EN: This Datasheet/Document is presented by the manufacturer.

Please visit our website for pricing and availability at www.hestore.hu.

PY32F002A series
ARM® 32-bit Cortex®-M0+ MCUs

Reference manual

Contents

1. List of abbreviations for register 14
2. System architecture 15
3. Memory and bus architecture 16
3.1. System architecture 16
3.2. Memory organization 16
3.3. Introduction to memory structure 16
3.4. Embedded SRAM 20
3.5. Flash memory 20
3.6. Boot mode 20
3.6.1. Memory physical mapping 20
3.6.2. Embedded boot loader 21
4. Embedded Flash memory 22
4.1. Key features 22
4.2. Flash memory function introduction 22
4.2.1. Flash structure 22
4.2.2. Flash read operation and access latency 22
4.2.3. Flash program and erase operations 22
4.3. Product Unique identity code (UID) register 26
4.4. Flash option byte 26
4.4.1. Flash option word 26
4.4.2. Flash option byte write 28
4.5. Flash configuration bytes 30
4.5.1. HSI_TRIMMING_FOR_USER 31
4.5.2. Calibration value of temperature sensor 31
4.5.3. HSI_8M/24M_EPPARA0 31
4.5.4. HSI_8M/24M_EPPARA1 31
4.5.5. HSI_8M/24M_EPPARA2 31
4.5.6. HSI_8M/24M_EPPARA3 32
4.5.7. HSI_8M/24M_EPPARA4 32
4.6. Flash protection 32
4.6.1. Flash software development kit (SDK) area protection 32
4.6.2. Flash read protection 33
4.6.3. Flash write protection 34
4.6.4. Option byte write protection 34
4.7. Flash interrupt 35
4.8. Flash register description 35
4.8.1. Flash access control register (FLASH_ACR) 35
4.8.2. Flash key register (FLASH_KEYR) 35

4.8.3. Flash option key register (FLASH_OPTKEYR) 36
4.8.4. Flash status register (FLASH_SR) 36
4.8.5. Flash control register (FLASH_CR) 36
4.8.6. Flash option register (FLASH_OPTR) 38
4.8.7. Flash SDK address register (FLASH_SDKR) 38
4.8.8. Flash WRP address register (FLASH_WRPR) 39
4.8.9. Flash sleep time configuration register (FLASH_STCR) 39
4.8.10. Flash TS0 register (FLASH_TS0) 40
4.8.11. Flash TS1 register (FLASH_TS1) 40
4.8.12. Flash TS2P register (FLASH_TS2P) 40
4.8.13. Flash TPS3 register (FLASH_TPS3) 41
4.8.14. Flash TS3 register (FLASH_TS3) 41
4.8.15. Flash page erase TPE register (FLASH_PERTPE) 42
4.8.16. Flash sector/mass erase TPE register (FLASH_SMERTPE) 42
4.8.17. Flash program TPE register (FLASH_PRGTPE) 42
4.8.18. Flash pre-program TPE register (FLASH_PRETPE) 43
4.8.19. Flash register mapping 43
5. Power control 46
5.1. Power supply 46
5.1.1. Power block diagram 46
5.2. Voltage regulator 46
5.3. Dynamic voltage value management 47
5.4. Power monitoring 47
5.4.1. Power-on reset (POR)/power-down reset (PDR)/brown-out reset (BOR) 47
6. Low-power control 49
6.1. Low-power mode 49
6.1.1. Introduction to low-power modes 49
6.1.2. Low-power mode switch 49
6.1.3. Functions in each working mode 50
6.2. Sleep mode 50
6.2.1. Entering sleep mode 50
6.2.2. Exiting sleep mode 51
6.3. Stop mode 51
6.3.1. Entering stop mode 51
6.3.2. Exiting stop mode 52
6.4. Decreasing system clock frequency 52
6.5. Peripheral clock gating 52
6.6. Power management register 53
6.6.1. Power control register 1 (PWR_CR1) 53
6.6.2. PWR register map 54

7. Reset 55
7.1. Reset source 55
7.1.1. Power reset 55
7.1.2. System reset 55
7.1.3. NRST pin (external reset) 55
7.1.4. Watchdog reset 56
7.1.5. Software reset 56
7.1.6. Option byte loader reset 56
8. Clock 57
8.1. Clock source 57
8.1.1. High-speed external clock (HSE) 57
8.1.2. High-speed internal clock (HSI) 57
8.1.3. Low-speed internal clock (LSI) 57
8.2. Clock tree 57
8.3. Clock security system (CSS) 58
8.4. Clock-out capability 59
8.5. Reset/clock register 59
8.5.1. Clock control register (RCC_CR) 59
8.5.2. Clock configuration register (RCC_CFGR) 60
8.5.3. Internal clock source calibration register (RCC_ICSCR) 61
8.5.4. External clock source control register (RCC_ECSCR) 62
8.5.5. Clock interrupt enable register (RCC_CIER) 62
8.5.6. Clock interrupt flag register (RCC_CIFR) 63
8.5.7. Clock interrupt clear register (RCC_CICR) 63
8.5.8. I/O interface reset register (RCC_IOPRSTR) 64
8.5.9. AHB peripheral reset register (RCC_AHBRSTR) 64
8.5.10. APB peripheral reset register 1 (RCC_APBRSTR1) 65
8.5.11. APB peripheral reset register 2 (RCC_APBRSTR2) 65
8.5.12. I/O interface clock enable register (RCC_IOPENR) 66
8.5.13. AHB peripheral clock enable register (RCC_AHBENR) 66
8.5.14. APB peripheral clock enable register 1 (RCC_APBENR1) 67
8.5.15. APB peripheral clock enable register 2 (RCC_APBENR2) 67
8.5.16. Peripheral independent clock configuration register (RCC_CCIPR) 68
8.5.17. RCC_BDCR 68
8.5.18. Control/status register (RCC_CSR) 69
8.5.19. RCC register address map 69
9. General-purpose I/Os (GPIO) 73
9.1. GPIO introduction 73
9.2. GPIO main features 73
9.3. GPIO functional description 73

9.3.1. General-purpose I/O (GPIO) 74
9.3.2. I/O pin alternate function multiplexer and mapping 74
9.3.3. I/O port control registers 75
9.3.4. I/O port data registers 75
9.3.5. I/O data bitwise handling 75
9.3.6. GPIO locking mechanism 76
9.3.7. I/O alternate function input/output 76
9.3.8. External interrupt/wakeup lines 76
9.3.9. I/O input configuration 76
9.3.10. I/O output configuration 77
9.3.11. Alternate function configuration 78
9.3.12. Analog configuration 79
9.3.13. Use the HSE oscillator pins as GPIOs 80
9.4. GPIO registers 80
9.4.1. GPIO port mode register (GPIOx_MODER) (x = A, B, F) 80
9.4.2. GPIO port output type register (GPIOx_OTYPER) (x = A, B, F) 80
9.4.3. GPIO port output speed register (GPIOx_OSPEEDR) (x = A, B, F) 81
9.4.4. GPIO port pull-up and pull-down register (GPIOx_PUPDR) (x = A, B, F) 81
9.4.5. GPIO port input data register (GPIOx_IDR) (x = A, B, F) 81
9.4.6. GPIO port output data register (GPIOx_ODR) (x = A, B, F) 82
9.4.7. GPIO port bit set/reset register (GPIOx_BSRR) (x = A, B, F) 82
9.4.8. GPIO port configuration lock register (GPIOx_LCKR) (x = A, B, F) 82
9.4.9. GPIO alternate function register (low) (GPIOx_AFRL) (x = A, B, F) 83
9.4.10. GPIO alternate function register (high) (GPIOx_AFRH) (x = A, B, F) 83
9.4.11. GPIO port bit reset register (GPIOx_BRR) (x = A, B, F) 84
9.4.12. GPIO register map 84
10. System configuration controller (SYSCFG) 87
10.1. System configuration register 87
10.1.1. SYSCFG configuration register 1 (SYSCFG_CFGR1) 87
10.1.2. SYSCFG configuration register 2 (SYSCFG_CFGR2) 87
10.1.3. SYSCFG register map 88
11. Interrupts and events 90
11.1. Nested vectored interrupt controller (NVIC) 90
11.1.1. NVIC main features 90
11.1.2. SysTick calibration value register 90
11.1.3. Interrupt and exception vectors 90
11.2. Extended interrupts and events controller (EXTI) 91
11.2.1. EXTI main features 91
11.2.2. EXTI diagram 91
11.2.3. EXTI connection between peripherals and CPU 92

11.2.4. EXTI configurable event trigger wake-up 92
11.2.5. EXTI direct type event input wakeup 93
11.2.6. External and internal interrupt/event line mapping 93
11.3. EXTI registers 94
11.3.1. Rising trigger selection register (EXTI_RTSR) 94
11.3.2. Falling trigger selection register (EXTI_FTSR) 95
11.3.3. Software interrupt event register (EXTI_SWIER) 96
11.3.4. Pending register (EXTI_PR) 98
11.3.5. External interrupt select register 1 (EXTI_EXTICR1) 101
11.3.6. External interrupt select register 2 (EXTI_EXTICR2) 101
11.3.7. External interrupt select register 3 (EXTI_EXTICR3) 102
11.3.8. Interrupt mask register (EXTI_IMR) 102
11.3.9. Event mask register (EXTI_EMR) 103
11.3.10. EXTI register map 105
12. Cyclic redundancy check calculation unit (CRC) 107
12.1. Introduction 107
12.2. CRC main features 107
12.3. CRC functional description 107
12.3.1. CRC block diagram 107
12.4. CRC registers 108
12.4.1. Data register (CRC_DR) 108
12.4.2. Independent data register (CRC_IDR) 108
12.4.3. Control register (CRC_CR) 108
12.4.4. CRC register map 109
13. Analog-to-digital converter (ADC) 110
13.1. Introduction 110
13.2. ADC main features 110
13.3. ADC functional description 111
13.3.1. ADC diagram 111
13.3.2. Calibration (ADCAL) 112
13.3.3. ADC on-off control (ADEN) 112
13.3.4. ADC click 113
13.3.5. Configuring the ADC 114
13.3.6. Channel selection (CHSEL, SCANDIR) 114
13.3.7. Programmable sampling time (SMP) 114
13.3.8. Single conversion mode (CONT = 0, DISCEN = 0) 115
13.3.9. Continuous conversion mode (CONT = 1) 115
13.3.10. Discontinuous conversion mode (DISCEN = 1) 116
13.3.11. Starting conversions (ADSTART) 116
13.3.12. Timings 117

13.3.13. Stopping an ongoing conversion (ADSTP) 117
13.4. Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) 118
13.4.1. Programmable resolution (RES) - fast conversion mode 118
13.4.2. End of conversion, end of sampling phase (EOC, EOSMP flags) 118
13.4.3. End of conversion sequence (EOSEQ flag) 119
13.4.4. Example timing diagrams 119
13.5. Data management 121
13.5.1. Data register and data alignment (ADC_DR, ALIGN) 121
13.5.2. ADC overrun (OVR, OVRMOD) 121
13.5.3. Managing a sequence of data converted 122
13.5.4. Managing converted data 122
13.6. Low-power features 123
13.6.1. Wait mode conversion 123
13.7. Analog window watchdog 123
13.7.1. ADC_AWD_OUT signal output generation 124
13.8. Temperature sensor and internal reference voltage 125
13.9. ADC interrputs 126
13.10. ADC registers 126
13.10.1. ADC interrupt and status register (ADC_ISR) 126
13.10.2. ADC interrupt enable register (ADC_IER) 127
13.10.3. ADC control register (ADC_CR) 128
13.10.4. ADC configuration register 1 (ADC_CFGR1) 129
13.10.5. ADC configuration register 2 (ADC_CFGR2) 131
13.10.6. ADC sampling time register (ADC_SMPR) 131
13.10.7. ADC watchdog threshold register (ADC_TR) 132
13.10.8. ADC channel selection register (ADC_CHSELR) 132
13.10.9. ADC data register (ADC_DR) 133
13.10.10. ADC calibration configuration and status registers (ADC_CCSR) 133
13.10.11. ADC common configuration register (ADC_CCR) 134
13.10.12. ADC register map 134
14. Advanced-control timer (TIM1) 136
14.1. TIM1 introduction 136
14.2. TIM1 main features 136
14.3. TIM1 functional description 137
14.3.1. Time-base unit 137
14.3.2. Counter modes 139
14.3.3. Repetition counter 146
14.3.4. Clock selection 148
14.3.5. Capture/compare channels 150
14.3.6. Input capture mode 151

14.3.7. PWM input mode 152
14.3.8. Forced output mode 153
14.3.9. Output compare mode 153
14.3.10. PWM mode 154
14.3.11. Complementary outputs and dead-time insertion 156
14.3.12. Using the break function 158
14.3.13. Clearing the OCxREF signal on an external event 160
14.3.14. 6-step PWM generation 161
14.3.15. One-pulse mode 162
14.3.16. Encoder interface mode 163
14.3.17. Timer input XOR function 165
14.3.18. Interfacing with Hall sensors 165
14.3.19. TIMx and external trigger synchronization 166
14.3.20. Timer synchronization 169
14.3.21. Debug mode 169
14.4. TIM1 registers 169
14.4.1. TIM1 control register 1 (TIM1_CR1) 169
14.4.2. TIM1 control register 2 (TIM1_CR2) 171
14.4.3. TIM1 slave mode control register (TIM1_SMCR) 172
14.4.4. TIM1 interrupt enable register (TIM1_DIER) 174
14.4.5. TIM1 status register (TIM1_SR) 174
14.4.6. TIM1 event generation register (TIM1_EGR) 176
14.4.7. TIM1 capture/compare mode register 1 (TIM1_CCMR1) 177
14.4.8. TIM1 capture/compare mode register 2 (TIM1_CCMR2) 180
14.4.9. TIM1 capture/compare enable register (TIM1_CCER) 181
14.4.10. TIM1 counter (TIM1_CNT) 183
14.4.11. TIM1 prescaler (TIM1_PSC) 184
14.4.12. TIM1 auto-reload register (TIM1_ARR) 184
14.4.13. TIM1 repetition counter register (TIM1_RCR) 184
14.4.14. TIM1 capture/compare register 1 (TIM1_CCR1) 185
14.4.15. TIM1 capture/compare register 2 (TIM1_CCR2) 185
14.4.16. TIM1 capture/compare register 3 (TIM1_CCR3) 186
14.4.17. TIM1 capture/compare register 4 (TIM1_CCR4) 186
14.4.18. TIM1 break and dead-time register (TIM1_BDTR) 187
14.4.19. TIM1 register map 188
15. Comparator (COMP) 191
15.1. Introduction 191
15.2. COMP main features 191
15.3. COMP function description 191
15.3.1. COMP diagram 191

15.3.2. COMP pins and internal signals 192
15.3.3. COMP reset and clock 192
15.3.4. COMP lock mechanism 192
15.3.5. Window comparator 193
15.3.6. Hysteresis 193
15.3.7. Power modes 193
15.3.8. Comparator filtering 194
15.3.9. COMP interrupt 194
15.4. COMP registers 194
15.4.1. COMP1 control and status registers (COMP1_CSR) 194
15.4.2. COMP1 filter register (COMP1_FR) 195
15.4.3. COMP2 control and status register (COMP2_CSR) 196
15.4.4. COMP2 filter register (COMP2_FR) 197
15.4.5. COMP register map 197
16. General-purpose timers (TIM16) 199
16.1. TIM16 main features 199
16.2. TIM16 functional description 199
16.2.1. Time-base unit 199
16.2.2. Counter operation 200
16.2.3. Repetition counter 203
16.2.4. Clock sources 204
16.3. TIM16 registers 204
16.3.1. TIM16 control register 1 (TIMx_CR1) 204
16.3.2. TIM16 interrupt enable register (TIM16_DIER) 205
16.3.3. TIM16 status register (TIM16_SR) 206
16.3.4. TIM16 event generation register (TIM16_EGR) 206
16.3.5. TIM16 counter (TIM16_CNT) 206
16.3.6. TIM16 prescaler (TIM16_PSC) 207
16.3.7. TIM16 auto-reload register (TIM16_ARR) 207
16.3.8. TIM16 repetition counter register (TIM16_RCR) 207
16.3.9. TIM16 register map 208
17. Low power timer (LPTIM) 210
17.1. Introduction 210
17.2. LPTIM main features 210
17.3. LPTIM functional description 210
17.3.1. LPTIM block diagram 210
17.3.2. LPTIM reset and clock 210
17.3.3. Prescaler 211
17.3.4. Operating mode 211
17.3.5. Register update 211

17.3.6. Enable timer 211
17.3.7. Counter reset INDANG 211
17.3.8. Debug mode 212
17.4. LPTIM low power mode 212
17.5. LPTIM interrupt 212
17.6. LPTIM register 212
17.6.1. LPTIM interrupt and status register (LPTIM_ISR) 212
17.6.2. LPTIM interrupt clear register (LPTIM_ICR) 213
17.6.3. LPTIM interrupt enable register (LPTIM_IER) 213
17.6.4. LPTIM configuration register (LPTIM_CFGR) 213
17.6.5. LPTIM control register (LPTIM_CR) 214
17.6.6. LPTIM auto-reload register (LPTIM_ARR) 214
17.6.7. LPTIM counter (LPTIM_CNT) 215
17.6.8. LPTIM register map 215
18. Independent watchdog (IWDG) 217
18.1. Introduction 217
18.2. IWDG main features 217
18.3. IWDG functional description 217
18.3.1. IWDG block diagram 217
18.3.2. Hardware watchdog 217
18.3.3. Register access protection 218
18.3.4. Debug mode 218
18.4. IWDG registers 218
18.4.1. Key register (IWDG_KR) 218
18.4.2. Prescaler register (IWDG_PR) 218
18.4.3. Reload register (IWDG_RLR) 219
18.4.4. Status register (IWDG_SR) 219
18.4.5. IWDG register map 219
19. Inter-integrated circuit (I2C) interface 221
19.1. Introduction 221
19.2. I2C main features 221
19.3. I2C functional description 221
19.3.1. I2C block diagram 222
19.3.2. Mode selection 222
19.3.3. I2C initialization 223
19.3.4. I2C slave mode 223
19.3.5. I2C master mode 225
19.3.6. Error stage 230
19.3.7. SDA/SCL control 231
19.4. I2C interrupts 231

