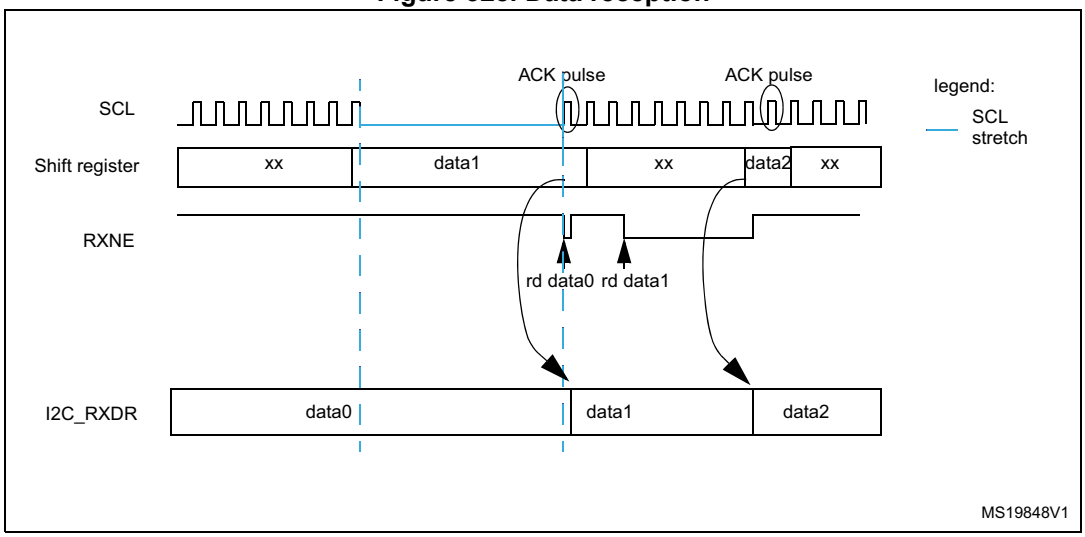
27.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

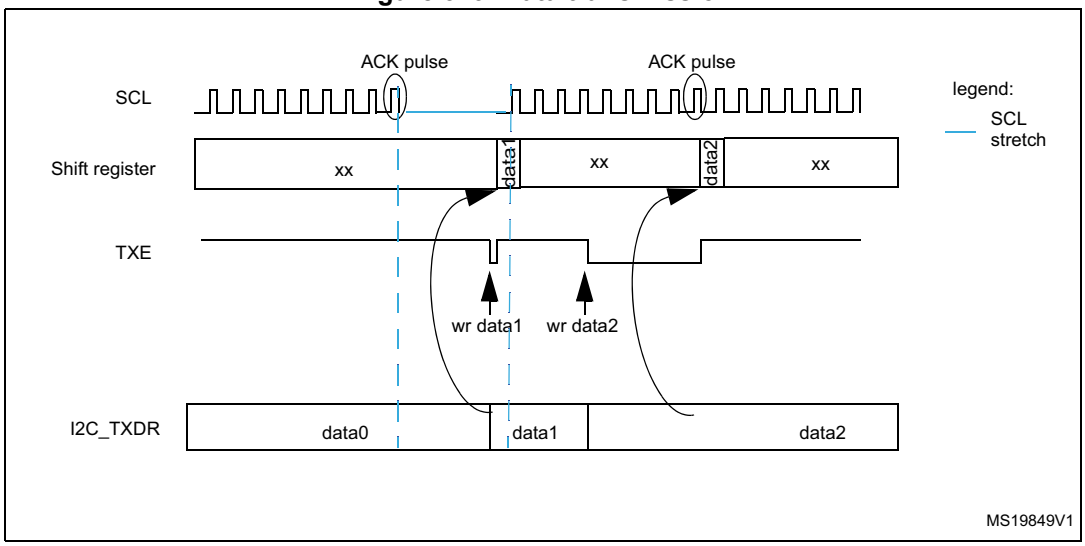
Figure 328. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 329. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = '0' in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 122. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

27.4.8 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

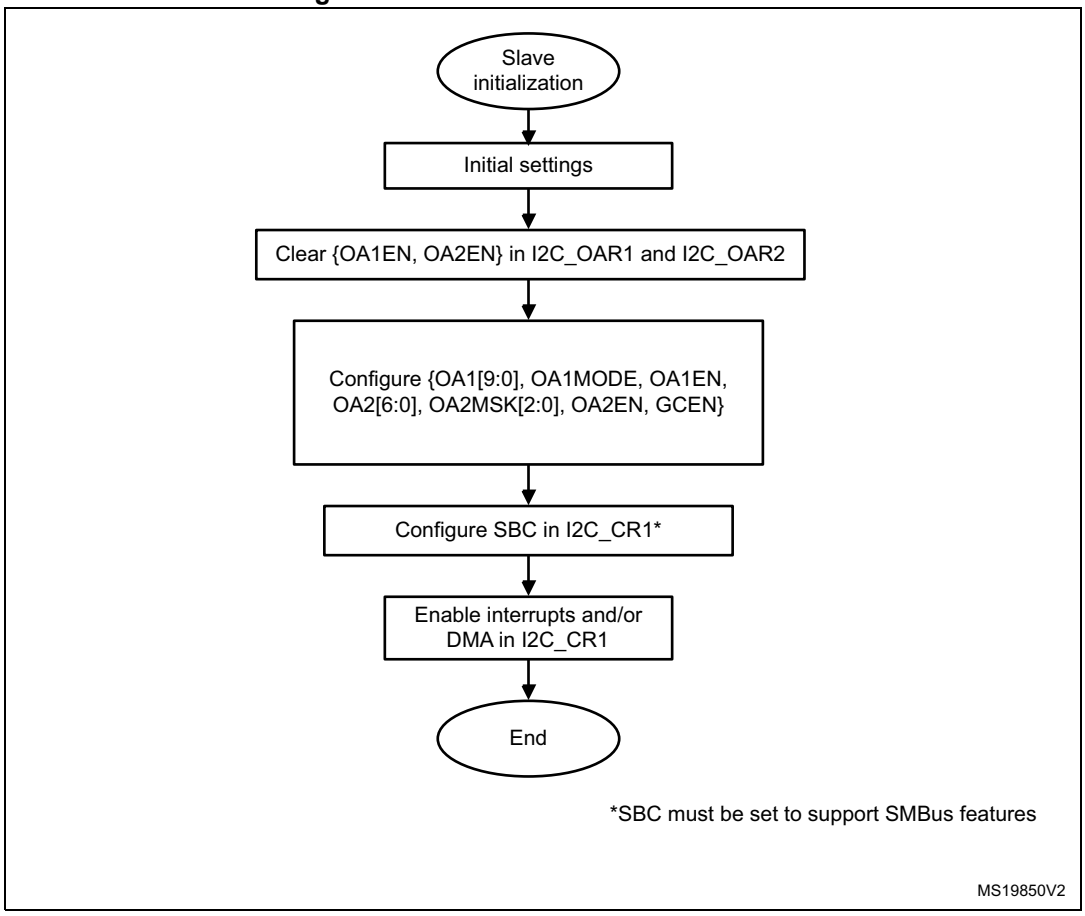
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 330. Slave initialization flowchart



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR=1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event is needed, (Transmit Interrupt or Transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 331. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0

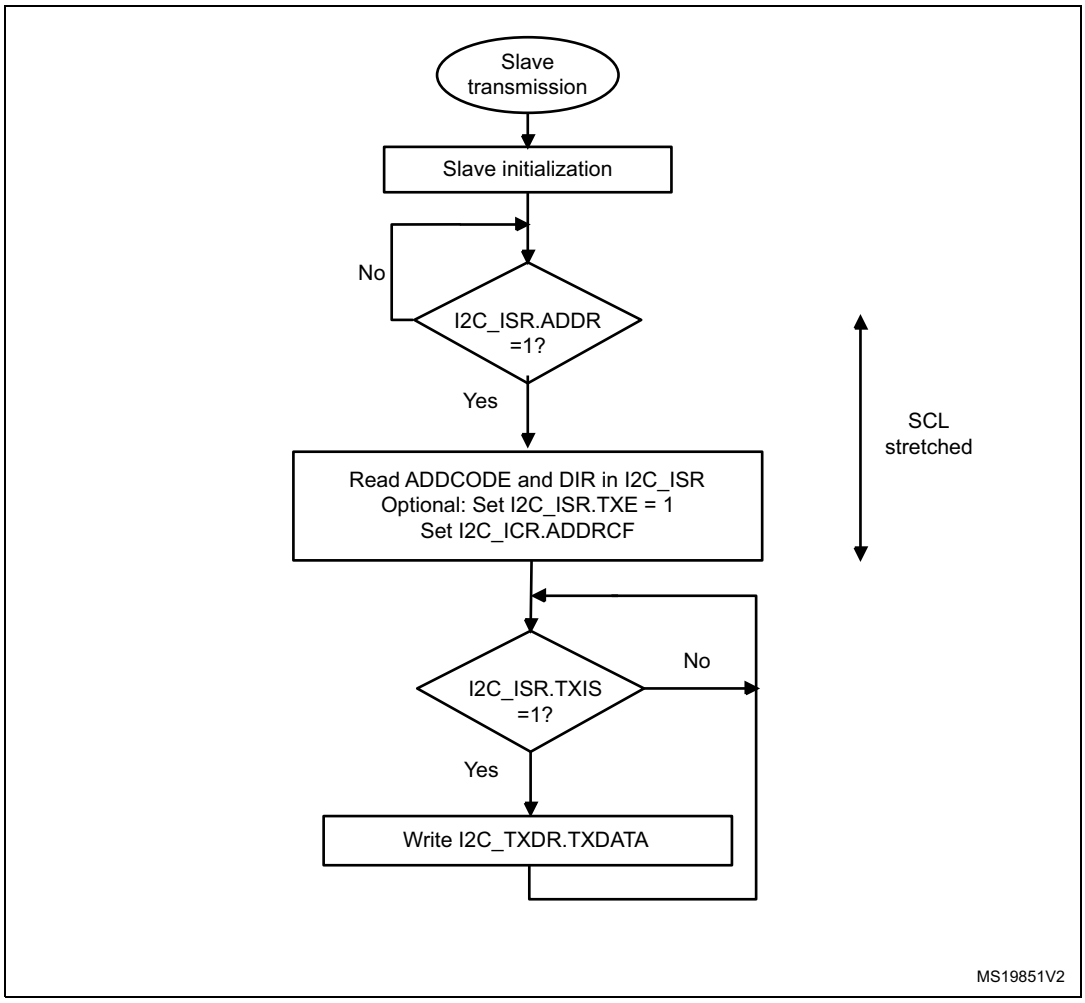


Figure 332. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1

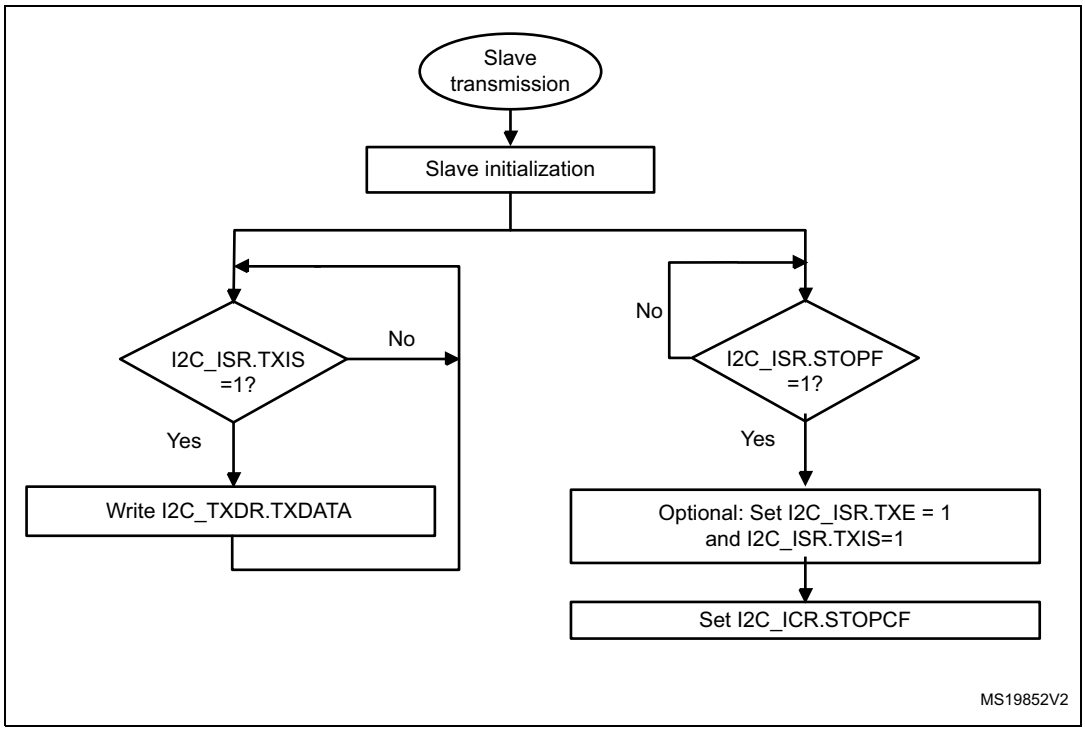
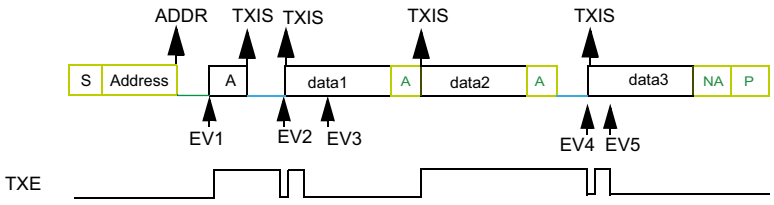


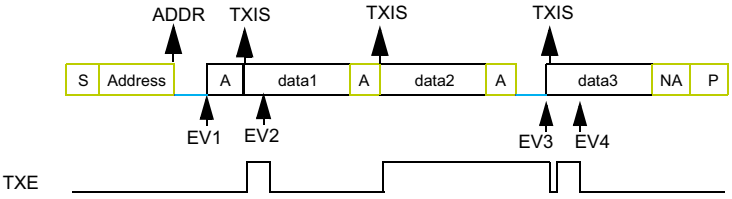
Figure 333. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



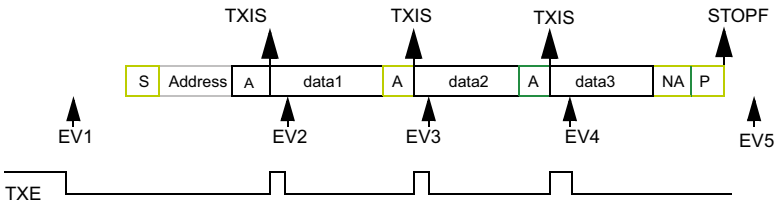
EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V2

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 334. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

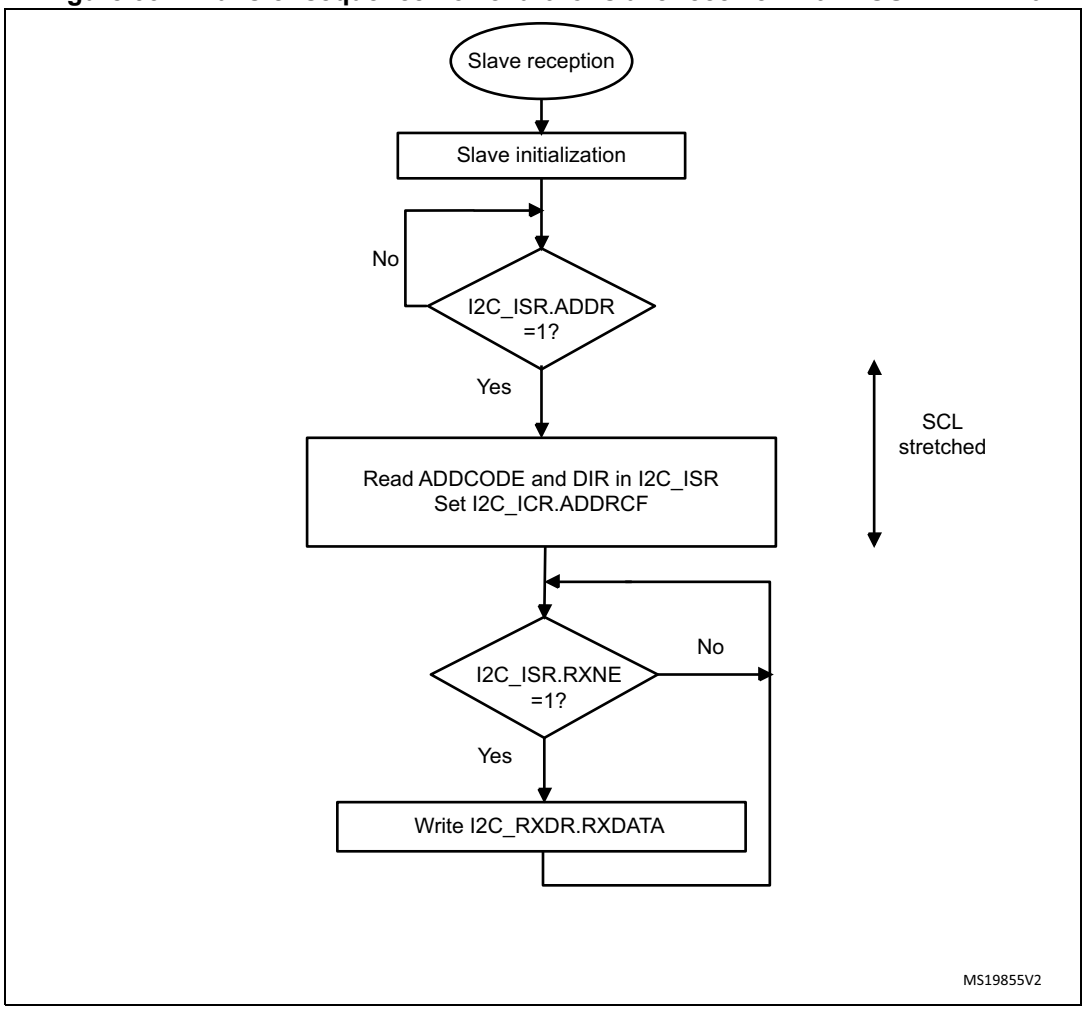


Figure 335. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

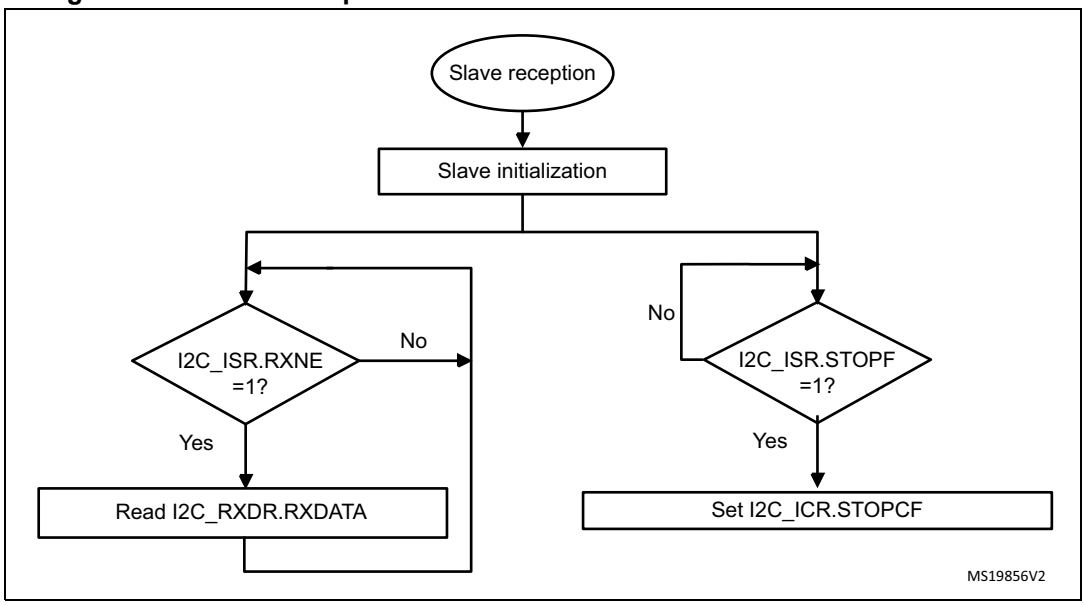
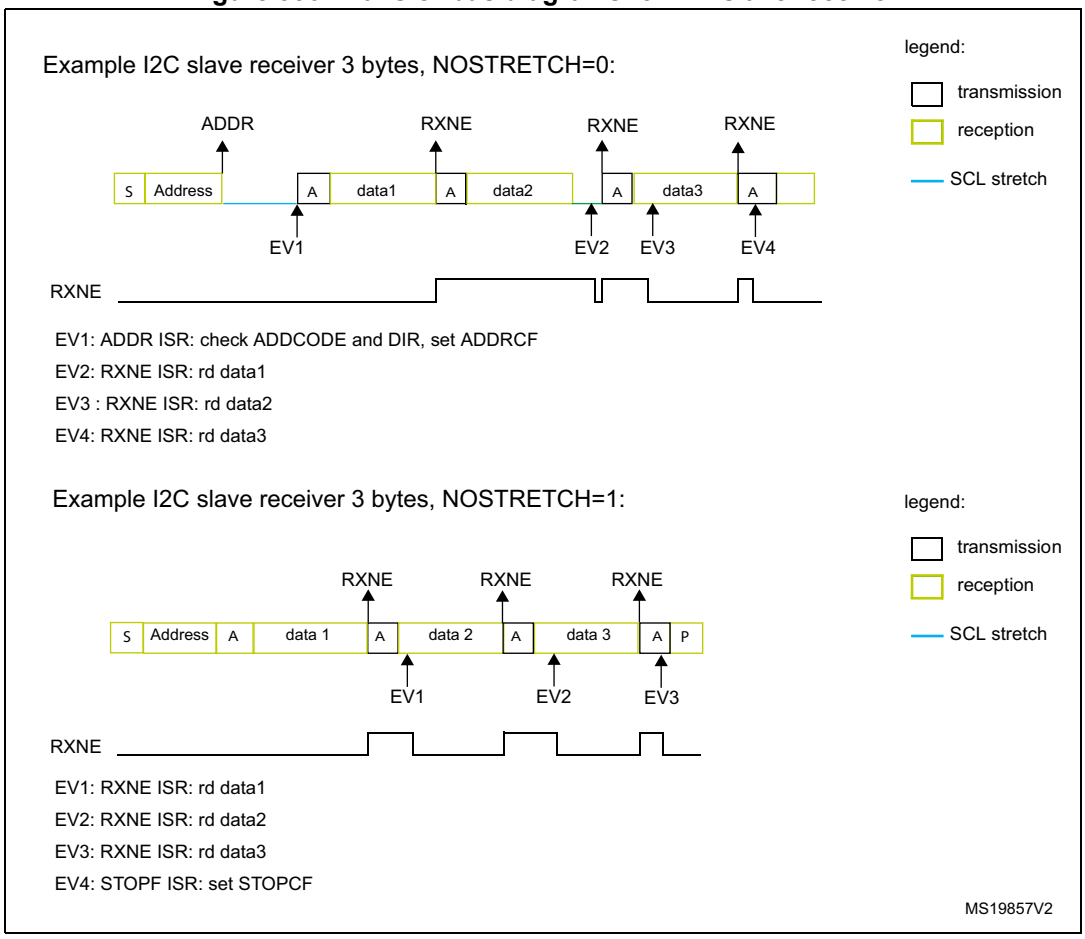


Figure 336. Transfer bus diagrams for I2C slave receiver



27.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

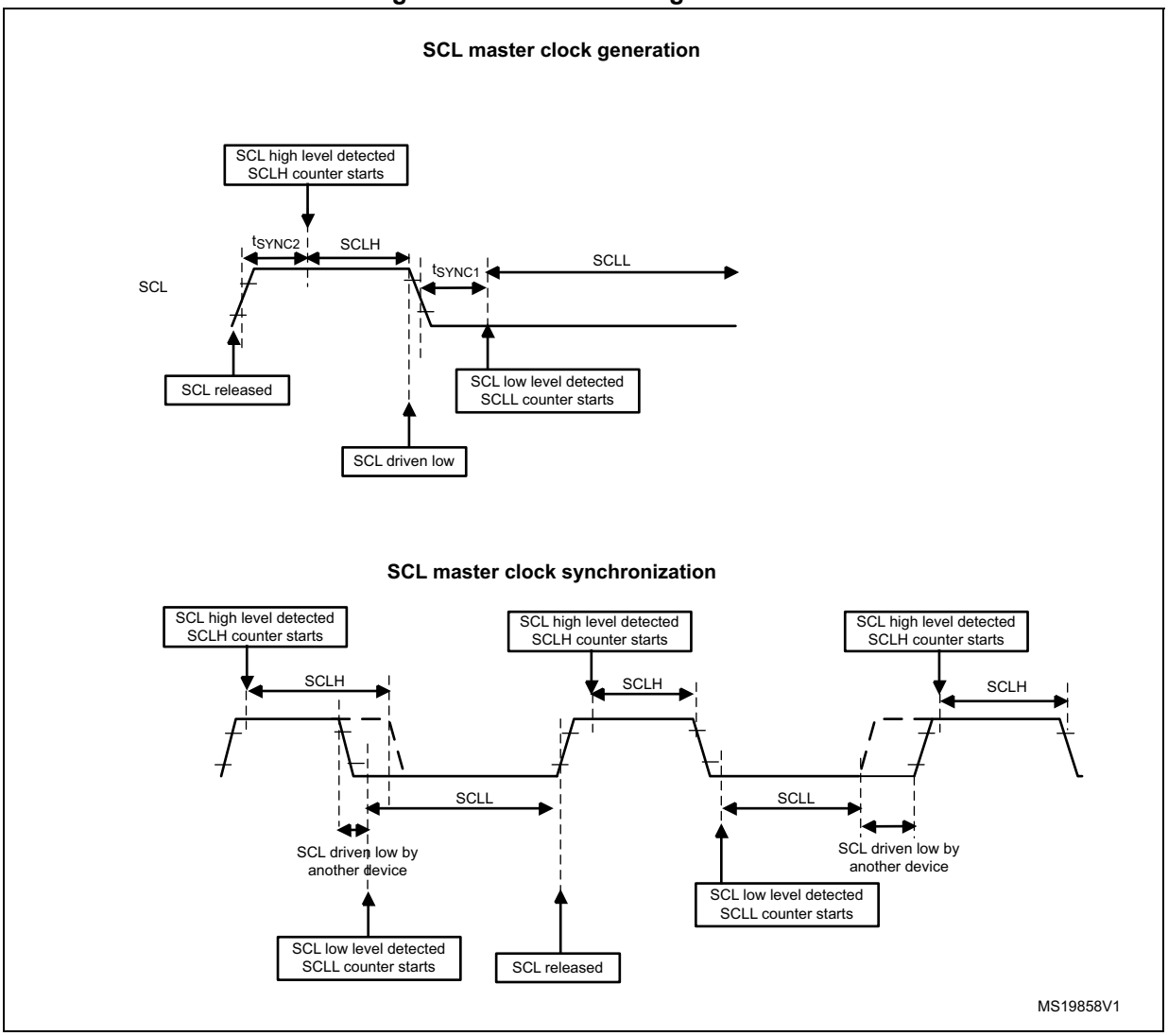
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 337. Master clock generation



Caution: In order to be I^{2}C or SMBus compliant, the master clock must respect the timings given the table below.

Table 123. I²C-SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	µs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 27.4.10: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

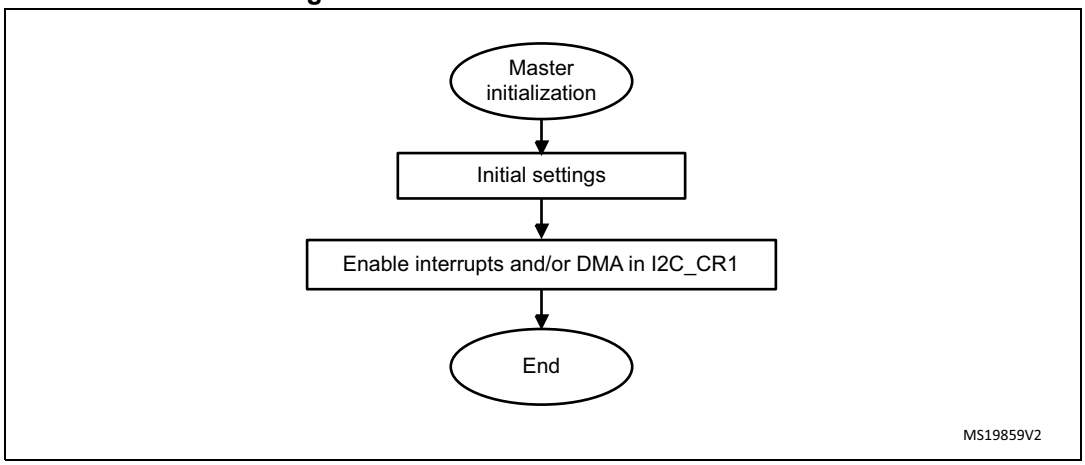
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.
In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the

master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.
If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

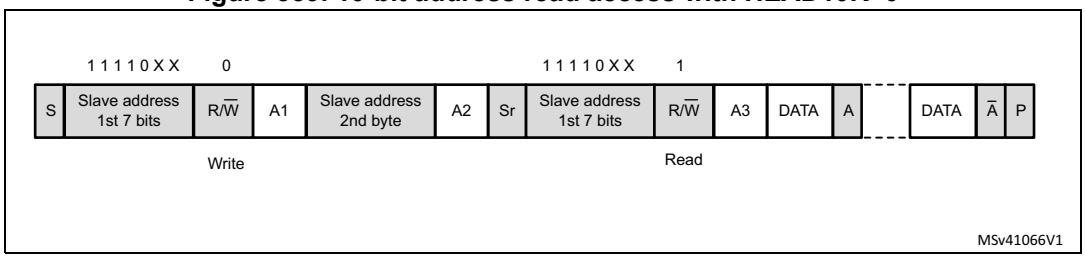
Figure 338. Master initialization flowchart



Initialization of a master receiver addressing a 10-bit address slave

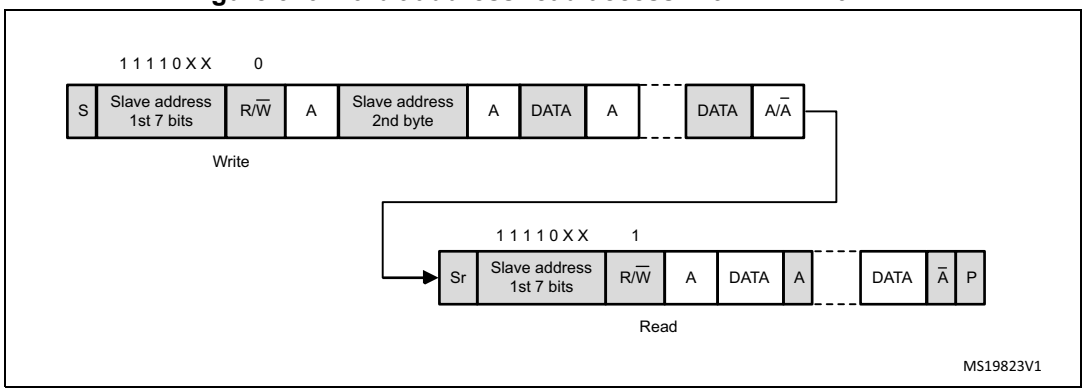
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

Figure 339. 10-bit address read access with HEAD10R=0



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

Figure 340. 10-bit address read access with HEAD10R=1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 341. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

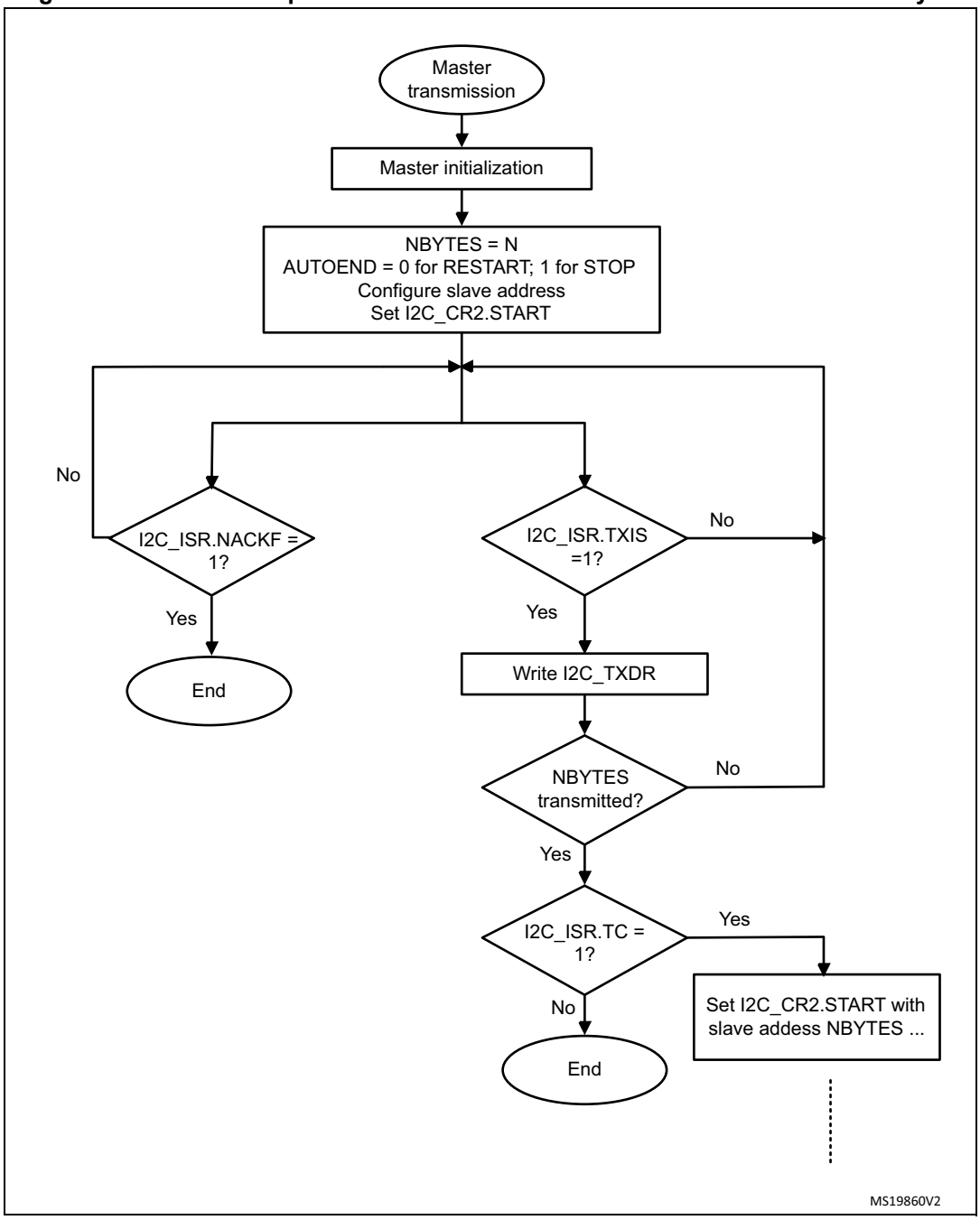


Figure 342. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

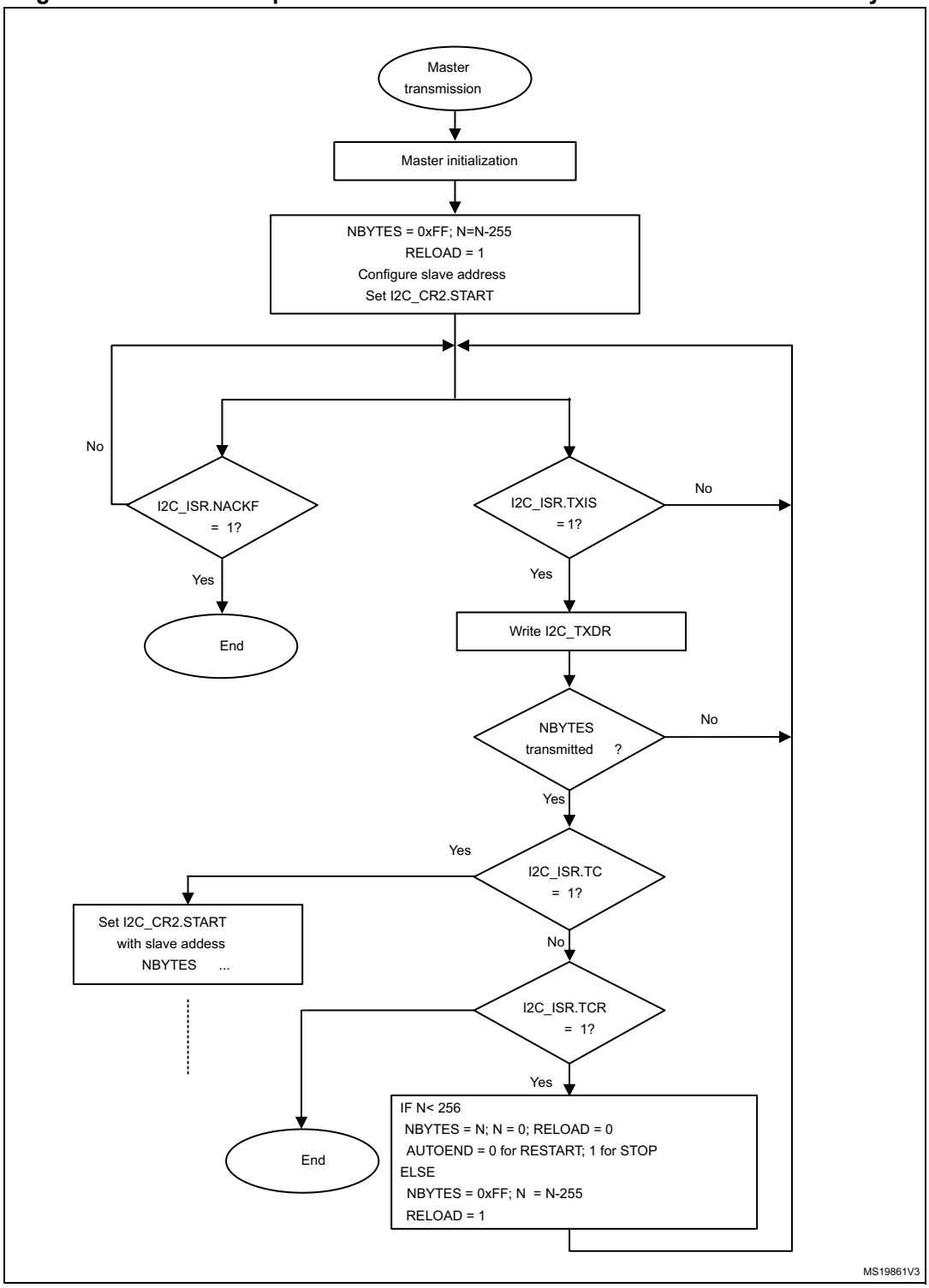
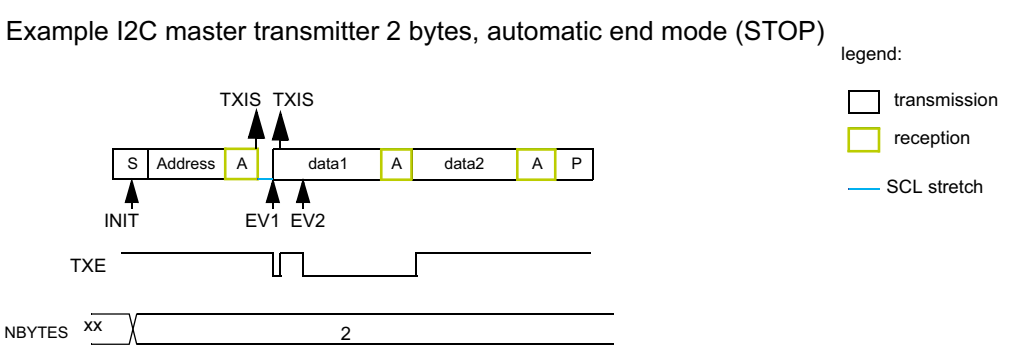


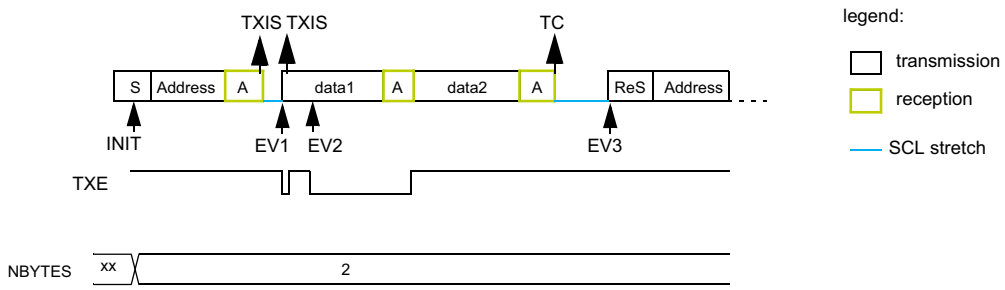
Figure 343. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V2

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 344. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

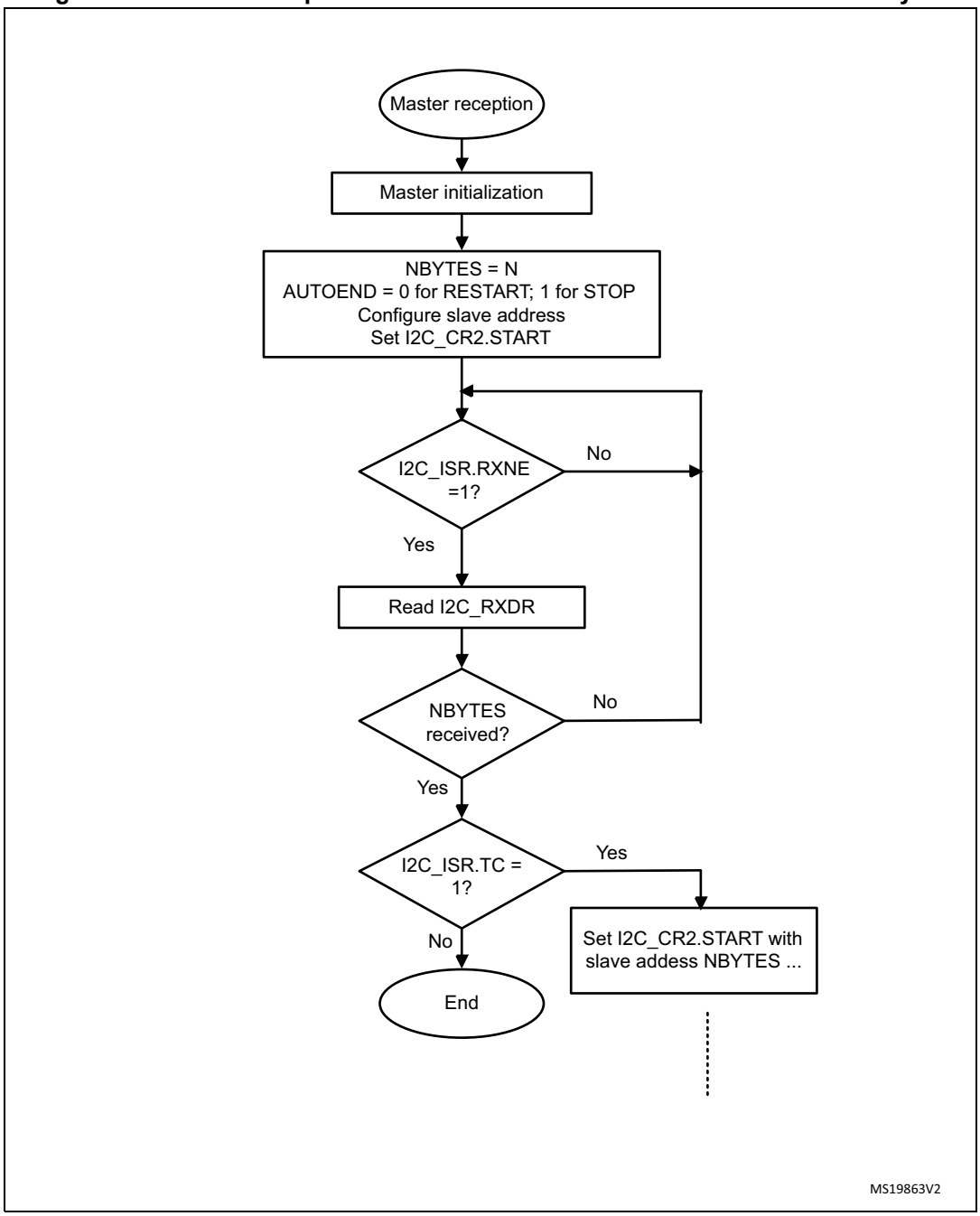


Figure 345. Transfer sequence flowchart for I2C master receiver for N >255 bytes

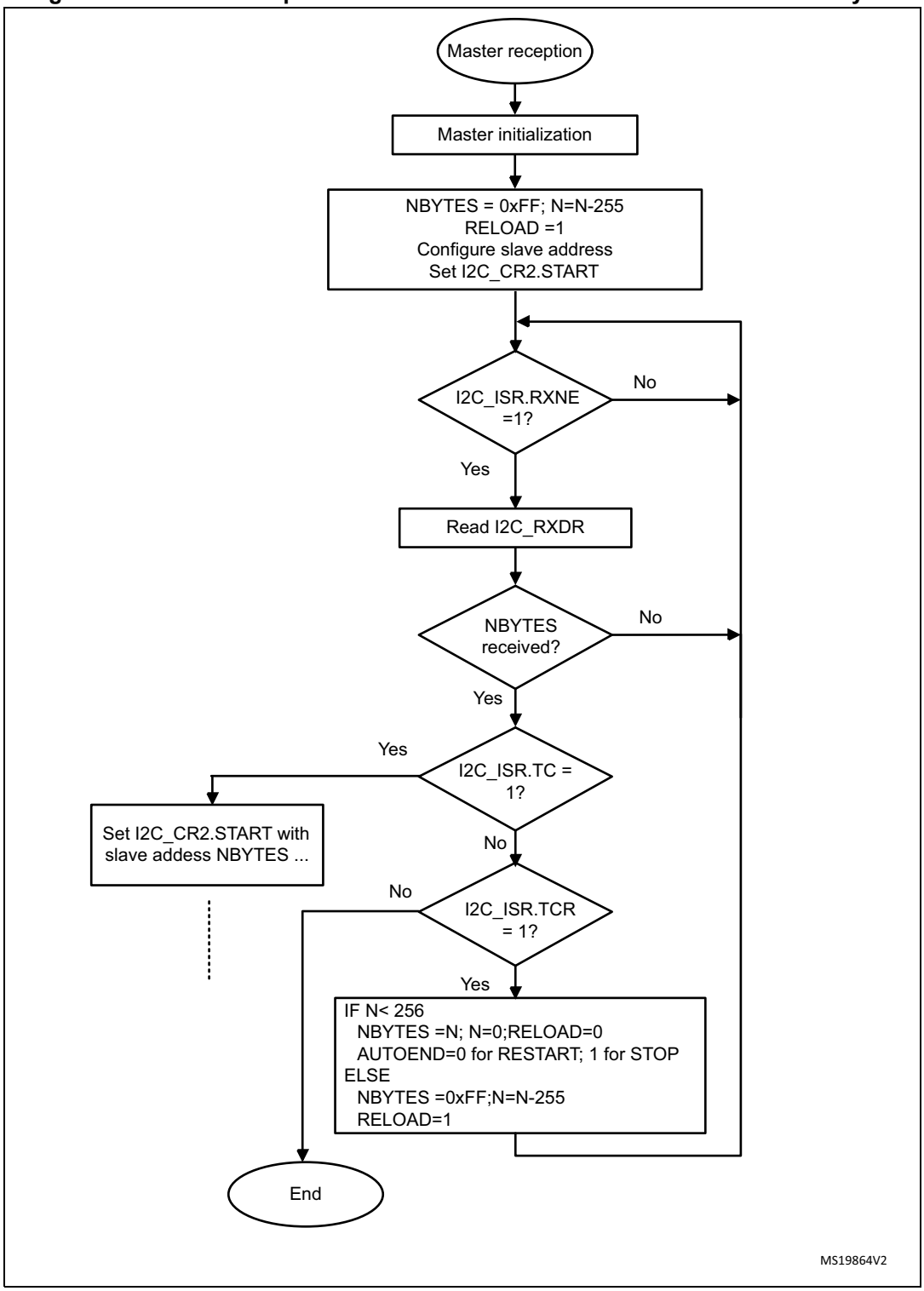
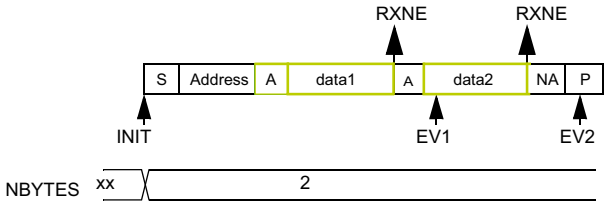


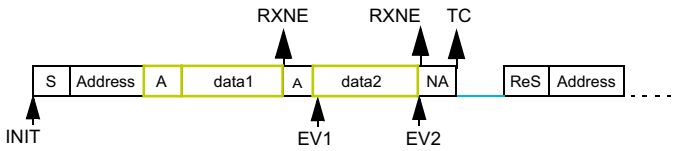
Figure 346. Transfer bus diagrams for I2C master receiver

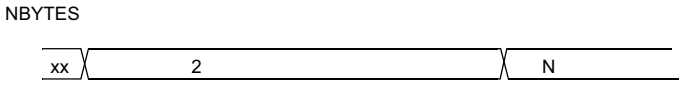
Example I2C master receiver 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)





INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

27.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

Table 124. Examples of timing settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125 ns = 500 ns	4x125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 125. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.

2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 500 ns.

Table 126. Examples of timings settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL} (1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. The SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 250 ns

27.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave Byte Control mode on page 892* for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host Address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a Not Acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

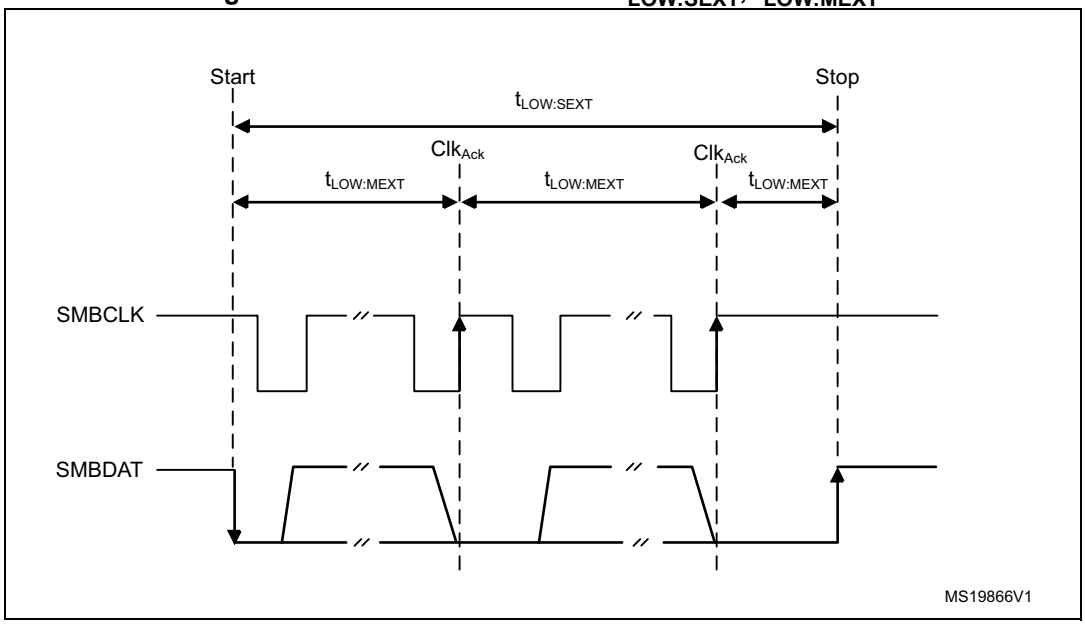
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 127. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (slave device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	ms

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 347. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 121: I2C-SMBus specification data setup and hold times*)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

27.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave Byte Control mode on page 892* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 915* for more details.

- The SMBus Device Default Address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host Address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 128. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 129: Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a slave and

$t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection on page 915* section, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 130: Examples of TIMEOUTB settings for various I2CCLK frequencies*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to ‘1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 131: Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

27.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 129. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 130. Examples of TIMEOUTB settings for various I2CCLK frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 131. Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

27.4.14 SMBus slave mode

This section is relevant only when SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 27.4.8: I2C slave mode*) some additional software flowcharts are provided to support SMBus.

SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 348. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

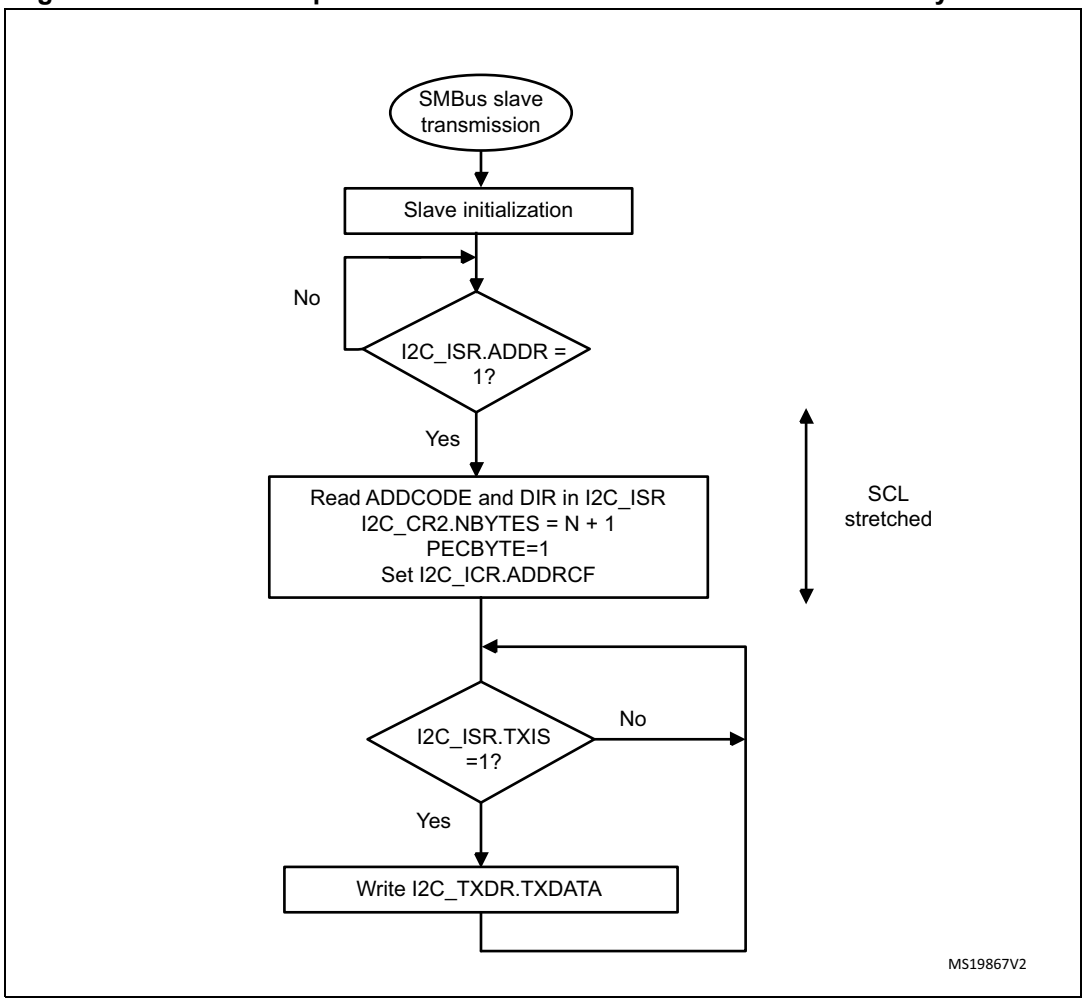
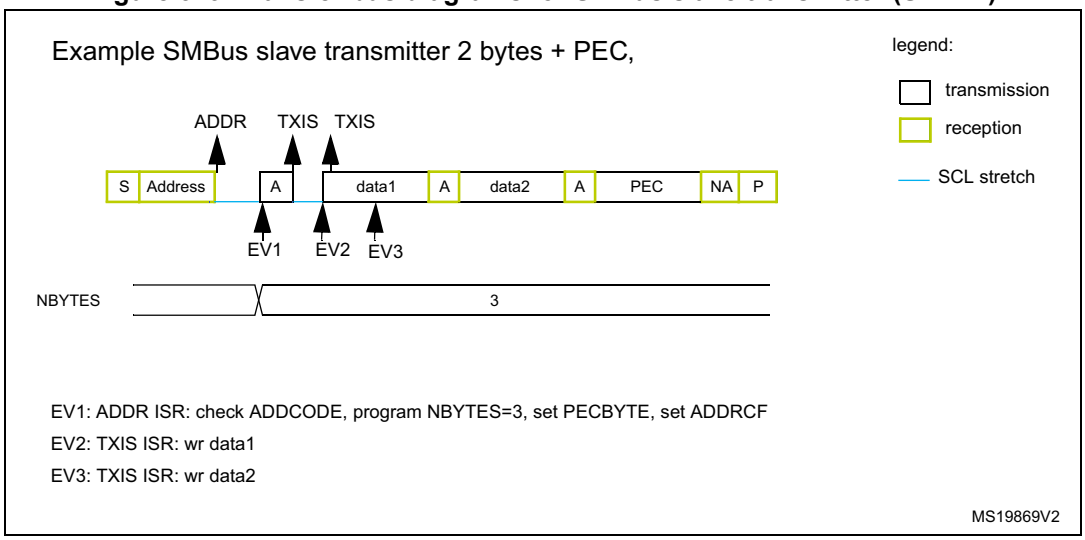


Figure 349. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Slave Byte Control mode on page 892* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 350. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

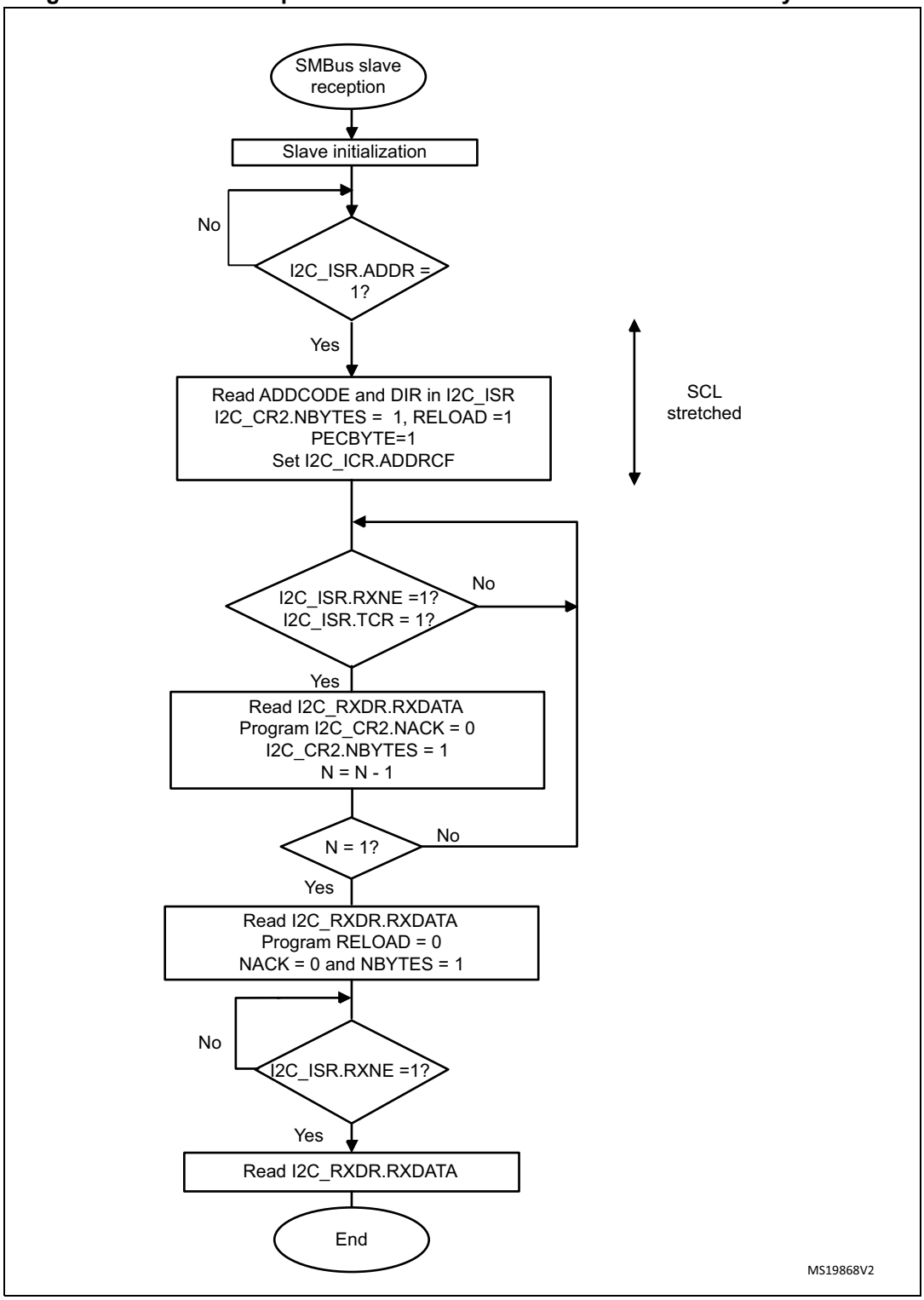
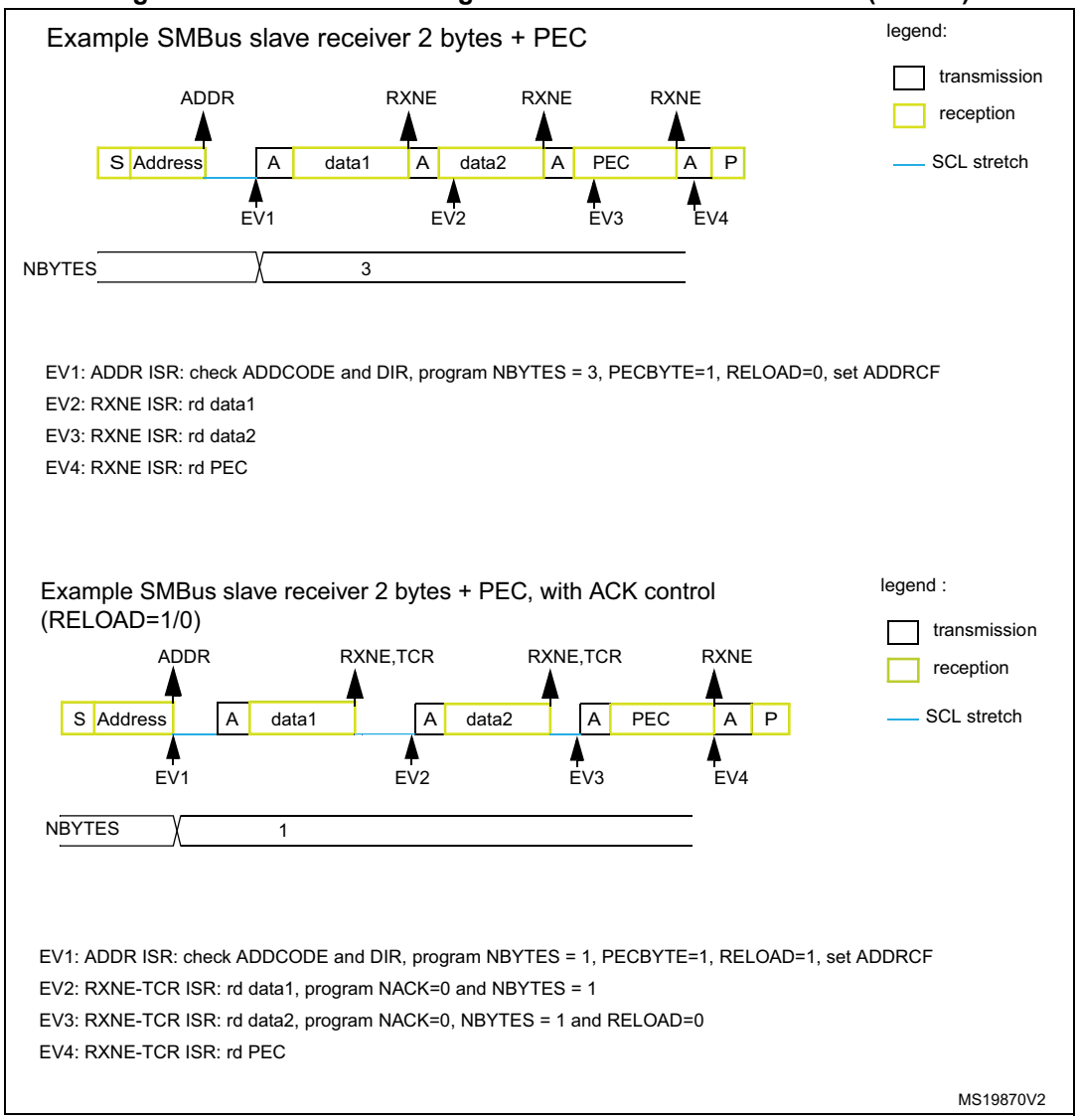


Figure 351. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 27.4.9: I2C master mode*) some additional software flowcharts are provided to support SMBus.

SMBus Master transmitter

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

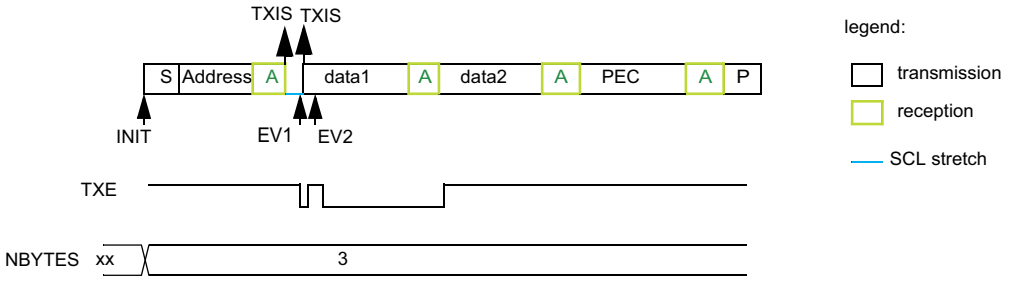
If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

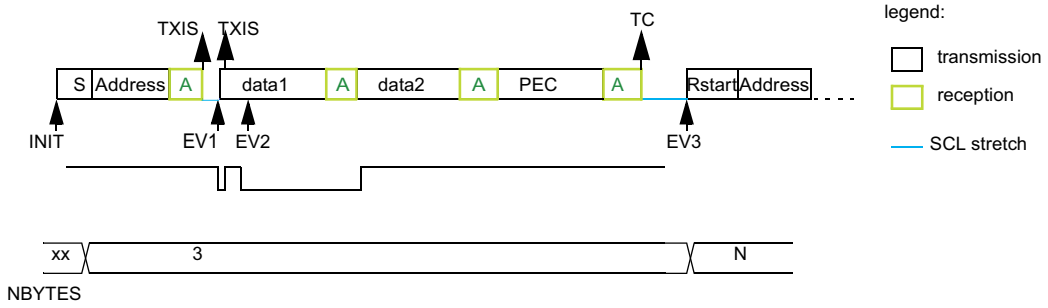
Figure 352. Bus transfer diagrams for SMBus master transmitter

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19871V2

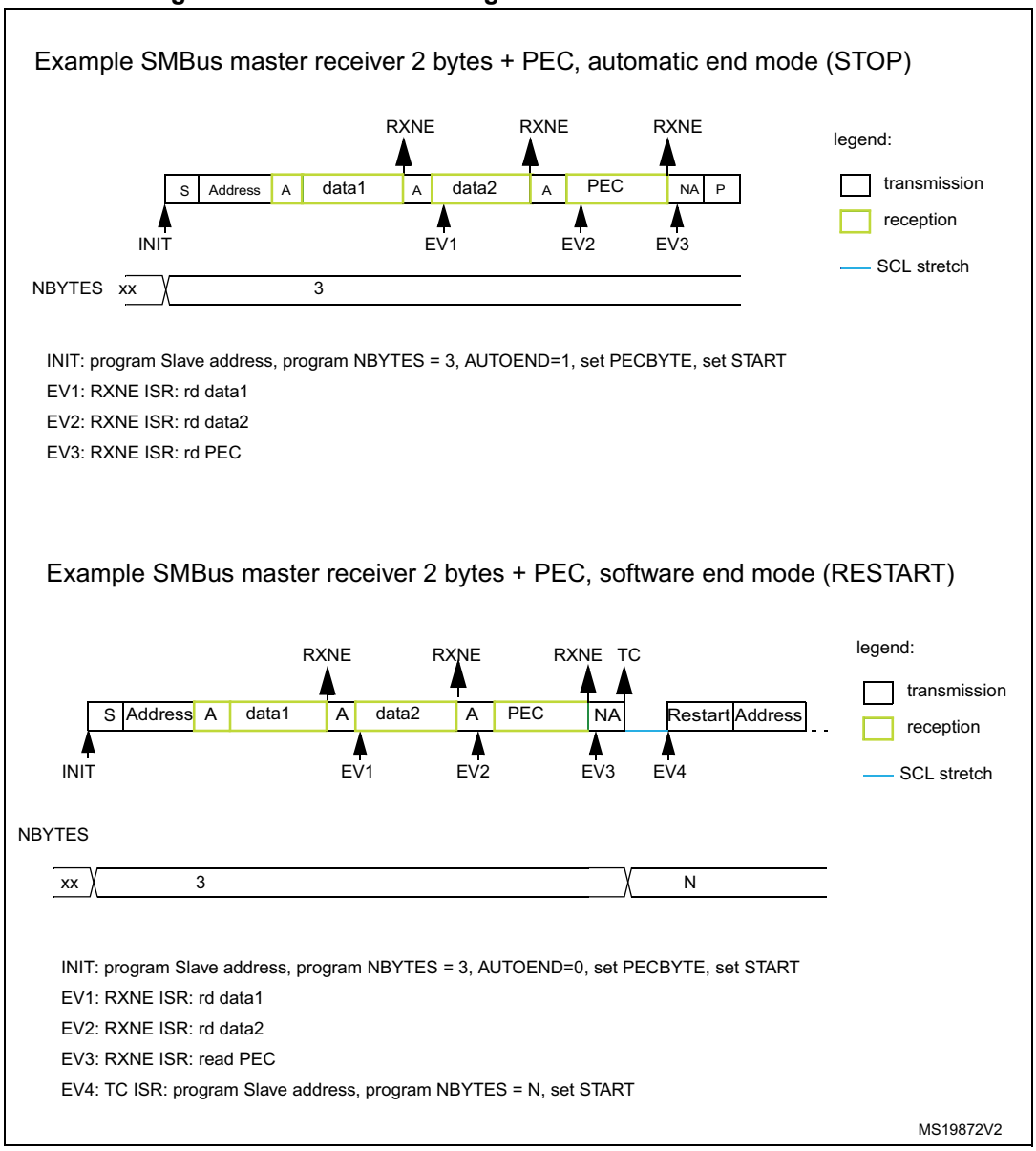
SMBus Master receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 353. Bus transfer diagrams for SMBus master receiver



27.4.15 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Refer to *Section 27.3: I2C implementation*.

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI is switched off. When a START is detected, the I2C interface switches the HSI on, and stretches SCL low until HSI is woken up.

HSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI is switched off again and the MCU is not woken up.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the HSI *is not switched on after a START is received.*

Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

Caution: If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

27.4.16 Error conditions

The following errors are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 27.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

27.4.17 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 14: Direct memory access controller (DMA) on page 479*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter on page 903*.
- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus Slave transmitter on page 918* and *SMBus Master transmitter on page 922*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 11: Direct memory access controller (DMA) on page 170*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the

DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 27.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver on page 920* and *SMBus Master receiver on page 924*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

27.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module.

27.5 I2C low-power modes

Table 132. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The I2C registers content is kept. If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 27.3: I2C implementation* for information about the Stop modes supported by each instance. If wakeup from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.

27.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 133. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby modes
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF=1			
		Transfer Complete Reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
		Transfer complete	TC		Write START=1 or STOP=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1		Yes(1)	
		NACK reception	NACKF	NACKIE	Write NACKCF=1		No	
	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	No	No
		Arbitration loss	ARLO		Write ARLOCF=1			
		Overrun/Under run	OVR		Write OVRCF=1			
		PEC error	PECERR		Write PECERRCF=1			
		Timeout/t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1			
		SMBus Alert	ALERT		Write ALERTCF=1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wakeup from Stop mode feature. Refer to *Section 27.3: I2C implementation*.

27.7 I2C registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

27.7.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **PECEN:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable
0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
1: The SMBus alert pin is supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).
Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus Device Default Address enable
0: Device Default Address disabled. Address 0b1100001x is NACKed.
1: Device Default Address enabled. Address 0b1100001x is ACKed.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus Host Address enable
0: Host Address disabled. Address 0b0001000x is NACKed.
1: Host Address enabled. Address 0b0001000x is ACKed.
Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wakeup from Stop mode enable
0: Wakeup from Stop mode disable.
1: Wakeup from Stop mode enable.
Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.
Note: WUPEN can be set only when DNF = '0000'

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Slave byte control
This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: Stop detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

27.7.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

Note: Writing '0' to this bit has no effect.

This bit has no effect when RELOAD is set.

This bit has no effect is slave mode when SBC=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)
0: The master operates in 7-bit addressing mode,
1: The master operates in 10-bit addressing mode
Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
0: Master requests a write transfer.
1: Master requests a read transfer.
Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Slave address (master mode)
In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] should be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.
In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] should be written with the 10-bit slave address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

27.7.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **OA1EN**: Own Address 1 enable

0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own Address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN=0.

Bits 9:0 **OA1[9:0]**: Interface own slave address

7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

Note: These bits can be written only when OA1EN=0.

27.7.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bit 15 **OA2EN**: Own Address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

27.7.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 884*) and for SCL high and low level counters (refer to *I2C master initialization on page 899*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

27.7.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
In master mode, the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
In slave mode, the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected
$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}
Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

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

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0
$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}
The bus idle condition (both SCL and SDA high) when TIDLE=1
t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}
Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 27.3: I2C implementation.

27.7.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

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

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

27.7.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

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

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 27.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

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

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

27.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

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

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

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

Bits 7:0 **PEC[7:0]** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 27.3: I2C implementation.

27.7.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^2C bus

27.7.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^2C bus
Note: These bits can be written only when TXE=1.

27.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 134. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**I2C_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	**I2C_CR2**	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	**I2C_OAR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	**I2C_OAR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	**I2C_TIMINGR**	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**I2C_TIMEOUTR**	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[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
0x18	**I2C_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	**I2C_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	**I2C_PECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	**I2C_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 134. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**I2C_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

28 Universal synchronous/asynchronous receiver transmitter (USART/UART)

28.1 Introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of Full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a programmable baud rate generator.

It supports synchronous one-way communication and Half-duplex Single-wire communication, as well as multiprocessor communications. It also supports the LIN (Local Interconnect Network), Smartcard protocol and IrDA (Infrared Data Association) SIR ENDEC specifications and Modem operations (CTS/RTS).

High speed data communication is possible by using the DMA (direct memory access) for multibuffer configuration.

28.2 USART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to give flexibility between speed and clock tolerance
- A common programmable transmit and receive baud rate of up to 9 Mbit/s when the clock frequency is 72 MHz and oversampling is by 8
- Dual clock domain allowing:
  - USART functionality and wakeup from Stop mode
  - Convenient baud rate programming independent from the PCLK reprogramming
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous mode and clock output for synchronous communications
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver

- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Fourteen interrupt sources with flags
- Multiprocessor communications

  The USART enters mute mode if the address does not match.
- Wakeup from mute mode (by idle line detection or address mark detection)

28.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11-bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition

28.4 USART implementation

Table 135. STM32F334xx USART features

USART modes/features[1]	USART1	USART2/ USART3
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode	X	X
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from Stop mode	X	-
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7[2], 8 and 9 bits	

1. X = supported.
2. In 7-bit data length mode, Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frame detection) are not supported.