19.5. I2C registers 231
19.5.1. I2C control register 1 (I2C_CR1) 231
19.5.2. I2C control register 2 (I2C_CR2) 233
19.5.3. I2C own address register 1 (I2C_OAR1) 234
19.5.4. I2C data register (I2C_DR) 234
19.5.5. I2C stage register 1 (I2C_SR1) 235
19.5.6. I2C stage registor 2 (I2C_SR2) 237
19.5.7. I2C clock control register (I2C_CCR) 238
19.5.8. I2C TRISE register (I2C_TRISE) 238
19.5.9. I2C register map 239
20. Universal synchronous asynchronous receiver transmitter (USART) 240
20.1. Introduction 240
20.2. USART main features 240
20.3. USART function description 241
20.3.1. USART character description 242
20.3.2. Transmitter 243
20.3.3. Recevier 246
20.3.4. USART baud rate generation 249
20.3.5. USART receiver's tolerance to clock deviation 249
20.3.6. USART auto baud rate detection 250
20.3.7. Multiprocessor communication using USART 251
20.3.8. USART synchronous mode 253
20.3.9. USART single-wire half-duplex communication 255
20.3.10. Hardware flow control 255
20.4. USART interrupt request 257
20.5. USART register 258
20.5.1. Status register (USART_SR) 258
20.5.2. UASRT data register (USART_DR) 260
20.5.3. Baud rate register (USART_BRR) 260
20.5.4. USART control register 1 (USART_CR1) 260
20.5.5. USART control register 2 (USART_CR2) 262
20.5.6. USART control register 3 (USART_CR3) 262
20.5.7. USART register map 263
21. Serial peripheral interface (SPI) 265
21.1. Introduction 265
21.2. SPI main features 265
21.3. SPI function description 265
21.3.1. Overview 265
21.3.2. Communications between one master and one slave 266
21.3.3. Multi-slave communication 268

21.3.4. Multi-master communication 269
21.3.5. Slave select (NSS) pin management 270
21.3.6. Communication formats 271
21.3.7. SPI configuration 272
21.3.8. Procedure for enabling SPI 273
21.3.9. Data transmission and reception procedures 273
21.3.10. Status flags 276
21.3.11. Error flags 277
21.3.12. SPI interrupts 278
21.4. SPI register 278
21.4.1. SPI control register 1 (SPI_CR1) 278
21.4.2. SPI control register 2 (SPI_CR2) 279
21.4.3. SPI status register (SPI_SR) 280
21.4.4. SPI data register (SPI_DR) 281
21.4.5. SPI register map 282
22. Debug support 283
22.1. Overview 283
22.2. Pinout and debug port pins 283
22.2.1. SWD port pins 283
22.2.2. SW-DP pin assignment 284
22.2.3. Internal pull-up & pull-down on SWD pins 284
22.3. ID codes and locking mechanism 284
22.4. SWD debug port 284
22.4.1. SWD protocol introduction 284
22.4.2. SWD protocol sequence 284
22.4.3. SW-DP state machine (reset, idle states, ID code) 285
22.4.4. DP and AP read/write accesses 285
22.4.5. SW-DP registers 285
22.4.6. SW-AP registers 286
22.5. Core debug 286
22.6. Break point unit (BPU) 286
22.6.1. BPU functionality 287
22.7. Data watchpoint (DWT) 287
22.7.1. DWT functionality 287
22.7.2. DWT program counter sample register 287
22.8. MCU debug component (DBGMCU) 287
22.8.1. Debug support for low-power modes 287
22.8.2. Debug support for times, watchdog and IIC 287
22.9. DBG register 288
22.9.1. DBG device ID code register (DBG_IDCODE) 288

22.9.2. Debug MCU configuration register (DBGMCU_CR) 288
22.9.3. DBG APB freeze register 1 (DBG_APB_FZ1) 288
22.9.4. DBG APB freeze register 2 (DBG_APB_FZ2) 289
22.9.5. DBG register map 289
23. Version history 291

1. List of abbreviations for register

Abbreviation	Describe
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 register. Writing to this bit has no effect.
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 clears it to ‘0’. Writing this bit has no effect on the bit value.
Read/set (rs)	Software can read as well as set this bit to ‘1’. Writing ‘0’ has no effect on the bit value.
Toggle (t)	Software can toggle this bit by writing ‘1’. Writing ‘0’ has no effect.
Reserved (Res)	Reserved bit, must be kept at reset value.

2. System architecture

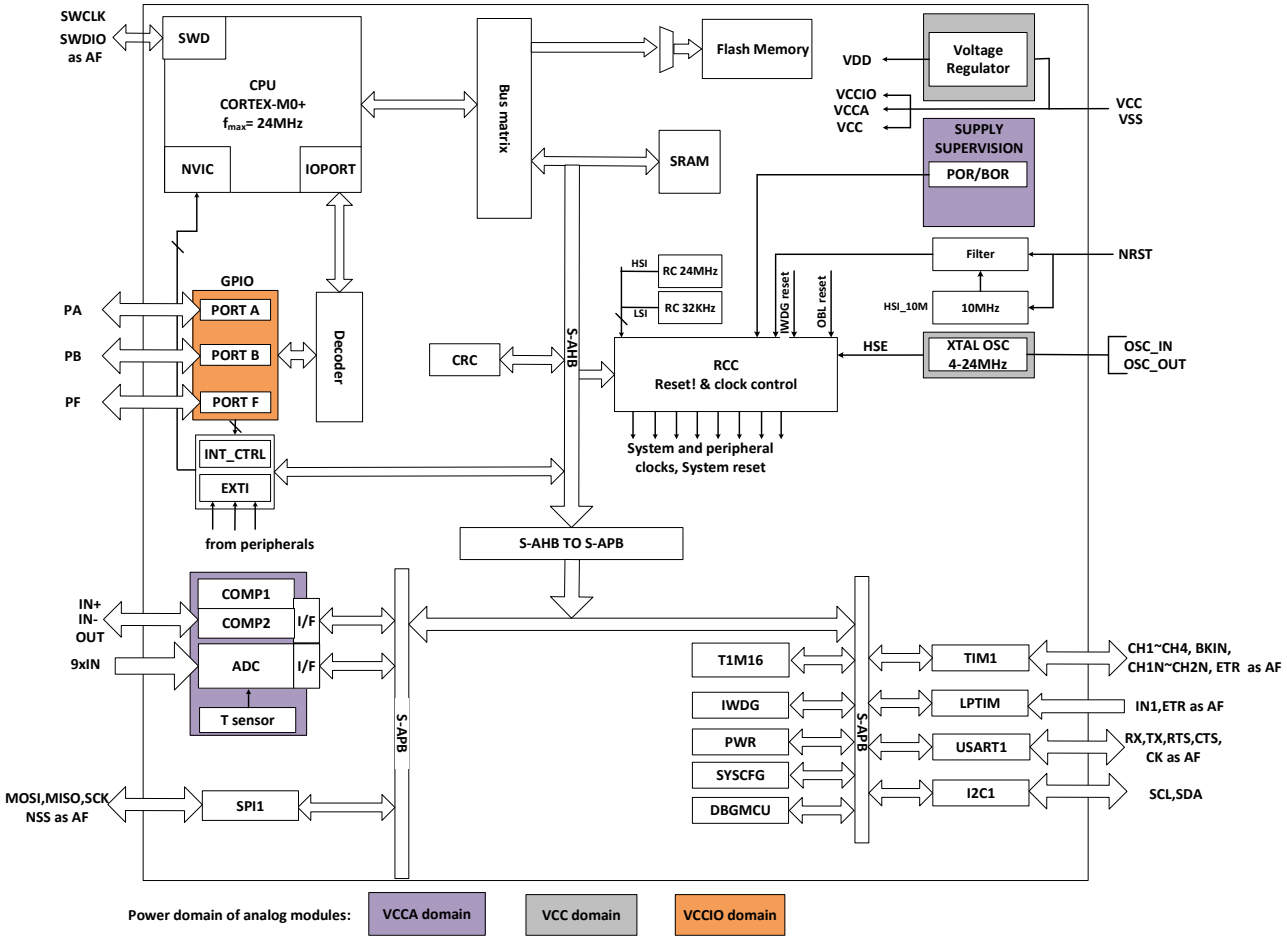


Figure 2-1 System architecture

3. Memory and bus architecture

3.1. System architecture

The system consists of the following parts:

- A masters:
  - Cortex-M0+
- Three Slaves
  - Internal SRAM
  - Internal Flash memory
  - AHB with AHB-APB Bridge

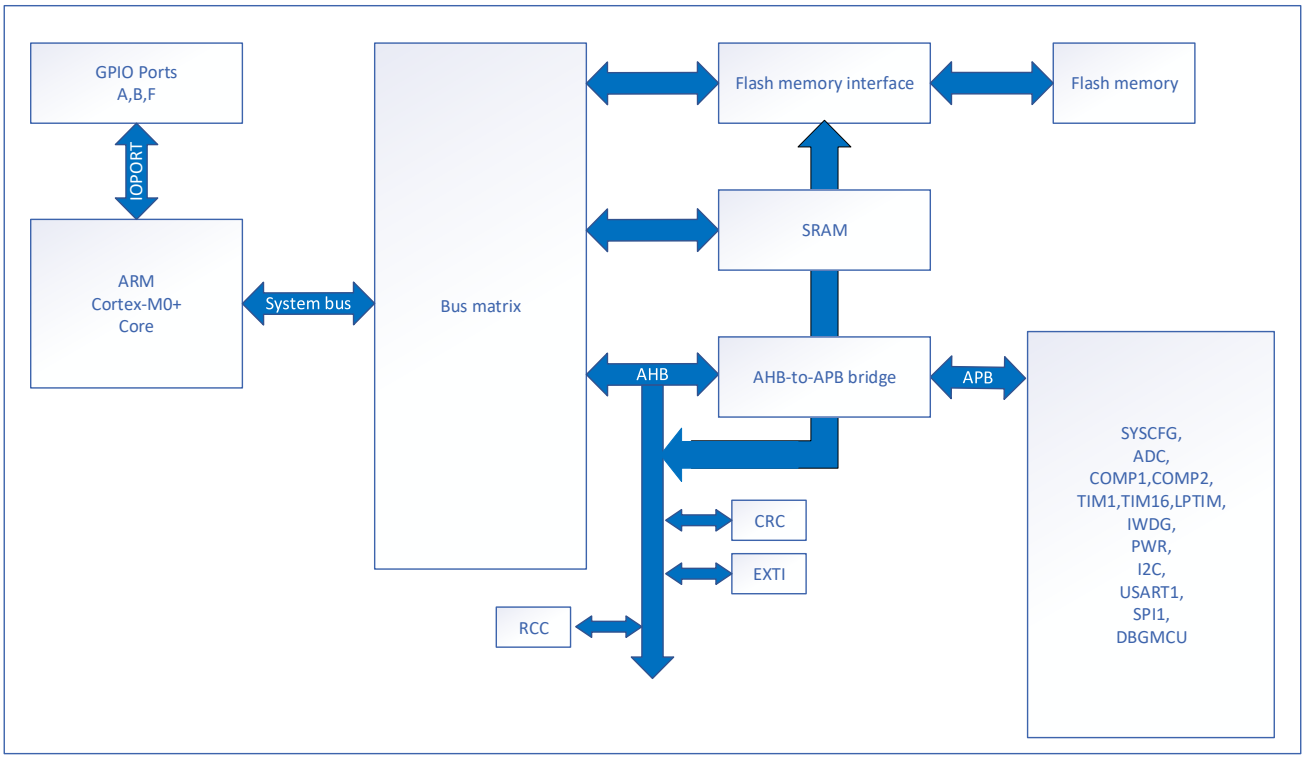


Figure 3-1 System architecture

- System bus

The bus architecture (M0 + of the system bus is connected to the bus matrix, which is used to manage the CPU to arbitration.

- Bus Matrix

The Bus Matrix manages the access arbitration the CPU bus. This arbitration uses the Round Robin algorithm. The Bus Matrix is composed of masters (CPU) and slaves (SRAM, Flash memory and AHB-to-APB bridge).

- AHB-to-APB bridge (APB)

The AHB-to-APB bridge provides a synchronous connection between the AHB and APB buses to the peripheral address mapping of the Bridge.

3.2. Memory organization

3.3. Introduction to memory structure

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbytes address space. The bytes are coded in memory in Little Endian format (in a word, the lowest numbered byte is considered the world's least sinificant byte).

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

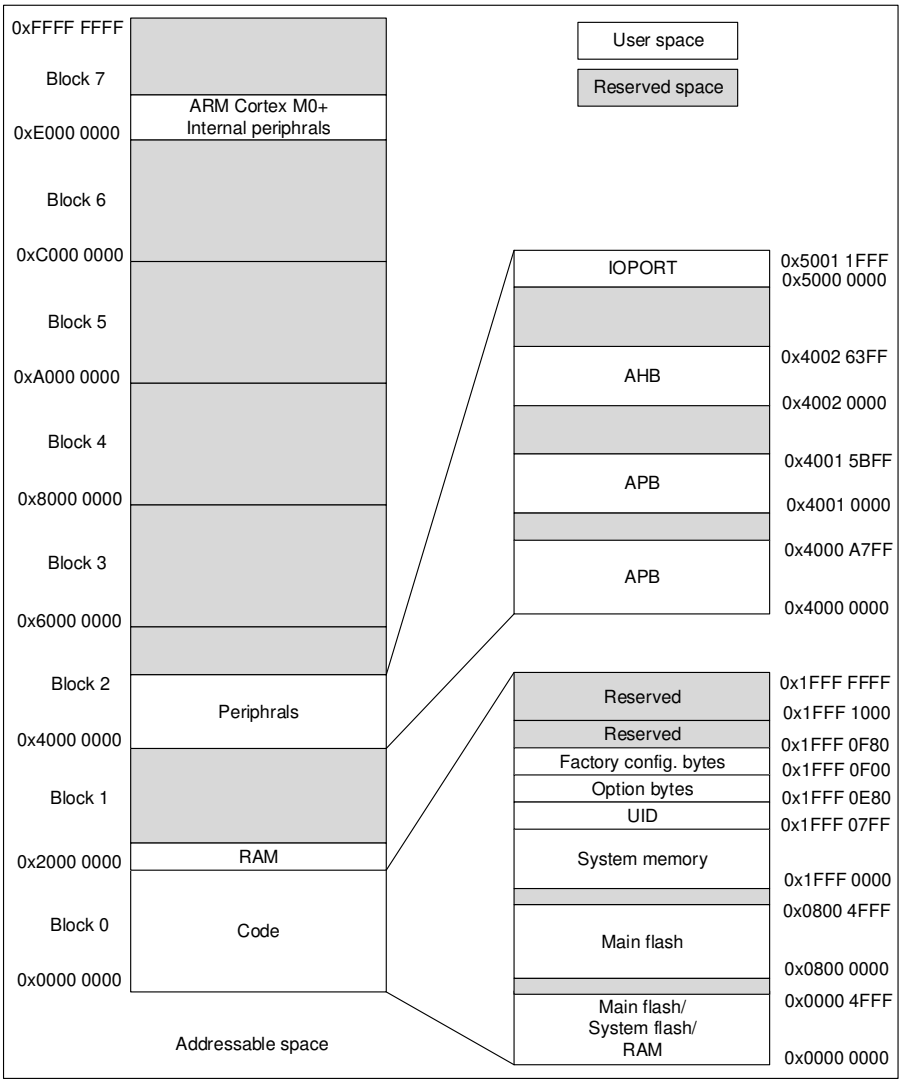


Figure 3-2 Memory map

Table 3-1 Memory boundary addresses

Type	Boundary Address	Size	Memory Area	Description
SRAM	0x2000 0C00-0x3FFF FFFF	512MBytes	Reserved	
	0x2000 0000-0x2000 0BFF	3KBytes	SRAM	根据硬件不同，SRAM 最大为 3kBytes
Code	0x1FFF 1000-0x1FFF FFFF	4KBytes	Reserved	
	0x1FFF 0F80-0x1FFF 0FFF	128Bytes	Reserved	
	0x1FFF 0F00-0x1FFF 0F7F	128Bytes	Factory config	存放 HSI triming 数据、flash 擦写时间配置参数
	0x1FFF 0E80-0x1FFF 0EFF	128Bytes	Option bytes	option bytes
	0x1FFF 0E00-0x1FFF 0E7F	128Bytes	UID	Unique ID
	0x1FFF 0000-0x1FFF 07FF	2KBytes	System memory	存放 boot loader
	0x0800 8000-0x1FFE FFFF	384MBytes	Reserved	
	0x0800 0000-0x0800 4FFF	20KBytes	Main flash memory	
	0x0000 5000-0x07FF FFFF	8MBytes	Reserved	

Type	Boundary Address	Size	Memory Area	Description
	0x0000 0000-0x0000 4FFF	20KBytes	根据 Boot 配置选择： 1）Main flash memory 2）System memory 3）SRAM	

Note:

Except for 0x1FFF 0E00 - 0x1FFF 0E7F, the above spaces are marked as reserved spaces, which cannot be written and read as 0 with a response error occurs.