28.5 USART functional description

Any USART bidirectional communication requires a minimum of two pins: Receive data In (RX) and Transmit data Out (TX):

- **RX**: Receive data Input.

  This is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.
- **TX:** Transmit data Output.

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire and Smartcard modes, this I/O is used to transmit and receive the data.

Serial data are transmitted and received through these pins in normal USART mode. The frames are comprised of:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (7, 8 or 9 bits) least significant bit first
- 0.5, 1, 1.5, 2 stop bits indicating that the frame is complete
- The USART interface uses a baud rate generator
- A status register (USART_ISR)
- Receive and transmit data registers (USART_RDR, USART_TDR)
- A baud rate register (USART_BRR)
- A guard-time register (USART_GTPR) in case of Smartcard mode.

Refer to *Section 28.8: USART registers on page 992* for the definitions of each bit.

The following pin is required to interface in synchronous mode and Smartcard mode:

- **CK:** Clock output. This pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX. This can be used to control peripherals that have shift registers. The clock phase and polarity are software programmable. In Smartcard mode, CK output can provide the clock to the smartcard.

The following pins are required in RS232 Hardware flow control mode:

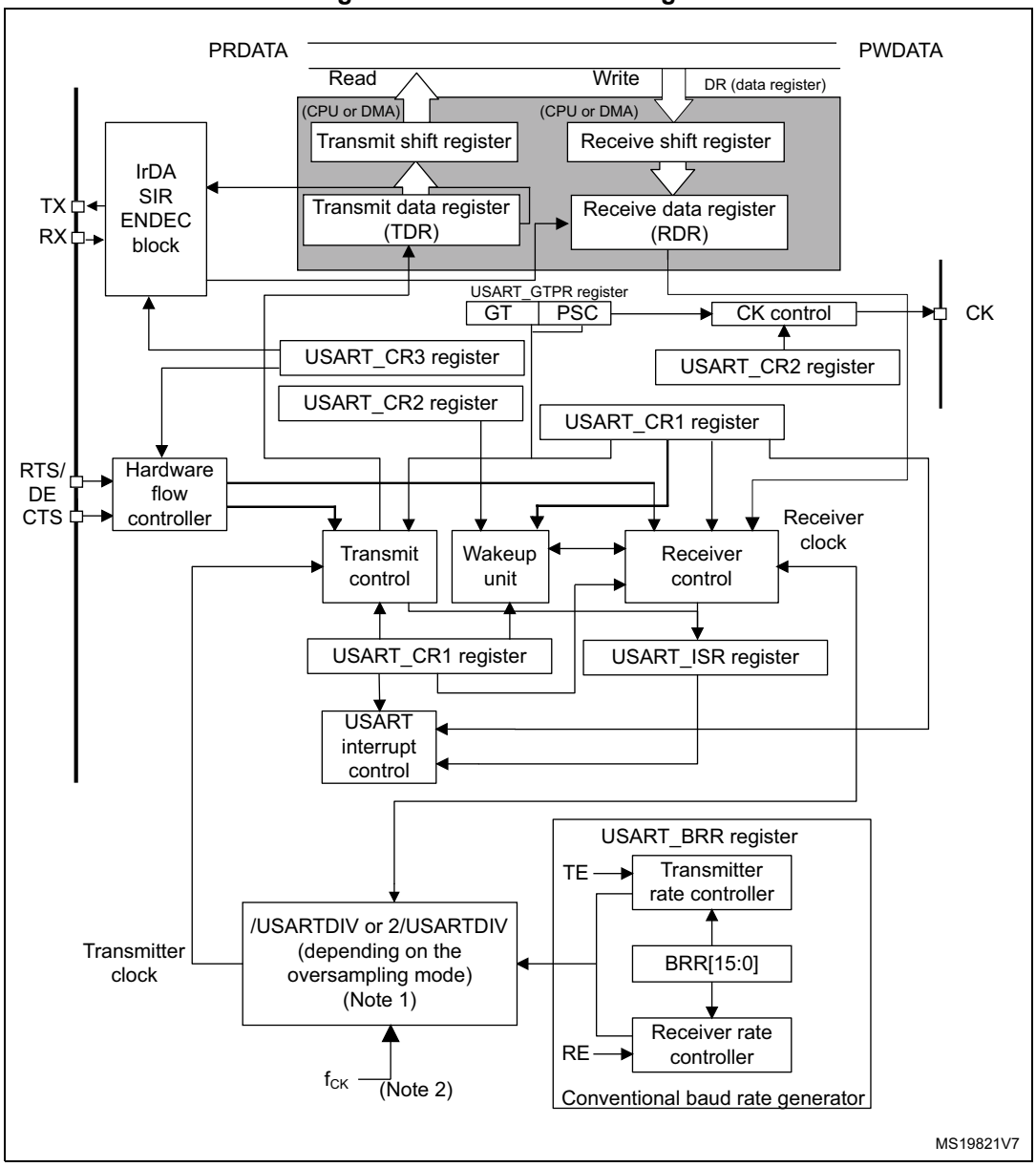
- **CTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **RTS:** Request to send indicates that the USART is ready to receive data (when low).

The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

Note: DE and RTS share the same pin.

Figure 354. USART block diagram



1. For details on coding USARTDIV in the USART_BRR register, refer to *Section 28.5.4: USART baud rate generation*.
2. f_{CK} can be f_{LSE}, f_{HSI}, f_{PCLK}, f_{SYS}.

28.5.1 USART character description

The word length can be selected as being either 7 or 8 or 9 bits by programming the M[1:0] bits in the USART_CR1 register (see *Figure 355*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: The 7-bit mode is supported only on some USARTs. In addition, not all modes are supported in 7-bit data length mode. Refer to Section 28.4: USART implementation for additional information.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

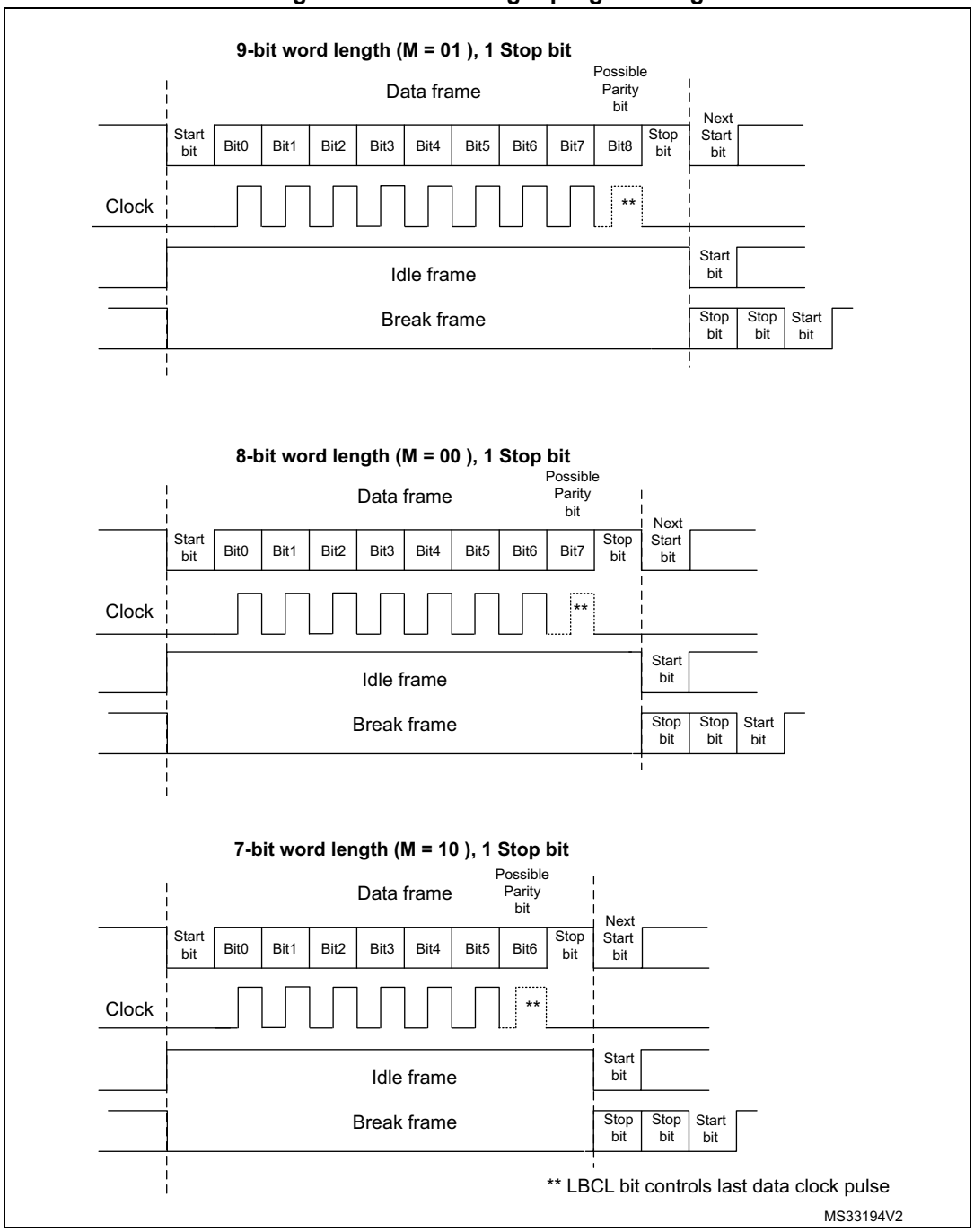
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 355. Word length programming



28.5.2 USART transmitter

The transmitter can send data words of either 7, 8 or 9 bits depending on the M bits status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 354*).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

An idle frame will be sent after the TE bit is enabled.

Configurable stop bits

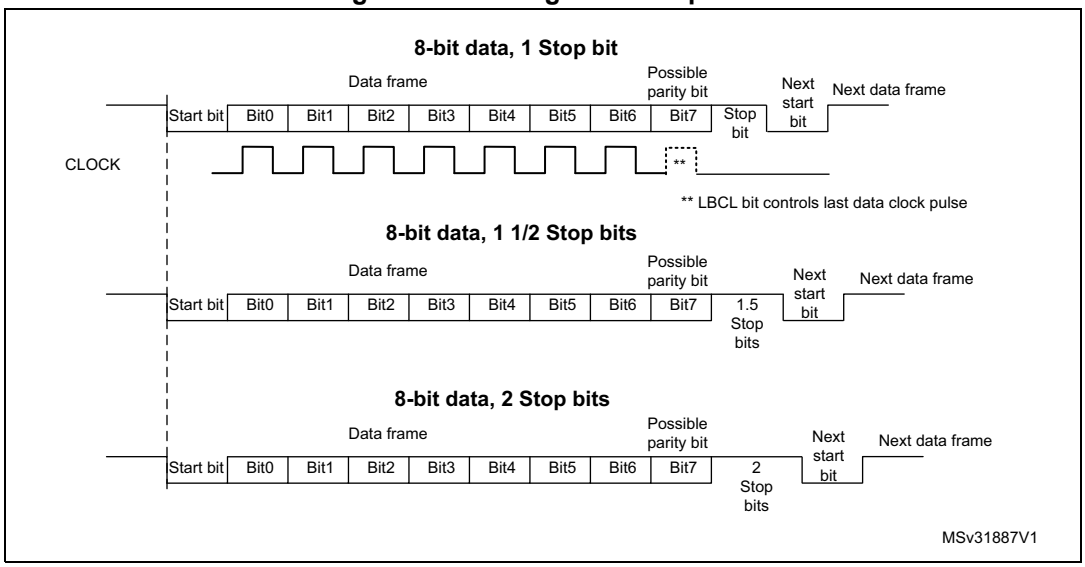
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit*****:** This is the default value of number of stop bits.
- ***2 stop bits*****:** This will be supported by normal USART, Single-wire and Modem modes.
- ***1.5 stop bits*****:** To be used in Smartcard mode.
- ***0.5 stop bit***: To be used when receiving data in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Figure 356*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 356. Configurable stop bits



Character transmission procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

Clearing the TXE bit is always performed by a write to the transmit data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from the USART_TDR register to the shift register and the data transmission has started.
- The USART_TDR register is empty.
- The next data can be written in the USART_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

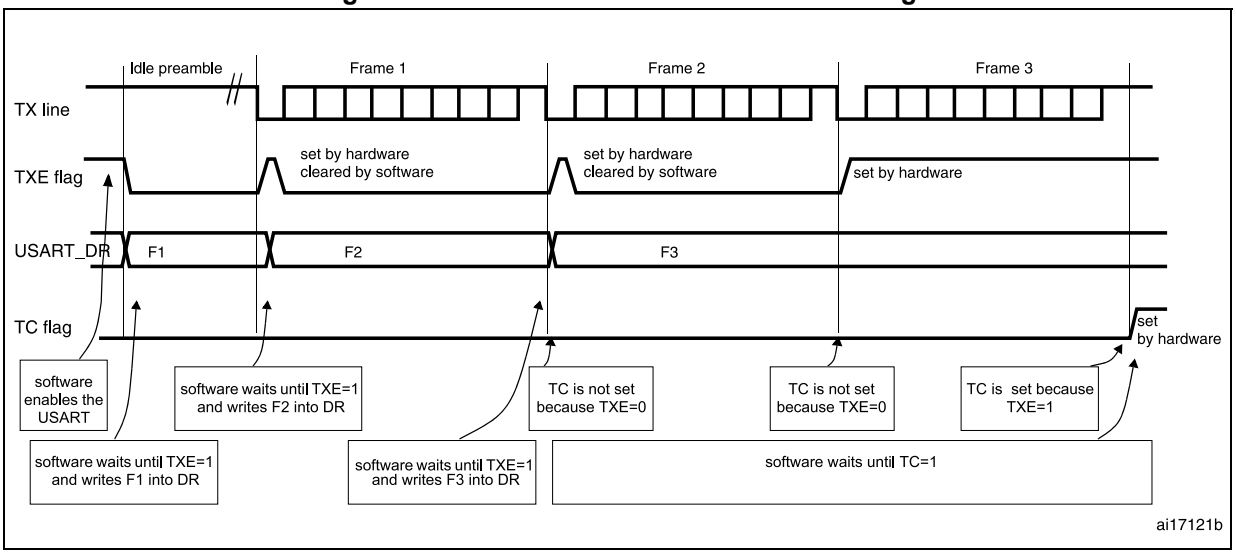
When a transmission is taking place, a write instruction to the USART_TDR register stores the data in the TDR register; next, the data is copied in the shift register at the end of the currently ongoing transmission.

When no transmission is taking place, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data in the USART_TDR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 357: TC/TXE behavior when transmitting*).

Figure 357. TC/TXE behavior when transmitting



Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 355*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

28.5.3 USART receiver

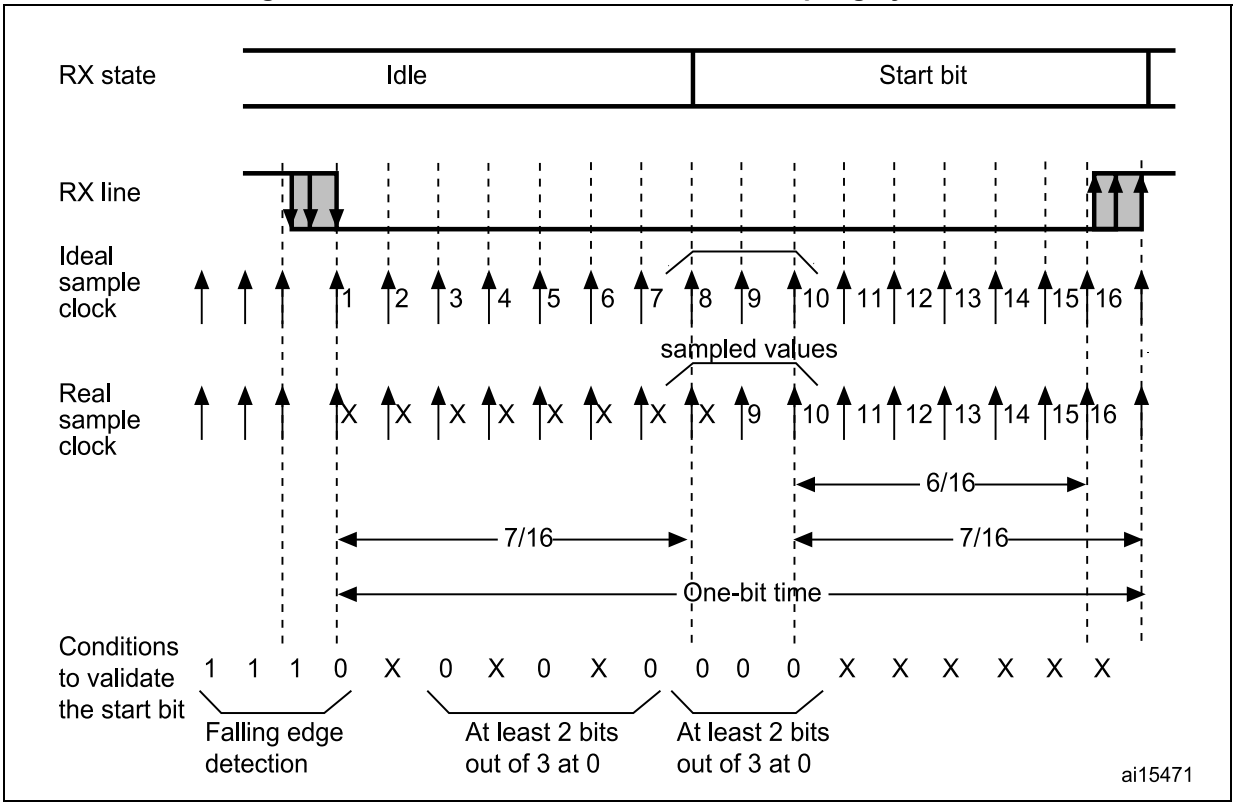
The USART can receive data words of either 7, 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 358. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated (RXNE flag set, interrupt generated if RXNEIE=1) but the NF noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither conditions a. or b. are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data shifts in least significant bit first (default configuration) through the RX pin. In this mode, the USART_RDR register consists of a buffer (RDR) between the internal bus and the receive shift register.

Character reception procedure

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- The RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXNEIE bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception. PE flag can also be set with RXNE.
- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read of the Receive data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.
- The RDR content will not be lost. The previous data is available when a read to USART_RDR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. There are two possibilities:

- if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,

- if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

Selecting the clock source and the proper oversampling method

The choice of the clock source is done through the Clock Control system (see Section Reset and clock control (RCC))). The clock source must be chosen before enabling the USART (by setting the UE bit).

The choice of the clock source must be done according to two criteria:

- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is f_{CK}.

When the dual clock domain with the wakeup from Stop mode is supported, the clock source can be one of the following sources: PCLK (default), LSE, HSI or SYSCLK. Otherwise, the USART clock source is PCLK.

Choosing LSE or HSI as clock source may allow the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the USART wakes up the MCU, when needed, in order to transfer the received data by software reading the USART_RDR register or by DMA.

For the other clock sources, the system must be active in order to allow USART communication.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques for data recovery by discriminating between valid incoming data and noise. This allows a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register and can be either 16 or 8 times the baud rate clock (*Figure 359* and *Figure 360*).

Depending on the application:

- Select oversampling by 8 (OVER8=1) to achieve higher speed (up to $f_{CK}/8$). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 28.5.5: Tolerance of the USART receiver to clock deviation on page 966*)
- Select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum $f_{CK}/16$ where f_{CK} is the clock source frequency.

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. There are two options:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- A single sample in the center of the received bit

  Depending on the application:

  – select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 136*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver's tolerance to clock deviations (see *Section 28.5.5: Tolerance of the USART receiver to clock deviation on page 966*). In this case the NF bit will never be set.

When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The NF bit is reset by setting NFCF bit in ICR register.

Note: Oversampling by 8 is not available in LIN, Smartcard and IrDA modes. In those modes, the OVER8 bit is forced to '0' by hardware.

Figure 359. Data sampling when oversampling by 16

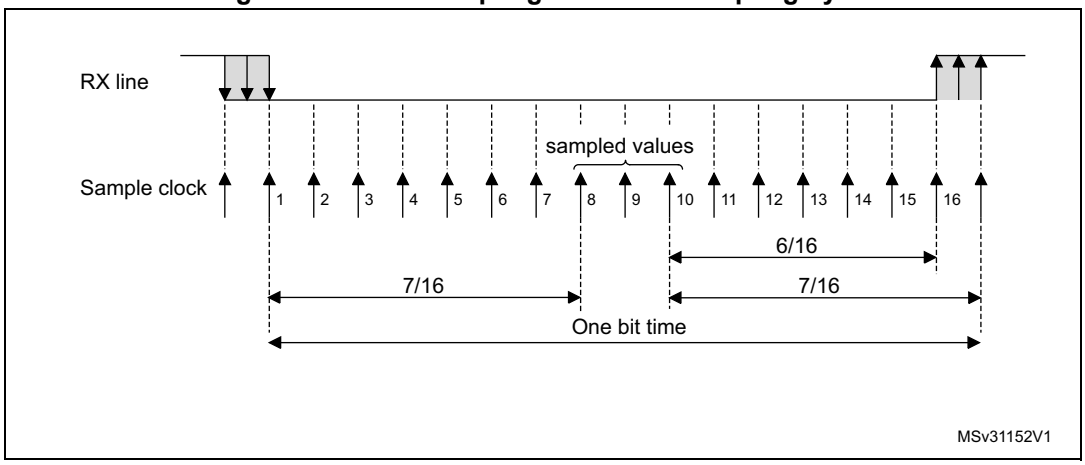


Figure 360. Data sampling when oversampling by 8

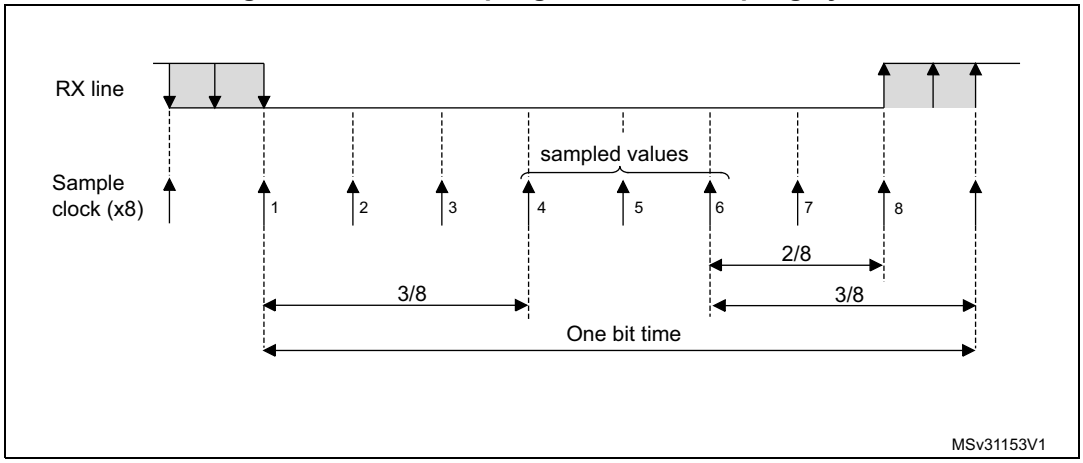


Table 136. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- ***0.5 stop bit (reception in Smartcard mode)****: No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.*
- ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
- ***1.5 stop bits (Smartcard mode)***: When transmitting in Smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) and the stop bit is checked to test if the smartcard has detected a parity error. In the event of a parity error, the smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 stop bits. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bits can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to *Section 28.5.13: USART Smartcard mode on page 977* for more details.
- ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

28.5.4 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the USART_BRR register.

Equation 1: Baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{CK}}{\text{USARTDIV}}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{CK}}{\text{USARTDIV}}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

In Smartcard, LIN and IrDA modes, only Oversampling by 16 is supported:

$$\text{Tx/Rx baud} = \frac{f_{CK}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 or 8, USARTDIV must be greater than or equal to 16d.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with f_{CK} = 8 MHz.

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 833d = 0341h
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h << 1 = 1h
  BRR = 0x681

Example 2

To obtain 921.6 Kbaud with f_{CK} = 48 MHz.

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 52d = 34h
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (104d = 68h)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 8h >> 1 = 4h
  BRR = 0x64

Table 137. Error calculation for programmed baud rates at f_{CK} = 72MHz in both cases of oversampling by 16 or by 8(1)

Baud rate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	BRR	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	BRR	% Error
1	2.4 KBps	2.4 KBps	0x7530	0	2.4 KBps	0xEA60	0
2	9.6 KBps	9.6 KBps	0x1D4C	0	9.6 KBps	0x3A94	0
3	19.2 KBps	19.2 KBps	0xEA6	0	19.2 KBps	0x1D46	0
4	38.4 KBps	38.4 KBps	0x753	0	38.4 KBps	0xEA3	0
5	57.6 KBps	57.6 KBps	0x4E2	0	57.6 KBps	0x9C2	0
6	115.2 KBps	115.2 KBps	0x271	0	115.2 KBps	0x4E1	0
7	230.4 KBps	230.03KBps	0x139	0.16	230.4 KBps	0x270	0
8	460.8 KBps	461.54KBps	0x9C	0.16	460.06KBps	0x134	0.16
9	921.6 KBps	923.08KBps	0x4E	0.16	923.07KBps	0x96	0.16
10	2 MBps	2 MBps	0x24	0	2 MBps	0x44	0
11	3 MBps	3 MBps	0x18	0	3 MBps	0x30	0
12	4MBps	4MBps	0x12	0	4MBps	0x22	0
13	5MBps	N.A	N.A	N.A	4965.51KBps	0x16	0.69
14	6MBps	N.A	N.A	N.A	6MBps	0x14	0
15	7MBps	N.A	N.A	N.A	6857.14KBps	0x12	2
16	9MBps	N.A	N.A	N.A	9MBps	0x10	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

28.5.5 Tolerance of the USART receiver to clock deviation

The asynchronous receiver of the USART works correctly only if the total clock system deviation is less than the tolerance of the USART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver's tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from Stop mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between:

1. The detection of start bit falling edge
2. The instant when clock (requested by the peripheral) is ready and reaching the peripheral and regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 138* and *Table 138* depending on the following choices:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 138. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 139. Tolerance of the USART receiver when BRR [3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 138 and Table 139 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit durations when M bits = 00 (11-bit durations when M bits =01 or 9- bit durations when M bits = 10).

28.5.6 USART auto baud rate detection

The USART is able to detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed (when oversampling by 16, the baud rate is between $f_{CK}/65535$ and $f_{CK}/16$. when oversampling by 8, the baud rate is between $f_{CK}/65535$ and $f_{CK}/8$).

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are various modes based on different character patterns.

They can be chosen through the ABRMOD[1:0] field in the USART_CR2 register. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are:

- **Mode 0**: Any character starting with a bit at 1. In this case the USART measures the duration of the Start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern. In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode). In this case, the baud rate is updated first at the end of the

start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit 0 to bit 6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3**: A 0x55 character frame. In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit 6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6.

  In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The RXNE interrupt will signal the end of the operation.

At any later time, the auto baud rate detection may be relaunched by resetting the ABRF flag (by writing a 0).

Note: If the USART is disabled (UE=0) during an auto baud rate operation, the BRR value may be corrupted.

28.5.7 Multiprocessor communication using USART

In multiprocessor communication, the following bits are to be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL, IREN and SCEN bits in the USART_CR3 register.

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the USART_CR1 register.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

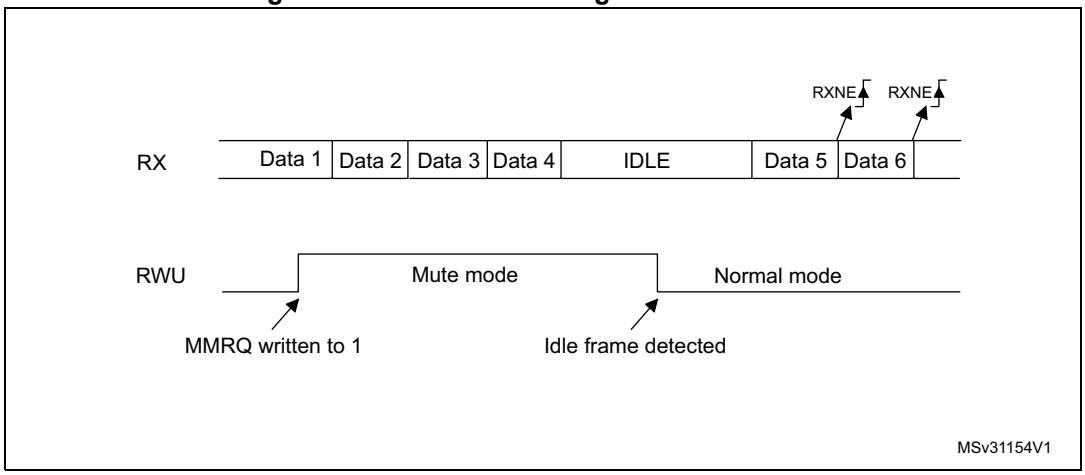
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 361*.

Figure 361. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode will not be entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

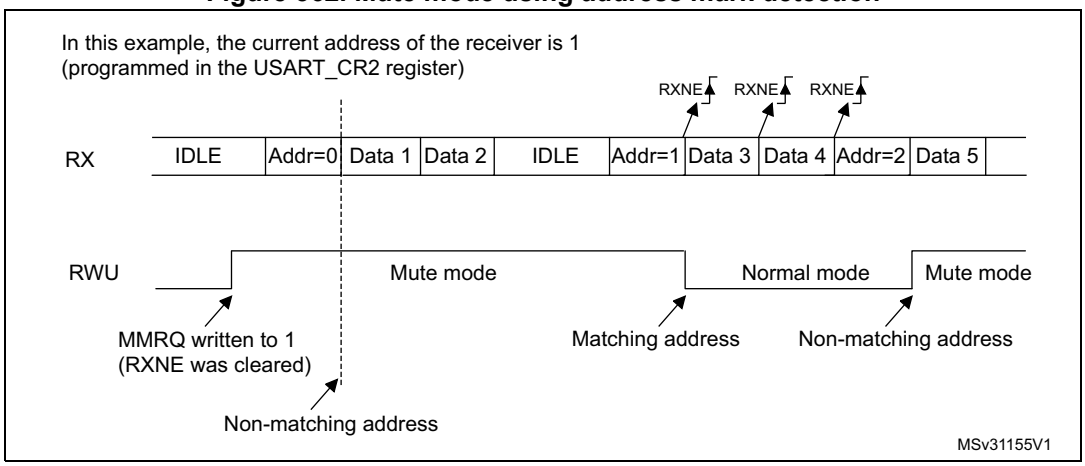
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode.

The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

An example of mute mode behavior using address mark detection is given in *Figure 362*.

Figure 362. Mute mode using address mark detection



28.5.8 Modbus communication using USART

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half duplex, block transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit duration) must be programmed in the RTO register. when the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

28.5.9 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 140*.

Table 140. Frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8-bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data | PB | STB |
10	0	| SB | 7-bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (9th, 8th or 7th, depending on the M bits value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bits values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bits values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

28.5.10 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Please refer to *Section 28.4: USART implementation on page 950*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure explained in *Section 28.5.2: USART transmitter* has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bits to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0’ bits as a break character. Then 2 bits of value ‘1’ are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 363: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 973*.

Examples of break frames are given on *Figure 364: Break detection in LIN mode vs. Framing error detection on page 974*.

Figure 363. Break detection in LIN mode (11-bit break length - LBDL bit is set)

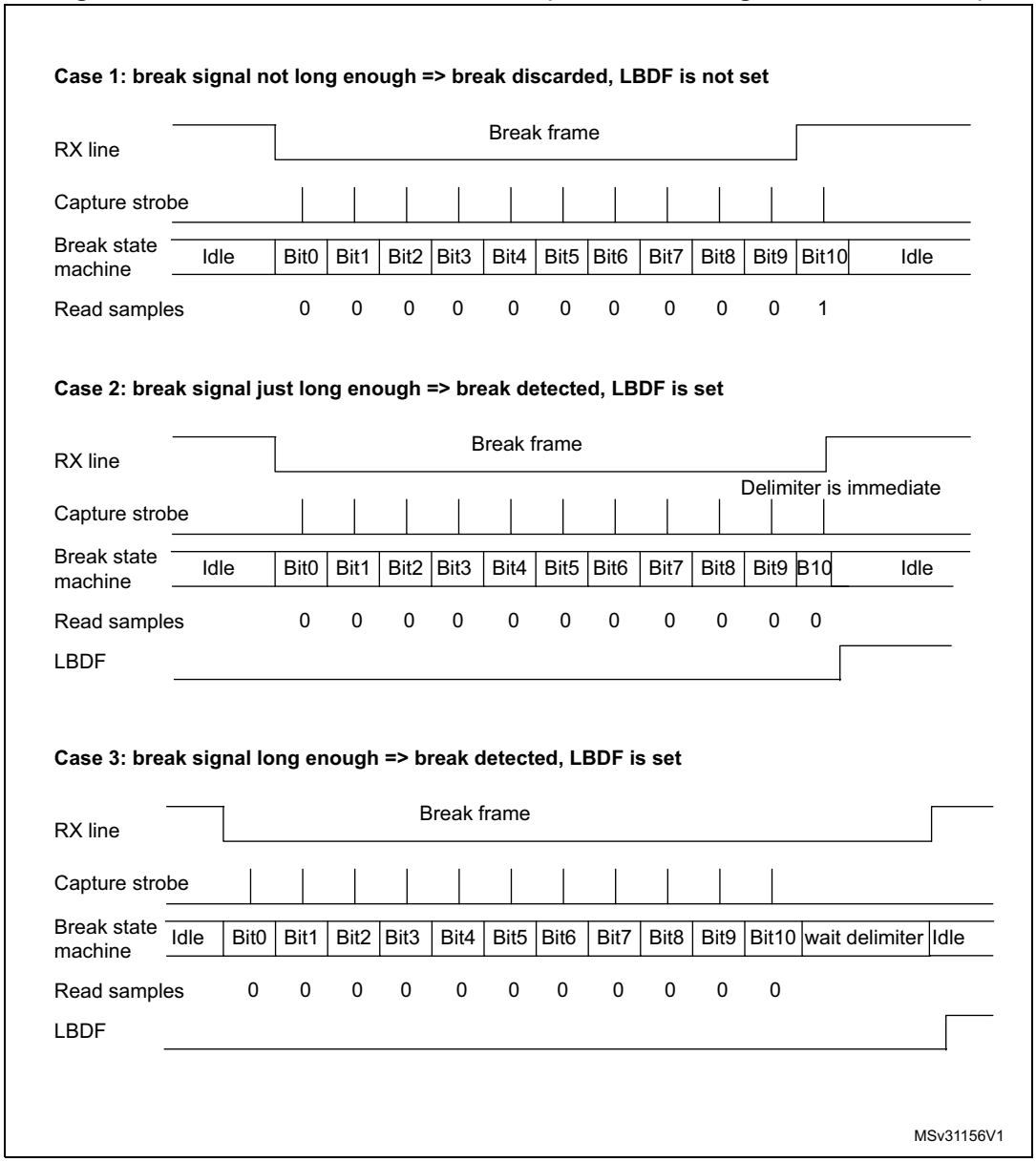
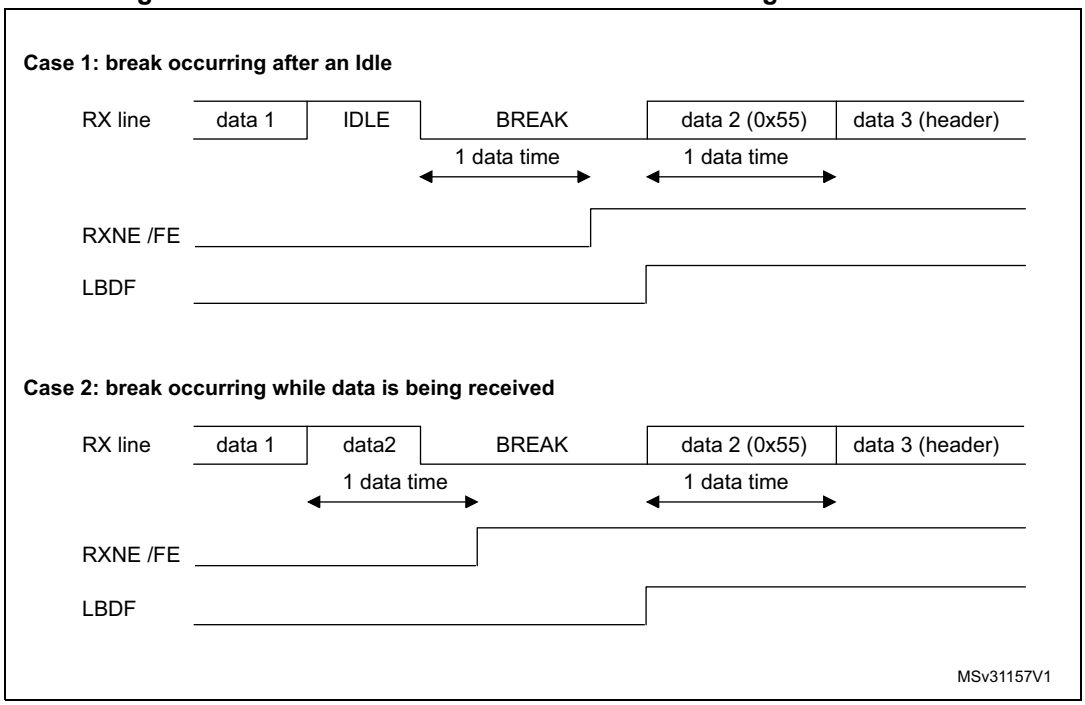


Figure 364. Break detection in LIN mode vs. Framing error detection



28.5.11 USART synchronous mode

The synchronous mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 365*, *Figure 366* and *Figure 367*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit duration).

Note: The CK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data is being transmitted (the data register USART_TDR written). This means that it is not possible to receive synchronous data without transmitting data.

The LBCL, CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

Figure 365. USART example of synchronous transmission

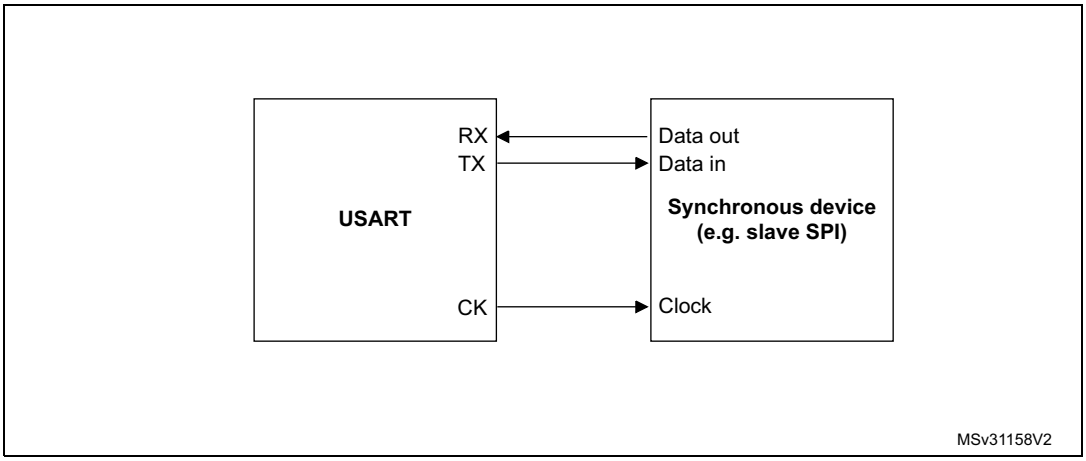


Figure 366. USART data clock timing diagram (M bits = 00)

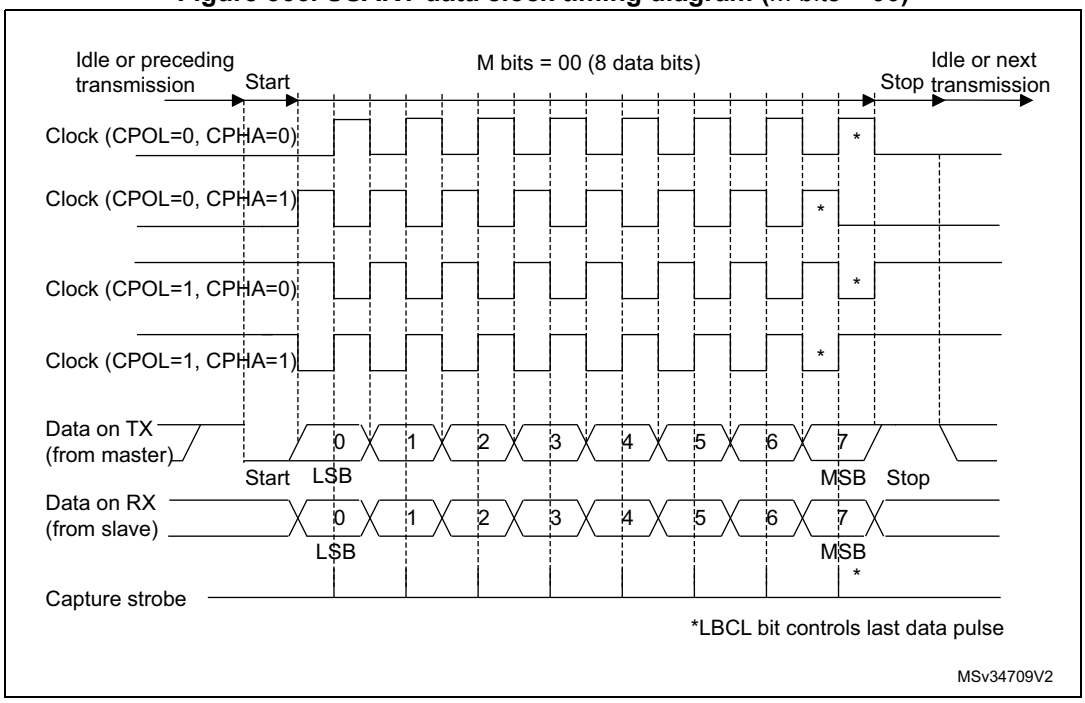


Figure 367. USART data clock timing diagram (M bits = 01)

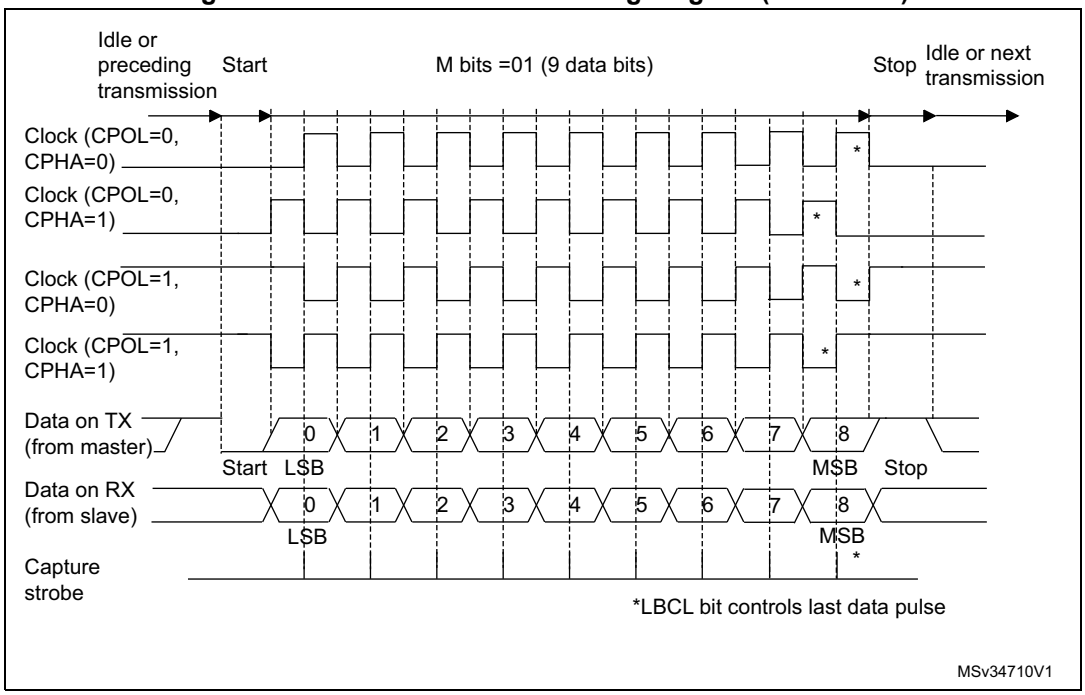
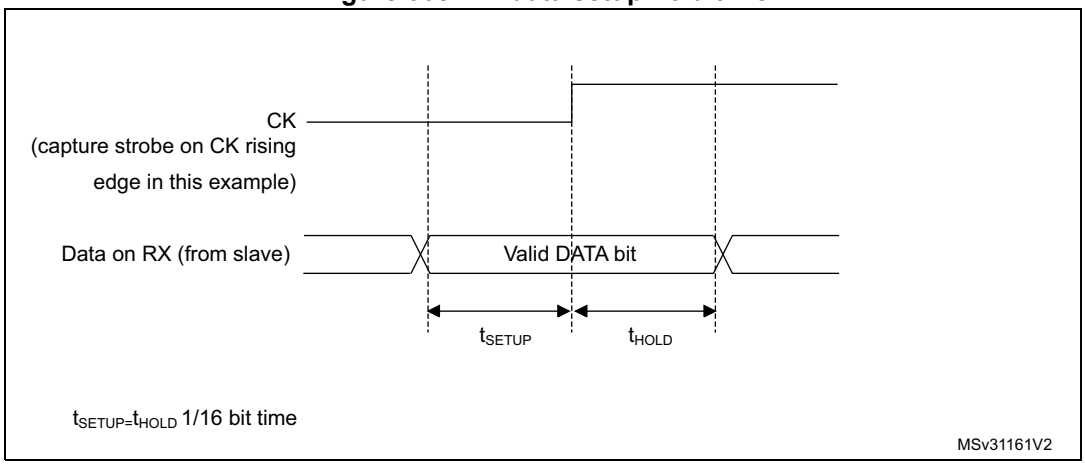


Figure 368. RX data setup/hold time



Note: The function of CK is different in Smartcard mode. Refer to Section 28.5.13: USART Smartcard mode for more details.

28.5.12 USART Single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

28.5.13 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Please refer to *Section 28.4: USART implementation on page 950*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous protocol for smartcards as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

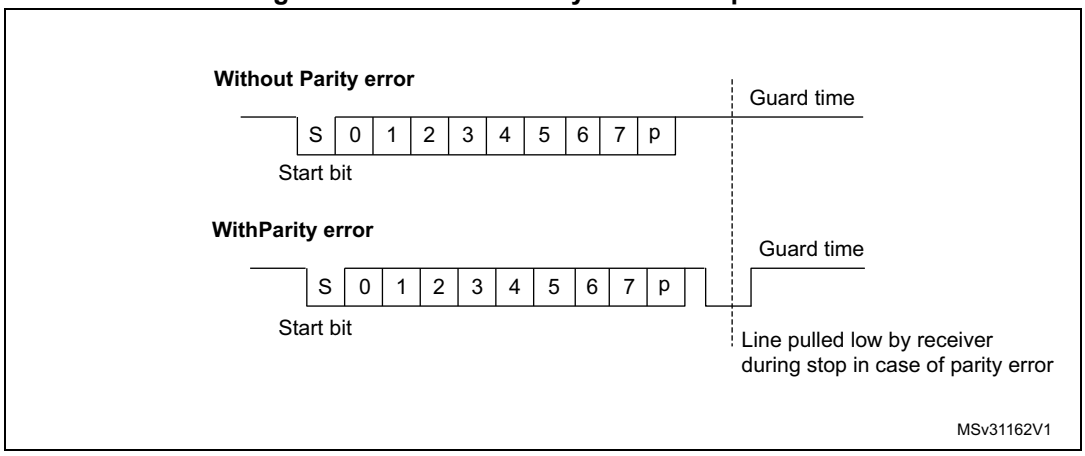
The USART should be configured as:

- 8 bits plus parity: where word length is set to 8 bits and PCE=1 in the USART_CR1 register
- 1.5 stop bits: where STOP=11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for receiving.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 369 shows examples of what can be seen on the data line with and without parity error.

Figure 369. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol. The number of retries is programmed in the SCARCNT bit field. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit can be set using the TXFRQ bit in the USART_RQR register.
- Smartcard auto-retry in transmission: a delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard-time). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bits period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bit field, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.

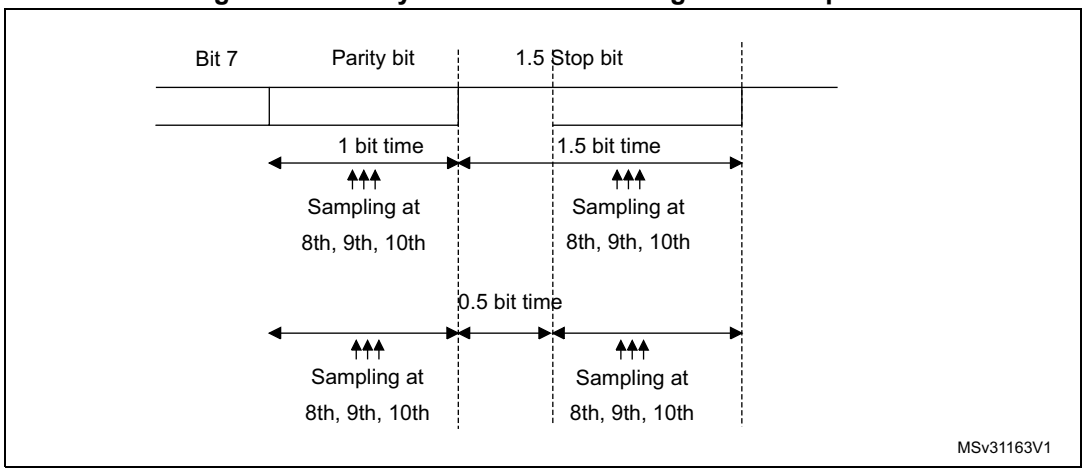
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: A break character is not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 370 details how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 370. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_GTPR. CK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission is deactivated, by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must enable the receiver Timeout feature by setting the RTOEN bit in the USART_CR2 register and program the RTO bits field in the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, the RTOF flag will be set and a timeout interrupt will be generated (if RTOIE bit in the USART_CR1 register is set). If the first character is received before the expiration of the period, it is signaled by the RXNE interrupt.

Note: The RXNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE interrupt), the RTO bit fields in the RTOR register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: The RTO counter starts counting:

- From the end of the stop bit in case STOP = 00.

- From the end of the second stop bit in case of STOP = 10.

- 1 bit duration after the beginning of the STOP bit in case STOP = 11.

- From the beginning of the STOP bit in case STOP = 01.

As in the Smartcard protocol definition, the BWT/CWT values are defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting (TXE=0). The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. when using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). with this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilogue bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). when decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). when decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character will be ‘03’ and the parity will be odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card didn’t answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it will be correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

28.5.14 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Please refer to *Section 28.4: USART implementation on page 950*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 371*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the IrDA decoder), data on the TX from the USART to IrDA is not encoded. while receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 372*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.

- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to "1 stop bit".

IrDA low-power mode

Transmitter

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz.

Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1 PSC period. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 371. IrDA SIR ENDEC- block diagram

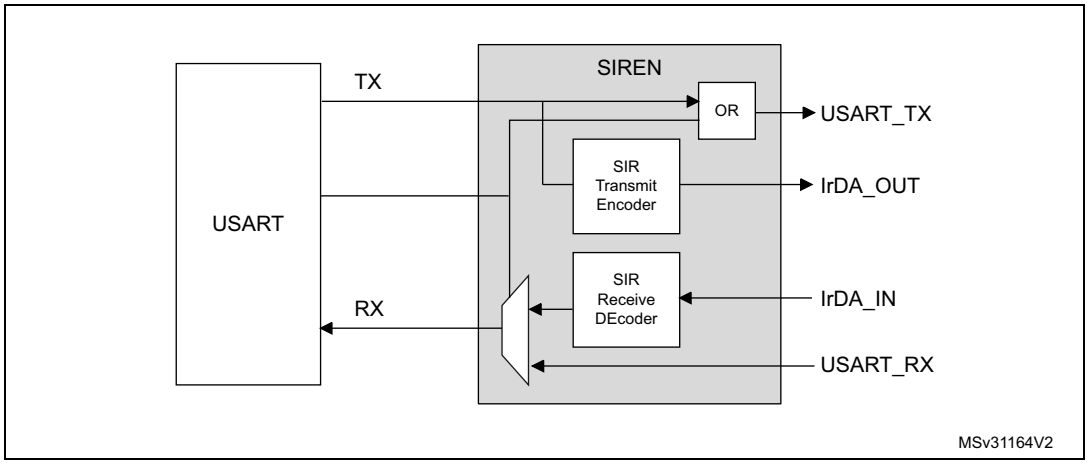
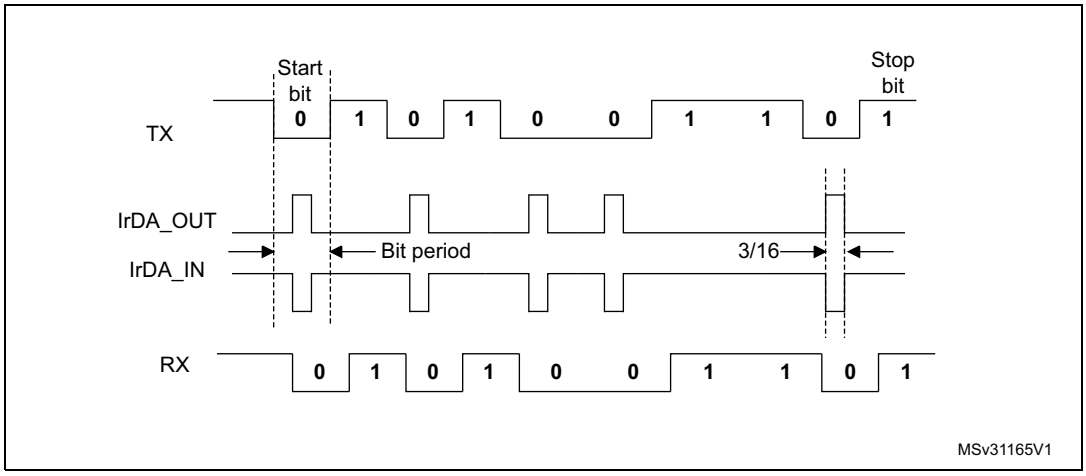


Figure 372. IrDA data modulation (3/16) -Normal Mode



28.5.15 USART continuous communication in DMA mode

The USART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Please refer to Section 28.4: USART implementation on page 950 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 28.5.2: USART transmitter or Section 28.5.3: USART receiver. To perform continuous communication, the user can clear the TXE/ RXNE flags In the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to *Section 11: Direct memory access controller (DMA) on page 170*) to the USART_TDR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

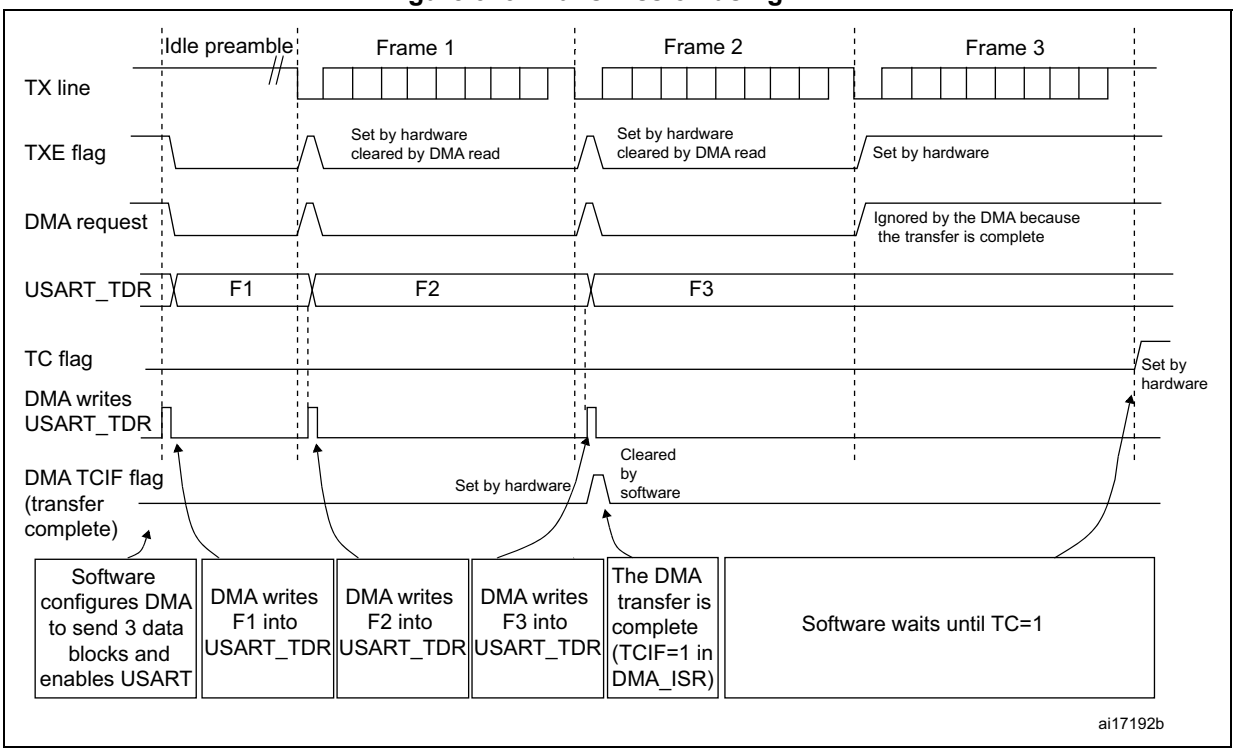
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART

communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Stop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 373. Transmission using DMA



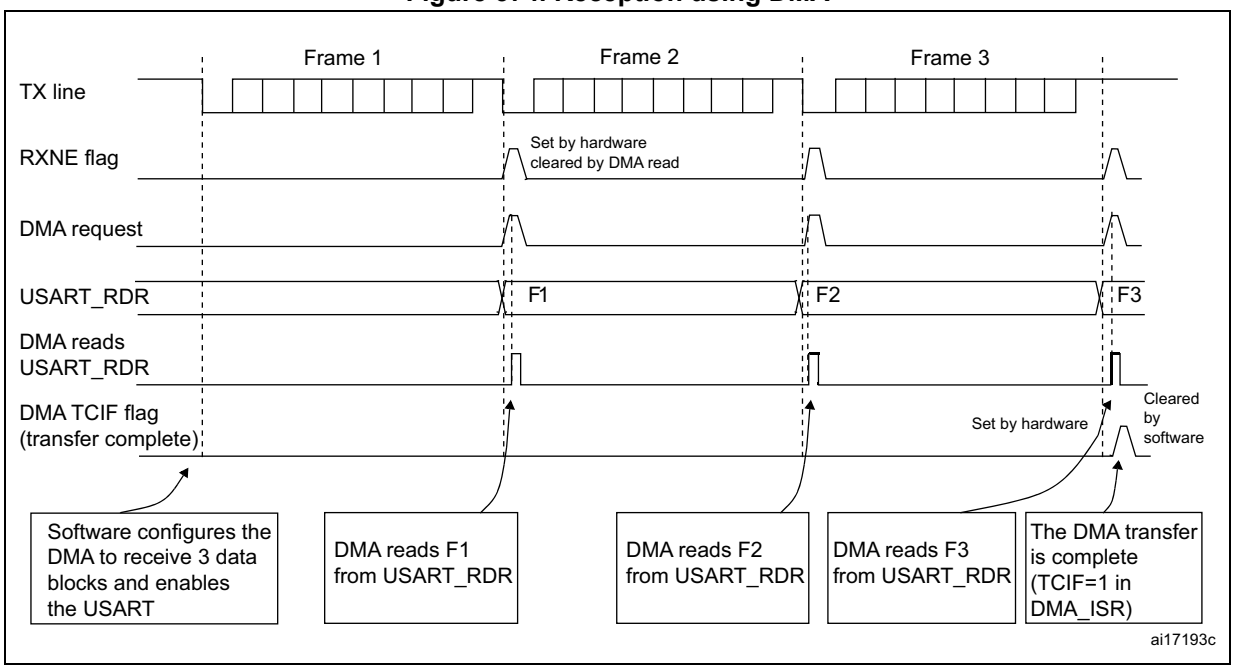
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_RDR register to a SRAM area configured using the DMA peripheral (refer to *Section 11: Direct memory access controller (DMA) on page 170*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 374. Reception using DMA



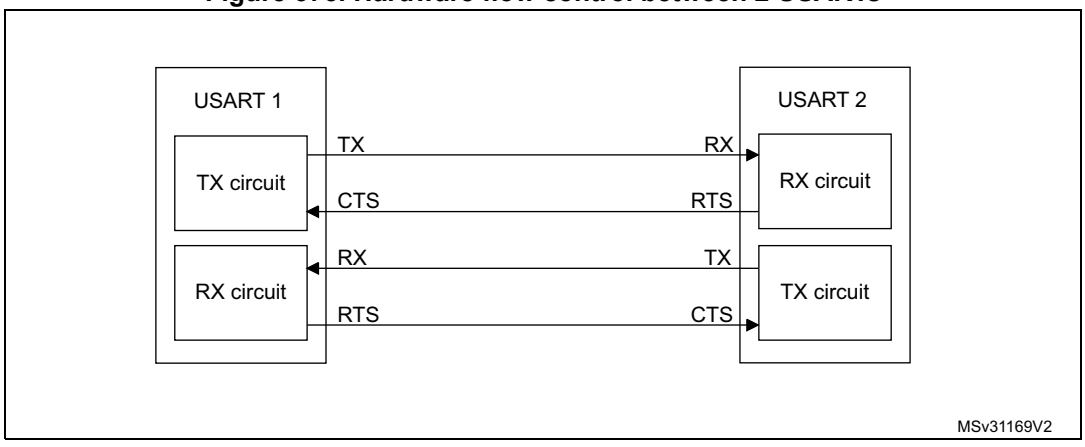
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

28.5.16 RS232 hardware flow control and RS485 driver enable using USART

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 375* shows how to connect 2 devices in this mode:

Figure 375. Hardware flow control between 2 USARTs

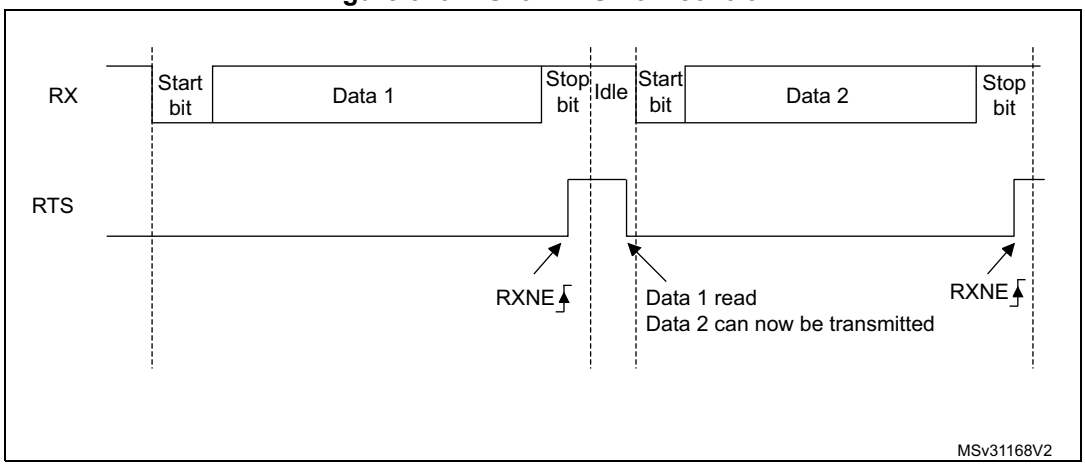


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the USART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is de-asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 376* shows an example of communication with RTS flow control enabled.

Figure 376. RS232 RTS flow control

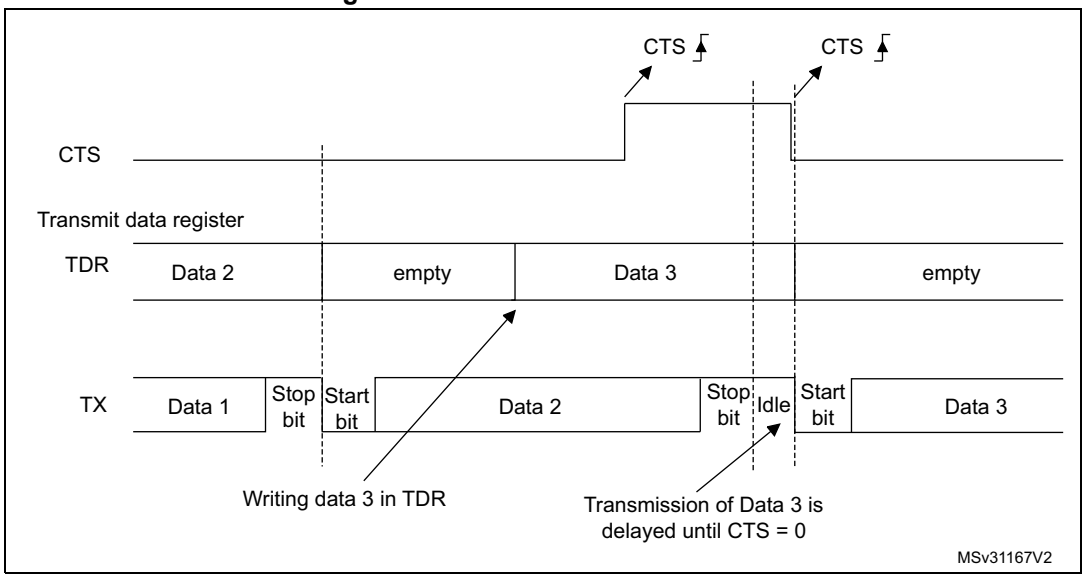


RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. when CTS is de-asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 377* shows an example of communication with CTS flow control enabled.

Figure 377. RS232 CTS flow control



Note: For correct behavior, CTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit duration, depending on the oversampling rate).

28.5.17 Wakeup from Stop mode using USART

The USART is able to wake up the MCU from Stopmode when the UESM bit is set and the USART clock is set to HSI or LSE (refer to Section Reset and clock control (RCC)).

- USART source clock is HSI

  If during stop mode the HSI clock is switched OFF, when a falling edge on the USART receive line is detected, the USART interface requests the HSI clock to be switched ON. The HSI clock is then used for the frame reception.

  – If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

  – If the wakeup event is not verified, the HSI clock is switched OFF again, the MCU is not waken up and stays in low-power mode and the clock request is released.

- USART source clock is LSE

  Same principle as described in case of USART source clock is HSI with the difference that the LSE is ON in stop mode, but the LSE clock is not propagated to USART if the

USART is not requesting it. The LSE clock is not OFF but there is a clock gating to avoid useless consumption.

The MCU wakeup from Stop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Stop mode.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Stop mode, the UESM bit in the USART_CR1 control register must be set prior to entering Stop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

Note: Before entering Stop mode, the user must ensure that the USART is not performing a transfer. BUSY flag cannot ensure that Stop mode is never entered during a running reception.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in Stop or in an active mode.

When entering Stop mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the USART is actually enabled.

When DMA is used for reception, it must be disabled before entering Stop mode and re-enabled upon exit from Stop mode.

The wakeup from Stop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

Using Mute mode with Stop mode

If the USART is put into Mute mode before entering Stop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Stop mode.
- If the wakeup from Mute mode on address match is used, then the source of wake-up from Stop mode must also be the address match. If the RXNE flag is set when entering the Stop mode, the interface will remain in mute mode upon address match and wake up from Stop.
- If the USART is configured to wake up the MCU from Stop mode on START bit detection, the WUF flag is set, but the RXNE flag is not set.

Determining the maximum USART baud rate allowing to wakeup correctly from Stop mode when the USART clock source is the HSI clock

The maximum baud rate allowing to wakeup correctly from stop mode depends on:

- the parameter $t_{WUUSART}$ provided in the device datasheet
- the USART receiver tolerance provided in the *Section 28.5.5: Tolerance of the USART receiver to clock deviation*.

Let us take this example: OVER8 = 0, M bits = 10, ONEBIT = 1, BRR [3:0] = 0000.

In these conditions, according to *Table 138: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance is 4.86 %.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver's tolerance

DWU max = $t_{WUUSART}$ / (9 x Tbit Min)

Tbit Min = $t_{WUUSART}$ / (9 x DWU max)

If we consider an ideal case where the parameters DTRA, DQUANT, DREC and DTCL are at 0%, the DWU max is 4.86 %. In reality, we need to consider at least the HSI inaccuracy.

Let us consider HSI inaccuracy = 1 %, $t_{WUUSART}$ = 3.125 µs (in case of wakeup from stop mode, with the main regulator in Run mode).

DWU max = 4.86 % - 1 % = 3.86 %

Tbit min = 3.125 µs / (9 x 3.86 %) = 9 µs

In these conditions, the maximum baud rate allowing to wakeup correctly from Stop mode is 1/9 µs = 111 Kbaud.

28.6 USART low-power modes

Table 141. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupt causes the device to exit Sleep mode.
Stop	The USART is able to wake up the MCU from Stop mode when the UESM bit is set and the USART clock is set to HSI or LSE. The MCU wakeup from Stop mode can be done using either a standard RXNE or a WUF interrupt.
Standby	The USART is powered down and must be reinitialized when the device has exited from Standby mode.

28.7 USART interrupts

Table 142. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout	RTOF	RTOIE
End of Block	EOBF	EOBIE
Wakeup from Stop mode	WUF(1)	WUFIE

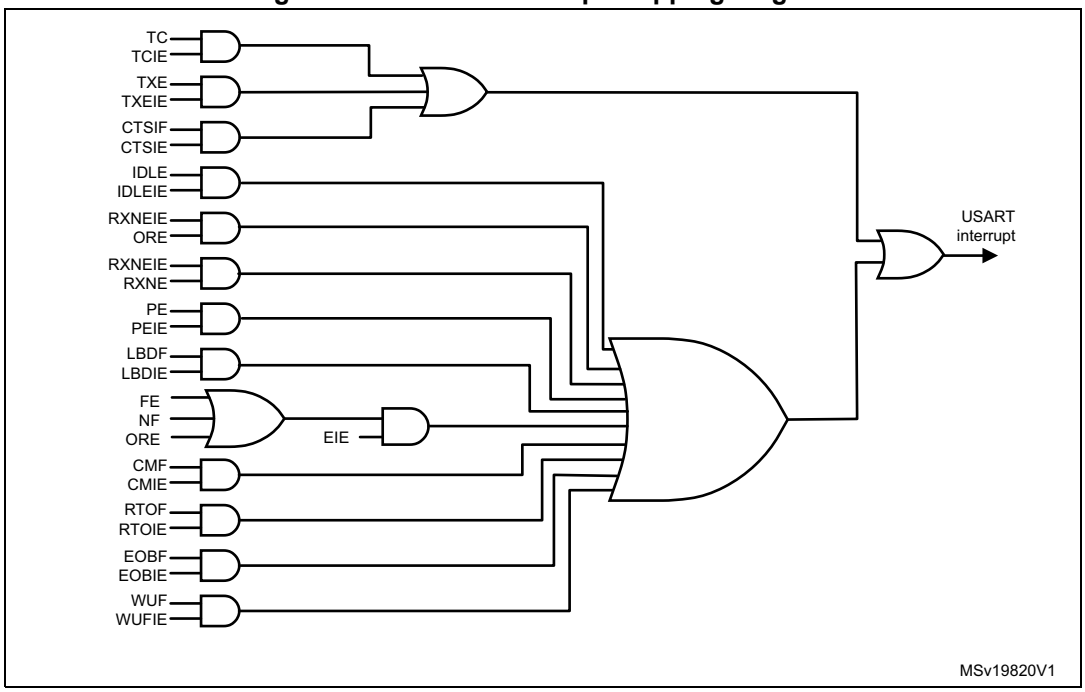
1. The WUF interrupt is active only in Stop mode.

The USART interrupt events are connected to the same interrupt vector (see *Figure 378*).

- During transmission: Transmission Complete, Clear to Send, Transmit data Register empty or Framing error (in Smartcard mode) interrupt.
- During reception: Idle Line detection, Overrun error, Receive data register not empty, Parity error, LIN break detection, Noise Flag, Framing Error, Character match, etc.

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 378. USART interrupt mapping diagram



28.8 USART registers

Refer to *Section 1.2 on page 43* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

28.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **M1**: Word length

This bit, with bit 12 (M0), determines the word length. It is set or cleared by software.
M[1:0] = 00: 1 Start bit, 8 data bits, n stop bits
M[1:0] = 01: 1 Start bit, 9 data bits, n stop bits
M[1:0] = 10: 1 Start bit, 7 data bits, n stop bits
This bit can only be written when the USART is disabled (UE=0).

Note: Not all modes are supported In 7-bit data length mode. Refer to Section 28.4: USART implementation for details.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated when the EOBF flag is set in the USART_ISR register.

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 28.4: USART implementation on page 950.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit duration, depending on the oversampling rate).
This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bits 20:16 **DEDT[4:0]**: Driver Enable de-assertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit duration, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

Note: In LIN, IrDA and modes, this bit must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the USART. when set, the USART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1), determines the word length. It is set or cleared by software. See Bit 28 (M1) description.

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever PE=1 in the USART_ISR register

Bit 7 **TXEIE**: interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever TXE=1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever TC=1 in the USART_ISR register

Bit 5 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever IDLE=1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
In Smartcard mode, when TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in Stop mode

When this bit is cleared, the USART is not able to wake up the MCU from Stop mode.

When this bit is set, the USART is able to wake up the MCU from Stop mode, provided that the USART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from Stop mode.

1: USART able to wake up the MCU from Stop mode. When this function is active, the clock source for the USART must be HSI or LSE (see Section Reset and clock control (RCC).

Note: It is recommended to set the UESM bit just before entering Stop mode and clear it on exit from Stop mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USART_ISR are set to their default values. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: In order to go into low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

28.8.2 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.
This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.
This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with address mark detection.
This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.
0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.
When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.
00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement. (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection
This bit field can only be written when ABREN = 0 or the USART is disabled (UE=0).

Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.
0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8/9) first, following the start bit.
This bit field can only be written when the USART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bit field can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the RX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another USART.

This bit field can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_RQR register, and to detect LIN Sync breaks.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit

10: 2 stop bits

11: 1.5 stop bits

This bit field can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

In order to provide correctly the CK clock to the Smartcard when CK is always available When CLKEN = 1, regardless of the UE bit value, the steps below must be respected:

- UE = 0

- SCEN = 1

- GTPR configuration (If PSC needs to be configured, it is recommended to configure PSC and GT in a single access to USART_ GTPR register).

- CLKEN= 1

- UE = 1

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 366* and *Figure 367*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bits in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE=0).
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

Note: The 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.

28.8.3 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	SCARCNT2	SCARCNT1	SCARCNT0	Res.
									rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wakeup from Stop mode interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An USART interrupt is generated whenever WUF=1 in the USART_ISR register
Note: WUFIE must be set before entering in Stop mode.
The WUF interrupt is active only in Stop mode.
If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value.

Bits 21:20 **WUS[1:0]**: Wakeup from Stop mode interrupt flag selection

This bit-field specify the event which activates the WUF (wakeup from Stop mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01:Reserved.

10: WuF active on Start bit detection

11: WUF active on RXNE.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bit-field specifies the number of retries in transmit and receive, in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE and PE bits set).

This bit field must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 28.4: USART implementation on page 950.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred (used for Smartcard mode).

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register.

This bit can only be written when the USART is disabled (UE=0).

Note: This control bit allows checking the communication flow without reading the data.

Bit 11 **ONEBIT**: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE=0).

Note: ONEBIT feature applies only to data bits, It does not apply to Start bit.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is asserted (tied to 0). If the CTS input is de-asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is de-asserted, the transmission is postponed until CTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.
0: Smartcard Mode disabled
1: Smartcard Mode enabled
This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USART_ISR register.

28.8.4 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bits 15:4 **BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

Bits 3:0 **BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

28.8.5 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit field can only be written when the USART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA Low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

In Smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bit field can only be written when the USART is disabled (UE=0).

Note: Bits [7:5] must be kept at reset value if Smartcard mode is used.