Table 3-2 Peripheral register address

Bus	Boundary Address	Size	Peripheral
	0xE000 0000-0xE00F FFFF	1Mbytes	M0+
IOPORT	0x5000 1800-0x5FFF FFFF	256MBytes	Reserved(1)
	0x5000 1400-0x5000 17FF	1KBytes	GPIOF
	0x5000 1000-0x5000 13FF	1KBytes	Reserved
	0x5000 0C00-0x5000 0FFF	1Kbytes	Reserved
	0x5000 0800-0x5000 0BFF	1Kbytes	Reserved
	0x5000 0400-0x5000 07FF	1Kbytes	GPIOB
	0x5000 0000-0x5000 03FF	1Kbytes	GPIOA
AHB	0x4002 3400-0x4FFF FFFF		Reserved
	0x4002 300C-0x4002 33FF	1Kbytes	Reserved
	0x4002 3000-0x4002 3008		CRC
	0x4002 2400-0x4002 2FFF		Reserved
	0x4002 2124-0x4002 23FF	1KBytes	Reserved
	0x4002 2000-0x4002 2120		Flash
	0x4002 1C00-0x4002 1FFF	3KBytes	Reserved
	0x4002 1888-0x4002 1BFF	1Kbytes	Reserved
	0x4002 1800-0x4002 1884		EXTI（2）
	0x4002 1400-0x4002 17FF	1Kbytes	Reserved
	0x4002 1064-0x4002 13FF	1KBytes	Reserved
	0x4002 1000-0x4002 1060		RCC（2）
	0x4002 0C00-0x4002 0FFF	1KBytes	Reserved
	0x4002 0040-0x4002 03FF	1KBytes	Reserved
	0x4002 0000-0x4002 003C		Reserved
APB	0x4001 5C00-0x4001 FFFF	32KBytes	Reserved
	0x4001 5880-0x4001 5BFF	1KBytes	Reserved
	0x4001 5800-0x4001 587F		DBG
	0x4001 4C00-0x4001 57FF	3KBytes	Reserved
	0x4001 4850-0x4001 4BFF	1KBytes	Reserved
	0x4001 4800-0x4001 484C		Reserved
	0x4001 4450-0x4001 47FF	1KBytes	Reserved
	0x4001 4400-0x4001 404C		TIM16
	0x4001 3C00-0x4001 43FF	2KBytes	Reserved
	0x4001 381C-0x4001 3BFF	1KBytes	Reserved
	0x4001 3800-0x4001 3018		USART1
	0x4001 3400-0x4001 37FF	1Kbytes	Reserved
	0x4001 3010-0x4001 33FF	1Kbytes	Reserved
	0x4001 3000-0x4001 300C		SPI1

Bus	Boundary Address	Size	Peripheral
	0x4001 2C50-0x4001 2FFF	1Kbytes	Reserved
	0x4001 2C00-0x4001 2C4C		TIM1
	0x4001 2800-0x4001 2BFF	1Kbytes	Reserved
	0x4001 270C-0x4001 27FF	1Kbytes	Reserved
	0x4001 2400-0x4001 2708		ADC
	0x4001 0400-0x4001 23FF	8Kbytes	Reserved
	0x4001 0220-0x4001 03FF	1KBytes	Reserved
	0x4001 0200-0x4001 021F		COMP1 and COMP2
	0x4001 0000-0x4001 01FF		SYSCFG
	0x4000 B400-0x4000 FFFF	19KBytes	Reserved
	0x4000 B000-0x4000 B3FF	1KBytes	Reserved
	0x4000 8400-0x4000 AFFF	11KBytes	Reserved
	0x4000 8000-0x4000 83FF	1KBytes	Reserved
	0x4000 7C28-0x4000 7FFF	1KBytes	Reserved
	0x4000 7C00-0x4000 7C24		LPTIM
	0x4000 7400-0x4000 7BFF	2KBytes	Reserved
	0x4000 7018-0x4000 73FF	1KBytes	Reserved
	0x4000 7000-0x4000 7014		PWR (3)
	0x4000 5800-0x4000 6FFF	6KBytes	Reserved
	0x4000 5434-0x4000 57FF	1KBytes	Reserved
	0x4000 5400-0x4000 5430		I2C
	0x4000 4800-0x4000 53FF	3KBytes	Reserved
	0x4000 441C-0x4000 47FF	1KBytes	Reserved
	0x4000 4400-0x4000 4418		Reserved
	0x4000 3C00-0x4000 43FF	1KBytes	Reserved
	0x4000 3800-0x4000 3BFF	1KBytes	Reserved
	0x4000 3400-0x4000 37FF	1KBytes	Reserved
	0x4000 3014-0x4000 33FF	1KBytes	Reserved
	0x4000 3000-0x4000 0010		IWDG
	0x4000 2C0C-0x4000 2FFF	1KBytes	Reserved
	0x4000 2C00-0x4000 2C08		Reserved
	0x4000 2830-0x4000 2BFF	1KBytes	Reserved
	0x4000 2800-0x4000 282C		Reserved
	0x4000 2400-0x4000 27FF	1KBytes	Reserved
	0x4000 2054-0x4000 23FF	1KBytes	Reserved
	0x4000 2000-0x4000 0050		Reserved
	0x4000 1800-0x4000 1FFF	2KBytes	Reserved
	0x4000 1400-0x4000 17FF	1KBytes	Reserved
	0x4000 1000-0x4000 13FF	1KBytes	Reserved
	0x4000 0800-0x4000 0FFF	2KBytes	Reserved
	0x4000 0450-0x4000 07FF	1Kbytes	Reserved
	0x4000 0400-0x4000 044C		Reserved
	0x4000 0000-0x4000 03FF	1KBytes	Reserved

Note :

(1) In the above table, the address space which marked as Reserved in AHB, cannot be written, read is 0 and a hardfault is generated. And in APB is marked as the reserved address space, cannot be written, read is 0, and no hardfault will be generated.

(2) Not only supports 32 bits word access, but also supports halfword and byte access.

(3) Not only supports 32 bits word access, but also supports half word access.

3.4. Embedded SRAM

The PY32F002A features up to 3 kbytes of SRAM. It can be accessed as bytes, half-word (16 bits) or full words (32 bits). A hard fault will be generated when the software reads and writes the space outside the setting range.

3.5. Flash memory

Flash memory consists of two physical areas:

- Main Flash area, 20 kbytes, it contains application and user data.
- Information area, 2.7 kbytes, it includes the following parts:
  - Factory config. bytes: 128 bytes, used to store trimming data (including HSI triming data), power-on reading check code, etc.
  - UID: 128 bytes, used to store the UID of the chip
  - Option bytes: 128 bytes, used to store the configuration values of hardware and storage protection
  - System memory (system memory): 2 kbytes, used to store Boot loader

Flash memory interface implements instruction of reading and data access based on the AHB protocol, and it also implements the basic program/erase operations of the Flash through registers.

3.6. Boot mode

Three different boot mode can be selected through the BOOT0 pin and boot selector option bit nBOOT1 (stored in the Option bytes), as shown in the following table:

Table 3-3 Boot mood

Boot mode selector pins		Mode
nBOOT1 bit	BOOT0 pin	
X	0	Main Flash memory is selected as the boot area
1	1	System memory is seleced as the boot area
0	1	Embedded SRAM is selected as the boot area

The values on the Boot pins are latched on the 4th SYSCLK after a reset. It is up to the user to set the boot mode to choose according to the table above.

After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code executes from the boot memory starting from 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 word, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF 0000).
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but still accessible at address 0x2000 0000.

3.6.1. Memory physical mapping

If boot mode is selected, the application software can modify the memory accessible in the program space. This modification is determined by the MEM_MODE bit selection in the SYSCFG_CFGR1 register (see the SYSCFG chapter for details).

3.6.2. Embedded boot loader

The embedded boot loader is located in the System memory, programmed during production. It is used to re-program the Flash memory using the following serial interface:

- USART corresponding to PA14/PA15 or PA9/PA10 or PA2/PA3.

4. Embedded Flash memory

4.1. Key features

- Main Flash block: maximum 20 kbytes (5 k x 32 bits)
- Information block: 2.7 kbytes (0.675 k x 32 bits)
- Page size: 128 bytes
- Sector size: 4 kbytes

The Flash control interface circuit features:

- Flash write and erase
- Read protection
- Write protection

4.2. Flash memory function introduction

4.2.1. Flash structure

Flash memory is composed of 32-bit wide storage units, which can be used for program and data storage.

In terms of function, Flash memory is divided into main Flash and information Flash, the former has a maximum capacity of 20 kbytes.Page erase operation can be applied to main Flash.

Mass erase can be applied to main Flash if there is no write protection setting, otherwise it cannot be applied to main Flash.

Table 4-1 Flash structure and boundary addresses

Block	扇区 sector	页 Page	Base address	Size
Main flash	Sector 0	Page 0-31	0x0800 0000-0x0800 0FFF	4Kbytes
	Sector 1	Page 32-63	0x0800 1000-0x0800 1FFF	4Kbytes
	Sector 2	Page 64-95	0x0800 2000-0x0800 2FFF	4Kbytes
	Sector 3	Page 96-127	0x0800 3000-0x0800 3FFF	4Kbytes
	Sector4	Page 128-159	0x0800 4000-0x0800 4FFF	4Kbytes
System flash	Sector 16	Page 0-27	0x1FFF 0000-0x1FFF 0DFF	3.5Kbytes
UID		Page 28	0x1FFF 0E00-0x1FFF 0E7F	128bytes
Option bytes		Page 29	0x1FFF 0E80-0x1FFF 0EFF	128bytes
Factory config		Page 30	0x1FFF 0F00-0x1FFF 0F7F	128bytes
Reserved		Page 31	0x1FFF 0F80-0x1FFF 0FFF	128bytes

4.2.2. Flash read operation and access latency

Flash can be used as a general memory space to accessed direct addressing. The contents of the Flash memory can be read through a special read control sequence.

The instruction fetch and data access are both done through the AHB bus. Read can mange through the Latency of the FLASH_ACR register, which is the read operation increase the wait state or not. When it is 0, the wait state of the Flash read operation is not added, when it is 1, the Flash read operation adds one wait state. This mechanism is specially designed to match high-speed system clock and relatively low-speed Flash read speed.

4.2.3. Flash program and erase operations

The Flash memory can be programmed by In -circuit programming (ICP) or In -application programming (IAP).

ICP: It is used to update the entire contents of the Flash memory, using the SWD protocol or the boot loader to load the user application into the MCU. ICP provides quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

IAP: It can use any communication interface supported by the microcontroller to download programming data into Flash memory. The IAP allows the user to re-program the Flash memory while the application is running. Then, part of the application has to have been previously programmed in the Flash memory using ICP.

If a reset occurs during Flash program and erase operations, the contents of the Flash memory are not protected.

During a program and erase operations to the Flash memory, any attempt to read the Flash memory will stall the bus. The read operation will proceed correctly once the program and erase operations has completed. This means that code or data fetches cannot be made while programming and erasing operations are in progress.

For program and erase operations, the HSI must be turned on.

Program and erase operations can be implemented through the following control interface-related registers :

- Access control register (FLASH_ACR)
- KEY register (FLASH_KEYR)
- Option byte key register (FLASH_OPTKEYR)
- Flash status register (FLASH_SR)
- Flash control register (FLASH_CR)
- Flash option register (FLASH_OPTR)
- Flash special area address register (FLASH_SAR)
- Flash w rite protection resister (FLASH_WRPR)
- Flash TS0 register (FLASH_TS0)
- Flash TS1 register (FLASH_TS1)
- TS2P register (FLASH_TS2P)
- Flash TPS3 register (FLASH_TPS3)
- Flash TS3 register (FLASH_TS3)
- Flash page erase TPE register (FLASH_PERTPE)
- Flash sector/mass erase TPE register (FLASH_SMERTPE)
- Flash program TPE register (FLASH_PRGTPE)
- Flash pre-program TPE register (FLASH_PRETPE)

4.2.3.1. Unlocking the Flash memory

After reset, the Flash memory is protected against unwanted (like caused by electrical interference) write or erase operations. The FLASH_CR register is not accessible in write mode, except for the OB L_LAUNCH bits, used to reload option bit. Every time to write or erase the Flash, must write the FLASH_KEYR register, to generate an unlock sequence, and to open the access to the FLASH_CR register.

This sequence consists of two steps:

Step 1: Write KEY1 = 0x4567 0123 to the FLASH_KEYR register

Step 2: Write KEY2 = 0xCDEF 89AB to the FLASH_KEYR register

Any wrong sequence locks up 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 FLASH_CR register can be locked again by user software by writing the LOCK bit in the FLASH_CR register. In addition, the FLASH_CR register cannot be written when the BSY bit of the FLASH_SR register is set. In the meantime, any attempt to write FLASH_CR register will cause the AHB bus to stall until the BSY1 bit is cleared.

4.2.3.2. Flash memory programming

The Flash memory can be programmed the entire page in units of 32 bits each time (hardfault will be generated when the half word or byte operation is performed). 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 non 32-bit write will cause a hard fault interrupt.

If the address 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_CR register. At the end of the program operation, the EOP bit in the FLASH_CR register will be set.

The Flash memory programming sequence is as follows:

1) Check that no Flash memory operation is ongoing by checking the BSY in the FLASH_SR register.
2) If no Flash memory erase or program operation is ongoing, the software reads out the 32 words of the page (if the page already has data stored, perform this step, otheRWise skip this step).
3) To release the protection of the FLASH_CR register by programming KEY1 and KEY2 to the FLASH_KEYR register.
4) Set the PG bit and the EOPIE bit in the FLASH_CR register.
5) Programming to the target address from the 1st to 31st word (only accept 32 bits program).
6) Set the PGSTRT in FLASH_CR register.
7) Write the 32nd word.
8) Wait until the BSY bit of the FLASH_SR register to be cleared.
9) Check the EOP flag in the FLASH_SR register (It is set when the programming operation has succeeded), and then clear it by software.
10) If there are no more program operations, software will clear the PG bit.

When the above step 7) is successfully executed, the program operation is automatically started, and the BSY bit is set by hardware at the same time.

Flash Erase Operation

The Flash memory can be erased by page, or sector and mass erase (sector and mass erase do not work for information memory).

4.2.3.3. Page erase

When a page is protected by WRP, it will not be erased and the WRPERR bit is set at this time. To excution the page erase operation, the following steps need to be performed:

1) Check that no Flash memory operation is ongoing by checking the BSY in the FLASH_SR register.
2) To release the protection of the FLASH_CR register by programming KEY1 and KEY2 to the FLASH_KEYR register.
3) Set the PER bit and the EOPIE bit in the FLASH_CR register.
4) Write arbitrary data (32-bit data) to the page.
5) Wait for the BSY bit to be cleared.
6) Check that the EOP flag is set.

7) Clear the EOP flag.

4.2.3.4. Mass erase

The Mass erase can used to completely erase the entire main Flash memory, but the information block is unaffected by this procedure. Additionally, when WRP is enabled, the mass erase function is disabled and no mass erase operation occurs, the WEPERR bit is set.

The following sequence for mass erase:

1) Check that no Flash memory operation is ongoing by checking the BSY.
2) To release the protection of the FLASH_CR register by programming KEY1 and KEY2 to the FLASH_KEYR register.
3) Set the MER bit and the EOPIE bit in the FLASH_CR register.
4) Write arbitrary data (32-bit data) to the main Flash memory.
5) Wait for the BSY bit to be cleared.
6) Check that the EOP flag is set.
7) Clear the EOP flag.

4.2.3.5. Sector erase

The wector erase can be used to erase the main Flash of 4 kbytes, but the information block is unaffected by this procedure. In addition, when a sector is protected by WRP, it will not be erased, and the WRPERR bit is set.

The following sequence for sector erase:

1) Check that no Flash memory operation is ongoing by checking the BSY.
2) To release the protection of the FLASH_CR register by programming KEY1 and KEY2 to the FLASH_KEYR register.
3) Set the SER bit and the EOPIE bit in the FLASH_CR register.
4) Write arbitrary data to the sector.
5) Wait for the BSY bit to be cleared.
6) Check that the EOP flag is set.
7) Clear the EOP flag.

System memory is read-only and will never be programmed/erased.

4.2.3.6. Program and erase time configuration

The program and erase time need to be strictly controlled, otherwise the operation will fail. By default, the hardware design sets the time parameters of program and erase operations that the HSI is 24 MHz. When the HSI output frequency is changed, the Flash program and erase time need to be configured the register correctly according to the table below.

Table 4-2 Program and erase time configuration

Register	4 MHz	8 MHz	16 MHz	22.12 MHz	24 MHz
TS0	0x1E	0x3C	0x78	0xA6	0xB4
TS1	0x48	0x90	0x120	0x18F	0x1B0
TS2P	0x1E	0x3C	0x78	0xA6	0xB4
TPS3	0x120	0x240	0x480	0x639	0x6C0
TS3	0x1E	0x3C	0x78	0xA6	0xB4
PERTPE	0x2EE0	0x5DC0	0XBB80	0x10338	0x11940
SMERTPE	0x2EE0	0x5DC0	0XBB80	0x10338	0x11940
PRGTPE	0XFA0	0x1F40	0x3E80	0x5668	0x5DC0
PRETPE	0x320	0x640	0xC80	0x1148	0x12C0

4.3. Product Unique identity code (UID) register

Typical application scenarios of unique ID codes:

- Used as a serial number
- When programming internal flash memory, use it as a key or encryption primitive to increase the security of your code
- Activate the security bootstrap process, etc

The product Unique ID provides a reference number that is unique to any device.

The user can never change these bits. Unique identifiers can also be read in different ways such as single byte/half word/word and then concatenated using a custom algorithm.