This bit field is reserved and must be kept at reset value when the Smartcard and IrDA modes are not supported. Please refer to Section 28.4: USART implementation on page 950.

28.8.6 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bit-field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE=0.

This bit-field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit-field gives the Receiver timeout value in terms of number of bit duration.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard section for more details.

In this case, the timeout measurement is done starting from the Start Bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on the fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Please refer to Section 28.4: USART implementation on page 950.

28.8.7 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

Writing 1 to this bit sets the TXE flag.

This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register.

If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Please refer to *Section 28.4: USART implementation on page 950*.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and sets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and request an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

28.8.8 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0200 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

When the wakeup from Stop mode is supported, the REACK flag can be used to verify that the USART is ready for reception before entering Stop mode.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from Stop mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE=1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in Stop mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_RQR register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character will be transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the USART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR1 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by a write to the USART_TDR register.

The TXE flag can also be cleared by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.

0: data is not transferred to the shift register

1: data is transferred to the shift register)

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 **RXNE**: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_RDR register. It is cleared by a read to the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE=1 in the USART_CR1 register.

0: data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE=1 or EIE = 1 in the USART_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multibuffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the OVRDIS bit is set in the USART_CR3 register.

Bit 2 **NF**: START bit Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multibuffer communication if the EIE bit is set.

Note: When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 28.5.5: Tolerance of the USART receiver to clock deviation on page 966).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

28.8.9 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											rc_w1			rc_w1	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
			rc_w1	rc_w1		rc_w1	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 20 **WUCF**: Wakeup from Stop mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USART_ISR register.

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Please refer to Section 28.4: USART implementation on page 950.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NCF**: Noise detected clear flag
Writing 1 to this bit clears the NF flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

28.8.10 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

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

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 354*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

28.8.11 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 354*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE=1.

28.8.12 USART register map

The table below gives the USART register map and reset values.

Table 143. USART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_CR1**	Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USART_CR2**	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0				
0x08	**USART_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS		SCARCNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_GTPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USART_RTOR**	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_RQR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0

Table 143. USART register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**USART_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	1	0	0	0	0	0	0
0x20	**USART_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
	Reset value												0			0					0	0		0	0		0		0	0	0	0	0
0x24	**USART_RDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x28	**USART_TDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X

Refer to *Section 2.2 on page 47* for the register boundary addresses.

29 Serial peripheral interface (SPI)

29.1 Introduction

The SPI interface can be used to communicate with external devices using the SPI protocol. SPI mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

29.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support

29.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 144. STM32F334xx SPI implementation

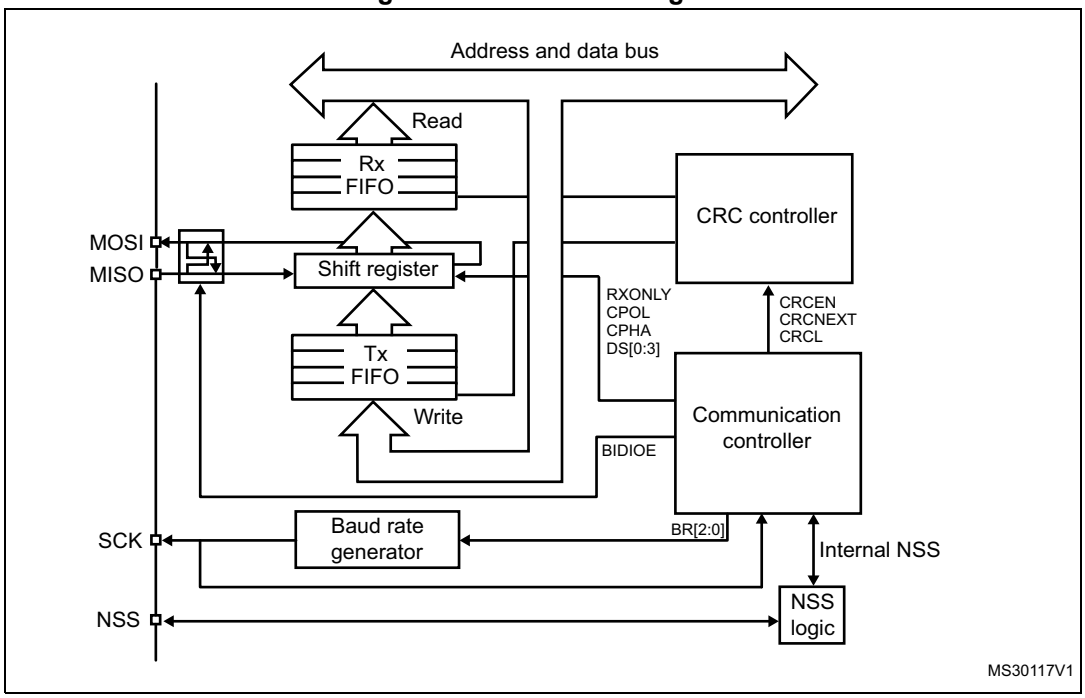
SPI Features	SPI1
Enhanced NSSP & TI modes	Yes
Hardware CRC calculation	Yes
Data size configurable	from 4 to 16-bit
Rx/Tx FIFO size	32-bit
Wakeup capability from Low-power Sleep	Yes

29.4 SPI functional description

29.4.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram *Figure 379*.

Figure 379. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 29.4.5: Slave select (NSS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

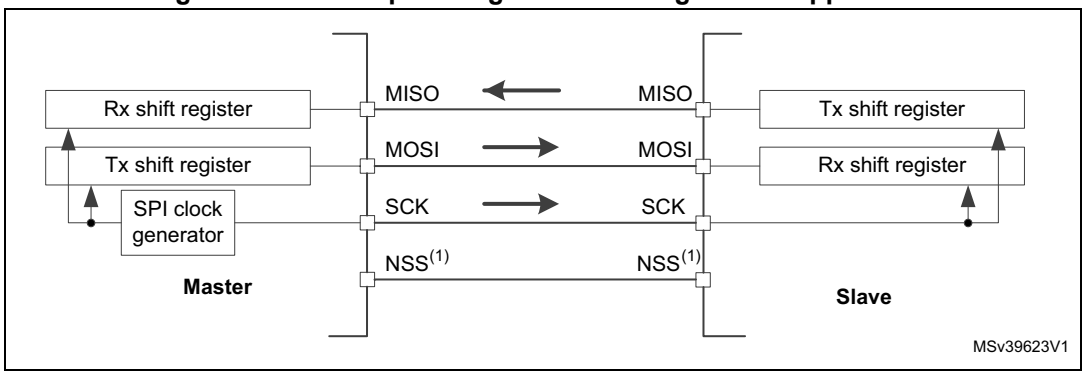
29.4.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 380. Full-duplex single master/ single slave application

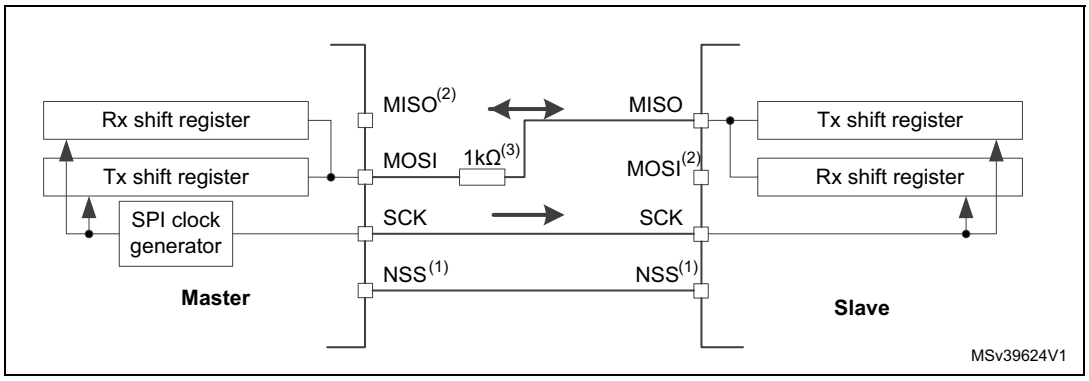


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 381. Half-duplex single master/ single slave application



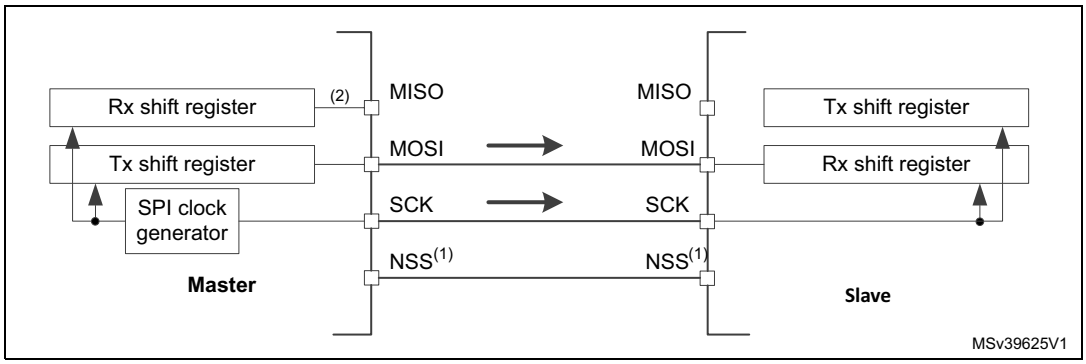
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectionnal mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1)**: The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *29.4.5: Slave select (NSS) pin management*). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 382. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



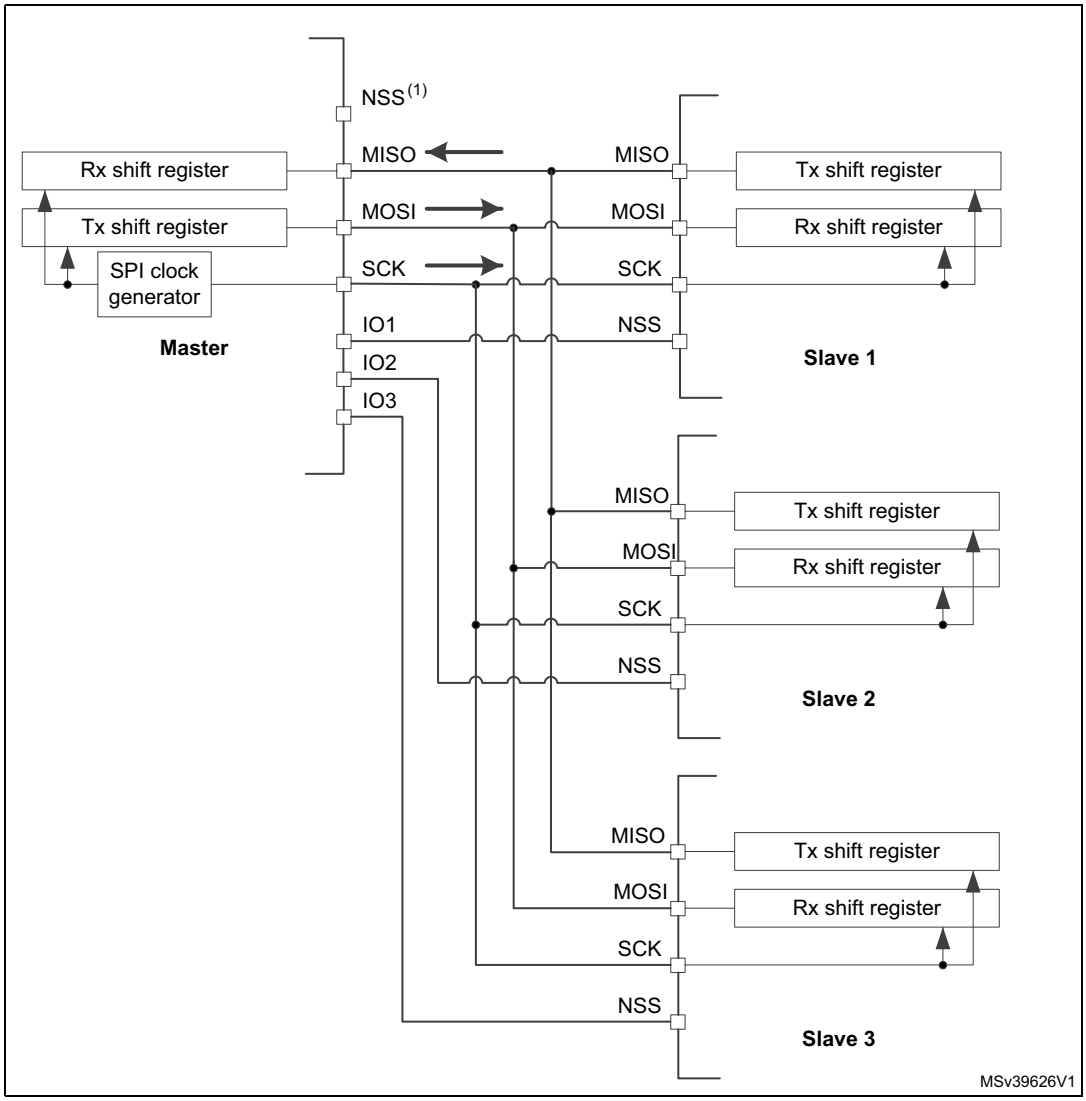
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 29.4.5: Slave select (NSS) pin management*.
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVF flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

Note: Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

29.4.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see *Figure 383.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 383. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see I/O alternate function input/output section (GPIO)).

29.4.4 Multi-master communication

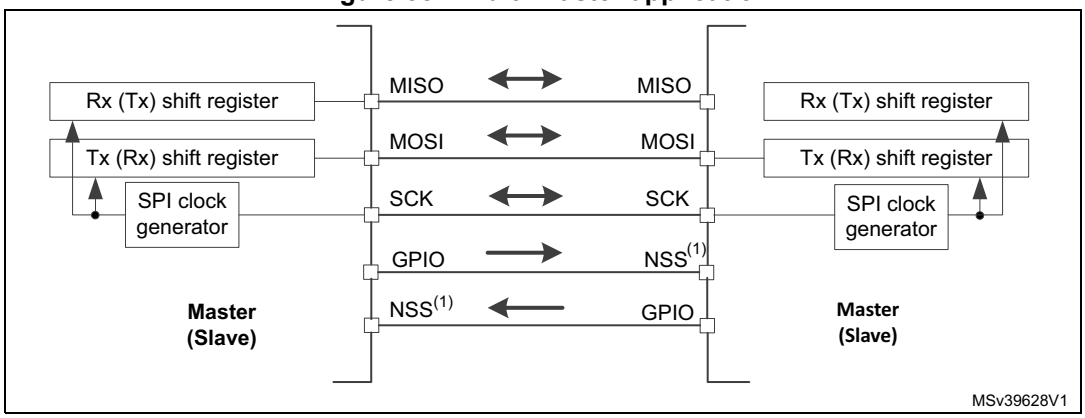
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 384. Multi-master application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

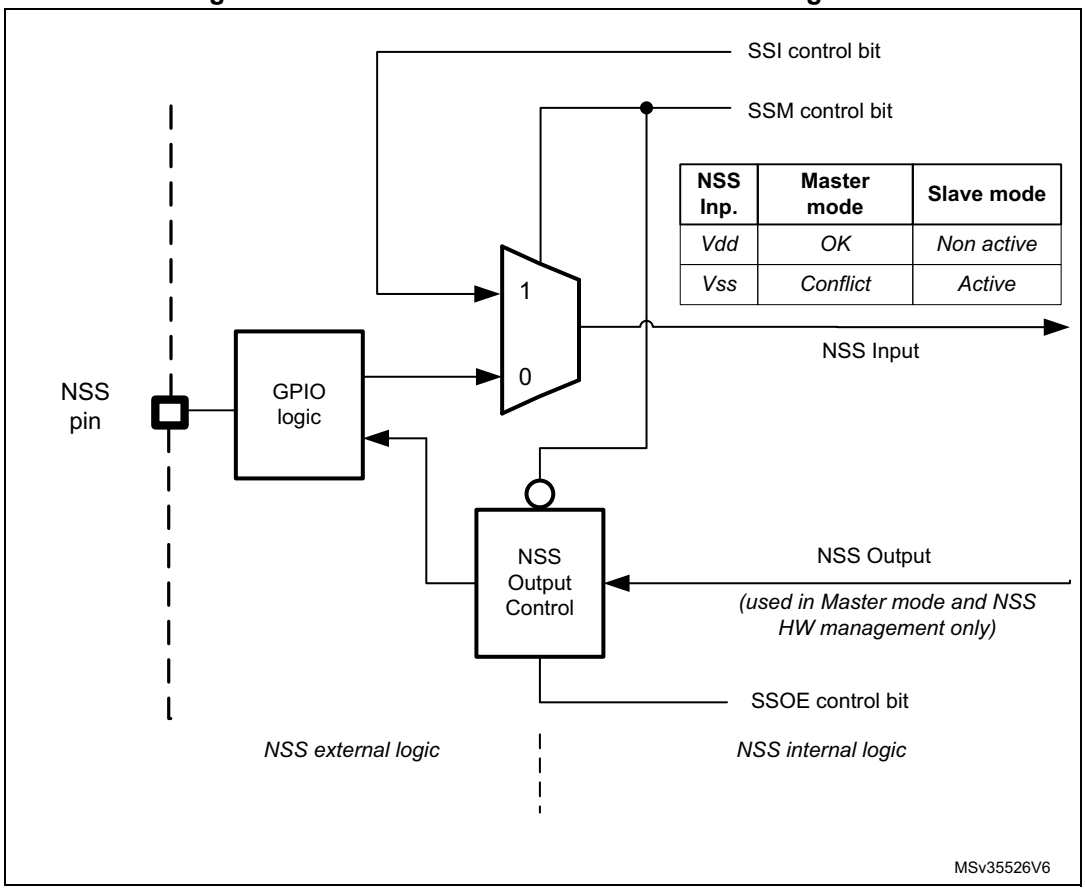
29.4.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
  - **NSS output enable (SSM=0,SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 385. Hardware/software slave select management



29.4.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

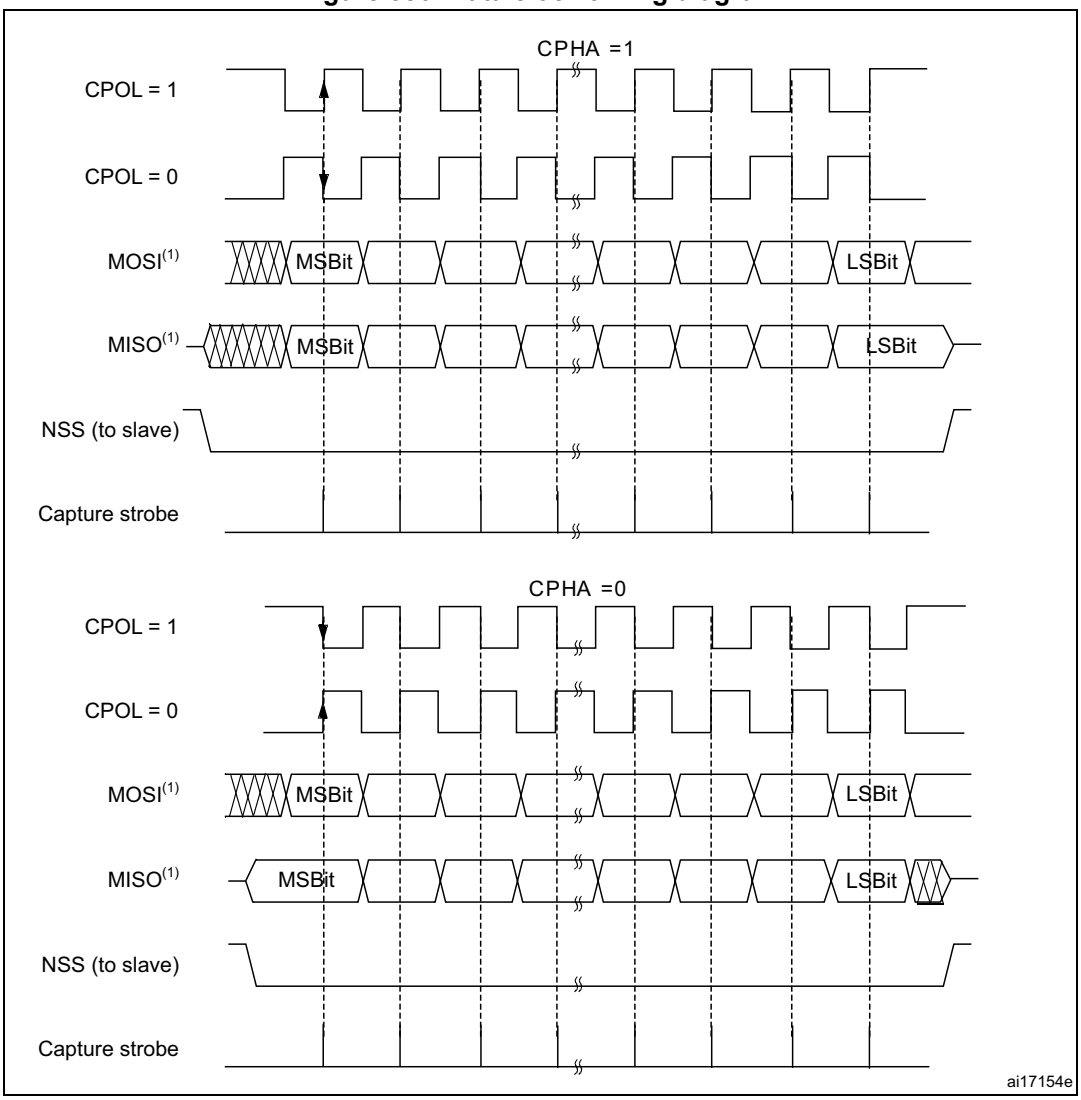
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 386, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 386. Data clock timing diagram

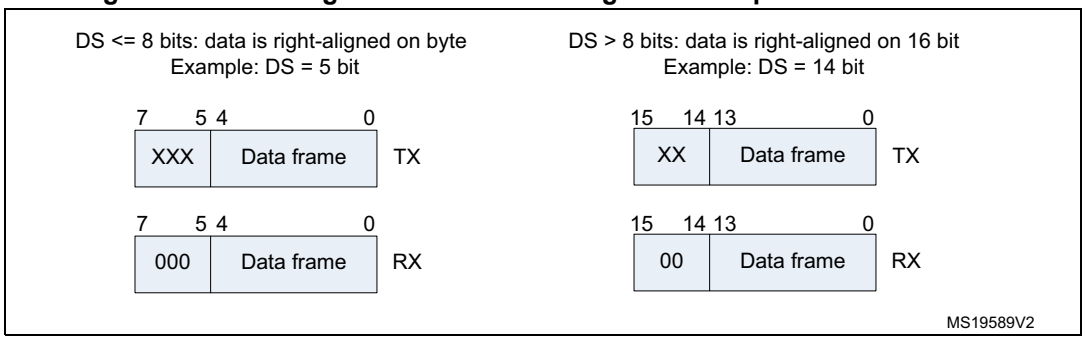


1. The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see *Figure 387*). During communication, only bits within the data frame are clocked and transferred.

Figure 387. Data alignment when data length is not equal to 8-bit or 16-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

29.4.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (Note: 2).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (Notes: 2 & 3).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (Notes: 1 & 2 & 3).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note: (1) Step is not required in slave mode.

(2) Step is not required in TI mode.

(3) Step is not required in NSSP mode.

(4) The step is not required in slave mode except slave working at TI mode

29.4.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

29.4.9 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see *Section 29.4.14: CRC calculation*).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see *Section 29.4.13: TI mode*).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See *Figure 389* through *Figure 392*.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 29.4.10: SPI status flags*). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see *Section 29.4.5: Slave select (NSS) pin management*).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to *Data packing* section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When

the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

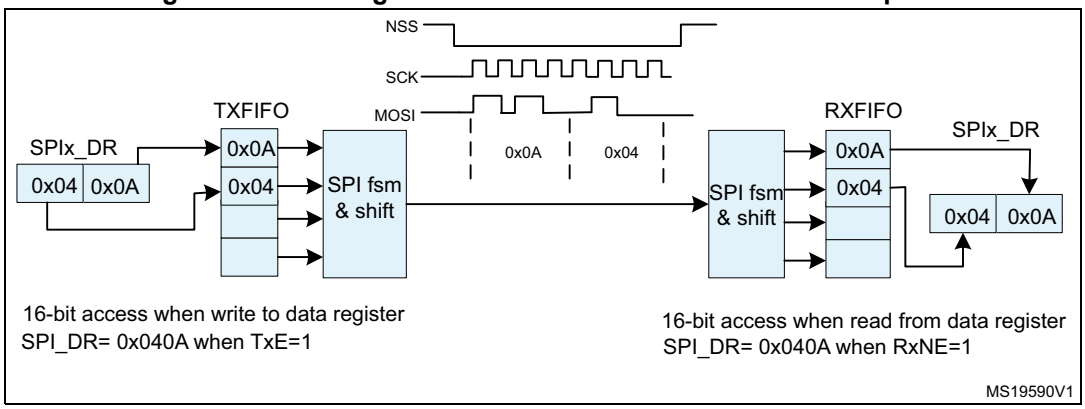
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. *Figure 388* provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

Figure 388. Packing data in FIFO for transmission and reception



Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See *Figure 389* through *Figure 392*.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to *Data packing on page 1027*.)

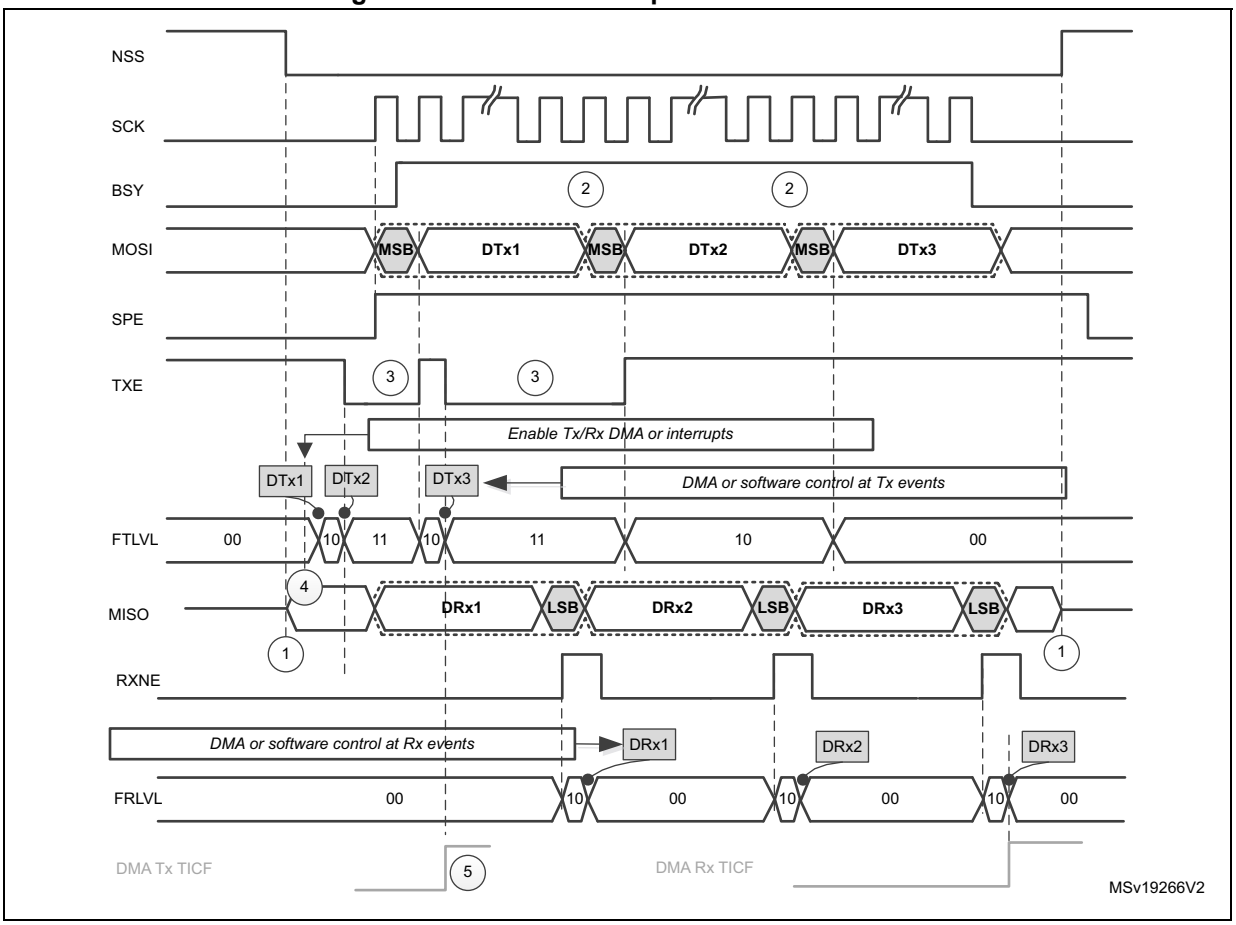
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for *Figure 389 on page 1031* through *Figure 392 on page 1034*:

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TXCRCR and SPIx_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 389. Master full-duplex communication



Assumptions for master full-duplex communication example:

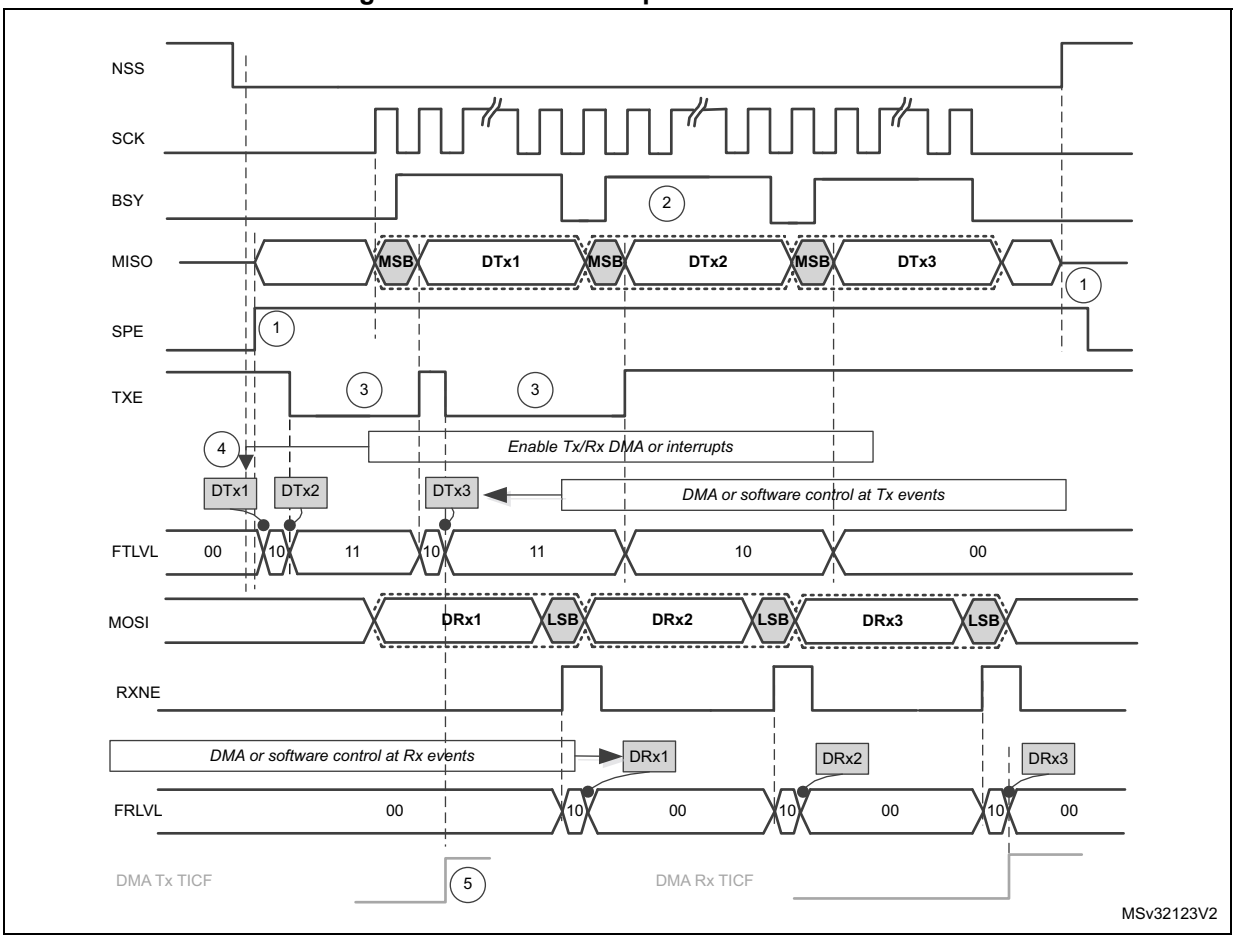
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 1030* for details about common assumptions and notes.

Figure 390. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

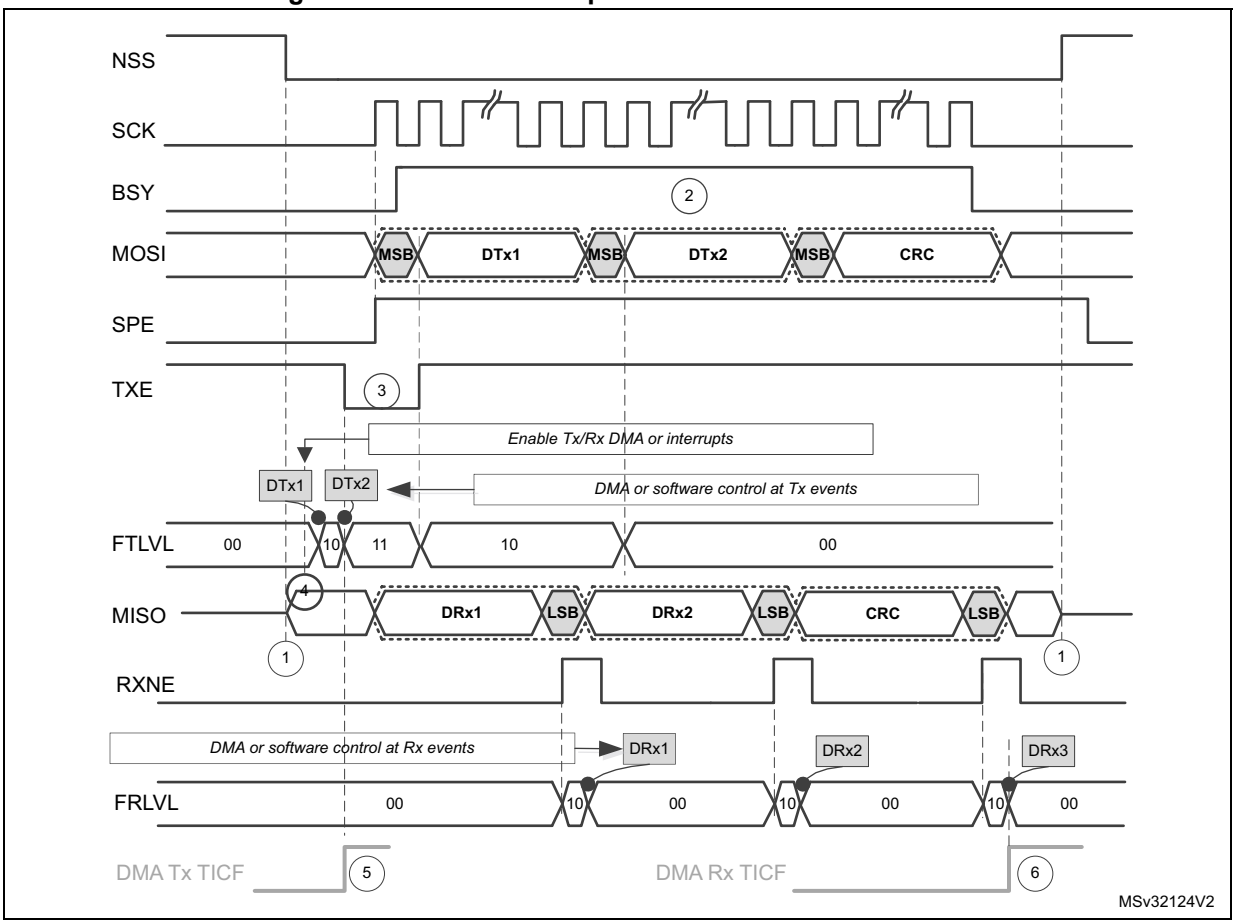
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 1030* for details about common assumptions and notes.

Figure 391. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

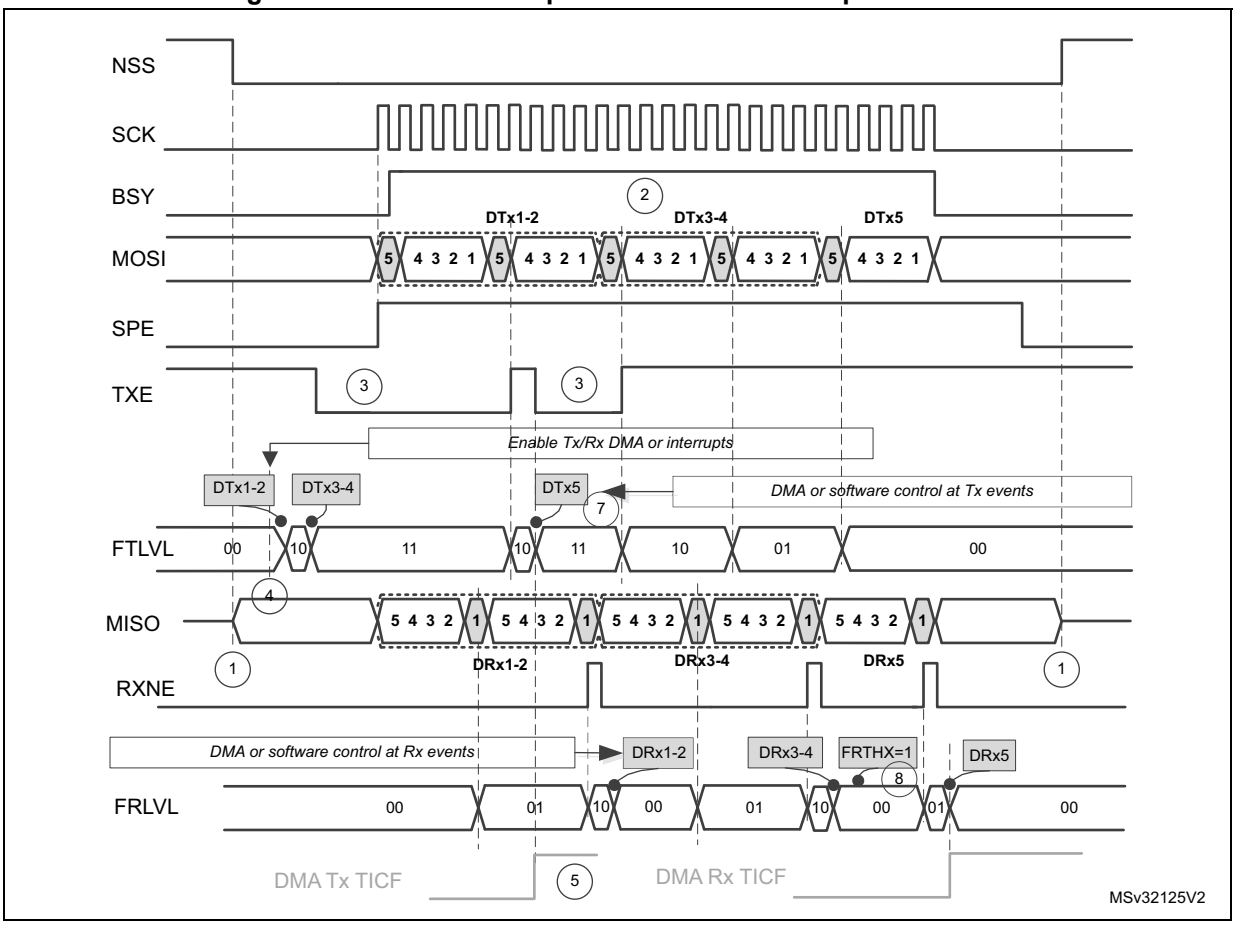
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also *: Communication diagrams on page 1030* for details about common assumptions and notes.