Base address: 0x1FFF 0E00

Table 4-3 UID format

Offset address	description	UID Bits							
		7	6	5	4	3	2	1	0
0	Lot Numer	Lot Number ASCII code							
1	Lot Numer	Lot Number ASCII code							
2	Lot Numer	Lot Number ASCII code							
3	Lot Numer	Lot Number ASCII code							
4	Wafer Number	Wafer Number							
5	Lot Numer	Lot Number ASCII code							
6	Lot Numer	Lot Number ASCII code							
7	Lot Numer	Lot Number ASCII code							
8	The internal encoding	The internal encoding							
9	Low y-coordinate	Low y-coordinate							
10	Low x-coordinate	Low x-coordinate							
11	X,Y coordinates high address	High y-coordinate				High x-coordinate			
12	Fixed code	0x78							
13	The internal encoding	The internal encoding							
14	The internal encoding	The internal encoding							
15	The internal encoding	The internal encoding							

4.4. Flash option byte

4.4.1. Flash option word

Part of the information area is used as an option byte, which is used to store the hardware configuration that the chip or the user needs to perform for the application. For example, the watchdog can be selected in hardware or software mode.

For data security, the option bytes are stored separately in the code and one's complement code.

Table 4-4 Option byte format

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

Option byte 1	Option byte 0

The option bytes can be read from the memory locations listed in the table option byte organization or from the relevant registers of the following option bytes :

- FLASH user option register (FLASH_OPTR)
- FLASH SDK area address register (FLASH_SDKR)
- FLASH WRP address register (FLASH_WRPR)

Table 4-5 Option byte organization

Word address	Describe
0x1FFF 0E80	Option byte for Flash User option and its complemented
0x1FFF 0E84	Option byte for Flash SDK area address and its complemented
0x1FFF 0E88	Reserved
0x1FFF 0E8C	Option byte for Flash WRP address and its complemented
0x1FFF 0E90	Reserved
0x1FFF 0E94	Reserved
…	Reserved
…	Reserved
…	Reserved
0x1FFF 0EFC	Reserved

- **Option byte for Flash User option**

Flash memory Address offset: 0x1FFF 0E80

Production value: 0x0155 BEAA

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash information memory and written to the corresponding option bit of the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~nBOOT1	~NRST_MODE	Res	~IWDG_SW	~BOR_LEV[2:0]			~BOR_EN	~RDP[7:0]							
R	R		R	R	R	R	R	R	R	R	R	R	R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
nBOOT1	NRST_MODE	Res	IWDG_SW	BOR_LEV[2:0]			BOR_EN	RDP[7:0]							
R	R		R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	~nBOOT1	R	One's complement of nBOOT1
30	~NRST_MODE	R	One's complement of NRST_MODE
29	Res	-	-
28	~IWDG_SW	R	One's complement of IWDG_SW
27:25	~BOR_LEV[2:0]	R	One's complement of BOR_LEV
24	~BOR_EN	R	One's complement of BOR_EN
23:16	~RDP	R	One's complement of RDP
15	nBOOT1	R	Select boot mode with BOOT PIN
14	NRST_MODE	R	0: Reset input only 1: GPIO function
13	Res	-	-
12	IWDG_SW	R	0: Hardware watchdog 1: Software watchdog
11:9	BOR_LEV[2:0]	R	000: BOR rising threshold is 1.8 V, falling threshold is 1.7 V 001: BOR rising threshold is 2.0 V, falling threshold is 1.9 V 010: BOR rising threshold is 2.2 V, falling threshold is 2.1 V 011: BOR rising threshold is 2.4 V, falling threshold is 2.3 V 100: BOR rising threshold is 2.6 V, falling threshold is 2.5 V 101: BOR rising threshold is 2.8 V, falling threshold is 2.7 V 110: BOR rising threshold is 3.0 V, falling threshold is 2.9 V 111: BOR rising threshold is 3.2 V, falling threshold is 3.1 V
8	BOR_EN	R	BOR enable 0: BOR is disabled 1: BOR is enabled, BOR_LEV works
7:0	RDP	R	0 xAA: level 0, read protection inactive Non 0xAA: level 1, read protection active

- Option byte for Flash SDK area address

Flash memory Address offset: 0x1FFF 0E84

Production value: 0x FF00 00FF

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash information memory and written to the corresponding option bit of the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	~SDK_END[4:0]					Res	Res	Res	~SDK_STRT[4:0]				
			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	SDK_END[4:0]					Res	Res	Res	SDK_STRT[4:0]				
			R	R	R	R	R				R	R	R	R	R

Bit	Name	R/W	Function
31:16	Reserved		
28:24	Complemented SDK_END[4:0]	R	One's complement of SDK_END
23:21	Reserved		
20:16	Complemented SDK_STRT[4:0]	R	One's complement of SDK_STRT
15:13	Reserved		
12:8	SDK_END[4:0]	R	SDK area end address, each corresponding STEP is 2 kbytes
7:5	Reserved		
4:0	SDK_STRT[4:0]	R	SDK area start address, each corresponding STEP is 2 kbytes

- Option byte for Flash WRP address

Flash memory Address offset: 0x1FFF 0E8C

Production value: 0x0000 FFFF

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash information memory and written to the corresponding option bit of the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~WRP[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**
WRP[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31:16	Complemented WRP	R	One's complement of WRP
15:0	WRP	R	0: sector [y] is protected 1: sector [y] unprotected y = 0 to 15

4.4.2. Flash option byte write

After reset, the bits in the FLASH_CR register associated with the option byte are write-protected. The OPT-LOCK bit in the FLASH_CR register must be cleared before the option byte can be manipulated.

The following steps are used to unlock this register:

1) Unlock sequence to unlock write protection of FLASH_CR register.
2) Write OPTKEY1 = 0x0819 2A3B to the FLASH_OPTKEYR register.
3) Write OPTKEY2 = 0x4C5D 6E7F to the FLASH_OPTKEYR register.

Any wrong sequence locks up 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.

User option (option bytes in information Flash memory) can be protected by software by writing the OPTLOCK bit of the FLASH_CR register to prevent unwanted erase/program operations.

If software sets the Lock bit, the OPTLOCK bit is also automatically set.

Modifying user option bytes

Programming operation of the option byte is different from the operation to the main Flash memory. To modify the option bytes, the following steps are required:

1) Using the steps described previously to clear the OPTLOCK bit.
2) Check that no Flash memory operation is ongoing by checking the BSY.
3) Write the desired value (1~3 words) to the option bytes register FLASH_OPTR/ FLASH_SAR/ FLASH_WRPR.
4) Set OPTSTRT bit.
5) Write any 32 bits data to the main Flash memory address 0x4002 2080 (trigger a formal program operation).
6) Wait for the BSY bit to be cleared.
7) Wait for EOP to be pulled high, software to be cleared.

Any change to the option bytes, the hardware will first erase the entire page to the option byte, and then program the value of the FLASH_OPTR, FLASH_SAR or FLASH_WRPR register to the option bytes. And, the hardware automatically calculates the corresponding complement, and programs the calculated value to the corresponding area of the option bytes.

Option byte loading

After the BSY bit is cleared, all new option bytes are written into the Flash information memory, but they are not applied to the system. The read operation of the option bytes register still returns the value in the last loaded option bytes. Once they are loaded with new values, it will work on the system.

The loading of option bytes is performed in the following two cases:

- OBL_LAUNCH bit in the FLASH_CR register is set.
- After power-on reset (POR, BOR)

Loading option bytes is: read the option bytes in the information memory area, and then store the read data in the internal option registers (FLASH_OPTR, FLASH_SAR and FLASH_WRPR). These internal registers configure the system and can be read by software. The OBL_LAUNCH bit is set to generate a reset, so that the loading of option bytes can be carried out under the reset of the system.

Each option bit has a corresponding complement at its same doubleword address (next half word). During the loading of the option bytes, the validation of the option bit and its complement ensures that the loading was performed correctly.

If the one's complement matches, the option bytes are copied into the option register.

If the one's complement does not match, the OPTVERR status bit in the FLASH_SR register is set. Unmatched values are written to the option register:

- For user option
  - BOR_LEV is written as 000 (the lowest threshold)
  - The BOR_EN bit is written as 0 (BOR is not enabled)
  - NRST_MODE bit written to 0 (reset input only)
  - RDP bit is written as 0xff (which is level 1)

- The rest of the mismatched values are written as 1

- For SDK area option, SDKR_STRT [4:0] = 0x00, SDKR_END [4:0] = 0x1F, all Flash memory is set as SDK
- For the WRP option, the unmatched value is the default "no protection"

After system reset, the contents of option bytes are copied to the following option registers (readable and writable by software):

- FLASH_OPTR
- FLASH_SDKR
- FLASH_WRPR

These registers are also used to modify option bytes. If these registers are not modified by the user, they reflect the state of the system option.

4.5. Flash configuration bytes

Part of the interval (one page in total) of the information area of the Flash memory is used as factory config. byte.

Page 0 is stored for software to read information (only code, no one's complement code is stored):

- HSI frequency selection control value, and corresponding trimming value.
- Erase and program time configuration parameter values corresponding to different frequencies of HSI.

Table 4-6 Factory config. byte organization

Page	Word	Address	Contents
0	0	0x1FFF 0F00	RESERVED
	1	0x1FFF 0F04	存放 HSI 8MHz 频率选择控制及对应的 Trimming 值
	2	0x1FFF 0F08	RESERVED
	3	0x1FFF 0F0C	RESERVED
	4	0x1FFF 0F10	存放 HSI 24MHz 频率选择控制及对应的 Trimming 值
	5	0x1FFF 0F14	TS_CAL1， 30℃温度传感器的校准值
	6	0x1FFF 0F18	TS_CAL2， 85℃温度传感器的校准值
	7	0x1FFF 0F1C	RESERVED
	8	0x1FFF 0F20	RESERVED
	9	0x1FFF 0F24	RESERVED
	10	0x1FFF 0F28	RESERVED
	11	0x1FFF 0F2C	RESERVED
	12	0x1FFF 0F30	存放 HSI 8MHz 频率下对应的 FLASH_TSO、FLASH_TS1 寄存器的配置值
	13	0x1FFF 0F34	存放 HSI 8MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	14	0x1FFF 0F38	存放 HSI 8MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	15	0x1FFF 0F3C	存放 HSI 8MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
	16	0x1FFF 0F40	存放 HSI 8MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
	17	0x1FFF 0F44	RESERVED
	18	0x1FFF 0F48	RESERVED
	19	0x1FFF 0F4C	RESERVED
	20	0x1FFF 0F50	RESERVED
	21	0x1FFF 0F54	RESERVED
	22	0x1FFF 0F58	RESERVED
	23	0x1FFF 0F5C	RESERVED
	24	0x1FFF 0F60	RESERVED
	25	0x1FFF 0F64	RESERVED
	26	0x1FFF 0F68	RESERVED
	27	0x1FFF 0F6C	存放 HSI 24MHz 频率下对应的 FLASH_TSO、FLASH_TS1 寄存器的配置值
	28	0x1FFF 0F70	存放 HSI 24MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	29	0x1FFF 0F74	存放 HSI 24MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	30	0x1FFF 0F78	存放 HSI 24MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值

	31	0x1FFF 0F7C	存放 HSI 24MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
1	0	0x1FFF 0F80- 0x1FFF 0FFF	RESERVED

4.5.1. HSI_TRIMMING_FOR_USER

Address offset: 0x1FFF 0F00 to 0x1FFF 0F10

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

The software needs to read data from this address, and then write to HSI_FS[2:0] and HSI_TRIM[12:0] corresponding to the RCC_ICSCR register to change the HSI frequency.

4.5.2. Calibration value of temperature sensor

Address offset: 0x1FFF 0F14 (30℃), 0x1FFF 0F18 (85℃)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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											
				TS_{CAL}[11:0]											

Software needs to read data from this address.

4.5.3. HSI_8M/24M_EPPARA0

Address offset: 0x1FFF 0F30 (8 MHz), 0x1FFF 0F6C (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	TS1[8:0]								
							R	R	R	R	R	R	R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS3[7:0]								TS0[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The software needs to set the HSI clock frequency according to the need, choose to read the data from the corresponding address, and then write the FLASH_TS0, FLASH_TS1, FLASH_TS3 registers to realize the configuration of the erasing and programming time required by the corresponding HSI frequency.

4.5.4. HSI_8M/24M_EPPARA1

Address offset: 0x1FFF 0F34 (8 MHz), 0x1FFF 0F70 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	TPS3[10:0]										
					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	TS2P[7:0]							
								R	R	R	R	R	R	R	R

The software needs to set the HSI clock frequency according to the need, choose to read the data from the corresponding address, and then write the FLASH_TS2P and FLASH_TPS3 registers to realize the configuration of the erasing and programming time required for the corresponding HSI frequency.

4.5.5. HSI_8M/24M_EPPARA2

Address offset: 0x1FFF 0F38 (8 MHz), 0x1FFF 0F74 (24 MHz)

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

															[16]
															R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PERTPE[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The software needs to set the HSI clock frequency according to the need, choose to read the data from the corresponding address, and then write it into the FLASH_PERTPE register to realize the configuration of the erasing and programming time required for the corresponding HSI frequency.

4.5.6. HSI_8M/24M_EPPARA3

Address offset: 0x1FFF 0F3C (8 MHz), 0x1FFF 0F78 (24 MHz)

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

The software needs to set the HSI clock frequency according to the need, choose to read the data from the corresponding address, and then write it into the FLASH_SMERTPE register to realize the configuration of the erasing and programming time required for the corresponding HSI frequency.

4.5.7. HSI_8M/24M_EPPARA4

Address offset: 0x1FFF 0F40 (8 MHz), 0x1FFF 0F7C (24 MHz)

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	Res	Res	Res	PRETPE[11:0]										
					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**
PRGTPE[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The software needs to set the HSI clock frequency according to the need, choose to read the data from the corresponding address, and then write it into the FLASH_PRGTPE and FLASH_PRETPE registers to realize the configuration of the erasing and programming time required for the corresponding HSI frequency.

4.6. Flash protection

The protection of Flash main memory includes the following mechanisms:

- Software design kit (SDK) is used to protect access to specific program areas, and the granularity is 2 kbytes.
- Read protection (RDP) is used to prevent access from outside.
- Wrtie protection (WRP) control is used to prevent unwanted writes (due to confusion of the program memory pointer PC). The granularity of write protection is designed to be 4 kbytes.
- Option byte write protection, special unlocking design.

4.6.1. Flash software development kit (SDK) area protection

The protection area is defined by SDKR _STRT[4:0], SDKR_END[4:0] of the FLASH_SDKR register, and each bit corresponds to 2 kbytes.

Start address

FLASH memory base address + SDK _STRT[4:0] x 0x800 (included)

End address

FLASH memory base address + (SDK _END[4:0] +1) x 0x800 (excluded)

When SDK _STRT[4:0] is greater than SDK_END[4:0], SDK protection is invalid. When SDK _STRT[4:0] is less than or equal to SDK _END[4:0], SDK protection is effective.

When the protection is in effect, when the FLASH_SDKR register is unprotected (writing SDK _STRT[4:0] is greater than SDK_END[4:0]), the hardware will first trigger mass erase (the protected program in the SDK area has been written before, and the mass erase is used to protect the program in the SDK area), and then the value of the SDK option in the Flash option byte is updated (the updated value at this time is that the SDK protection is invalid).

At this time, the content of the FLASH_SDKR register will not be updated, until the power-on reset (P OR/BOR/PDR) or OBL reset, the register content will be loaded from the SDK option in the Flash option byte into the register.

4.6.2. Flash read protection

By setting RDP option byte, and perform system reset (POR/BOR or OPL reset) to load a new RDP option byte to activate the read protection function. RDP protects main Flah memory, option byte, and SRAM.

If the read protection is set while the debug by SWD is still connected, a power-on reset is required instead of a system reset.

When the RDP option byte and the two's complement code exist in the option byte, the Flash memory will be protected.

Table 4-7 Flash read protection status

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
Any value except the combination of (0xAA and 0x55)		Level 1

Regardless of any protection level, system memory is access only and program and erase operations cannot be performed.

Level 0: No protection

To read, program and erase the main Flash memory, as well as any operation to the option byte.

Level 1: Read protection

When the RDP and its two's complement in the option byte contain any combination rather than 0xAA, 0x55, the level 1 read protection takes effect, and the level 1 is the default protection level.

- User mode: The program executed in user mode (boot from main Flash memory) can perform all operations on main Flash and option byte.
- Debug, boot from SRAM and boot from system memory mode (Boot loader): In debug mode, or when booting from SRAM or system memory (Boot loader), the main Flash memory cannot be accessed. In these modes, a bus error is generated for a read or program access to the main Flash, and a hard fault interrupt is generated.

When it is already at Level 1 (any number rather than 0xAA), changing to Level 0 by programming 0xAA, the hardware will perform a mass erase operation on the main Flash memory.

Table 4-8 The relationship between access status and protection level and execution mode

Area	READ Protection level	SDK Area Protection level	Boot From Main Flash(CPU)						Debug/ excuted From RAM/ excuted From System memory		
			User execution (From Non SDK Area)			User execution (From SDK Area)					
			Read	Write	Erase	Read	Write	Erase	Read	Write	Erase
Non SDK Area	0	Disable	Yes	Yes	Yes	N/A	N/A	N/A	Yes	Yes	Yes
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SDK Area		Disable	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Enable	No	No	No	Yes	Yes	Yes	No	No	No
Non SDK Area	1	Disable	Yes	Yes	Yes	N/A	N/A	N/A	No	No	No
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
SDK Area		Disable	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Enable	No	No	No	Yes	Yes	Yes	No	No	No
System memory	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No
		Enable	Yes	No	No	Yes	No	No	Yes	No	No
Option bytes area	x	Disable	Yes	Yes	Yes	N/A	N/A	N/A	Yes	Yes	Yes
		Enable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Factory bytes	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No
		Enable	Yes	No	No	Yes	No	No	Yes	No	No
UID	x	Disable	Yes	No	No	N/A	N/A	N/A	Yes	No	No
		Enable	Yes	No	No	Yes	No	No	Yes	No	No

Note:

(1) Mass erase command issued from any area will erase the SDK area.

(2) Any modification of level 1 to level 0 will trigger the hardware mass erase of the main Flash memory.