Figure 392. Master full-duplex communication in packed mode



Assumptions for master full-duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also *: Communication diagrams on page 1030* for details about common assumptions and notes.

29.4.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

29.4.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see *Section 29.4.14: CRC calculation*).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

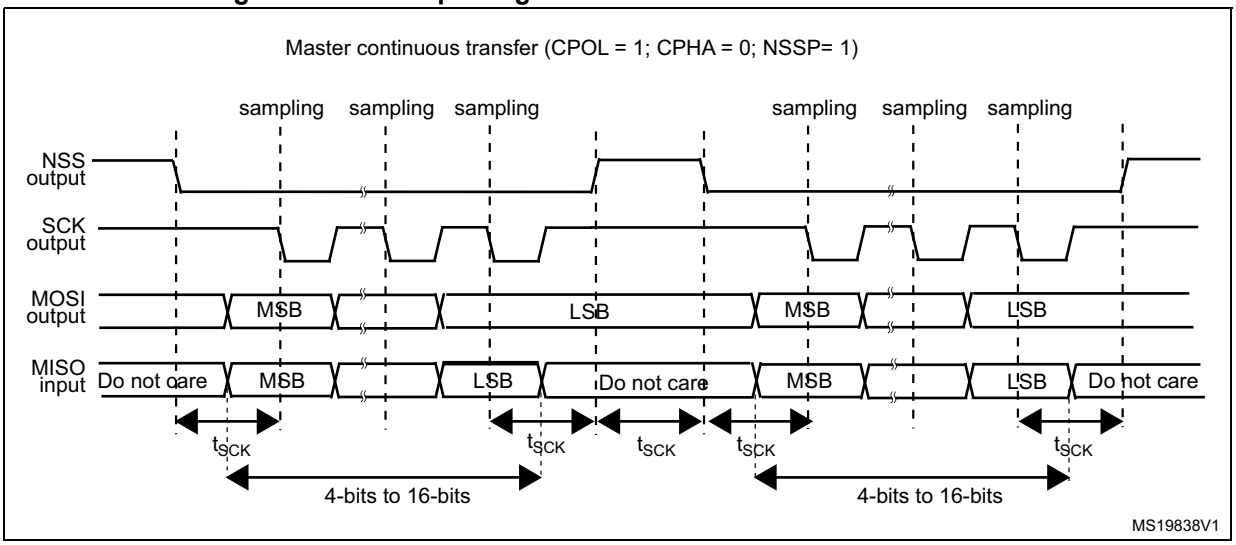
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

29.4.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 393 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 393. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the *rising* edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

29.4.13 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 394*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

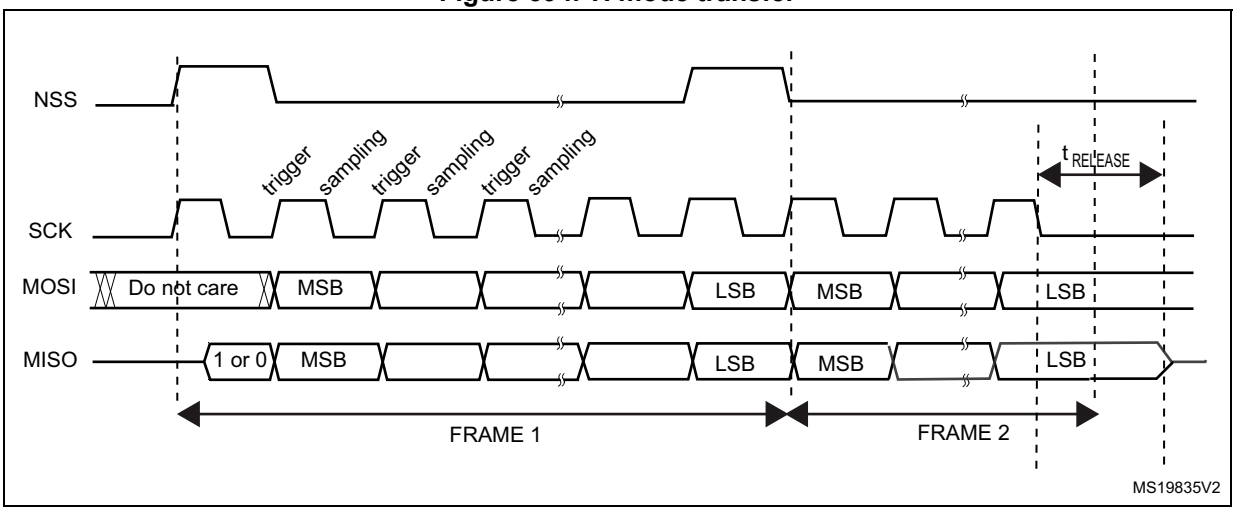
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Figure 394: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 394. TI mode transfer



29.4.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

DMA_RX = Numb_of_data + 2

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.

At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.

29.5 SPI interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

Table 145. SPI interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

29.6 SPI registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition can be accessed by 8-bit access.

29.6.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable.

This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Bit 14 **BIDIOE:** Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

Bit 12 **CRCNEXT:** Transmit CRC next

0: Next transmit value is from Tx buffer.

1: Next transmit value is from Tx CRC register.

Note: This bit has to be written as soon as the last data is written in the SPIx_DR register.

Bit 11 **CRCL:** CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

Bit 10 **RXONLY:** Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full-duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in SPI TI mode.

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

Note: This bit is not used in SPI TI mode.

Bit 7 **LSBFIRST***:* Frame format

0: data is transmitted / received with the MSB first

1: data is transmitted / received with the LSB first

Note: 1. This bit should not be changed when communication is ongoing.

2. This bit is not used in SPI TI mode.

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: When disabling the SPI, follow the procedure described in Procedure for disabling the SPI on page 1026.

Bits 5:3 **BR[2:0]:** Baud rate control

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

Note: This bit should not be changed when communication is ongoing.

Bit 1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit should not be changed when communication is ongoing.
This bit is not used in SPI TI mode except the case when CRC is applied at TI mode.

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit should not be changed when communication is ongoing.
This bit is not used in SPI TI mode except the case when CRC is applied at TI mode.

29.6.2 SPI control register 2 (SPIx_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA_TX:** Last DMA transfer for transmission
This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 1026 if the CRCEN bit is set.

Bit 13 **LDMA_RX**: Last DMA transfer for reception
This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).
0: Number of data to transfer is even
1: Number of data to transfer is odd
Note: Refer to Procedure for disabling the SPI on page 1026 if the CRCEN bit is set.

Bit 12 **FRXTH**: FIFO reception threshold
This bit is used to set the threshold of the RXFIFO that triggers an RXNE event
0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)
1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

Bits 11:8 **DS[3:0]**: Data size

These bits configure the data length for SPI transfers.

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit
1000: 9-bit
1001: 10-bit
1010: 11-bit
1011: 12-bit
1100: 13-bit
1101: 14-bit
1110: 15-bit
1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked
1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked
1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).

0: Error interrupt is masked
1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode
1 SPI TI mode

Note: This bit must be written only when the SPI is disabled (SPE=0).

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. it allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = ’1’, or FRF = ’1’.

0: No NSS pulse
1: NSS pulse generated

Note: 1. This bit must be written only when the SPI is disabled (SPE=0).
2. This bit is not used in SPI TI mode.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in SPI TI mode.

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

29.6.3 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	Res.	Res.	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0			r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]:** FIFO transmission level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full

Note: These bits are not used in SPI receive-only mode while CRC calculation is enabled.

Bit 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode. Refer to *Section 29.4.11: SPI error flags*.

This flag is set by hardware and reset when SPIx_SR is read by software.

0: No frame format error

1: A frame format error occurred

Bit 7 **BSY:** Busy flag
0: SPI not busy
1: SPI is busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.
Note: The BSY flag must be used with caution: refer to Section 29.4.10: SPI status flags and Procedure for disabling the SPI on page 1026.

Bit 6 **OVR:** Overrun flag
0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a software sequence.

Bit 5 **MODF:** Mode fault
0: No mode fault occurred
1: Mode fault occurred
This flag is set by hardware and reset by a software sequence. Refer to *Section : Mode fault (MODF) on page 1036* for the software sequence.

Bit 4 **CRCERR:** CRC error flag
0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value
Note: This flag is set by hardware and cleared by software writing 0.

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

Bit 1 **TXE:** Transmit buffer empty
0: Tx buffer not empty
1: Tx buffer empty

Bit 0 **RXNE:** Receive buffer not empty
0: Rx buffer empty
1: Rx buffer not empty

29.6.4 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]:** Data register
Data received or to be transmitted
The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See *Section 29.4.9: Data transmission and reception procedures*).
Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

29.6.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

29.6.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RXCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

A read to this register when the BSY Flag is set could return an incorrect value.

29.6.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **TXCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TXCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

A read to this register when the BSY flag is set could return an incorrect value.

29.6.8 SPI register map

Table 146 shows the SPI register map and reset values.

Table 146. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPIx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPIx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0x08	**SPIx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	Res.	Res.	TXE	RXNE
	Reset value																				0	0	0	0	0	0	0	0	0			1	0
0x0C	**SPIx_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPIx_CRCPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPIx_RXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPIx_TXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2 on page 47* for the register boundary addresses.

30 Controller area network (bxCAN)

30.1 Introduction

The **Basic Extended CAN** peripheral, named **bxCAN**, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. It has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the CAN Time Triggered Communication option.

30.2 bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  - 14 filter banks for single CAN
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

30.3 bxCAN general description

In today CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

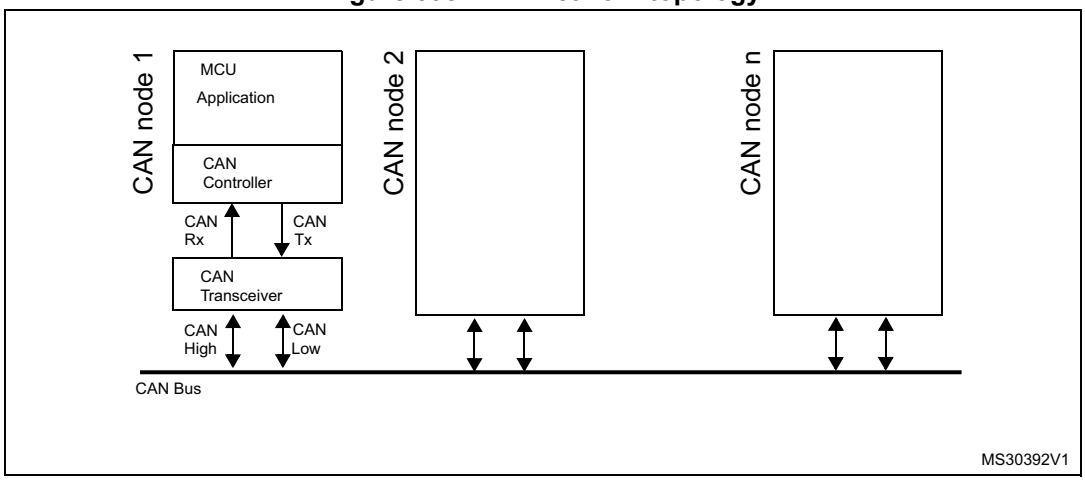
- An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

Figure 395. CAN network topology



30.3.1 CAN 2.0B active core

The bxCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

30.3.2 Control, status and configuration registers

The application uses these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

30.3.3 Tx mailboxes

Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first.

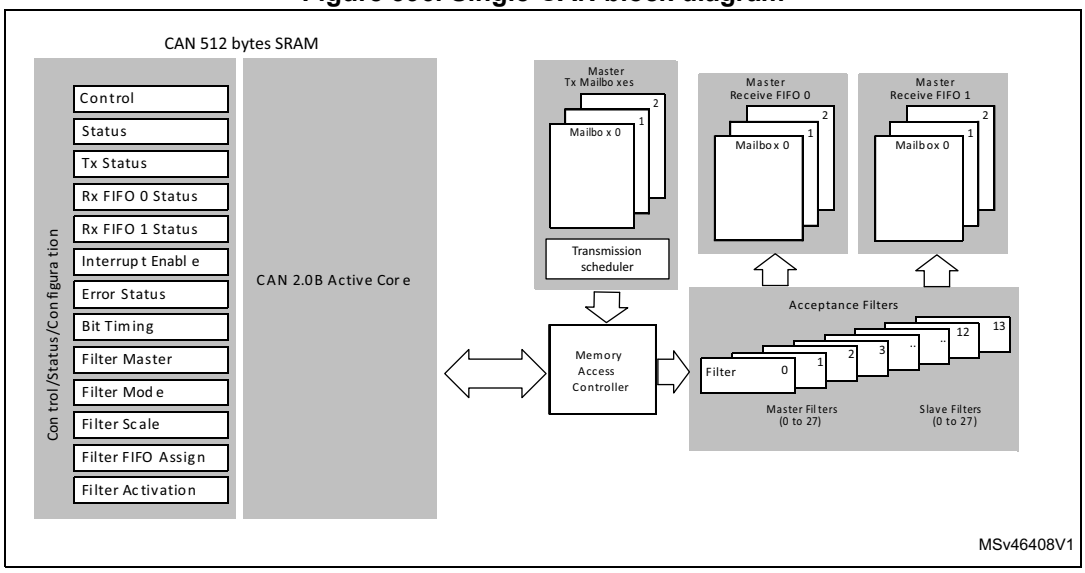
30.3.4 Acceptance filters

The bxCAN provides up to 14 scalable/configurable identifier filter banks in single CAN configuration, for selecting the incoming messages, that the software needs and discarding the others.

Receive FIFO

Two receive FIFOs are used by hardware to store the incoming messages. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware.

Figure 396. Single-CAN block diagram



30.4 bxCAN operating modes

bxCAN has three main operating modes: **initialization**, **normal** and **Sleep**. After a hardware reset, bxCAN is in Sleep mode to reduce power consumption and an internal pull-up is active on CANTX. The software requests bxCAN to enter **initialization** or **Sleep** mode by setting the INRQ or SLEEP bits in the CAN_MCR register. Once the mode has been entered, bxCAN confirms it by setting the INAK or SLAK bits in the CAN_MSR register and the internal pull-up is disabled. When neither INAK nor SLAK are set, bxCAN is in **normal** mode. Before entering **normal** mode bxCAN always has to **synchronize** on the CAN bus. To synchronize, bxCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

30.4.1 Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CAN_MCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CAN_MSR register.

To leave Initialization mode, the software clears the INQR bit. bxCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization Mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing (CAN_BTR) and CAN options (CAN_MCR) registers.

To initialize the registers associated with the CAN filter banks (mode, scale, FIFO assignment, activation and filter values), software has to set the FINIT bit (CAN_FMR). Filter initialization also can be done outside the initialization mode.

Note: When FINIT=1, CAN reception is deactivated.

The filter values also can be modified by deactivating the associated filter activation bits (in the CAN_FA1R register).

If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).

30.4.2 Normal mode

Once the initialization is complete, the software must request the hardware to enter Normal mode to be able to synchronize on the CAN bus and start reception and transmission.

The request to enter Normal mode is issued by clearing the INRQ bit in the CAN_MCR register. The bxCAN enters Normal mode and is ready to take part in bus activities when it has synchronized with the data transfer on the CAN bus. This is done by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). The switch to Normal mode is confirmed by the hardware by clearing the INAK bit in the CAN_MSR register.

The initialization of the filter values is independent from Initialization Mode but must be done while the filter is not active (corresponding FACTx bit cleared). The filter scale and mode configuration must be configured before entering Normal Mode.

30.4.3 Sleep mode (low-power)

To reduce power consumption, bxCAN has a low-power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CAN_MCR register. In this mode, the bxCAN clock is stopped, however software can still access the bxCAN mailboxes.

If software requests entry to **initialization** mode by setting the INRQ bit while bxCAN is in **Sleep** mode, it must also clear the SLEEP bit.

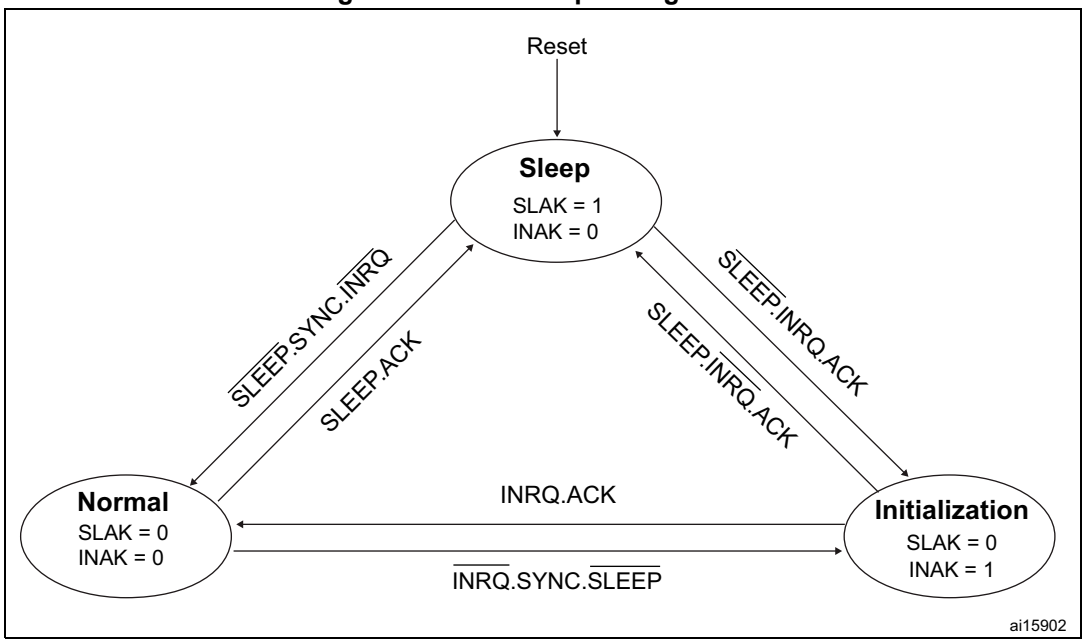
bxCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLEEP bit if the AWUM bit in the CAN_MCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wakeup interrupt occurs, in order to exit from Sleep mode.

Note: If the wakeup interrupt is enabled (WKUIE bit set in CAN_IER register) a wakeup interrupt is generated on detection of CAN bus activity, even if the bxCAN automatically performs the wakeup sequence.

After the SLEEP bit has been cleared, Sleep mode is exited once bxCAN has synchronized with the CAN bus, refer to *Figure 397: bxCAN operating modes*. The Sleep mode is exited once the SLAK bit has been cleared by hardware.

Figure 397. bxCAN operating modes



1. ACK = The wait state during which hardware confirms a request by setting the INAK or SLAK bits in the CAN_MSR register
2. SYNC = The state during which bxCAN waits until the CAN bus is idle, meaning 11 consecutive recessive bits have been monitored on CANRX

30.5 Test mode

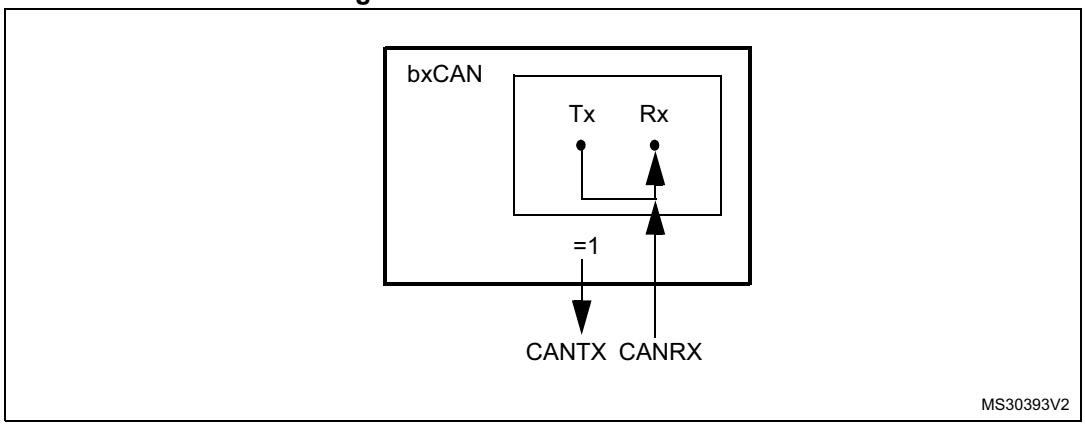
Test mode can be selected by the SILM and LBKM bits in the CAN_BTR register. These bits must be configured while bxCAN is in Initialization mode. Once test mode has been selected, the INRQ bit in the CAN_MCR register must be reset to enter Normal mode.

30.5.1 Silent mode

The bxCAN can be put in Silent mode by setting the SILM bit in the CAN_BTR register.

In Silent mode, the bxCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

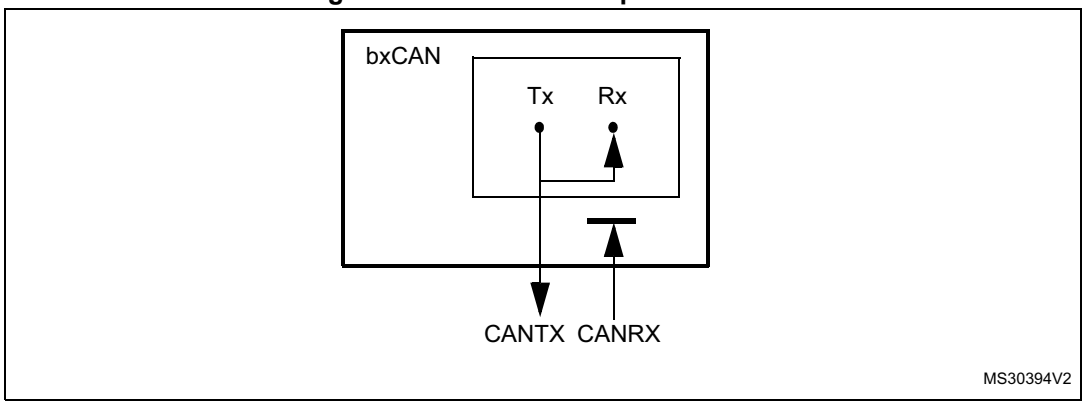
Figure 398. bxCAN in silent mode



30.5.2 Loop back mode

The bxCAN can be set in Loop Back Mode by setting the LBKM bit in the CAN_BTR register. In Loop Back Mode, the bxCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in a Receive mailbox.

Figure 399. bxCAN in loop back mode

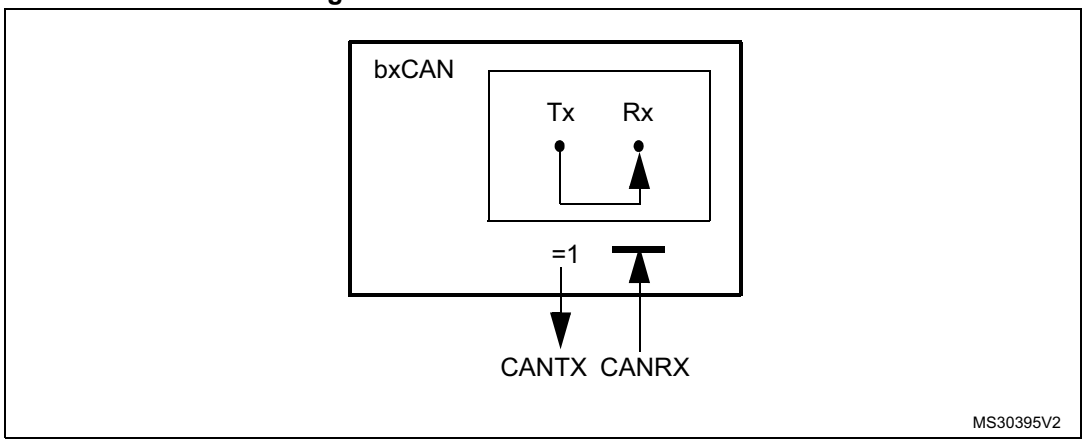


This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the bxCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the bxCAN. The transmitted messages can be monitored on the CANTX pin.

30.5.3 Loop back combined with silent mode

It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CAN_BTR register. This mode can be used for a “Hot Selftest”, meaning the bxCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the bxCAN and the CANTX pin is held recessive.

Figure 400. bxCAN in combined mode



30.6 Behavior in debug mode

When the microcontroller enters the debug mode (Cortex®-M4 core halted), the bxCAN continues to work normally or stops, depending on:

- the DBG_CAN1_STOP bit in the DBG module for the single mode.
- the DBF bit in CAN_MCR. For more details, refer to *Section 30.9.2: CAN control and status registers*.

30.7 bxCAN functional description

30.7.1 Transmission handling

In order to transmit a message, the application must select one **empty** transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the CAN_TIxR register. Once the mailbox has left **empty** state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters **pending** state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it is **scheduled** for transmission. The transmission of the message of the scheduled mailbox starts (enter **transmit** state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it becomes **empty** again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the CAN_TSR register.

If the transmission fails, the cause is indicated by the ALST bit in the CAN_TSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

Transmit priority

By identifier

When more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number is scheduled first.

By transmit request order

The transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CAN_MCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

Abort

A transmission request can be aborted by the user setting the ABRQ bit in the CAN_TSR register. In **pending** or **scheduled** state, the mailbox is aborted immediately. An abort request while the mailbox is in **transmit** state can have two results. If the mailbox is transmitted successfully the mailbox becomes **empty** with the TXOK bit set in the CAN_TSR register. If the transmission fails, the mailbox becomes **scheduled,** the transmission is aborted and becomes **empty** with TXOK cleared. In all cases the mailbox becomes **empty** again at least at the end of the current transmission.

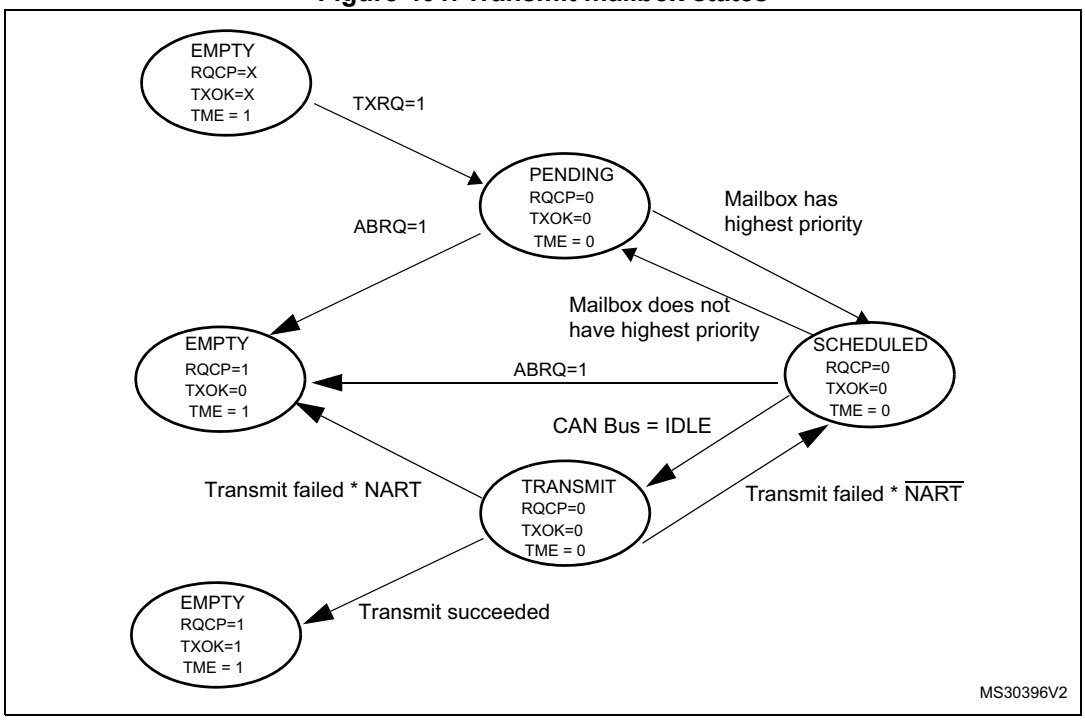
Non automatic retransmission mode

This mode has been implemented in order to fulfill the requirement of the Time Triggered Communication option of the CAN standard. To configure the hardware in this mode the NART bit in the CAN_MCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware does not automatically restart the message transmission.

At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the CAN_TSR register. The result of the transmission is indicated in the CAN_TSR register by the TXOK, ALST and TERR bits.

Figure 401. Transmit mailbox states



30.7.2 Time triggered communication mode

In this mode, the internal counter of the CAN hardware is activated and used to generate the Time Stamp value stored in the CAN_RDTxR/CAN_TDTxR registers, respectively (for Rx and Tx mailboxes). The internal counter is incremented each CAN bit time (refer to *Section 30.7.7: Bit timing*). The internal counter is captured on the sample point of the Start Of Frame bit in both reception and transmission.

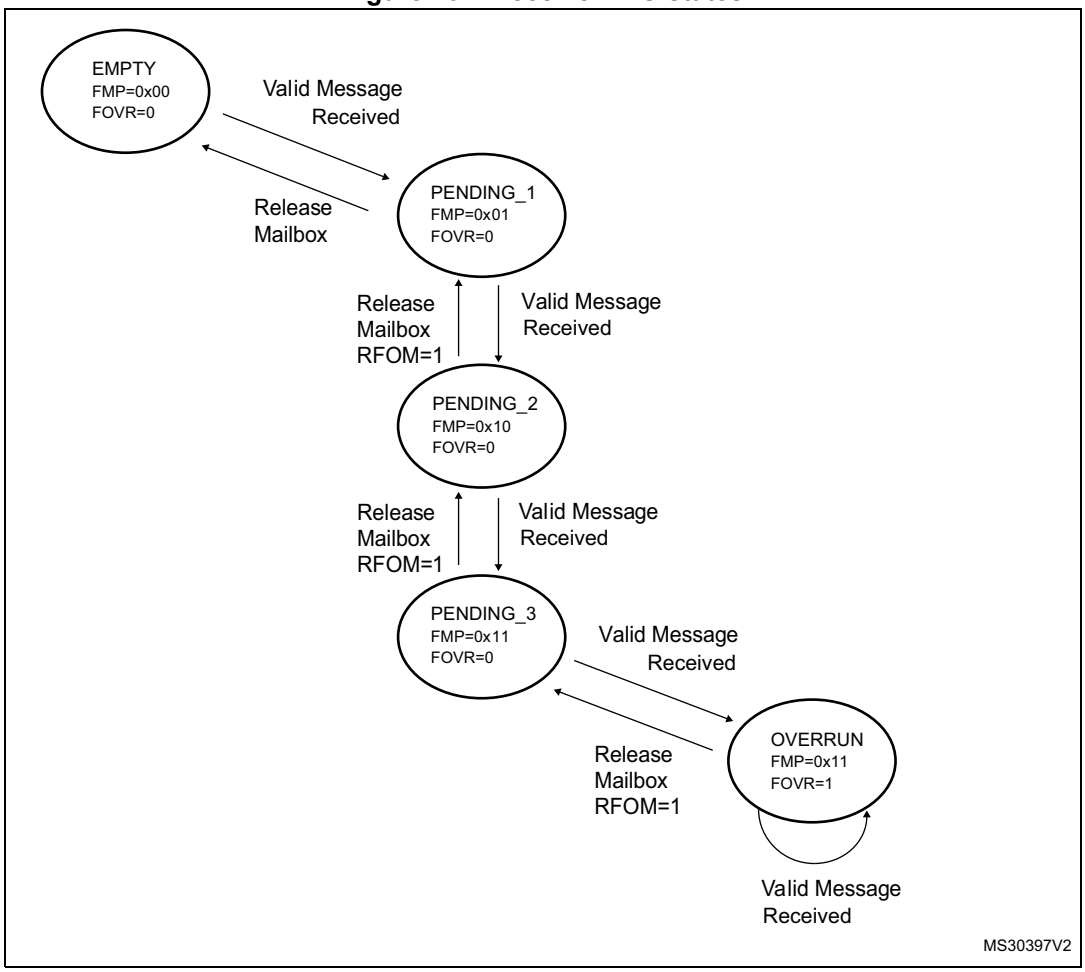
30.7.3 Reception handling

For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

Valid message

A received message is considered as valid **when** it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) **and** It passed through the identifier filtering successfully, see *Section 30.7.4: Identifier filtering*.

Figure 402. Receive FIFO states



FIFO management

Starting from the **empty** state, the first valid message received is stored in the FIFO which becomes **pending_1**. The hardware signals the event setting the FMP[1:0] bits in the CAN_RFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CAN_RFR register. The FIFO becomes **empty** again. If a new valid message has been received in the meantime, the FIFO stays in **pending_1** state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message is stored in the FIFO which enters **pending_2** state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into **pending_3** state (FMP[1:0] = 11b). At this point, the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received causes a loss of message.

Refer also to *Section 30.7.5: Message storage*

Overrun

Once the FIFO is in **pending_3** state (i.e. the three mailboxes are full) the next valid message reception leads to an **overrun** and a message is lost. The hardware signals the overrun condition by setting the FOVR bit in the CAN_RFR register. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (RFLM bit in the CAN_MCR register cleared) the last message stored in the FIFO is overwritten by the new incoming message. In this case the latest messages are always available to the application.
- If the FIFO lock function is enabled (RFLM bit in the CAN_MCR register set) the most recent message is discarded and the software has the three oldest messages in the FIFO available.

Reception related interrupts

Once a message has been stored in the FIFO, the FMP[1:0] bits are updated and an interrupt request is generated if the FMPIE bit in the CAN_IER register is set.

When the FIFO becomes full (i.e. a third message is stored) the FULL bit in the CAN_RFR register is set and an interrupt is generated if the FFIE bit in the CAN_IER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CAN_IER register is set.

30.7.4 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the SRAM. If not, the message must be discarded without intervention by the software.

To fulfill this requirement the bxCAN Controller provides 14 configurable and scalable filter banks (13-0) to the application, in order to receive only the messages the software needs.

This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank x consists of two 32-bit registers, CAN_FxR0 and CAN_FxR1.

Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], EXTID[17:0], IDE and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR, IDE and EXTID[17:15] bits.

Refer to *Figure 403*.

Furthermore, the filters can be configured in mask mode or in identifier list mode.

Mask mode

In **mask** mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

Identifier list mode

In **identifier list** mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

Filter bank scale and mode configuration

The filter banks are configured by means of the corresponding CAN_FMR register. To configure a filter bank it must be deactivated by clearing the FACT bit in the CAN_FAR register. The filter scale is configured by means of the corresponding FSCx bit in the CAN_FS1R register, refer to *Figure 403*. The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FBMx bits in the CAN_FMR register.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

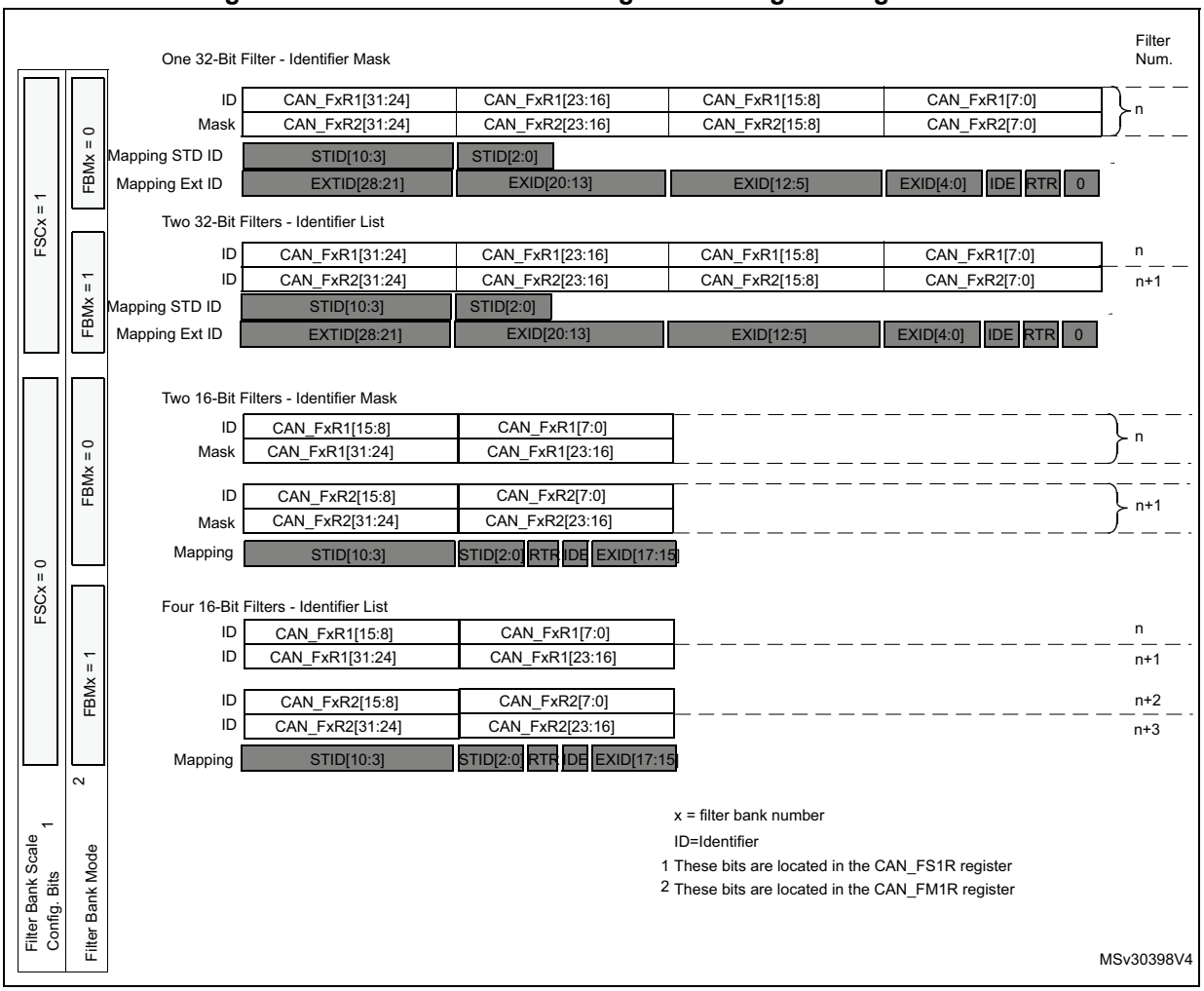
To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Each filter within a filter bank is numbered (called the *Filter Number*) from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

Concerning the filter configuration, refer to *Figure 403*.

Figure 403. Filter bank scale configuration - register organization



Filter match index

Once a message has been received in the FIFO it is available to the application. Typically, application data is copied into SRAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this, and to ease the access to the SRAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated filter match index.

The Filter Match index can be used in two ways:

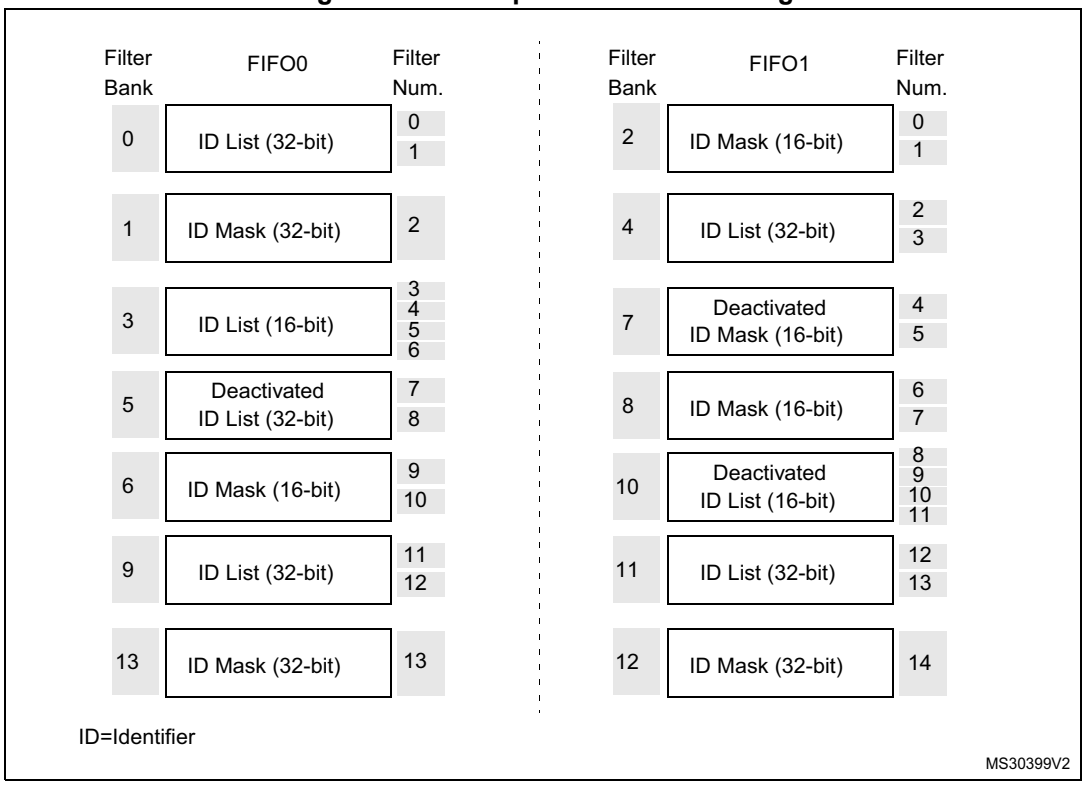
- Compare the Filter Match index with a list of expected values.
- Use the Filter Match Index as an index on an array to access the data destination location.

For non masked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

The index value of the filter number does not take into account the activation state of the filter banks. In addition, two independent numbering schemes are used, one for each FIFO. Refer to *Figure 404* for an example.

Figure 404. Example of filter numbering

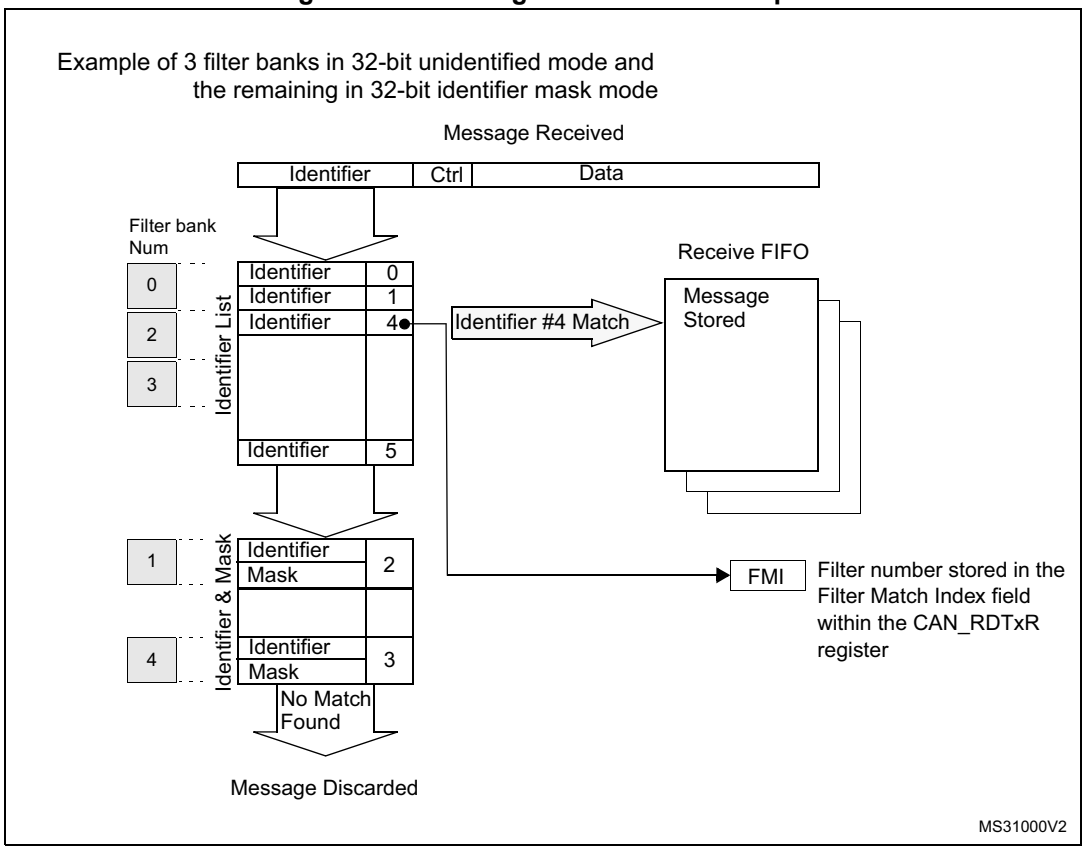


Filter priority rules

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter.
- For filters of equal scale, priority is given to the Identifier List mode over the Identifier Mask mode
- For filters of equal scale and mode, priority is given by the filter number (the lower the number, the higher the priority).

Figure 405. Filtering mechanism - example



The example above shows the filtering principle of the bxCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the associated FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and FMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without disturbing the software.

30.7.5 Message storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control, status and time stamp information.

Transmit mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the CAN_TSR register.

Table 147. Transmit mailbox mapping

Offset to transmit mailbox base address	Register name
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

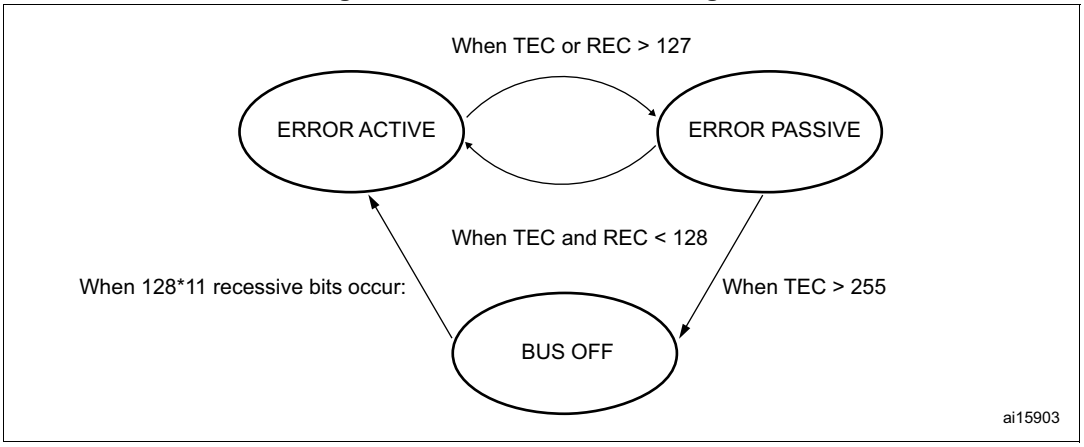
Receive mailbox

When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CAN_RFR register to make the next incoming message available. The filter match index is stored in the MFMI field of the CAN_RDTxR register. The 16-bit time stamp value is stored in the TIME[15:0] field of CAN_RDTxR.

Table 148. Receive mailbox mapping

Offset to receive mailbox base address (bytes)	Register name
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

Figure 406. CAN error state diagram



30.7.6 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TEC value, in CAN_ESR register) and a Receive Error Counter (REC value, in the CAN_ESR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CAN_ESR register. By means of the CAN_IER register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by BOFF bit in CAN_ESR register. In Bus-Off state, the bxCAN is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CAN_MCR register, bxCAN recovers from Bus-Off (become error active again) either automatically or on software request. But in both cases the bxCAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the bxCAN starts the recovering sequence automatically after it has entered Bus-Off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting bxCAN to enter and to leave initialization mode.

Note: In initialization mode, bxCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. ***To recover, bxCAN must be in normal mode****.*

30.7.7 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- **Synchronization segment (SYNC_SEG)**: a bit change is expected to occur within this time segment. It has a fixed length of one time quantum (1 x t_q).
- **Bit segment 1 (BS1)**: defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- **Bit segment 2 (BS2)**: defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing Register (CAN_BTR) is only possible while the device is in Standby mode.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898 standard.

Figure 407. Bit timing

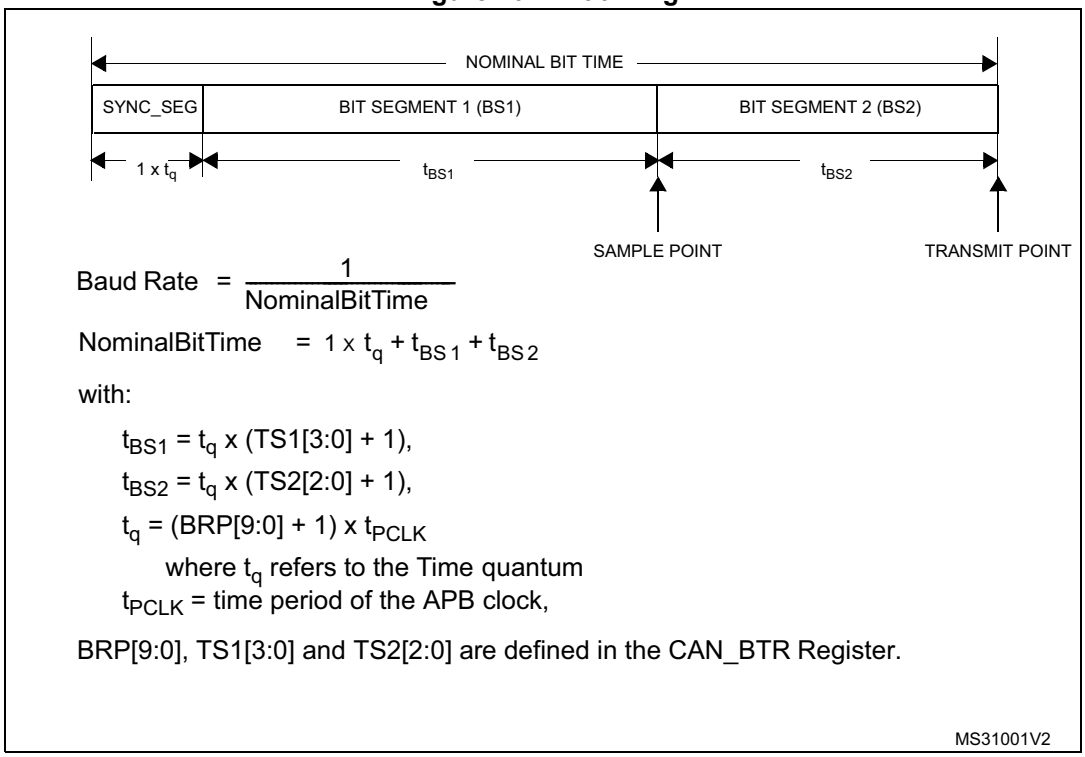
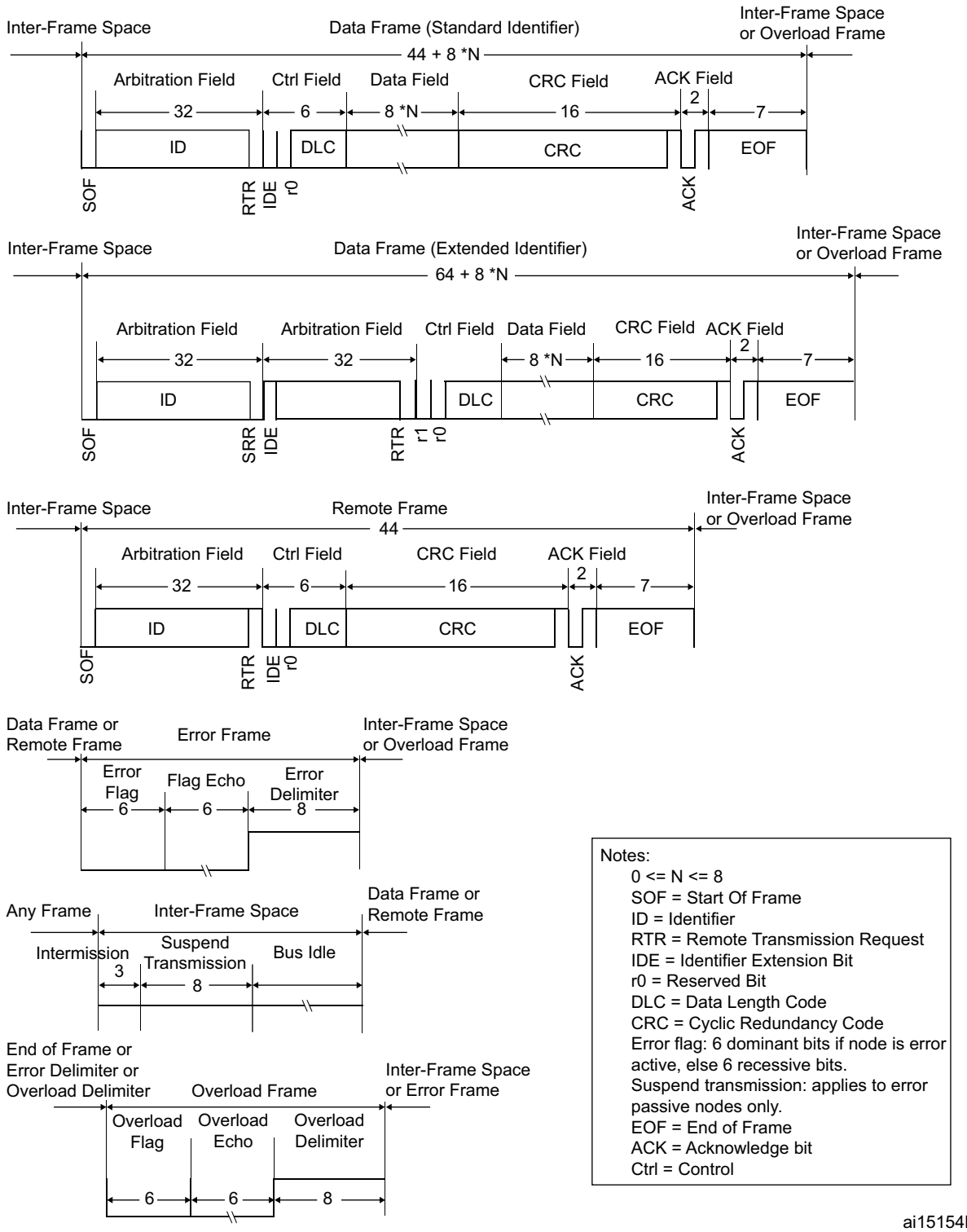


Figure 408. CAN frames



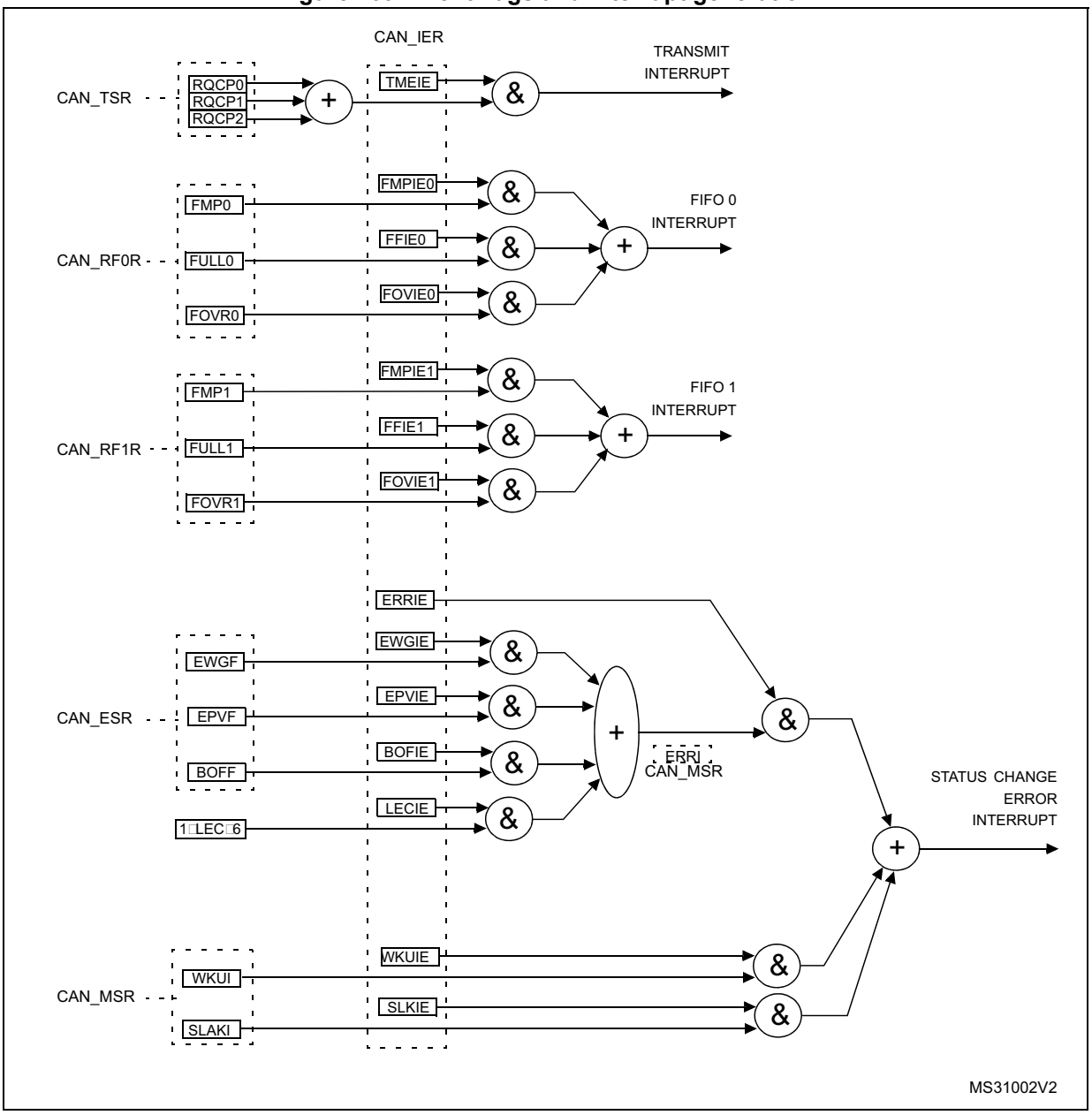
Notes:
- 0 <= N <= 8
- SOF = Start Of Frame
- ID = Identifier
- RTR = Remote Transmission Request
- IDE = Identifier Extension Bit
- r0 = Reserved Bit
- DLC = Data Length Code
- CRC = Cyclic Redundancy Code
- Error flag: 6 dominant bits if node is error active, else 6 recessive bits.
- Suspend transmission: applies to error passive nodes only.
- EOF = End of Frame
- ACK = Acknowledge bit
- Ctrl = Control

ai15154b

30.8 bxCAN interrupts

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN_IER).

Figure 409. Event flags and interrupt generation



- The **transmit interrupt** can be generated by the following events:
  - Transmit mailbox 0 becomes empty, RQCP0 bit in the CAN_TSR register set.
  - Transmit mailbox 1 becomes empty, RQCP1 bit in the CAN_TSR register set.
  - Transmit mailbox 2 becomes empty, RQCP2 bit in the CAN_TSR register set.
- The **FIFO 0 interrupt** can be generated by the following events:
  - Reception of a new message, FMP0 bits in the CAN_RF0R register are not '00'.
  - FIFO0 full condition, FULL0 bit in the CAN_RF0R register set.
  - FIFO0 overrun condition, FOVR0 bit in the CAN_RF0R register set.
- The **FIFO 1 interrupt** can be generated by the following events:
  - Reception of a new message, FMP1 bits in the CAN_RF1R register are not '00'.
  - FIFO1 full condition, FULL1 bit in the CAN_RF1R register set.
  - FIFO1 overrun condition, FOVR1 bit in the CAN_RF1R register set.
- The **error and status change interrupt** can be generated by the following events:
  - Error condition, for more details on error conditions refer to the CAN Error Status register (CAN_ESR).
  - Wakeup condition, SOF monitored on the CAN Rx signal.
  - Entry into Sleep mode.

30.9 CAN registers

The peripheral registers have to be accessed by words (32 bits).

30.9.1 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the CAN_BTR register can be modified by software only while the CAN hardware is in initialization mode.

Although the transmission of incorrect data does not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state, refer to *Figure 401: Transmit mailbox states*.

The filter values can be modified either deactivating the associated filter banks or by setting the FINIT bit. Moreover, the modification of the filter configuration (scale, mode and FIFO assignment) in CAN_FMxR, CAN_FSxR and CAN_FFAR registers can only be done when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

30.9.2 CAN control and status registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

CAN master control register (CAN_MCR)

Address offset: 0x00
Reset value: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBF
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **DBF:** Debug freeze

0: CAN working during debug

1: CAN reception/transmission frozen during debug. Reception FIFOs can still be accessed/controlled normally.

Bit 15 **RESET:** bxCAN software master reset

0: Normal operation.

1: Force a master reset of the bxCAN -> Sleep mode activated after reset (FMP bits and CAN_MCR register are initialized to the reset values). This bit is automatically reset to 0.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **TTCM**: Time triggered communication mode

0: Time Triggered Communication mode disabled.

1: Time Triggered Communication mode enabled

Note: For more information on Time Triggered Communication mode, refer to Section 30.7.2: Time triggered communication mode.

Bit 6 **ABOM:** Automatic bus-off management

This bit controls the behavior of the CAN hardware on leaving the Bus-Off state.

0: The Bus-Off state is left on software request, once 128 occurrences of 11 recessive bits have been monitored and the software has first set and cleared the INRQ bit of the CAN_MCR register.

1: The Bus-Off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.

For detailed information on the Bus-Off state refer to *Section 30.7.6: Error management*.

Bit 5 **AWUM**: Automatic wakeup mode

This bit controls the behavior of the CAN hardware on message reception during Sleep mode.

0: The Sleep mode is left on software request by clearing the SLEEP bit of the CAN_MCR register.

1: The Sleep mode is left automatically by hardware on CAN message detection.

The SLEEP bit of the CAN_MCR register and the SLAK bit of the CAN_MSR register are cleared by hardware.

Bit 4 **NART**: No automatic retransmission

0: The CAN hardware automatically retransmits the message until it has been successfully transmitted according to the CAN standard.

1: A message is transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 **RFLM**: Receive FIFO locked mode

0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message overwrites the previous one.
1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message is discarded.

Bit 2 **TXFP**: Transmit FIFO priority

This bit controls the transmission order when several mailboxes are pending at the same time.
0: Priority driven by the identifier of the message
1: Priority driven by the request order (chronologically)

Bit 1 **SLEEP**: Sleep mode request

This bit is set by software to request the CAN hardware to enter the Sleep mode. Sleep mode is entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.
This bit is cleared by software to exit Sleep mode.
This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.
This bit is set after reset - CAN starts in Sleep mode.

Bit 0 **INRQ**: Initialization request

The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit in the CAN_MSR register.
Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CAN_MSR register.

CAN master status register (CAN_MSR)

Address offset: 0x04
Reset value: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RX	SAMP	RXM	TXM	Res.	Res.	Res.	SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RX**: CAN Rx signal

Monitors the actual value of the **CAN_RX** Pin.

Bit 10 **SAMP**: Last sample point

The value of RX on the last sample point (current received bit value).

Bit 9 **RXM**: Receive mode

The CAN hardware is currently receiver.

Bit 8 **TXM**: Transmit mode
The CAN hardware is currently transmitter.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SLAKI**: Sleep acknowledge interrupt
When SLKIE=1, this bit is set by hardware to signal that the bxCAN has entered Sleep Mode. When set, this bit generates a status change interrupt if the SLKIE bit in the CAN_IER register is set.
This bit is cleared by software or by hardware, when SLAK is cleared.
Note: When SLKIE=0, no polling on SLAKI is possible. In this case the SLAK bit can be polled.

Bit 3 **WKUI**: Wakeup interrupt
This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in Sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CAN_IER register is set.
This bit is cleared by software.

Bit 2 **ERRI**: Error interrupt
This bit is set by hardware when a bit of the CAN_ESR has been set on error detection and the corresponding interrupt in the CAN_IER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CAN_IER register is set.
This bit is cleared by software.

Bit 1 **SLAK**: Sleep acknowledge
This bit is set by hardware and indicates to the software that the CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set SLEEP bit in CAN_MCR register).
This bit is cleared by hardware when the CAN hardware has left Sleep mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.
Note: The process of leaving Sleep mode is triggered when the SLEEP bit in the CAN_MCR register is cleared. Refer to the AWUM bit of the CAN_MCR register description for detailed information for clearing SLEEP bit

Bit 0 **INAK**: Initialization acknowledge
This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CAN_MCR register).
This bit is cleared by hardware when the CAN hardware has left the initialization mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

CAN transmit status register (CAN_TSR)

Address offset: 0x08
Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Res.	Res.	Res.	TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Res.	Res.	Res.	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res.	Res.	Res.	TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **LOW2**: Lowest priority flag for mailbox 2
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 2 has the lowest priority.

Bit 30 **LOW1**: Lowest priority flag for mailbox 1
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.

Bit 29 **LOW0**: Lowest priority flag for mailbox 0
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.
Note: The LOW[2:0] bits are set to zero when only one mailbox is pending.

Bit 28 **TME2**: Transmit mailbox 2 empty
This bit is set by hardware when no transmit request is pending for mailbox 2.

Bit 27 **TME1**: Transmit mailbox 1 empty
This bit is set by hardware when no transmit request is pending for mailbox 1.

Bit 26 **TME0**: Transmit mailbox 0 empty
This bit is set by hardware when no transmit request is pending for mailbox 0.

Bits 25:24 **CODE[1:0]**: Mailbox code
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.
In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.

Bit 23 **ABRQ2**: Abort request for mailbox 2
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

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

Bit 19 **TERR2**: Transmission error of mailbox 2
This bit is set when the previous TX failed due to an error.

Bit 18 **ALST2**: Arbitration lost for mailbox 2
This bit is set when the previous TX failed due to an arbitration lost.

Bit 17 **TXOK2**: Transmission OK of mailbox 2
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 2 has been completed successfully. Refer to *Figure 401*.

Bit 16 **RQCP2**: Request completed mailbox2
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ2 set in CAN_TMID2R register).
Clearing this bit clears all the status bits (TXOK2, ALST2 and TERR2) for Mailbox 2.

Bit 15 **ABRQ1**: Abort request for mailbox 1
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

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

Bit 11 **TERR1**: Transmission error of mailbox1
This bit is set when the previous TX failed due to an error.

Bit 10 **ALST1**: Arbitration lost for mailbox1
This bit is set when the previous TX failed due to an arbitration lost.

Bit 9 **TXOK1**: Transmission OK of mailbox1
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to *Figure 401*

Bit 8 **RQCP1**: Request completed mailbox1
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ1 set in CAN_TI1R register).
Clearing this bit clears all the status bits (TXOK1, ALST1 and TERR1) for Mailbox 1.

Bit 7 **ABRQ0**: Abort request for mailbox0
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **TERR0**: Transmission error of mailbox0
This bit is set when the previous TX failed due to an error.

Bit 2 **ALST0**: Arbitration lost for mailbox0
This bit is set when the previous TX failed due to an arbitration lost.

Bit 1 **TXOK0**: Transmission OK of mailbox0
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to *Figure 401*

Bit 0 **RQCP0**: Request completed mailbox0
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ0 set in CAN_TI0R register).
Clearing this bit clears all the status bits (TXOK0, ALST0 and TERR0) for Mailbox 0.

CAN receive FIFO 0 register (CAN_RF0R)

Address offset: 0x0C
Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM0**: Release FIFO 0 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR0**: FIFO 0 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.
This bit is cleared by software.

Bit 3 **FULL0**: FIFO 0 full

Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP0[1:0]**: FIFO 0 message pending

These bits indicate how many messages are pending in the receive FIFO.
FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM0 bit.

CAN receive FIFO 1 register (CAN_RF1R)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM1**: Release FIFO 1 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR1**: FIFO 1 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.
This bit is cleared by software.

Bit 3 **FULL1**: FIFO 1 full
Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending
These bits indicate how many messages are pending in the receive FIFO1.
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

CAN interrupt enable register (CAN_IER)

Address offset: 0x14
Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Res.	Res.	Res.	LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **SLKIE**: Sleep interrupt enable
0: No interrupt when SLAKI bit is set.
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable
0: No interrupt when WKUI is set.
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable
0: No interrupt is generated when an error condition is pending in the CAN_ESR.
1: An interrupt is generation when an error condition is pending in the CAN_ESR.

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

Bit 11 **LECIE**: Last error code interrupt enable
0: ERRI bit is not set when the error code in LEC[2:0] is set by hardware on error detection.
1: ERRI bit is set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 10 **BOFIE**: Bus-off interrupt enable
0: ERRI bit is not set when BOFF is set.
1: ERRI bit is set when BOFF is set.

Bit 9 **EPVIE**: Error passive interrupt enable
0: ERRI bit is not set when EPVF is set.
1: ERRI bit is set when EPVF is set.

Bit 8 **EWGIE**: Error warning interrupt enable
0: ERRI bit is not set when EWGF is set.
1: ERRI bit is set when EWGF is set.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FOVIE1**: FIFO overrun interrupt enable
0: No interrupt when FOVR is set.
1: Interrupt generation when FOVR is set.

Bit 5 **FFIE1**: FIFO full interrupt enable
0: No interrupt when FULL bit is set.
1: Interrupt generated when FULL bit is set.

Bit 4 **FMPIE1**: FIFO message pending interrupt enable
0: No interrupt generated when state of FMP[1:0] bits are not 00b.
1: Interrupt generated when state of FMP[1:0] bits are not 00b.

Bit 3 **FOVIE0**: FIFO overrun interrupt enable
0: No interrupt when FOVR bit is set.
1: Interrupt generated when FOVR bit is set.

Bit 2 **FFIE0**: FIFO full interrupt enable
0: No interrupt when FULL bit is set.
1: Interrupt generated when FULL bit is set.

Bit 1 **FMPIE0**: FIFO message pending interrupt enable
0: No interrupt generated when state of FMP[1:0] bits are not 00b.
1: Interrupt generated when state of FMP[1:0] bits are not 00b.

Bit 0 **TMEIE**: Transmit mailbox empty interrupt enable
0: No interrupt when RQCPx bit is set.
1: Interrupt generated when RQCPx bit is set.

Note: Refer to Section 30.8: bxCAN interrupts.

CAN error status register (CAN_ESR)

Address offset: 0x18
Reset value: 0x0000 0000

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

Bits 31:24 **REC[7:0]**: Receive error counter
The implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

Bits 23:16 **TEC[7:0]**: Least significant byte of the 9-bit transmit error counter
The implementing part of the fault confinement mechanism of the CAN protocol.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **LEC[2:0]**: Last error code

This field is set by hardware and holds a code which indicates the error condition of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field is cleared to 0.
The LEC[2:0] bits can be set to value 0b111 by software. They are updated by hardware to indicate the current communication status.
000: No Error
001: Stuff Error
010: Form Error
011: Acknowledgment Error
100: Bit recessive Error
101: Bit dominant Error
110: CRC Error
111: Set by software

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOFF**: Bus-off flag

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TEC overflow, greater than 255, refer to *Section 30.7.6 on page 1065*.

Bit 1 **EPVF**: Error passive flag

This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter>127).

Bit 0 **EWGF**: Error warning flag

This bit is set by hardware when the warning limit has been reached (Receive Error Counter or Transmit Error Counter≥96).

CAN bit timing register (CAN_BTR)

Address offset: 0x1C
Reset value: 0x0123 0000

This register can only be accessed by the software when the CAN hardware is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Res.	Res.	Res.	Res.	SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SILM**: Silent mode (debug)

0: Normal operation
1: Silent Mode

Bit 30 **LBKM**: Loop back mode (debug)

0: Loop Back Mode disabled
1: Loop Back Mode enabled

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **SJW[1:0]**: Resynchronization jump width
These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.
$t_{RJW} = t_q \times (SJW[1:0] + 1)$

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **TS2[2:0]**: Time segment 2
These bits define the number of time quanta in Time Segment 2.
$t_{BS2} = t_q \times (TS2[2:0] + 1)$

Bits 19:16 **TS1[3:0]**: Time segment 1
These bits define the number of time quanta in Time Segment 1
$t_{BS1} = t_q \times (TS1[3:0] + 1)$
For more information on bit timing, refer to *Section 30.7.7: Bit timing on page 1065*.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **BRP[9:0]**: Baud rate prescaler
These bits define the length of a time quanta.
$t_q = (BRP[9:0]+1) \times t_{PCLK}$

30.9.3 CAN mailbox registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to *Section 30.7.5: Message storage on page 1063* for detailed register mapping.

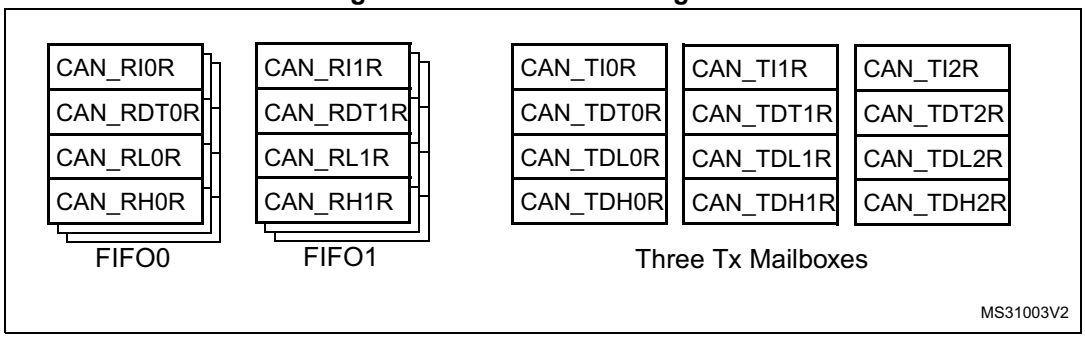
Transmit and receive mailboxes have the same registers except:

- The FMI field in the CAN_RDTxR register.
- A receive mailbox is always write protected.
- A transmit mailbox is write-enabled only while empty, corresponding TME bit in the CAN_TSR register set.

There are 3 TX Mailboxes and 2 RX Mailboxes. Each RX Mailbox allows access to a 3 level depth FIFO, the access being offered only to the oldest received message in the FIFO.

Each mailbox consist of 4 registers.

Figure 410. CAN mailbox registers



CAN TX mailbox identifier register (CAN_TIxR) (x = 0..2)

Address offsets: 0x180, 0x190, 0x1A0
Reset value: 0xXXXX XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

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

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bit 20:3 **EXID[17:0]**: Extended identifier
The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension
This bit defines the identifier type of message in the mailbox.
0: Standard identifier.
1: Extended identifier.

Bit 1 **RTR**: Remote transmission request
0: Data frame
1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request
Set by software to request the transmission for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.

CAN mailbox data length control and time stamp register (CAN_TDTxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x184, 0x194, 0x1A4
Reset value: 0xXXXX XXXX

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

Bits 31:16 **TIME[15:0]**: Message time stamp
This field contains the 16-bit timer value captured at the SOF transmission.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TGT**: Transmit global time
This bit is active only when the hardware is in the Time Trigger Communication mode, TTCM bit of the CAN_MCR register is set.
0: Time stamp TIME[15:0] is not sent.
1: Time stamp TIME[15:0] value is sent in the last two data bytes of the 8-byte message: TIME[7:0] in data byte 7 and TIME[15:8] in data byte 6, replacing the data written in CAN_TDHxR[31:16] register (DATA6[7:0] and DATA7[7:0]). DLC must be programmed as 8 in order these two bytes to be sent over the CAN bus.