(3) The meaning of N/A is that when the SDK Area is disabled, since there is no SDK Area, no situation in which programs can be read out from the SDK Area in the above table, and no situation in which the programs read out from other areas can access the SDK Area.

(4) There are two cases for executing programs from SRAM or system memory: one is Boot from, the other is boot from other memory, and the program jumps to SRAM or system memory.

4.6.3. Flash write protection

Flash can be set to be write-protected against unwanted writes. Define the control granularity of each bit of the WRP register as a write protection (WRP) area of 4 kbytes, that is, the size of 1 sector. See the description of the WRP register for details.

When the WRP area is activated, erase or program operations are not allowed. Accordingly, the mass erase function does not work even if only one area is set as write-protected.

In addition, if an attempt is made to erase or program a write-protected area, the write -protection error flag (WRPERR) of the FLASH_SR register will be set.

Note: Write protection only works on main Flash, and read doesn't work on system memory.

4.6.4. Option byte write protection

By default, Option bytes are readable and write-protected. To gain erase or program access to option bytes, the correct sequence needs to be written to the OPTKEYR register.

4.7. Flash interrupt

Table 4-9 Flash interrupt request

Interrupt event	Event flag	Time stamp/interrupt clear method	Control bit enable
End of operation	EOP	Write EOP = 1	EOPIE
Write protection	WRPERR	Write WRPERR = 1	ERRIE

Note: The following events do not have a separate interrupt flag, but will generate a Hard fault:

- Sequence error of FLASH_CR register of unlock Flash memory.
- Unlock Flash option bytes write sequence error.
- Flash program operation is not aligned with 32-bit data.
- Flash erase (including page erase, sector erase and mass erase) operations do not perform 32-bit data alignment.
- To the option byte register is not aligned with 32-bit data.

4.8. Flash register description

4.8.1. Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	Reserved			
0	LATENCY	RW	0	The wait state corresponding to the read operation: 0: There is no wait state for Flash read operation (system clock is 24 MHz and below). 1: The Flash read operation has one wait state, which is two system clock cycles are required for each Flash read (the system clock is at 48 MHz).

4.8.2. Flash key register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

All register bits are write-only and read as 0.

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

Bit	Name	R/W	Reset Value	Function
31:0	KEY[31:0]	W	0x0000	The following values must be written consecutively to unlock the FLASH_CR register and enable the program/erase operation of the Flash KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

4.8.3. Flash option key register (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

All register bits are write-only and read as 0.

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

Bit	Name	R/W	Reset Value	Function
31:0	OPTKEY[31:0]	W	0x0000 0000	The following values must be written consecutively to unlock the option register of the Flash and enable the program/erase operation of the option byte KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

4.8.4. Flash status register (FLASH_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	Res	BSY
															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WRP ERR	Res	Res	Res	EOP
RC_W1											RC_W1				RC_W1

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	BSY	R	0	Busy bit This bit indicates that the operation of the Flash is in progress. This bit is set by hardware at the beginning of a Flash operation, and is cleared by hardware when the operation is completed or an error occurs.
15	OPTVERR	RC_W1	0	Option and trimming bits loading validity error when the option and trimming bits and their one's complements do not match. Load unmatched option bytes, coerced to safe values. Software writes 1 to clear.
14:5	Reserved			
4	WRPERR	RC_W1	0	Write protection error This bit is set by hardware when the address to be programmed/erased is in a write-protected Flash region (WRP). Write 1 to clear this bit.
3:1	Reserved			
0	EOP	RC_W1	0	When the program/erase operation of the Flash completes successfully. This bit is only set if the EOPIE bit in the FLASH_CR register is enabled. Write 1 to clear this bit.

4.8.5. Flash control register (FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

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

LOCK	OPTLOCK	Res	Res	OBL_LAUNCH	Res	ERRIE	EOPIE	Res	Res	Res	Res	PGSTRT	Res	OPTSTRT	Res
RS	RS			RC_W1		RW	RW					RW		RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	SER	Res	Res	Res	Res	Res	Res	Res	Res	MER	PER	PG
				RW									RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Lock	RS		FLASH_CR Lock bit. Software can only set this bit. When set, the FLASH_CR register is locked. When the unlock timing is successfully given, this bit is cleared by hardware, and the FLASH_CR register is unlocked. The software should set this bit after the program/erase operation is completed. When an unsuccessful unlock sequence is given, this bit remains set until the next system reset.
30	OPTLOCK	RS		Option bytes Lock bit. Software can only set this bit. When set, the bits related to option bytes in the FLASH_CR register are locked. When the unlock timing is successfully given, this bit is cleared by hardware, and the FLASH_CR register is unlocked. The software should set this bit after the program/erase operation is completed. When an unsuccessful unlock sequence is given, this bit remains set until the next system reset.
29:28	Reserved			
27	OBL_LAUNCH	RC_W1		Force the option bytes loading. When set, this bit forces the system to perform a reload of option bytes. This bit is only cleared by hardware when the option byte load has been completed. This bit cannot be written if the OPTLOCK bit is set. 0: Option byte loading completed 1: Option byte loading request is generated, the system resets, and the option byte is reloaded.
25	ERRIE	RW		Error interrupt enable bit, when the WRPERR bit in the FLASH_SR register is set, if this bit is enabled, an interrupt request is generated. 0: No interrupt is generated 1: An interrupt is generated
24	EOPIE	RW		End of operation interrupt enable This bit enables interrupt generation when the EOP bit in the FLASH_SR register is set. 0: EOP interrupt disabled 1: EOP interrupt enable
23:18	Reserved	RW		
19	PGSTRT	RW		The start bit of the program operation of the Flash main memory. Program operation of the main Flash memory, and is set by software. After the BSY bit of the FLASH_SR register is cleared, the hardware clears this bit.
18	Reserved			
17	OPTSTRT	RW		Flash option bytes modified start bit This bit initiates modification of option bytes. Set by software and cleared by hardware after the BSY bit in the FLASH_SR register is cleared. Note: When modifying the Flash option bytes, the hardware will automatically perform the erase operation on the entire page of 128 bytes, and then perform the program operation, which also includes the automatic writing of the two's complement code.
16:12	Reserved			
11	SER	RW		4 kbyte Sector erase operation 0: Sector erase operation of Flash is not selected

				1: Select the sector erases operation of Flash Note: 1） Sector erase will not work on Flash information memory. 2） Sector erase has no effect on areas set to WRP.
10:3	Reserved			
2	MER	RW		Mass erase operation 0: Mass erase operation of Flash is not selected 1: Select the mass erases operation of Flash Note: Mass erase will not work on Flash information memory. Mass erase does not work when WRP is set
1	PER	RW		Page erase operation 0: Page erase operation of the Flash is not selected 1: Select the page erase operation of Flash
0	PG	RW		Program operation 0: Program operation of Flash is not selected 1: Select the program operation of Flash

4.8.6. Flash option register (FLASH_OPTR)

Address offset: 0x20

Reset value: 0x0000 xxxx

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash in formation memory and written to the corresponding option bit of the 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**
nBOOT1	NRST_MODE	Res	IWDG_SW	BOR_LEV[2:0]			BOR_EN	RDP[7:0]							
RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15	nBOOT1	RW		与 BOOT PIN 一起，选择芯片启动模式
14	NRST_MODE	RW		0：仅复位输入 1：GPIO：GPIO 功能
13	Reserved			
12	IWDG_SW	RW		0：硬件 watchdog 1：软件 watchdog
11：9	BOR_LEV[2:0]	RW		000：BOR 上升阈值为 1.8V，下降阈值位 1.7V 001：BOR 上升阈值为 2.0V，下降阈值位 1.9V 010：BOR 上升阈值为 2.2V，下降阈值位 2.1V 011：BOR 上升阈值为 2.4V，下降阈值位 2.3V 100：BOR 上升阈值为 2.6V，下降阈值位 2.5V 101：BOR 上升阈值为 2.8V，下降阈值位 2.7V 110：BOR 上升阈值为 3.0V，下降阈值位 2.9V 111：BOR 上升阈值为 3.2V，下降阈值位 3.1V
8	BOR_EN	RW		BOR enable 0：BOR 不使能 1：BOR 使能，BOR_LEV 起作用
7：0	RDP	RW		0xAA：level 0, read protection inactive 非 0xAA：level 1, read protection active

4.8.7. Flash SDK address register (FLASH_SDKR)

Address offset: 0x24

Reset value: 32'b0000 0000 0000 0000 000X _ XXXX 000X XXXX

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash information memory and written to the corresponding option bit of the register.

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

Bit	Name	R/W	Reset Value	Function
31:13	Reserved			
12:8	SDK_END [4:0]	RW		SDK area end addres, each corresponding STEP is 2 kbytes
7:5	Reserved			
4:0	SDK_STRT[4:0]	RW		SDK area start address, each corresponding STEP is 2 kbytes

4.8.8. Flash WRP address register (FLASH_WRPR)

Address offset: 0x2C

Reset value: 0x0000 XXXX

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option bytes area of the Flash in formation memory and written to the corresponding option bit of the register.

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

Bit	Name	R/W	Reset Value	Function
31：5	Reserved			
4	WRP	RW	1	0：sector 4，有写保护，不允许进行 program 和 erase 1：sector 4，无写保护
3	WRP	RW	1	0：sector 3，有写保护，不允许进行 program 和 erase 1：sector 3，无写保护
2	WRP	RW	1	0：sector 2，有写保护，不允许进行 program 和 erase 1：sector 2，无写保护
1	WRP	RW	1	0：sector 1，有写保护，不允许进行 program 和 erase 1：sector 1，无写保护
0	WRP	RW	1	0：sector 0，有写保护，不允许进行 program 和 erase 1：sector 0，无写保护

4.8.9. Flash sleep time configuration register (FLASH_STCR)

Address offset: 0x90

Reset value: 0x0000 6400

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

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			

15:8	SLEEP_TIME	RW	0x64	FLASH sleep time count (counter based on HSI_10M clock) When the system clock selects LSI, in order to obtain more optimized power consumption in Run mode, which can use the function of this register (it is only recommended to use this function when LSI is the system clock). When this function is enabled, the time width of the Flash in the Sleep state in each half system clock low period is: t_{HSI_10M}*SLEEP_TIME Note : t_{HSI_10M} is the period of HSI_10M. To ensure the correct Flash function, the maximum setting value of this register is recommended to be set to 0x28.
7:1	Reserved			
0	SLEEP_EN	RW	0	FLASH Sleep enable 1: Enable Flash sleep 0: Disable Flash sleep

4.8.10. Flash TS0 register (FLASH_TS0)

Address offset: 0x100

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**
Res	Res	Res	Res	Res	Res	Res	Res	TS0							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS0	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F6C The 8MHz calibration value is stored at 0x1FFF 0F30

4.8.11. Flash TS1 register (FLASH_TS1)

Address offset: 0x104

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**
Res	Res	Res	Res	Res	Res	Res	TS1								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	Reserved			
8：0	TS1	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F6C The 8MHz calibration value is stored at 0x1FFF 0F30

4.8.12. Flash TS2P register (FLASH_TS2P)

Address offset: 0x108

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
Res	Res	Res	Res	Res	Res	Res	Res	TS2P							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			
7:0	TS2P	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F70 The 8MHz calibration value is stored at 0x1FFF 0F34

4.8.13. Flash TPS3 register (FLASH_TPS3)

Address offset: 0x10C

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
Res	Res	Res	Res	Res	TPS3										
					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:11	Reserved			
10:0	TPS3	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F70 The 8MHz calibration value is stored at 0x1FFF 0F34

4.8.14. Flash TS3 register (FLASH_TS3)

Address offset: 0x110

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
Res	Res	Res	Res	Res	Res	Res	Res	TS3							
															RW

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			
7:0	TS3	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corre-

				sponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F6C The 8MHz calibration value is stored at 0x1FFF 0F30

4.8.15. Flash page erase TPE register (FLASH_PERTPE)

Address offset: 0x114

Reset value: 0x0001 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	PERTPE
															RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERTPE															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16:0	PERTPE	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F74 The 8MHz calibration value is stored at 0x1FFF 0F38

4.8.16. Flash sector/mass erase TPE register (FLASH_SMERTPE)

Address offset: 0x118

Reset value: 0x0001 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	SMERTPE
															RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMERTPE															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16:0	SMERTPE	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F78 The 8MHz calibration value is stored at 0x1FFF 0F3C

4.8.17. Flash program TPE register (FLASH_PRGTPE)

Address offset: 0x11C

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
PRGTPE															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PRGTPE	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F7C The 8MHz calibration value is stored at 0x1FFF 0F40

4.8.18. Flash pre-program TPE register (FLASH_PRETPE)

Address offset: 0x120

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
Res	Res	PRETPE[13:0]													
		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	Reserved			
13:0	PRETPE	RW	0xXXXX	By reading out the data stored at the corresponding address in the information area and writing it into the corresponding register, the software can realize the configuration of the erasable time required by the corresponding HSI frequency. Save it in the following Flash address: The 24MHz calibration value is stored at 0x1FFF 0F7C The 8MHz calibration value is stored at 0x1FFF 0F40

4.8.19. Flash register mapping

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LA-
	Reset value																																0
0x008	FLASH_KEYR	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_OPTKEYR	OPTKEY[31:16]																OPTKEY[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
0x010	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY	OPTVER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPER	Res.	Res.	Res.	EOP
	Reset value																0	0											0				0
0x014	FLASH_CR	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNC	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	PGSTRT	Res.	OPTSTRT	Res.	Res.	Res.	Res.	Res.	SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	PER	PG

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
	Reset value	0	0			0								0		0						0									0	0	0
0x200	FLASH_OPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	nBOOT1	NRST_MODE	Res	IWDG_SW	BOR_LEV[2:0]			BOR_EN	RDP[7:0]							
	Reset value																	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X
0x224	FLASH_SDKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SA_END[4:0]					Res.	Res.	Res.	SA_STRT[4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X
0x22C	FLASH_WRPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x900	FLASH_STCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_TIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_E
	Reset value																	0	1	1	0	0	1	0	0								0
0x100	FLASH_TS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x104	FLASH_TS1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1[8:0]							
	Reset value																								1	1	1	0	1	1	0	0	0
0x108	FLASH_TS2P	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS2P[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x10C	FLASH_TPS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASH_TPS3[10:0]										
	Reset value																						1	1	0	1	1	0	0	0	0	0	0
0x110	FLASH_TS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS3[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x114	FLASH_PERTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERTPE	WRP[15:0]															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x118	FLASH_SMERTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMERTPE[16:0]																
	Reset value																0	1	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0
0x11C	FLASH_PRGTPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRGTPE[15:0]															
	Reset value																	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0
0x1	FLASH_PRETPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRETPE[13:0]													

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

5. Power control

5.1. Power supply

5.1.1. Power block diagram

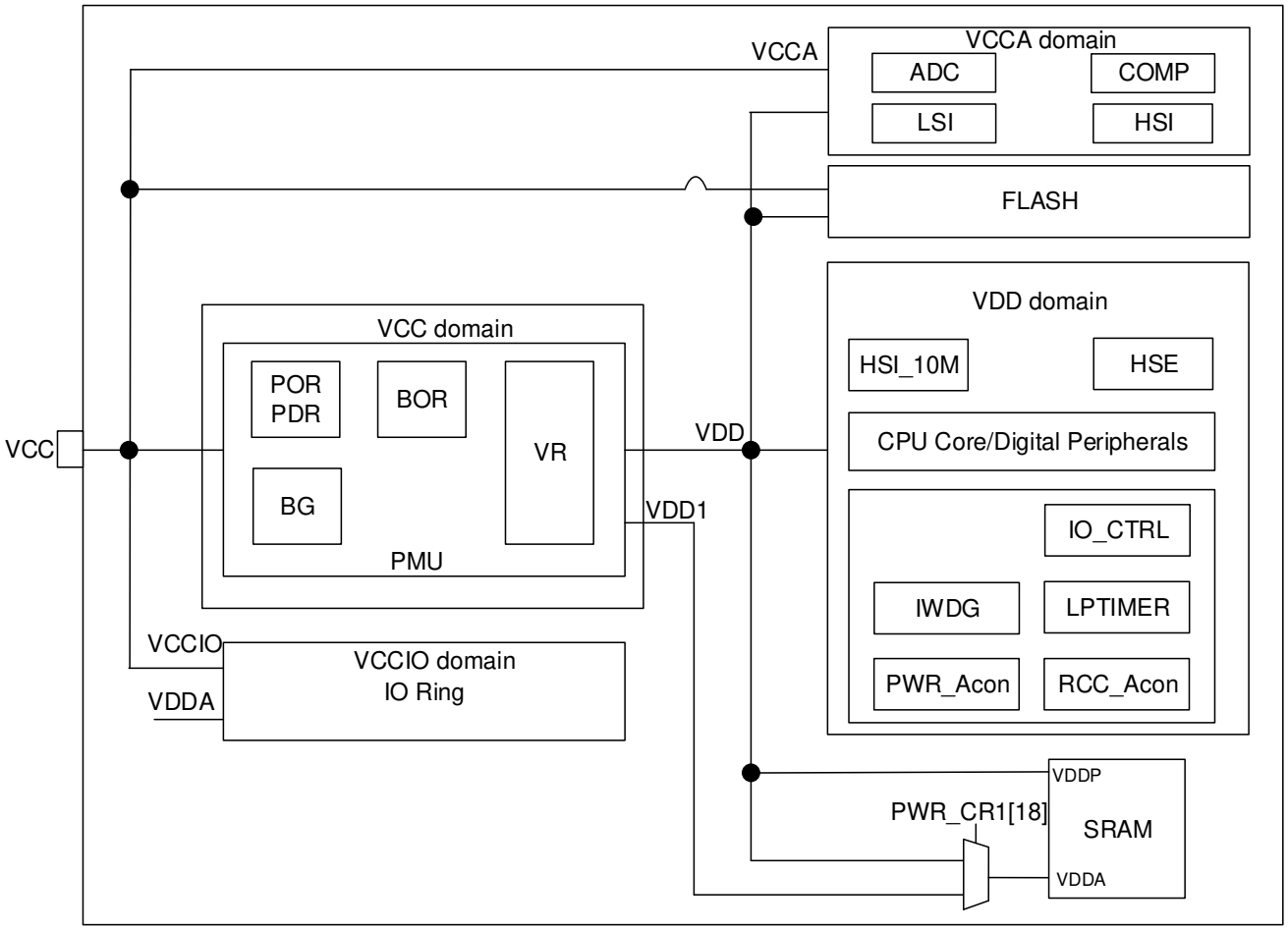


Figure 5-1 Power block diagram

Table 5-1 Power block

Numbering	Power supply	Power value	Describe
1	VCC	1.7 to 5.5 V	Provide power to the chip through the power pins, and its power supply analog circuit module.
2	VCCA	1.7 to 5.5 V	Provide power to most of the analog modules from VCC PAD (a separate power supply PAD can also be designed).
3	VCCIO	1.7 to 5.5 V	Power supply to IO, from VCC PAD
4	VDD	1.2/1.0 V ± 10%	Output from VR which supplies power to the main logic circuits and SRAM inside the chip. When the MR is powered, it outputs 1.2 V. When entering the stop mode, according to the software configuration, it can be powered by MR or LPR, and the LPR output is determined to be 1.2 V or 1.0 V according to the software configuration.