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

Bits 3:0 **DLC[3:0]**: Data length code
This field defines the number of data bytes a data frame contains or a remote frame request. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.

CAN mailbox data low register (CAN_TDLxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8
Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA3[7:0]**: Data byte 3
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0
Data byte 0 of the message.
A message can contain from 0 to 8 data bytes and starts with byte 0.

CAN mailbox data high register (CAN_TDHxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC
Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA7[7:0]**: Data byte 7
Data byte 7 of the message.
Note: If TGT of this message and TTCM are active, DATA7 and DATA6 are replaced by the TIME stamp value.

Bits 23:16 **DATA6[7:0]**: Data byte 6
Data byte 6 of the message.

Bits 15:8 **DATA5[7:0]**: Data byte 5
Data byte 5 of the message.

Bits 7:0 **DATA4[7:0]**: Data byte 4
Data byte 4 of the message.

CAN receive FIFO mailbox identifier register (CAN_RIxR) (x = 0..1)

Address offsets: 0x1B0, 0x1C0
Reset value: 0xXXXX XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bits 20:3 **EXID[17:0]**: Extended identifier
The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension
This bit defines the identifier type of message in the mailbox.
0: Standard identifier.
1: Extended identifier.

Bit 1 **RTR**: Remote transmission request
0: Data frame
1: Remote frame

Bit 0 Reserved, must be kept at reset value.

CAN receive FIFO mailbox data length control and time stamp register (CAN_RDTxR) (x = 0..1)

Address offsets: 0x1B4, 0x1C4
Reset value: 0xXXXX XXXX

All RX registers are write protected.

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

Bits 31:16 **TIME[15:0]**: Message time stamp
This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index
This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering refer to *Section 30.7.4: Identifier filtering on page 1059* - **Filter Match Index** paragraph.

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

Bits 3:0 **DLC[3:0]**: Data length code
This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

CAN receive FIFO mailbox data low register (CAN_RDLxR) (x = 0..1)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x1B8, 0x1C8
Reset value: 0xXXXX XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA3[7:0]**: Data Byte 3
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data Byte 2
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data Byte 1
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data Byte 0
Data byte 0 of the message.
A message can contain from 0 to 8 data bytes and starts with byte 0.

CAN receive FIFO mailbox data high register (CAN_RDHxR) (x = 0..1)

Address offsets: 0x1BC, 0x1CC
Reset value: 0xXXXX XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA7[7:0]**: Data Byte 7
Data byte 3 of the message.

Bits 23:16 **DATA6[7:0]**: Data Byte 6
Data byte 2 of the message.

Bits 15:8 **DATA5[7:0]**: Data Byte 5
Data byte 1 of the message.

Bits 7:0 **DATA4[7:0]**: Data Byte 4
Data byte 0 of the message.

30.9.4 CAN filter registers

CAN filter master register (CAN_FMR)

Address offset: 0x200
Reset value: 0x2A1C 0E01

All bits of this register are set and cleared by software.

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

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

Bit 0 **FINIT**: Filter initialization mode
Initialization mode for filter banks
0: Active filters mode.
1: Initialization mode for the filters.

CAN filter mode register (CAN_FM1R)

Address offset: 0x204
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: Refer to Figure 403: Filter bank scale configuration - register organization on page 1061.

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

Bits 13:0 **FBM*x***: Filter mode

Mode of the registers of Filter x.
0: Two 32-bit registers of filter bank x are in Identifier Mask mode.
1: Two 32-bit registers of filter bank x are in Identifier List mode.

CAN filter scale register (CAN_FS1R)

Address offset: 0x20C
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **FSC*x***: Filter scale configuration

These bits define the scale configuration of Filters 13-0.
0: Dual 16-bit scale configuration
1: Single 32-bit scale configuration

Note: Refer to Figure 403: Filter bank scale configuration - register organization on page 1061.

CAN filter FIFO assignment register (CAN_FFA1R)

Address offset: 0x214
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **FFA*x***: Filter FIFO assignment for filter x

The message passing through this filter is stored in the specified FIFO.
0: Filter assigned to FIFO 0
1: Filter assigned to FIFO 1

CAN filter activation register (CAN_FA1R)

Address offset: 0x21C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 13:0 **FACT*x***: Filter active

The software sets this bit to activate Filter x. To modify the Filter x registers (CAN_FxR[0:7]), the FACTx bit must be cleared or the FINIT bit of the CAN_FMR register must be set.
0: Filter x is not active
1: Filter x is active

Filter bank i register x (CAN_FiRx) (i = 0..13, x = 1, 2)

Address offsets: 0x240 to 0x2AC
Reset value: 0xXXXX XXXX

There are 14 filter banks, i= 0 to 13. Each filter bank i is composed of two 32-bit registers, CAN_FiR[2:1].

This register can only be modified when the FACTx bit of the CAN_FAxR register is cleared or when the FINIT bit of the CAN_FMR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

In all configurations:

Bits 31:0 **FB[31:0]**: Filter bits

Identifier
Each bit of the register specifies the level of the corresponding bit of the expected identifier.
0: Dominant bit is expected
1: Recessive bit is expected

Mask
Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.
0: Do not care, the bit is not used for the comparison
1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier register of the filter.

Note: Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to Section 30.7.4: Identifier filtering on page 1059.

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list** mode.

For the register mapping/addresses of the filter banks refer to Table 149 on page 1090.

30.9.5 bxCAN register map

Refer to *Section 2.2 on page 47* for the register boundary addresses.

Table 149. bxCAN register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**CAN_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBF	RESET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	1	0
0x004	**CAN_MSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	SAMP	RXM	TXM	Res.	Res.	Res.	SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	**CAN_TSR**	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Res.	Res.	Res.	TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Res.	Res.	Res.	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res.	Res.	Res.	TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	**CAN_RF0R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
	Reset value																											0	0	0		0	0
0x010	**CAN_RF1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
	Reset value																											0	0	0		0	0
0x014	**CAN_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLKIE	WKUIE	ERRIE	Res.	Res.	Res.	LECIE	BOFIE	EPVIE	EWGIE	Res.	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value															0	0	0				0	0	0	0		0	0	0	0	0	0	0
0x018	**CAN_ESR**	REC[7:0]								TEC[7:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEC[2:0]			Res.	BOFF	EPVF	EWGF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0	0		0	0	0
0x01C	**CAN_BTR**	SILM	LBKM	Res.	Res.	Res.	Res.	SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]					Res.	Res.	Res.	Res.	Res.	Res.	BRP[9:0]									
	Reset value	0	0					0	0		0	1	0	0	0	1	1							0	0	0	0	0	0	0	0	0	0
0x020-0x17F	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x180	**CAN_TI0R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0

Table 149. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	**CAN_TDT0R**	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x188	**CAN_TDL0R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x18C	**CAN_TDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	**CAN_TI1R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x194	**CAN_TDT1R**	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x198	**CAN_TDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	**CAN_TDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	**CAN_TI2R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	**CAN_TDT2R**	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	x	-	-	-	-	x	x	x	x
0x1A8	**CAN_TDL2R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	**CAN_TDH2R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	**CAN_RI0R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Res.
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-

Table 149. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1B4	**CAN_RDT0R**	TIME[15:0]																FMI[7:0]								Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	x	x	x	x
0x1B8	**CAN_RDL0R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	**CAN_RDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	**CAN_RI1R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Res.
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-
0x1C4	**CAN_RDT1R**	TIME[15:0]																FMI[7:0]								Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	x	x	x	x
0x1C8	**CAN_RDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1CC	**CAN_RDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x200	**CAN_FMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINIT
	Reset value																																1
0x204	**CAN_FM1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FBM[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	-																																
0x20C	**CAN_FS1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 149. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x214	**CAN_FFA1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FFA[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x21C	**CAN_FA1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FACT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x224-0x23F	-	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x240	**CAN_F0R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	**CAN_F0R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	**CAN_F1R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	**CAN_F1R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . . .	. . . .	.																															
0x318	**CAN_F27R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	**CAN_F27R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Refer to *Section 2.2 on page 47* for the register boundary addresses.

31 Debug support (DBG)

31.1 Overview

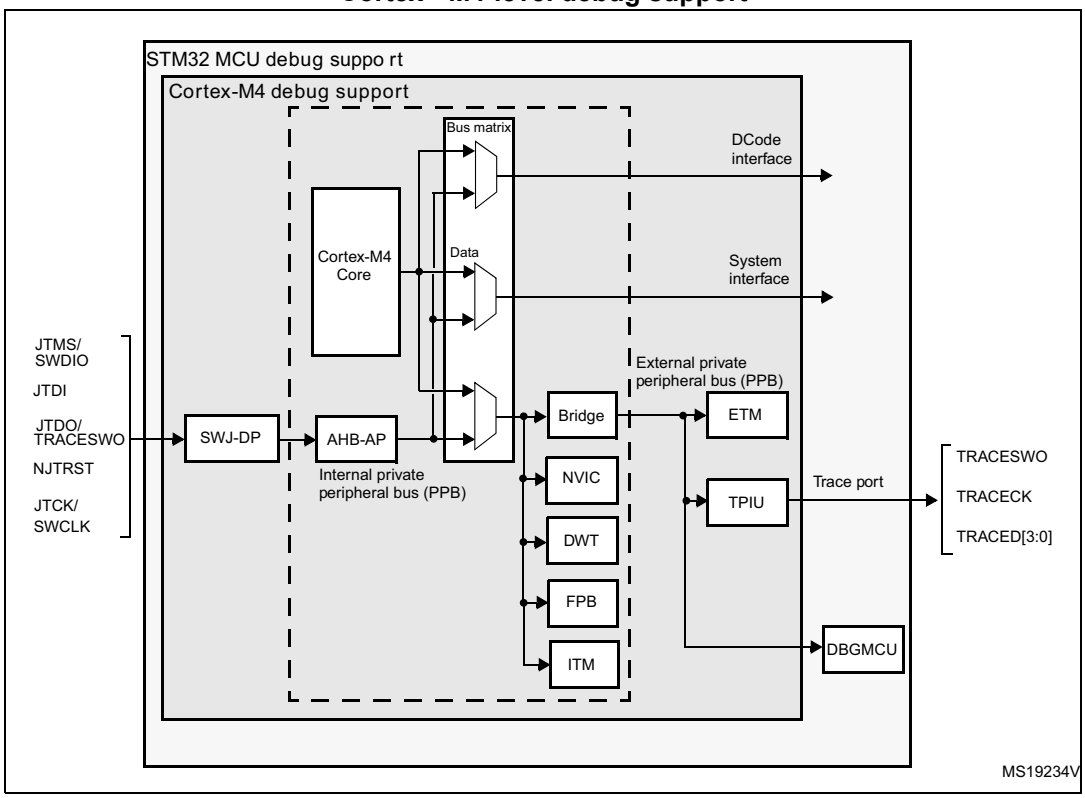
The STM32F334xx devices are built around a Cortex®-M4 core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32F334xx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 411. Block diagram of STM32 MCU and Cortex®-M4-level debug support



Note: The debug features embedded in the Cortex-M4 core are a subset of the Arm CoreSight Design Kit.

The Cortex®-M4 core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F334xx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Note: For further information on the debug feature supported by the Cortex®-M4 core, refer to the Cortex®-M4 with FPU-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see Section 31.2: Reference Arm documentation).

31.2 Reference Arm documentation

- Cortex®-M4 r0p1 Technical Reference Manual (TRM)
  It is available from: http://infocenter.arm.com
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r0p1 Technical Reference Manual

31.3 SWJ debug port (serial wire and JTAG)

The STM32F334xx core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an Arm standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG debug port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The serial wire debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 412. SWJ debug port

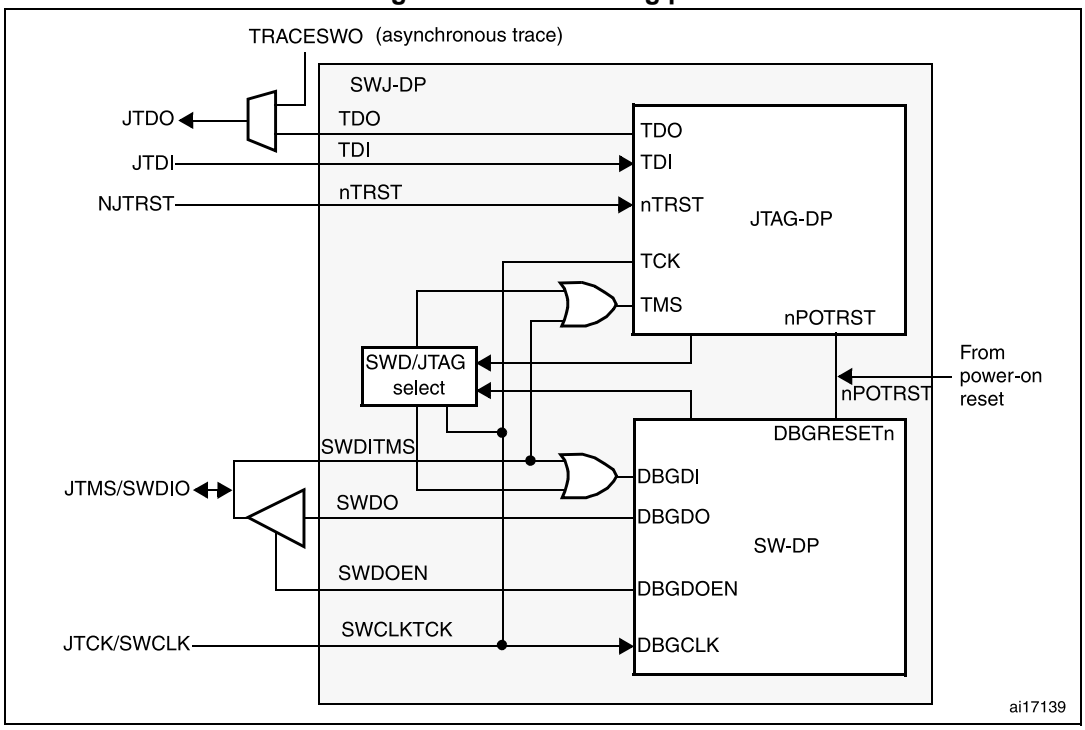


Figure 412 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

31.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

31.4 Pinout and debug port pins

The STM32F334xx MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

31.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F334xx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 150. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assign ment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

31.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, it is possible to disable some or all of the SWJ-DP ports and so, to release (in gray in the table below) the associated pins for general-purpose I/O(GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to *Section 9.3.2: I/O pin alternate function multiplexer and mapping*.

Table 151. Flexible SWJ-DP pin assignment

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

Note: When the APB bridge write buffer is full, it takes one extra APB cycle when writing the AFIO_MAPR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

31.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- JTCK/SWCLK: Input pull-down
- JTDO: Input floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

31.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they are first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

31.5 STM32F334xx JTAG TAP connection

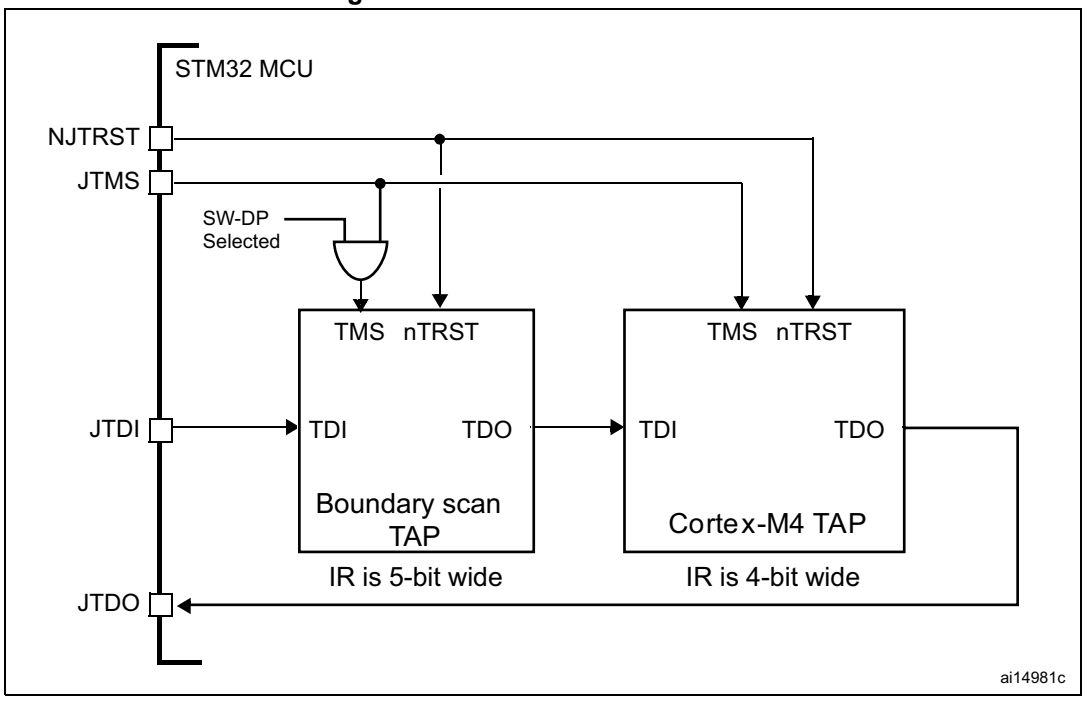
The STM32F334xx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex®-M4 TAP (IR is 4-bit wide).

To access the TAP of the Cortex®-M4 for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note: **Important**: Once Serial-Wire is selected using the dedicated Arm JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).*

Figure 413. JTAG TAP connections



31.6 ID codes and locking mechanism

There are several ID codes inside the STM32F334xx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

31.6.1 MCU device ID code

The STM32F334xx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see *Section 31.15 on page 1111*). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

DBGMCU_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only

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

This code is read as 0x10000438 for Revision 1.0.

Bits 31:16 **REV_ID[15:0]** Revision identifier
This field indicates the revision of the device. For example, it is read as 0x1000 for Revision 1.

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

Bits 11:0 **DEV_ID[11:0]**: Device identifier
This field indicates the device and its revision.
The device ID is 0x438.

31.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F334xx BSC (boundary scan) integrates a JTAG ID code equal to 0x06432041.

31.6.3 Cortex®-M4 TAP

The TAP of the Cortex®-M4 integrates a JTAG ID code. This ID code is the Arm® default one and has not been modified. This code is only accessible by the JTAG Debug Port.

This code is **0x4BA00477** (corresponds to Cortex®-M4 r0p1, see *Section 31.2: Reference Arm documentation*).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

31.6.4 Cortex®-M4 JEDEC-106 ID code

The Cortex®-M4 integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

31.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex®-M4r0p1 Technical Reference Manual (TRM), for references, please see *Section 31.2: Reference Arm documentation*).

Table 152. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID CODE 0x3BA00477 (Cortex®-M4 r0p1 ID Code)

Table 152. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1010	DPACC [35 bits]	Debug port access register This initiates a debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to *Table 153* for a description of the A(3:2) bits
1011	APACC [35 bits]	Access port access register Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved There are many AP Registers (see AHB-AP) addressed as the combination of: – The shifted value A[3:2] – The current value of the DP SELECT register
1000	ABORT [35 bits]	Abort register – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 153. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	DP CTRL/STAT register. Used to: – Request a system or debug power-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-up acknowledges)

Table 153. 32-bit debug port registers addressed through the shifted value A[3:2] (continued)

Address	A(3:2) value	Description
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

31.8 SW debug port

31.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 kΩ recommended by Arm).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

31.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 154. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request

Table 154. Packet request (8-bits) (continued)

Bit	Name	Description
4:3	A(3:2)	Address field of the DP or AP registers (refer to *Table 153*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up

Refer to the Cortex®-M4 r0p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 155. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 156. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

31.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm® one and is set to **0x1BA01477** (corresponding to Cortex®-M4 r0p1).

Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target issues a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex®-M4 r0p1 TRM* and the *CoreSight Design Kit r0p1TRM*.

31.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
  The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
  This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it fails.

31.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 157. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read	-	IDCODE	The manufacturer code is not set to ST code **0x2BA01477** (identifies the SW-DP)
00	Write	-	ABORT	-
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read	-	READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.

Table 157. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
10	Write	-	SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write	-	READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

31.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

31.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP resisters are 6-bits wide (up to 64 words or 256 bytes) and consists of:

g) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register

h) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex®-M4 includes 9 x 32-bits registers:

Table 158. Cortex®-M4 AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	AHB-AP Transfer Address	-
0x0C	AHB-AP Data Read/Write	-
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	-

Refer to the *Cortex®-M4 r0p1 TRM* for further details.

31.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 159. Core debug registers

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status Register This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	The 17-bit Debug Core Register Selector Register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core Register Data Register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control Register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named ***TRCENA*** which enable the use of a TRACE.

Note: ***Important****: these registers are not reset by a system reset. They are only reset by a power-on reset.*

Refer to the *Cortex®-M4* r0p*1 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register.

31.11 Capability of the debugger host to connect under system reset

The STM32F334xx MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex®-M4 differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

31.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

31.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

31.14 ITM (instrumentation trace macrocell)

31.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex®-M4 clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before programming or using the ITM.

31.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: If the SYNENA bit is not set, the DWT generates synchronization triggers to the TPIU which sends only TPIU synchronization packets and not ITM synchronization packets.

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 160. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000-E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to *Section 31.15.3: Debug MCU configuration register*)
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

31.15 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

31.15.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This feeds HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This enables the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

31.15.2 Debug support for timers, watchdog, bxCAN and I^2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I^2C, the user can choose to block the SMBUS timeout during a breakpoint.

31.15.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- bxCAN communication support
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004.

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

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

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

Bit 2 **DBG_STANDBY:** Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.
From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)
1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP:** Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.
1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP:** Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.
In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.
1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

31.15.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under DEBUG. It concerns the APB1 peripherals:

- Timer clock counter freeze
- I2C SMBUS timeout freeze
- Window watchdog and independent watchdog counter freeze support

This DBGMCU_APB1_FZ is mapped on the external PPB bus at address 0xE0042008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 2008

Only 32-bit access are supported.

Power on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	DBG_CAN_STOP	Res	Res	Res	DBG_I2C1_SMBUS_TIMEOUT	Res	Res	Res	Res	Res
						rw				rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res	Res	Res	Res	DBG_TIM7_STOP	DBG_TIM6_STOP	Res	Res	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DBG_CAN_STOP:** Debug CAN stopped when core is halted
0: Same behavior as in normal mode
1: The CAN2 receive registers are frozen

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

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

Bit 12 **DBG_IWDG_STOP:** Debug independent watchdog stopped when core is halted
0: The independent watchdog counter clock continues even if the core is halted
1: The independent watchdog counter clock is stopped when the core is halted

Bit 11 **DBG_WWDG_STOP:** Debug window watchdog stopped when core is halted
0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted

Bit 10 **DBG_RTC_STOP**: Debug RTC stopped when core is halted
0: The clock of the RTC counter is fed even if the core is halted
1: The clock of the RTC counter is stopped when the core is halted

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 counter stopped when core is halted
0: The counter clock of TIM7 is fed even if the core is halted
1: The counter clock of TIM7 is stopped when the core is halted

Bit 4 **DBG_TIM6_STOP**: TIM6 counter stopped when core is halted
0: The counter clock of TIM6 is fed even if the core is halted
1: The counter clock of TIM6 is stopped when the core is halted

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

Bit 1 **DBG_TIM3_STOP**: TIM3 counter stopped when core is halted
0: The counter clock of TIM3 is fed even if the core is halted
1: The counter clock of TIM3 is stopped when the core is halted

Bit 0 **DBG_TIM2_STOP:** TIM2 counter stopped when core is halted
0: The counter clock of TIM2 is fed even if the core is halted
1: The counter clock of TIM2 is stopped when the core is halted

31.15.5 Debug MCU APB2 freeze register (DBGMCU_APB2_FZ)

The DBGMCU_APB2_FZ register is used to configure the MCU under DEBUG. It concerns APB2 peripherals:

- Timer clock counter freeze

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	DBG_HRTIM1_STOP	Res	Res	Res	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res	DBG_TIM1_STOP
							rw				rw	rw	rw		rw

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

Bit 8 **DBG_HRTIM1_STOP:** HRTIM1 counter stopped when core is halted
0: The clock of the HRTIM1 timer counters is fed even if the core is halted
1: The clock of the HRTIM1 timer counters is stopped when the core is halted

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=1, 8,15..17)
0: The clock of the involved timer counter is fed even if the core is halted
1: The clock of the involved timer counter is stopped when the core is halted
Note: Bit1 is reserved.

31.16 TPIU (trace port interface unit)

31.16.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Table 161. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_IOEN	**TRACE_MODE [1:0]**		**PB3 /** JTDO/ TRACESWO	**PE2 /** TRACECK	**PE3 /** TRACED[0]	**PE4 /** TRACED[1]	**PE5 /** TRACED[2]	**PE6 /** TRACED[3]
0	XX	No Trace (default state)	Released (1)	-				
1	00	Asynchronous Trace	TRACESWO	-	-	Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released (1)	TRACECK	TRACED[0]	-	-	-
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

31.17 DBG register map

Table 162. DBG register map and reset values

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE0042000	**DBGMCU_ IDCODE**	REV_ID																Res	Res	Res	Res	DEV_ID											
	Reset value[1]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE0042004	**DBGMCU_CR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.		Res.	Res	Res	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																														0	0	0
0xE004 2008	**DBGMCU_ APB1_FZ**	Res	Res	Res	Res	Res	Res	DBG_CAN_STOP	Res	Res	Res	DBG_I2C1_SMBUS_TIMEOUT	Res	Res	Res	Res	Res	Res	Res	Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res	Res	Res	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res	Res.	Res.DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value							0				0									0	0	0					0	0			0	0
0xE004 200C	**DBGMCU_ APB2_FZ**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_HRTIM1_STOP	Res	Res	Res	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res	DBG_TIM1_STOP
	Reset value																								0				0	0	0		0

1. The reset value is product dependent. For more information, refer to *Section 31.6.1: MCU device ID code*.

32 Device electronic signature

The device electronic signature is stored in the System memory area of the Flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the STM32F334xx microcontroller.

32.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF F7AC

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

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

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

32.2 Flash memory size data register

Base address: 0x1FFF F7CC

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This bitfield indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x040 corresponds to 64 Kbytes.

33 Revision history

Table 163. Document revision history

Date	Revision	Changes
29-Jun-2014	1	Initial release.
29-Sep-2015	2	Added *Section 9: Peripheral interconnect matrix* Updated Section 18.4.24: TIM1 option registers (TIM1_OR), Section 21: High Resolution Timer (HRTIM), *Section 5: Power control (PWR)*, *Section 14: Digital-to-analog converter (DAC1 and DAC2)*
06-Sep-2017	3	Bits 1,2,3,18,21,24,26,27,28,29,31 updated in the following sections: – *Section 11.2.6: External and internal interrupt/event line mapping* and note updated – *Section 11.3.1: Interrupt mask register (EXTI_IMR1)* and note updated – *Section 11.3.2: Event mask register (EXTI_EMR1)* – *Section 11.3.3: Rising trigger selection register (EXTI_RTSR1)* – *Section 11.3.4: Falling trigger selection register (EXTI_FTSR1)* – *Section 11.3.5: Software interrupt event register (EXTI_SWIER1)* – *Section 11.3.6: Pending register (EXTI_PR1)*,*Section 11.3.7: Interrupt mask register (EXTI_IMR2)* and note removed – *Section 11.3.8: Event mask register (EXTI_EMR2)* – *Section 11.3.9: Rising trigger selection register (EXTI_RTSR2)* – *Section 11.3.10: Falling trigger selection register (EXTI_FTSR2)* – *Section 11.3.11: Software interrupt event register (EXTI_SWIER2)* – *Section 11.3.12: Pending register (EXTI_PR2)* Updated *Section 26.6.16: RTC tamper and alternate function configuration register (RTC_TAFCR)* to modify bits 5 and 6.
16-Jun-2020	4	Updated: – Sections order (homogeneous STM32 reference manuals) – *Section 1: Documentation conventions* – *Section 2.2: Memory organization* – *Section 11: Direct memory access controller (DMA)* – *Section 17.6.10: TSC I/O group x counter register (TSC_IOGxCR)* – *Section 21.3.6: Set/reset events priorities and narrow pulses management* – *Section 18.4.7: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)* – *Section 18.4.9: TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)* – *Figure 321: Watchdog block diagram* – *Section 26.3.4: Real-time clock and calendar* – *Section 27.4.1: I2C block diagram* – *Section 27.5: I2C low-power modes* – *Section 27.6: I2C interrupts*

Index

A

ADCx_AWD2CR 305
ADCx_AWD3CR 306
ADCx_CALFACT 307
ADCx_CCR 310
ADCx_CDR 313
ADCx_CFGR 287
ADCx_CR 284
ADCx_CSR 308
ADCx_DIFSEL 306
ADCx_DR 301
ADCx_IER 282
ADCx_ISR 280
ADCx_JDRy 305
ADCx_JSQR 302
ADCx_OFRy 304
ADCx_SMPR1 291
ADCx_SMPR2 293
ADCx_SQR1 296
ADCx_SQR2 297
ADCx_SQR3 299
ADCx_SQR4 300
ADCx_TR1 293
ADCx_TR2 294
ADCx_TR3 295

C

CAN_BTR 1078
CAN_ESR 1077
CAN_FA1R 1088
CAN_FFA1R 1087
CAN_FiRx 1089
CAN_FM1R 1086
CAN_FMR 1086
CAN_FS1R 1087
CAN_IER 1076
CAN_MCR 1069
CAN_MSR 1071
CAN_RDHxR 1085
CAN_RDLxR 1085
CAN_RDTxR 1084
CAN_RF0R 1074
CAN_RF1R 1075
CAN_RIxR 1083
CAN_TDHxR 1082
CAN_TDLxR 1082
CAN_TDTxR 1081
CAN_TIxR 1079
CAN_TSR 1072
COMP2_CSR 347
COMP4_CSR 348
COMP6_CSR 350
CRC_CR 80
CRC_DR 79
CRC_IDR 79
CRC_INIT 80
CRC_POL 81

D

DAC_CR 331
DAC_DHR12L1 336
DAC_DHR12L2 337
DAC_DHR12LD 338
DAC_DHR12R1 335
DAC_DHR12R2 336
DAC_DHR12RD 338
DAC_DHR8R1 336
DAC_DHR8R2 337
DAC_DHR8RD 338
DAC_DOR1 339
DAC_DOR2 339
DAC_SR 339
DAC_SWTRIGR 335
DBGMCU_APB1_FZ 1113
DBGMCU_APB2_FZ 1114
DBGMCU_CR 1112
DBGMCU_IDCODE 1100
DMA_CCRx 186
DMA_CMARx 190
DMA_CNDTRx 189
DMA_CPARx 189
DMA_IFCR 185
DMA_ISR 182

E

EXTI_EMR 202, 206
EXTI_FTSR 203, 207
EXTI_IMR 201, 206
EXTI_PR 205, 208
EXTI_RTSR 202, 206
EXTI_SWIER 204, 207

F

FLASH_ACR 66
FLASH_CR 68
FLASH_KEYR 66
FLASH_OPTKEYR 67
FLASH_SR 67

G

GPIOx_AFRH 153
GPIOx_AFRL 153
GPIOx_BRR 154
GPIOx_BSRR 151
GPIOx_IDR 150
GPIOx_LCKR 151
GPIOx_MODER 148
GPIOx_ODR 151
GPIOx_OSPEEDR 149
GPIOx_OTYPER 149
GPIOx_PUPDR 150

H

HRTIM_ADC1R 782
HRTIM_ADC2R 783
HRTIM_ADC3R 784
HRTIM_ADC4R 786
HRTIM_BDMADR 795
HRTIM_BDMUPR 793
HRTIM_BDTxUPR 794
HRTIM_BMCMPR 777
HRTIM_BMCR 773
HRTIM_BMPER 777
HRTIM_BMTRGR 775
HRTIM_CHPxR 754
HRTIM_CMP1CxR 739
HRTIM_CMP1xR 738
HRTIM_CMP2xR 739
HRTIM_CMP3xR 740
HRTIM_CMP4xR 740
HRTIM_CNTxR 737
HRTIM_CPT1xCR 756
HRTIM_CPT1xR 741
HRTIM_CPT2xCR 757
HRTIM_CPT2xR 741
HRTIM_CR1 764
HRTIM_CR2 766
HRTIM_DTxR 742
HRTIM_EECR1 778
HRTIM_EECR2 780
HRTIM_EECR3 781
HRTIM_EEFxR1 748
HRTIM_EEFxR2 750
HRTIM_FLTINR1 789
HRTIM_FLTINR2 791
HRTIM_FLTxR 763
HRTIM_ICR 768
HRTIM_IER 769
HRTIM_ISR 767
HRTIM_MCMP1R 724
HRTIM_MCMP2R 725
HRTIM_MCMP3R 725
HRTIM_MCMP4R 726
HRTIM_MCNTR 723
HRTIM_MCR 716
HRTIM_MDIER 721
HRTIM_MICR 720
HRTIM_MISR 719
HRTIM_MPER 723
HRTIM_MREP 724
HRTIM_ODISR 771
HRTIM_ODSR 772
HRTIM_OENR 770
HRTIM_OUTxR 760
HRTIM_PERxR 737
HRTIM_REPxR 738
HRTIM_RSTAR 751
HRTIM_RSTBR 753
HRTIM_RSTCR 753
HRTIM_RSTDR 753
HRTIM_RSTER 754
HRTIM_RSTx1R 746
HRTIM_RSTx2R 747
HRTIM_SETx1R 744
HRTIM_SETx2R 746
HRTIM_TIMxCR 727
HRTIM_TIMxDIER 734
HRTIM_TIMxICR 733
HRTIM_TIMxISR 731

I

I2C_CR1 931
I2C_CR2 934
I2C_ICR 943
I2C_ISR 941
I2C_OAR1 937
I2C_OAR2 938
I2C_PECR 944
I2C_RXDR 945
I2C_TIMEOUTR 940
I2C_TIMINGR 939
I2C_TXDR 945-946
IWDG_KR 822

IWDG_PR 823
IWDG_RLR 824
IWDG_SR 825
IWDG_WINR 826

O

OPAMP2_CSR 360

P

PWR_CR 93
PWR_CSR 94

R

RCC_AHBENR 126
RCC_AHBRSTR 134
RCC_APB1ENR 129
RCC_APB1RSTR 124
RCC_APB2ENR 127
RCC_APB2RSTR 123
RCC_BDCR 131
RCC_CFGR 118
RCC_CFGR2 135
RCC_CFGR3 136
RCC_CIR 120
RCC_CR 116
RCC_CSR 132
RTC_ALRMAR 862
RTC_ALRMASSR 873
RTC_ALRMBR 863
RTC_ALRMBSSR 874
RTC_BKPxR 875
RTC_CALR 869
RTC_CR 854
RTC_DR 852
RTC_ISR 857
RTC_PRER 860
RTC_SHIFTR 865
RTC_SSR 864
RTC_TAFCR 870
RTC_TR 851
RTC_TSDR 867
RTC_TSSSR 868
RTC_TSTR 866
RTC_WPR 864
RTC_WUTR 861

S

SPIx_CR1 1041
SPIx_CR2 1043
SPIx_CRCPR 1047
SPIx_DR 1046
SPIx_RXCRCR 1047
SPIx_SR 1045
SPIx_TXCRCR 1047
SYSCFG_EXTICR1 160
SYSCFG_EXTICR2 161
SYSCFG_EXTICR3 163
SYSCFG_EXTICR4 164
SYSCFG_MEMRMP 157, 167

T

TIM1_ARR 464
TIM1_BDTR 467
TIM1_CCER 461
TIM1_CCMR1 454-455
TIM1_CCMR2 458-459
TIM1_CCMR3 472
TIM1_CCR1 465
TIM1_CCR2 466
TIM1_CCR3 466
TIM1_CCR4 467
TIM1_CCR5 473
TIM1_CCR6 474
TIM1_CNT 464
TIM1_CR1 443
TIM1_CR2 444
TIM1_DCR 470
TIM1_DIER 449
TIM1_DMAR 471
TIM1_EGR 453
TIM1_OR 472
TIM1_PSC 464
TIM1_RCR 465
TIM1_SMCR 447
TIM1_SR 451
TIM15_ARR 600
TIM15_BDTR 602
TIM15_CCER 597
TIM15_CCMR1 593-594
TIM15_CCR1 601
TIM15_CCR2 602
TIM15_CNT 600
TIM15_CR1 585
TIM15_CR2 586
TIM15_DCR 604
TIM15_DIER 589
TIM15_DMAR 605
TIM15_EGR 592
TIM15_PSC 600
TIM15_RCR 601

TIM15_SMCR 588
TIM15_SR 590
TIM16_OR 624
TIMx_ARR 542, 619, 817
TIMx_BDTR 621
TIMx_CCER 539, 616
TIMx_CCMR1 . . . . . .533, 535, 613-614
TIMx_CCMR2 537-538
TIMx_CCR1 . . . . . .542, 620
TIMx_CCR2 543
TIMx_CCR3 543
TIMx_CCR4 544
TIMx_CNT 540-541, 618, 816
TIMx_CR1 . . . . . 524, 608, 813
TIMx_CR2 . . . . . 525, 609, 815
TIMx_DCR 545, 623
TIMx_DIER 530, 610, 815
TIMx_DMAR 545, 623
TIMx_EGR 532, 612, 816
TIMx_PSC 541, 619, 817
TIMx_RCR 620
TIMx_SMCR 527
TIMx_SR 531, 611, 816
TSC_CR 373
TSC_ICR 376
TSC_IER 375
TSC_IOASCR 378
TSC_IOCCR 379
TSC_IOGCSR 379
TSC_IOGxCR 380
TSC_IOHCR 377
TSC_IOSCR 378
TSC_ISR 377

U

USART_BRR 1003
USART_CR1 992
USART_CR2 995
USART_CR3 999
USART_GTPR 1003
USART_ICR 1011
USART_ISR 1006
USART_RDR 1012
USART_RQR 1005
USART_RTOR 1004
USART_TDR 1012

W

WWDG_CFR 832
WWDG_CR 831
WWDG_SR 832