5.2. Voltage regulator

The microcontroller designs two voltage regulators:

- Main regulator (MR) keeps working when the chip is in normal operating state.

- Low power regulator (LPR) provides a lower power consumption option in stop mode.

VDD comes from MR or LPR depending on the working mode.

In run mode, MR keeps working, outputs is 1.2 V, and LPR is turned off.

In stop mode, power can be supplied from MR or LPR as determined by software. Likewise, it is up to the software to decide whether VDD is 1.2 V or 1.0 V in the case of LPR power supply after entering stop.

5.3. Dynamic voltage value management

Dynamic voltage value management refers to adjusting the output VDD voltage of VR, to obtain corresponding performance and power consumption with different voltages according to application requirements.

- **Range 1: High performance range**

The typically output of MR is 1.2V (VDD), and the system clock frequency can run as fast as 48 MHz.

- **Range 2: Low power range**

Only in stop mode, it is allowed to enter the low power range, and the range only works for LPR.

By default, the typical output of LPR is 1.2 V (VDD). When the VOS bit of the register is set and the chip enters the stop mode, the MR switches to the low power supply (if the software selects the stop mode, it is powered by LPR powered), and LPR is switched to a typical value of 1.0V (VDD). At this time, part of the logic circuit (LPTIMER) in working state can run under LSI.

When the chip exits the stop mode, the chip restores the MR power supply, and the VOS bit is also cleared by hardware. When the next time enter the stop mode, the VOS bit need to be set by software to ensure the LPR power supply in stop mode is 1.0V, which can get lower power consumption.

5.4. Power monitoring

5.4.1. Power-on reset (POR)/power-down reset (PDR)/brown-out reset (BOR)

The POR/PDR module is designed in the chip and placed under the VDD power domain to provide power-on and power-off reset for the chip. The module keeps working in all modes.

In addition to POR/PDR, BOR (brown out reset) is also implemented. BOR can only be enabled and disabled through the option byte.

When the BOR is turned on, the BOR threshold can be selected by the option byte, and both the rising and falling detection points can be configured individually.

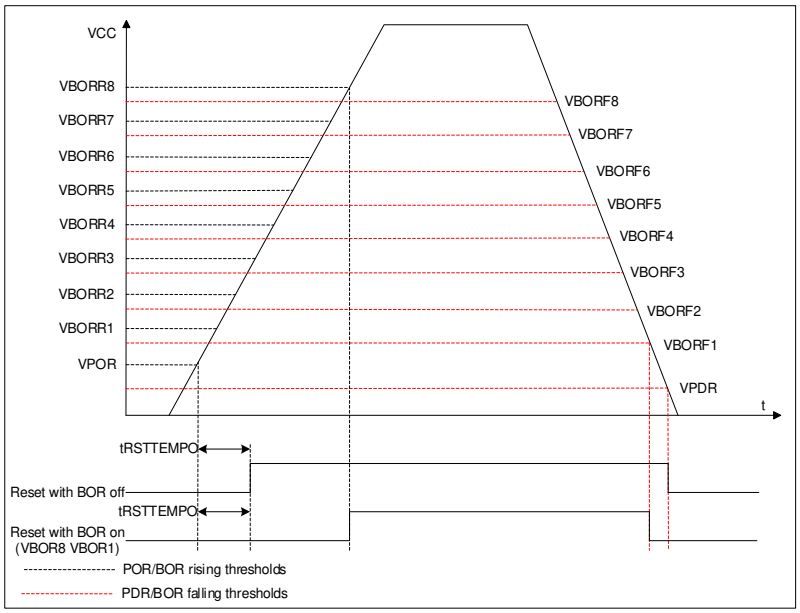


Figure 5-2 POR/PDR/BOR threshold

6. Low-power control

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. Software can choose between power consumption, startup time, and wakeup sources.

6.1. Low-power mode

6.1.1. Introduction to low-power modes

There are two low-power modes:

- **Sleep mode:** The CPU clock is off, NVIC, Sys Tick, peripherals can be configured to keep working. (It is recommended to enable only the modules that must work, and close the modules after the modules work).
- **Stop mode:** In this mode, the contents of SRAM and registers are maintained, HSI and HSE are turned off, and the clocks of most modules in the VDD domain are stopped.

In stop mode, LSI , LPTIMER, etc. can keep working. For details on the working conditions of each module in this mode.

In stop mode, the corresponding of VR state can be controlled by software and set to MR or LPR power supply. When LPR is powered, the power consumption greatly reduced, but the wake-up time is long. When the MR power is maintained, the power consumption is large, but it has the ability to wake up quickly in several cycles.

In addition, in run mode, the power consumption can be reduced by the following methods:

- Decrease system clock frequency
- For unused peripherals, turn off peripheral clocks (system clock and module clock)

In summary, the low-power mode transition diagram of this project is as follows.

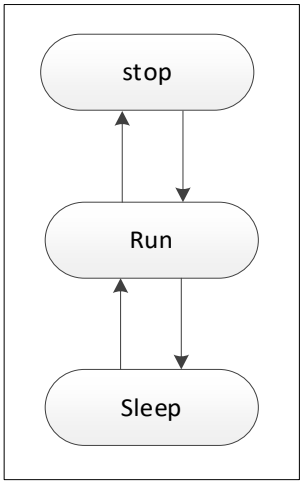


Figure 6-1Low-power mode

6.1.2. Low-power mode switch

Table 6-1 Low power mode switch

Mode	Entry	Wakeup	Wake-up clock	Effects on the clock	Voltage regulator	
					MR	LPR
Sleep (sleep-now or	WFI or Return from ISR	Any interruption	Same as before entering sleep mode	The CPU clock is off and has no effect on other clocks and clock sources.	On(1)	Close
	WFE	Wakeup event				

sleep-on-exit)						
Stop	SLEEPDEEP bit 1. WFI 2. Return from ISR 3. WFE Note: The system clock cannot select LSI	Any EXTI line configured to wake up, IWDG, NRST	HSISYS HSI maintains the frequency configuration before entering stop and does not divide the frequency	HSI close. HSE close. LSI can be turned on or off. LPTIMER, IWDG: Whether it is configured by software or not. Low-power wakeup and some modules such as RCC keep working. Clocks of the remaining modules are turned off.	Software configuration switch	Software configuration switch, if open, output voltage 1.2/1.0 V can be configured

Note: The software must configure the VR state as MR mode to enter sleep mode.

6.1.3. Functions in each working mode

Table 6-2 Functions in each working mode(1)

Peripheral	Run	Sleep	Stop	
			VR@LPR or VR@MR	Wakeup ability
CPU	Y	-	-	-
Flash memory	Y	Y	- (2)	-
SRAM	Y	O (3)	- (4)	-
Brown-out reset (BOR)	Y	Y	O	O
HSI	O	O	-	-
HSE	O	O	-	-
LSI	O	O	O	-
HSE Clock Security System (CSS)	O	O	-	-
USART1	O	O	-	-
SPI1	O	O	-	-
ADC	O	O	-	-
COMP1/COMP2	O	O	O	O
Temperature sensor	O	O	-	-
Timers(TIM1/ TIM16)	O	O	-	-
LPTIM	O	O	O	O
IWDG	O	O	O	O
SysTick timer	O	O	-	-
CRC	O	O	-	-
GPIOs	O	O	O	O

Note:

(1) Y = Yes (enable), O = Optional (default disabled, can be enabled by software), - = Not available.

(2) Flash is not powered off, but no clock is provided, and it enters the lowest power consumption state.

(3) SRAM clock can be turned on or off.

(4) SRAM is not powered off, but no clock is provided, and it enters the lowest power consumption state.

6.2. Sleep mode

6.2.1. Entering sleep mode

The sleep mode is entered by executing the WFI (wait for interrupt) or WFE (wait for event) instructions. Two option are available to select the sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex M0+ System Control Register.

- Sleep-now: If the SLEEPONEXIT bit is 0, to enter sleep mode as soon as WFI or WFE instruction is excuted.
- Sleep-on-exit: If the SLEEPONEXIT bit is 1, to enter sleep mode as soon as it exits the low priority ISR.

In the sleep mode, all IO pins keep the same state as in the run mode.

6.2.2. Exiting sleep mode

If the WFI instruction is used to enter sleep mode, any peripheral interrupt acknowledged by 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 events can be generated in the following ways:

- Enable interrupts in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex M0+. 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) must 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 corresponding to the event line is not set.

This mode offer the shortest wakeup time, and no time is wasted in interrupt entry and exit.

Table 6-3 Sleep-now

Sleep-now mode	Description
Mode entry	WFI or WFE while: - SLEEPDEEP = 0 and - SLEEPONEXIT = 0
Mode exit	Enter the sleep mode through WFI, the exit method is: interrupt. Enter the sleep mode through WFE, the exit method is: wakeup event.
Wakeup latency	None

Table 6-4 Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI while: - SLEEPDEEP = 0 and - SLEEPONEXIT = 1
Mode exit	Interrupt
Wakeup latency	None

6.3. Stop mode

The stop mode is based on the Cortex-M0+ deep sleep mode combined with peripheral clock gating, and the VR can be configured as MR or LPR power supply. In stop mode, HSI and HSE are turned off, SRAM and register contents are kept in a state, LSI, LPTIMER, IWDG can be configured by software whether to work, low-power wakeup and some RCC logic and so on, the clock inputs to the digital blocks of the remaining VCORE domains are turned off.

In the stop mode, all IO pins keep the same state as in the run mode.

6.3.1. Entering stop mode

To further reduce power consumption in stop mode, when PWR_CR.LPR = 1, VR can enter LPR to supply power.

If Flash memory programming is ongoing, the stop mode entry is delayed until the memory access is finished (the BSY bit of the FLASH_SR register is read by software to determine whether the current erase and program operations have been completed).

If an access to the APB domain is ongoing, the stop mode entry is delayed until the APB access is finished (controlled by software).

6.3.2. Exiting stop mode

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

In stop mode, if VR is in LPR state, there is an additional stabilization delay for wakup in stop mode.

In stop mode, if VR is in MR state, the current consumption will be large, but the wakeup time will be reduced.

Table 6-5 Stop mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: Configuration settings: 1）Configure the LPR bit of PWR_CR, to select VR to work under MR or LPR. 2）Configure the VOS bit of PWR_CR, to select LPR mode to provide 1.2 V or 1.0 V. 3）Configure the SRAM_RETV bit of PWR_CR, to select the retention voltage of SRAM. 4）Configure the MRRDY_TIME and FLS_SLPTIME of PWR_CR, to set wake-up time of MR and Flash. Set the SLEEPDEEP bit of Cortex M0+ **Note:** To enter stop mode, all EXTI line pending bits (EXTI_PR register), all peripheral interrupt pending bits must be reset. OtheRWise, the stop mode entry proceduce is ignored and program execution continues. If the application needs to disable HSE before entering stop mode, the system clock source must be first switched to HSI and then clear the HSEON bit. To make the change of chip power consumption as balanced as possible, the software needs to follow the principle of gradual shutdown: gradually shut down the clock of each module, select HSI as the system clock, close HSE. To shorten the wakeup time, before entering the stop mode, the system clock should be configured to select the HSI high-frequency clock, and the HPRE of the RCC_CFGR register is set to 0, otheRWise the hardware switching clock after wake-up will consume extra clocks.
Mode exit	If using WFI to enter stop mode: - Any EXTI Line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). If using WFE to enter stop mode: - Any EXTI Line configured in event mode. - Interrupt pending bit when the CPU SEVONPEND bit is set.
Wakeup latency	LPR to MR wakeup time + HSI wakeup time + Flash wakeup time

6.4. Decreasing system clock frequency

In run mode, the frequency of the system clock (SYSCLK, HCLK, PCLK) can be reduced by frequency division through the prescaler register configuration. These prescalers can also be used to reduce the frequency of peripherals before entering sleep mode.

When the system runs at a lower frequency (32.768 kHz), to obtain less power consumption, software can set the drive capability configuration bit of the voltage regulator (MR) (PWR_CR1 register BIAS_CR [3:0]), which greatly reduces the power consumption of MR itself. But it should be noted that the system clock frequency should be reduced first, and then the driving capability of the MR should be adjusted. On the contrary, when exiting the lower frequency and entering the higher operating frequency, it need increase the driving capacity of the MR first, and then change the operating frequency of the system clock.

6.5. Peripheral clock gating

In run mode, the AHB clock (HCLK) and APB clock (PCLK) for individual peripherals and memories can be stopped at any time to reduce power consumption.

To reduce the power consumption in sleep mode, peripheral clocks can be stopped before executing WFI or WFE instructions.

6.6. Power management register

The peripheral 's registers can be accessed through half-word or word.

6.6.1. Power control register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0003 0000 (reset by POR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	HSION_CTRL	SRAM_RETV[2:0]		
												RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LPR	FLS_SLP-TIME[1:0]		MRRDY_TIME[1:0]		VOS	DBP	Res	Res	Res	BIAS_CR_SEL	BIAS_CR[3:0]			
	RW	RW	RW	RW	RW	RW	RW				RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19	HSION_CTRL	RW	0	HSI turns on time control when wakeup from stop mode. 0: After waiting for MR to stabilize, enable HIS. 1: Turn on the VR, as well as enable HSI when wakeup
18:16	SRAM_RETV _ [2:0]	RW	111	SRAM retention voltage control in stop mode 000: Reserved 001: Reserved 010: Reserved 011: Supply 0.9 V voltage to SRAM 1xx: Supply 1.2 V or 1.0 V to SRAM (depending on VOS bit)
15	Reserved			
14	LPR	RW	0	Low power regulator 0: Main regulator works in stop mode 1: Low power regulator works in stop mode
13:12	FLS_SLPTIME	RW	2'b00	Wakeup sequence from stop mode, after the HSI is stable, a waiting time is required before the Flash operation. 2'b00: 5 us 2'b01: 2 us 2'b10: 3 us 2'b11: 0 us Note: When this register is set to 2'b11, it means that the program is executed from SRAM instead of Flash after wakeup. And the program guarantees that Flash will not be accessed within 3 us after waking up the execution program.
11:10	MRRDY_TIME	RW	2'b00	During the stop mode, the VDD voltage is LP-VR, and the time control of switching from LP-VR to stable Main-VR during wakeup. 2'b00: 2 us 2'b01: 3 us 2'b10: 4 us 2'b11: 5 us
9	VOS	RW	0	Voltage scaling range selection 0: After entering stop mode, VDD = 1.2 V 1: After entering stop mode, VDD = 1.0 V
8:5	Reserved	-	-	Reserved
4	BIAS_CR_SEL	RW	0	Select the MR bias current from the configuration of the BIAS_CR register or from the loading of the Factory config.bytes area of the information memory 0: Select the load from the Factory config.bytes area 1: Select from BIAS_CR register
3:0	BIAS_CR	RW	4'b0000	MR bias current configuration. 4'b0000:

				…

6.6.2. PWR register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSION_CTRL	SRAM_RETV[2:0]			Res.	LPR	FLS_SLP-TIME[1:0]		MRRD_TIME[1:0]		VOS	Res.	Res.	Res.	Res.	BIAS_CR_SEL	BIAS_CR[3:0]			
	Reset value													0	1	1	1		0	0	0	0	0	0	0				0	0	0	0	0

7. Reset

There are two types of resets defined as power reset and system reset.

7.1. Reset source

7.1.1. Power reset

A power reset sets all registers to their reset value, which occurs in the following situations:

- Power on reset (POR/ PDR)
- Brown-out reset (BOR)

7.1.2. System reset

A system reset sets most registers to their reset values, except some special registers, such as the reset flag register.

A system reset generates when the following events occur:

- Reset of NRST pin
- Independent watchdog reset (IWDG)
- SYSRESETREQ software reset
- Option byte load reset (OBL)
- Power reset (POR/PDR, BOR)

The reset source can be identified by checking the reset flag bits of the RCC_CSR register.

7.1.3. NRST pin (external reset)

By loading the option byte (NRST_MODE bit), the NRST pin can be configured in the following modes (see option byte description for specific configuration):

- Reset input

In this mode, any valid reset signal on the NRST pin is passed to the internal logic, but the reset generated inside the chip is not output on the NRST pin.

In this configuration mode, the PF2 function of the GPIO is invalid.

There is burr filtering for NRST pin. The design ensures that NRST must meet the minimum width of 20 us, and the signal less than this width will be filtered out.

- GPIO

In this mode, the PIN can be used as a standard GPIO, like PF2. The reset function on the pin does not work. Resets are only generated internally by the chip and cannot be passed to the pin.

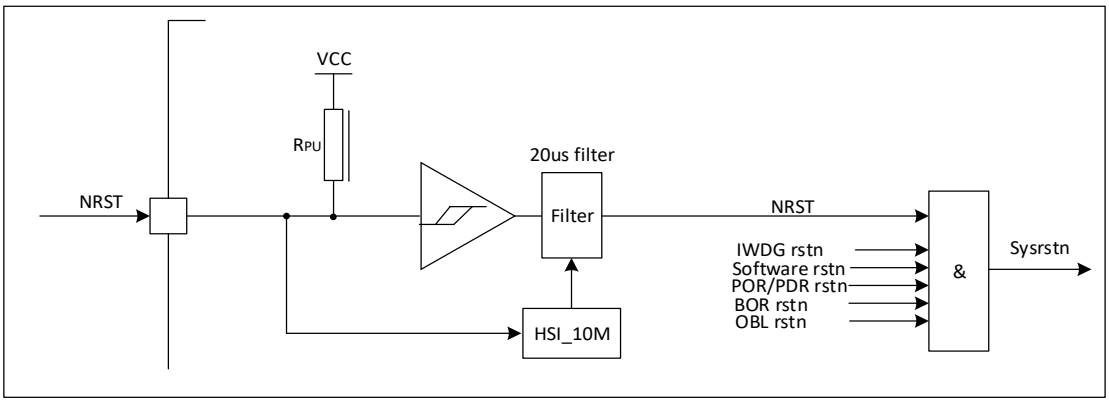


Figure 7-1 Simplified diagram of the reset circuit

7.1.4. Watchdog reset

See independent watchdog for details.

7.1.5. Software reset

A software reset can be achieved by setting the SYSRESETREQ bit in the ARM M0+ interrupt and reset control register.

7.1.6. Option byte loader reset

By configuring FLASH_CR.OBL_LAUNCH = 1, the software generates an option byte load reset, thereby starting the option byte load again.

8. Clock

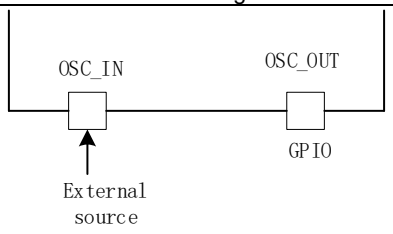
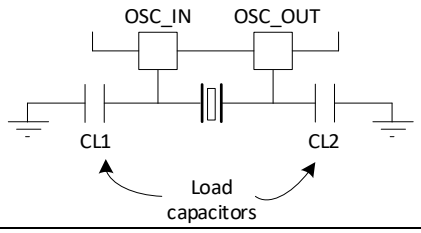
8.1. Clock source

8.1.1. High-speed external clock (HSE)

The high-speed external clock comes from two sources:

- External crystal (crystal), with the internal start-up circuit, a clock signal of 4 to 32 MHz is generated.
- Input high-speed clock source directly from external.

Table 8-1HSE clock sources

Clock source	Hardware configuration
External clock	
External crystal	

External crystal

The 4 to 24 MHz crystals have very high accuracy. The HSERDY flag of RCC_CR shows whether the HSE is stable. HSE can be turned on or off by the HSEON bit.

External clock source (HSE bypass)

In this mode, the external clock source is directly supplied to the chip. Software selects this mode by the HSEBYP and HSEON bits of RCC_CR. The external clock source will be input into the chip through PF0, and PF1 is used as GPIO.

8.1.2. High-speed internal clock (HSI)

The high-speed internal clock is the most important source of the chip system clock. The center frequency of the HSI clock source is designed to be 24 MHz.

8.1.3. Low-speed internal clock (LSI)

The low-speed internal clock, as the clock for IWDG and LPTIM, and as the system clock when the chip is running at low speed. The clock center frequency is designed at 32.768 kHz.

8.2. Clock tree

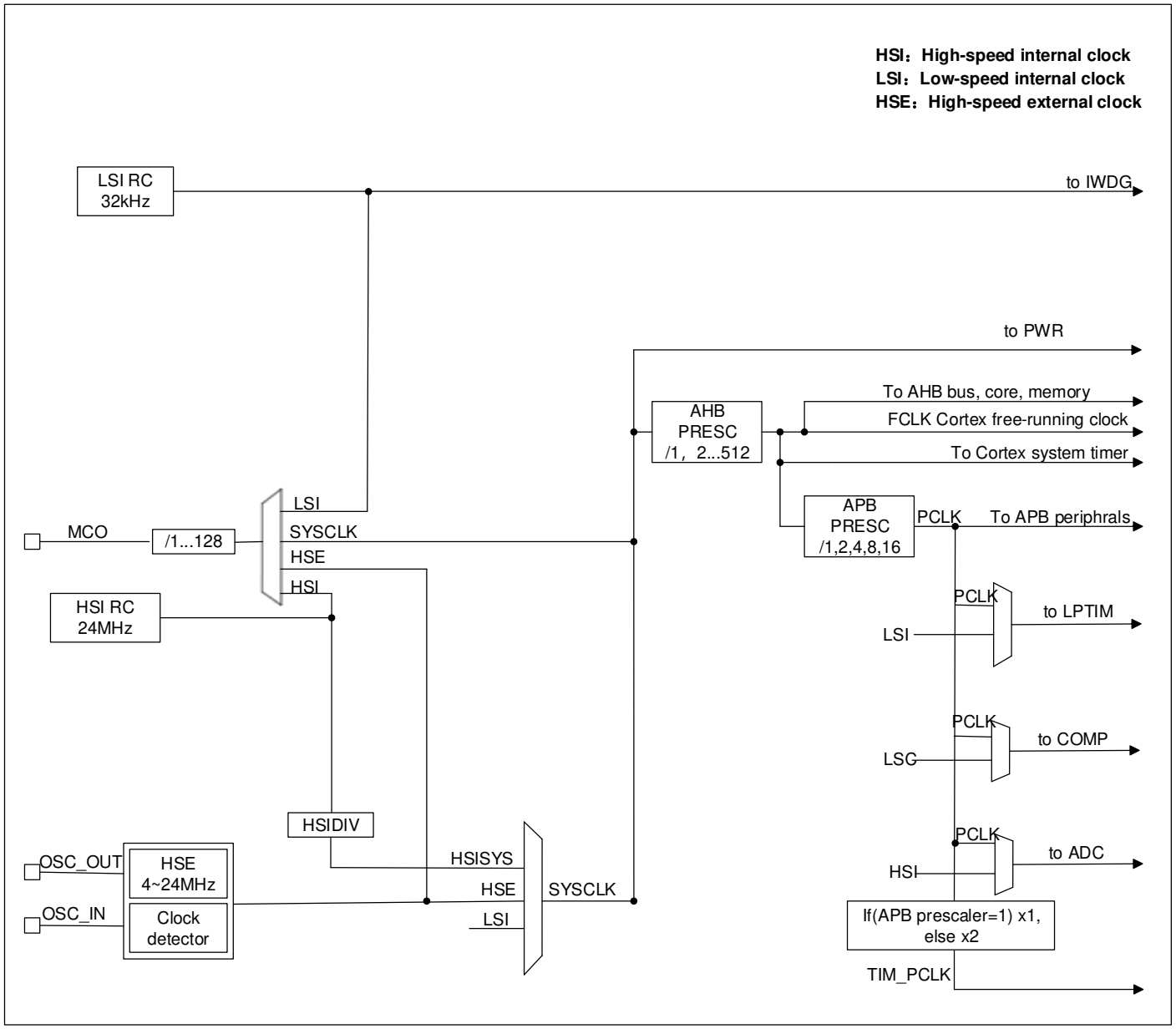


Figure 8-1 System clock structure

8.3. Clock security system (CSS)

The clock security system can be activated by software. In this case, the clock detection is enable 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 disable, a clock failure event is sent to the break input of advanced-control timer (TIM1) and general purpose timers (TIM16) and an interrupt is generated to inform the software of the failure (Clock Security System Interrupt CSSI), which allows the MCU to perform rescue operations. CSSI is linked to the Cortex-M0+ 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 will be executed indefinitely unless the CSS interrupt pending bit is cleared. Therefore, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the clock interrupt register (RCC_CICR).

If the HSE oscillator is used directly as the system clock, a detected failure cause a switch of the system clock to the HIS oscillator and the disabling of the HSE oscillator.

8.4. Clock-out capability

In order to facilitate board-level applications, save BOM costs and debug requirements, the chip needs to provide a clock output function. That is, the MCO signal (parallel frequency division) in the following table is used to realize the clock output function through the multiplexing function of GPIO.

Table 8-2 Output clock selection

Clock source	MCO output clock source
HSI	√
SYSCLK	√
HSE	√
LSI	√

Note: When switching the MCO clock source and selecting the GPIO AF function as the initial stage of the MCO, the MCO may generate glitches, and this period of time needs to be avoided.

8.5. Reset/clock register

The registers of this module can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit).

8.5.1. Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CSS ON	HSE BYP	HSE RDY	HSE ON
												RS	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	HSIDIV[2:0]			HSI RDY	Res	HSION	Res	Res	Res	Res	Res	Res	Res	Res
		RW			R		RW								

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19	HSE_CSSON	RS	0	Clock safety system enabled. Set by software to enable the clock security system. When this bit is set, if HSE is ready, the hardware will perform clock detection. When the clock is found to be invalid, the hardware disables the clock detection. This bit can only be set and cleared by reset. 0: Clock security system OFF (clock detection OFF) 1: Clock safety system ON (clock detection ON if HSE is stable, otheRWise OFF)
18	HSEBYP	RW	0	Bypass HSE to connect external crystal, select the pin input clock. It is set and cleared by software, the circumstance of by-passing the external crystal, and connecting the external pin to input the clock. The external clock must be enabled with HSEON. The HSEBYP bit is set only when the HSE external crystal is disabled. 0: HSE external crystal is not dropped by bypass 1: HSE external crystal is bypassed, and external pin input clock
17	HSERDY	R	0	HSE clock ready flag Set by hardware, indicating that the HSE is stable. 0: HSE not ready 1: HSE ready Note: When HSEON is cleared, HSERDY is cleared immediately
16	HSEON	RW	0	HSE clock enable

				Software can be set and cleared. Enter stop mode, the hardware clears this bit. This bit cannot be reset if HSE is used directly or indirectly as the system clock. 0: HSE OFF 1: HSE ON
15:14	Reserved	-	-	Reserved
13:11	HSIDIV[2:0]	RW	0	HSI clock division factor. Software controls these bits to set the frequency division factor of HSI to generate the HSISYS clock 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128
10	HSIRDY	R	0	HSI clock ready flag. Set by hardware to indicate HSI OSC is stable. This bit is only valid when HSION = 1. 0: HSI OSC not ready, 1: HSI OSC ready, When HSION is cleared, HSIRDY will be pulled low immediately.
9	Reserved	-	-	Reserved
8	HSION	RW	1	HSI clock enable bit. Software can set and clear this bit. When entering s top mode, the hardware clears this bit to stop HSI. When the HSI is used directly or indirectly as the system clock (also when exiting stop mode, or when the HSE is used as the system clock and fails). 0: HSI OFF 1: HSI ON
7:0	Reserved	-	-	Reserved

8.5.2. Clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	MCOPRE[2:0]			Res	MCOSEL[2:0]			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	PPRE[2:0]			HPRE[3:0]				Res	Res	SWS[2:0]			SW[2:0]		
	RW			RW						R			RW		

Bit	Name	R/W	Reset Value	Function
31	Reserved	-	-	Reserved
30:28	MCOPRE[2:0]	RW	0	Microcontroller clock output (MCO) frequency division factor. Software controls these bits to set the division factor of the MCO output: 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128 Set these bits before enabling the MCO output.
27	Reserved	-	-	Reserved
26:24	MCOSEL[2:0]	RW	0	MCO selection 000: No clock, MCO output disabled 001: SYSCLK 010: Reserved

				011: HSI 100: HSE 101: Reserved 110: LSI 111: Reserved Note: Incomplete output clock conditions may occur during the clock startup or switchover phase.
23:15	Reserved	-	-	Reserved
14:12	PPRE[2:0]	RW	0	This bit is controlled by software. To generate the PCLK clock, it sets the division factor of HCLK as follows: 0xx: 1 100: 2 101: 4 110: 8 111: 16
11:8	HPRE[3:0]	RW	0	AHB clock division factor. Software controls this bit. In order to generate the HCLK clock, it sets the frequency division factor of SYSCLK as follows: 0xxx: 1 1000: 2 1001: 4 1010: 8 1011: 16 1100: 64 1101: 128 1110: 256 1111: 512 In order to ensure the normal operation of the system, it is necessary to configure an appropriate frequency according to the VR power supply. Note: It is recommended to switch the frequency division factor step by step.
7:6	Reserved	-	-	Reserved
5:3	SWS[2:0]	R	0	System clock switch status bits These bits are controlled by hardware and indicate which clock source is currently being used as the system clock: 000: HSISYS 001: HSE 010: Reserved 011: LSI Others: Reserved
2:0	SW[2:0]	RW	0	System clock source selection bits. Controlled by software and hardware, these bits select the system clock: 000: HSISYS 001: HSE 010: Reserved 011: LSI Others: Reserved The hardware is configured as HSISYS include: 1) The system exits from stop mode 2) Software configuration 001 (HSE), HSE failure occurs (HSE is the system clock source.)

8.5.3. Internal clock source calibration register (RCC_ICSCR)

Address offset: 0x04

Reset value: 0x00FF 10FF, reset by POR/BOR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	LSI_STARTUP		Res	LSI_TRIM[8:0]								
				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**
HSI_FS[2:0]			HSI_TRIM[12:0]												
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function

31:28	Reserved		-	Reserved
27:26	LSI_STARTUP	RW	2'b00	Low-speed internal clock (LSI) stabilization time selection: 11: 256 LSI clock cycles 10: 64 LSI clock cycles 01: 16 LSI clock cycles 00: 4 LSI clock cycles
25	Reserved			
24:16	LSI_TRIM	RW	0x0FF	Low-speed internal clock frequency calibration. After power-on, the chip hardware will write the factory information (stored in 0x1FFF 0FA4) into this register, so that the LSI can output an accurate 32.768 kHz frequency. By rewriting the value of this register, the software increases (decrease) the output frequency of LSI by about 0.2% for each increase (decrease) by 1.
15:13	HSI_FS	RW	3'b000	HSI frequency selection: 001: 8 MHz 100: 24 MHz others: reserved Power on the default 8M
12:0	HSI_TRIM	RW	0x10FF	Clock frequency calibration value. After power-on, and the factory information (stored in 0 x1FFF 0FA0) will be written into this register when trimming. The software reads out the data stored in the corresponding address in the information area and writes it into the register to realize the calibration under the specific output frequency of the HSI. Save it in the following address of Flash: 24 MHz calibration value storage Address offset: 0x1FFF 0F10 8 MHz calibration value storage Address offset: 0x1FFF 0F04 After writing the calibration value to this register, the value of this register can also be the central value, and the value of this register can be modified. For each increase (decrease) by 1, the output frequency of the HSI will increase (decrease) by about 0.1%.

8.5.4. External clock source control register (RCC_ECSCR)

Address offset: 0x10

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	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	HSE_FREQ		Res	
												RW	RW		

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	RES	-	Resserved
3:2	HSE_FREQ	RW	0x0	HSE crystal oscillator operating frequency. 00: HSE off 01: 4 MHz to 8 MHz 10: 8 MHz to 16 MHz 11: 16 MHz to 24 MHz
1:0	Resserved	-	-	-

8.5.5. Clock interrupt enable register (RCC_CIER)

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	HSE RDYIE	HSI RDYIE	Res	Res	LSI RDYIE
											RW	RW			RW

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	-	-	Reserved
4	HSERDYIE	RW	0	HSE clock ready interrupt enable. 0: Disable 1: Enable
3	HSIRDYIE	RW	0	HSI clock ready interrupt enable. 0: Disable 1: Enable
2	Reserved	-	-	Reserved
1	Reserved	-	-	Reserved
0	LSIRDYIE	RW	0	LSI clock ready interrupt enable. 0: Disable 1: Enable

8.5.6. Clock interrupt flag register (RCC_CIFR)

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	CSSF	Res	Res	Res	HSE RDYF	HSI RDYF	Res	Res	LSI RDYF
							R				R	R			R

Bit	Name	R/W	Reset Value	Function
31:9	Reserved	-	-	Reserved
8	CSSF	R	0	HSE clock security system interrupt flag. When hardware detects HSE, this register is set when the OSC clock fails. 0: HSE clock detection failure interrupt is not generated, 1: HSE clock detection failure interrupt generation, Programming CSSC register 1 clears this bit.
7:5	Reserved	-	-	Reserved
4	HSERDYF	R	0	HSE ready interrupt flag This bit is set by hardware when HSE is stable and HSER-DYIE is enabled. Software clears this bit by setting the HSERDYC bit. 0: No clock ready interrupt caused by HSE 1: Clock ready interrupt caused by HSE
3	HSIRDYF	R	0	HIS ready interrupt flag This bit is set by hardware when HSI is stable and HSIRDYIE is enabled. Software clears this bit by setting the HSIRDYC bit. 0: No clock ready interrupt caused by HSI 1: Clock ready interrupt caused by HSI
2:1	Res	-	-	Reserved
0	LSIRDYF	R	0	LSI ready interrupt flag This bit is set by hardware when LSI is stable and LSIRDYIE is enabled. Software clears this bit by setting the LSIRDYC bit. 0: No clock ready interrupt caused by LSI 1: Clock ready interrupt caused by LSI

8.5.7. Clock interrupt clear register (RCC_CICR)

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	CSSC	Res	Res	Res	HSE RDYC	HSI RDYC	Res	Res	LSI RDYC
							W				W	W			W

Bit	Name	R/W	Reset Value	Function
31:9	Reserved	-	-	Reserved
8	CSSC	W	0	Clock safe interrupt clear bit. 0: No effect. 1: Clear the CSSF flag.
7:5	Reserved	-	-	Reserved
4	HSERDYC	W	0	HSE ready flag is cleared. 0: No effect. 1: Clear the HSERDYF bit.
3	HSIRDYC	W	0	HSI ready flag is cleared. 0: No effect. 1: Clear the HSIRDYF bit.
2	Reserved	-	-	Reserved
1	Reserved	-	-	Reserved
0	LSIRDYC	W	0	LSI ready flag is cleared. 0: No effect. 1: Clear the LSIRDYF bit.

8.5.8. I/O interface reset register (RCC_IOPRSTR)

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	GPIOF RST	Res	Res	Res	GPIOB RST	GPIOA RST
										RW				RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	Reserved	-	-	Reserved
5	GPIOFRST	RW	0	I/O PortF reset. 0: no effect, 1: PortF I/O reset
4:2	Reserved	-	-	Reserved
1	GPIOBRST	RW	0	I/O PortB resets. 0: no effect, 1: Port B I/O reset
0	GPIOARST	RW	0	I/O PortA resets. 0: no effect, 1: PortA I/O reset

8.5.9. AHB peripheral reset register (RCC_AHBRSTR)

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	CRC RST	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
			RW												

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	-	-	Reserved
12	CRCRST	RW	0	CRC module reset. 0: no effect, 1: CRC module reset,
11:9	Reserved	-	-	Reserved
8:0	Reserved	-	-	Reserved

8.5.10. APB peripheral reset register 1 (RCC_APBRSTR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM RST	Res	Res	PWR RST	DBG RST	Res	Res	Res	Res	Res	I2C RST	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	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31	LPTIMRST	RW	0	LP Timer module reset. 0: no effect, 1: The module is reset,
30:29	Reserved	-	-	Reserved
28	PWRRST	RW	0	Power interface module reset. 0: no effect, 1: The module is reset,
27	DBGRST	RW	0	MCU Debug module reset. 0: no effect, 1: The module is reset,
26:22	Reserved	-	-	Reserved
21	I2CRST	RW	0	I2C1 module reset. 0: no effect, 1: The module is reset,
20:0	Reserved	-	-	Reserved

8.5.11. APB peripheral reset register 2 (RCC_APBRSTR2)

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	COMP2 RST	COMP1 RST	ADC RST	Res	Res	TIM16 RST	Res
									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	SYSCFG RST
	RW		RW	RW											RW

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	-	Reserved
22	COMP2RST	RW	0	COMP2 module reset. 0: no effect, 1: The module is reset,
21	COMP1RST	RW	0	COMP1 module reset. 0: no effect, 1: The module is reset,
20	ADCRST	RW	0	ADC module reset. 0: no effect, 1: The module is reset,

19:18	Reserved	-	-	Reserved
17	TIM16RST	RW	0	TIM16 module reset. 0: no effect, 1: The module is reset,
16	Reserved	-	-	Reserved
15:14	USART1RST	RW	0	USART1 module reset. 0: no effect, 1: The module is reset,
13	Reserved	-	-	Reserved
12	SPI1RST	RW	0	SPI1 module reset. 0: no effect, 1: The module is reset,
11	TIM1RST	RW	0	TIM1 module reset. 0: no effect, 1: The module is reset,
10:1	Reserved	-	-	Reserved
0	SYSCFGRST	RWs	0	SYSCFG module reset. 0: no effect, 1: The module is reset,

8.5.12. I/O interface clock enable register (RCC_IOPENR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	GPIOFEN	Res	Res	Res	GPIOBEN	GPIOAEN
										RW				RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	Reserved	-	-	Reserved
5	GPIOFEN	RW	0	I/O PortF clock enable. 0: Clock disabled, 1: Clock enable
4:2	Reserved	-	-	Reserved
1	GPIOBEN	RW	0	I/O PortB clock enable. 0: Clock disabled, 1: Clock enable
0	GPIOAEN	RW	0	I/O PortA clock enable. 0: Clock disabled, 1: Clock enable

8.5.13. AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x38

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	CRCEN	Res	Res	SRAMEN	FLASHEN	Res	Res	Res	Res	Res	Res	Res	Res
			RW			RW	RW								

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	-	-	Reserved
12	CRCEN	RW	0	CRC module clock enable. 0: Disable 1: Enable
11:10	Reserved	-	-	Reserved
9	SRAMEN	RW	1	In sleep mode, the clock enable control of SRAM

				0: The module clock is disabled in sleep mode 1: The module clock is enabled in sleep mode Note: This bit only affects the clock enable of this module in sleep mode, in run mode, the clock of this module will not be disabled
8	FLASHEN	RW	1	In sleep mode, the clock enable control of FLASH 0: The module clock is disabled in sleep mode 1: The module clock is enabled in sleep mode Note: This bit only affects the clock enable of this module in sleep mode, in run mode, the clock of this module will not be disabled
7:0	Reserved	-	-	Reserved

8.5.14. APB peripheral clock enable register 1 (RCC_APBENR1)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM EN	Res	Res	PWR EN	DBG EN	Res	Res	Res	Res	Res	I2C EN	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	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31	LTIMEN	RW	0	LP Timer1 module clock enable. 0: Disable 1: Enable
30:29	Reserved	-	-	Reserved
28	PWREN	RW	0	Low power control block clock enable. 0: Disable 1: Enable
27	DBGEN	RW	0	Debug module clock enable. 0: Disable 1: Enable
26:22	Reserved	-	-	Reserved
21	I2CEN	RW	0	I2C1 module clock enable. 0: Disable 1: Enable
20:0	Reserved	-	-	Reserved

8.5.15. APB peripheral clock enable register 2 (RCC_APBENR2)

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	COMP2 EN	COMP1 EN	ADC EN	Res	Res	TIM16 EN	Res
									RW	RW	RW			RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	USART1 EN	Res	SPI1 EN	TIM1 EN	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SYSCFG EN
	RW		RW	RW											RW

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	-	Reserved
22	COMP2EN	RW	0	COMP2 module clock enable. 0: Disable 1: Enable
21	COMP1EN	RW	0	COMP1 module clock enable.

				0: Disable 1: Enable
20	ADCEN	RW	0	ADC module clock enable. 0: Disable 1: Enable
19:18	Reserved	-	-	Reserved
17	TIM16EN	RW	0	TIM16 module clock enable. 0: Disable 1: Enable
16:15	Reserved	-	-	Reserved
14	USART1EN	RW	0	USART1 module clock enable. 0: Disable 1: Enable
13	Reserved	-	-	Reserved
12	SPIEN	RW	0	SPI1 module clock enable. 0: Disable 1: Enable
11	TIM1EN	RW	0	TIM1 module clock enable. 0: Disable 1: Enable
10:1	Reserved	-	-	Reserved
0	SYSCFGEN	RW	0	SYSCFG module clock enable. 0: Disable 1: Enable

8.5.16. Peripheral independent clock configuration register (RCC_CCIPR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPTIM1SEL[1:0]		Res	Res
												RW	RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		Res		Res		COMP2SEL	COMP1SEL	Res	Res	Res	Res	Res	Res	Res	Res
						RW									

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19 :18	LPTIMSEL[1:0]	RW	2'b00	LPTIM1 internal clock source selection. 00: PCLK 01: LSI 10: No clock 11: Reserved
17:10	Reserved	-	-	Reserved
9	COMP2SEL	RW	0	COMP2 module clock source selection. 0: PCLK 1: LSC (Clock selected by RCC_BDCR.LSCOSEL) Note: Configure the selected LSC clock before enabling FLTEN.
8	COMP1SEL	RW	0	COMP1 module clock source selection. 0: PCLK 1: LSC (Clock selected by RCC_BDCR.LSCOSEL) Configure this register to select the clock before enabling COMP2_FR2.FLTEN.
7:0	Reserved	-	-	Reserved

8.5.17. RCC_BDCR

Address offset: 0x5C

Reset value: 0x0000 0000, reset by POR/BOR

When PWR_CR1.DBP is 1, it is allowed to write to this register.

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

							RW								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res		Res	Res	Res	Res		Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	-	-	Reserved
24	LSCOEN	RW	0	Low-speed clock enable. 0: Disable 1: Enable
23:0	Reserved	-	-	Reserved

8.5.18. Control/status register (RCC_CSR)

Address offset: 0x60

Reset value: 0x0000 0000

Reset sources: 1) [30:25]: POR reset, 2) LSION: system reset, 3) NRST_FLTIDS will not be reset by system reset reset

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	IWDG RSTF	SFT RSTF	PWR RSTF	PIN RSTF	OBL RSTF	Res	RMVF	Res	Res	Res	Res	Res	Res	Res
		R	R	R	R	R		RW							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	NRST_FLT-DIS	Res	Res	Res	Res	Res	Res	LSIRDY	LSION
							RW							R	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	IWDGRSTF	R	0	IWDG reset flag. Setting RMVF to 1 clears this bit.
28	SFTRSTF	R	0	Soft reset flag. Setting RMVF to 1 clears this bit.
27	PWRRSTF	R	0	BOR/POR/PDR reset flag. Setting RMVF to 1 clears this bit.
26	PINRSTF	R	0	External NRST pin reset flag. Setting RMVF to 1 clears this bit.
25	OBLRSTF	R	0	Option byte loader reset flag. Setting RMVF to 1 clears this bit.
24	Reserved			-
23	RMVF	RW	0	Reset flags [30:25] need to be cleared by software.
8	NRST_FLTDIS	RW	0	NRST filter disabled 0: HSI_10M is enabled, and the filter 20 us width function is enabled 1: The filtering function is disabled, and HSI_10M remains off
7:2	Reserved	-	-	Reserved
1	LSIRDY	R	0	LSI OSC stable flag. 0: LSI is not stable 1: LSI has stabilized
0	LSION	RW	0	LSI OSC enabled. 0: Disable 1: Enable Set by software, cleared by software. This bit is set by hardware when IWDG is enabled by hardware (via option byte).

8.5.19. RCC register address map

Offset	Register	0x000		0x004		0x008		0x010		0x014	0x018		0x01C		0x020		0x024	
		RCC_CR	Reset value	RCC_ICSCR	Reset value	RCC_CFGR	Reset value	RCC_ECSCR	Reset value	Reserved	RCC_CIER	RCC_CIFR	Reset value	RCC_CICR	Reset value	RCC_IOPRSTR	Reset value
31		Res.		Res.		Res.		Res.		Res.	Res.	Res.		Res.		Res.	
30		Res.		Res.		MCO-PRE[2:0]		Res.		Res.	Res.	Res.		Res.		Res.	
29		Res.		Res.				Res.		Res.	Res.	Res.		Res.		Res.	
28		Res.		Res.				Res.		Res.	Res.	Res.		Res.		Res.	
27		Res.		LSI_STARTUP[1:0]	0 0	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
26		Res.				MCO-SEL[2:0]		Res.		Res.	Res.	Res.		Res.		Res.	
25		Res.		Res.				Res.		Res.	Res.	Res.		Res.		Res.	
24		Res.		LSI_TRIM[8:0]	0		0	Res.		Res.	Res.	Res.		Res.		Res.	
23		Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
22		Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
21		Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
20		Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
19	HSE_CSS	Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
18	HSEBYP	Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
17	HSERDY	Res.			1	Res.		Res		Res.	Res.	Res.		Res.		Res.	
16	HSEON	Res.			1	Res.		Res		Res.	Res.	Res.		Res.		Res.	
15	Res.	Res.		HSI_FS[2:0]	0	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
14	Res.	Res.			0	PPRE[2:0]	0	Res.		Res.	Res.	Res.		Res.		Res.	
13	HSIDIV[2:0]	Res.			0		0	Res.		Res.	Res.	Res.		Res.		Res.	
12		Res.		HSI_TRIM[12:0]	1		0	Res.		Res.	Res.	Res.		Res.		Res.	
11		Res.			0	HPRE[3:0]	0	Res.		Res.	Res.	Res.		Res.		Res.	
10	HSIRDY	Res.			0		0	Res.		Res.	Res.	Res.		Res.		Res.	
9	Res.	Res.			0		0	Res.		Res.	Res.	Res		Res.		Res.	
8	HSION	Res.			0		0	Res.		Res.	Res.	Res.		Res.		Res.	
7	Res.	Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
6	Res.	Res.			1	Res.		Res.		Res.	Res.	Res.		Res.		Res.	
5	Res.	Res.			1	SWS[2:0]	0	Res.		Res.	Res.	Res.		Res.		GPIOFRS	0
4	Res.	HSER-	0		1		0	Res.		Res.	Res.	HSER-	0	HSER-	0	Res.	
3	Res.	HSIRDYI	0		1		0	HSE_FREQ[1:0]	0 0	Res.	Res.	HSIRDYI	0	HSIRDYI	0	Res.	
2	Res.	Res.			1	SW[2:0]	0			Res.	Res.	Res.		Res.			
1	Res.	Res.			1		0	Res.		Res.	Res.	Res		Res.		GPIOBRS	0
0	Res.	LSIRDYI	0		1		0	Res.		Res.	Res.	LSIRDYI	0	LSIRDYI	0	GPIOARS	0

Offset	0x28		0x2C		0x30		0x34		0x38		0x3C		0x40		0x44-0x50
Register	RCC	Reset value	RCC	Reset value	RCC	Reset value	RCC	Reset value	RCC	Reset value	RCC	Reset value	RCC	Reset value	Reserved
31	Res.		LPTIMRST		Res.		Res.		Res.		LPTIMEN	0	Res.		Res.
30	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
29	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
28	Res.		PWRRST	0	Res.		Res.		Res.		PWREN	0	Res.		Res.
27	Res.		DBGRST	0	Res.		Res.		Res.		DBGEN	0	Res.		Res.
26	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
25	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
24	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
23	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
22	Res.		Res.		COMP2RST	0	Res.		Res.		Res.		COMP2EN	0	Res.
21	I2CRST	0	Res.		COMP1RST	0	Res.		Res.		I2CEN	0	COMP1EN	0	Res.
20	Res.		Res.		ADCRST		Res.		Res.		Res.		ADCEN	0	Res.
19	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
18	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
17	Res.		TIM16RST	0	Res.		Res.		Res.		Res.		TIM16EN	0	Res.
16	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
15	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
14	Res.		USART1RST	0	Res.		Res.		Res.		Res.		USART1EN	0	Res.
13	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
12	CRCRST	0	Res.		SPI1RST	0	Res.		CRCEN	0	Res.		SPI1EN	0	Res.
11	Res.		Res.		TIM1RST	0	Res.		Res.		Res.		TIM1EN	0	Res.
10	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
9	Res.		Res.		Res.		Res.		SRAMEN	1	Res.		Res.		Res.
8	Res.		Res.		Res.		Res.		FLASHE	1	Res.		Res.		Res.
7	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
6	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
5	Res.		Res.		Res.		GPIOFEN	0	Res.		Res.		Res.		Res.
4	RST	0	Res.		Res.		Res.		Res.			0	Res.		Res.
3	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
2	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.
1	Res.		Res.		Res.		GPIOBEN	0	Res.		Res.		Res.		Res.
0	Res.		SYSCFGRST	0	Res.		GPIOAEN	0	Res.		Res.		SYSCFGEN	0	Res.

Offset	0x54		0x58	0x5C		0x60	
Register	RCC_CCIPR	Reset value	Reserved	RCC_BDCR	Reset value	RCC_CSR	Reset value
31	Res.		Res.	Res.		Res.	
30	Res.		Res.	Res.		Res.	
29	Res.		Res.	Res.		IWDGRSTF	0
28	Res.		Res.	Res.		SFTRSTF	0
27	Res.		Res.	Res.		PWRRSTF	0
26	Res.		Res.	Res.		PINRSTF	0
25	Res.		Res.	LSCOSEL	0	OBLRSTF	0
24	Res.		Res.	LSCOEN	0	Res.	
23	Res.		Res.	Res.		RMVF	0
22	Res.		Res.	Res.		Res.	
21	Res.		Res.	Res.		Res.	
20	Res.		Res.	Res.		Res.	
19	LPTIMSEL[1:0]	0 0	Res.	Res.		Res.	
18			Res.	Res.		Res.	
17	Res.		Res.	Res.		Res.	
16	Res.		Res.	BDRST	0	Res.	
15	Res.		Res.	Res.		Res.	
14	Res.		Res.	Res.		Res.	
13	Res.		Res.	Res.		Res.	
12	Res.		Res.	Res.		Res.	
11	Res.		Res.	Res.		Res.	
10	Res.		Res.	Res.		Res.	
9	COMP2SEL	0	Res.	Res		Res.	
8	COMP1SEL		Res.			NRST_FLT-	0
7	Res.		Res.	Res.		Res.	
6	Res.		Res.	Res.		Res.	
5	Res.		Res.	Res.	0	Res.	
4	Res.		Res.	Res.		Res.	
3	Res.		Res.	Res.		Res.	
2	Res.		Res.	Res.		Res.	
1	Res.		Res.	Res.		LSIRDY	0
0	Res.		Res.	Res.		LSION	0

9. General-purpose I/Os (GPIO)

9.1. GPIO introduction

Each GPIO port has:

Four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR)

Two 32-bit data registers (GPIOx_IDR and GPIOx_ODR)

One 32-bit set/reset register (GPIOx_BSRR)

One 32-bit lock register (GPIOx_LCKR)

Two alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

9.2. GPIO main features

- Output status: 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 status: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set/reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configuration function
- Analog function
- Alternate functions selection registers (at most 16 AFs per I/O port)
- Fast toggle capable of changing every single cycle
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral function

9.3. GPIO functional description

Each port bit of the GPIO ports can be individually configured by software in several modes:

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

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

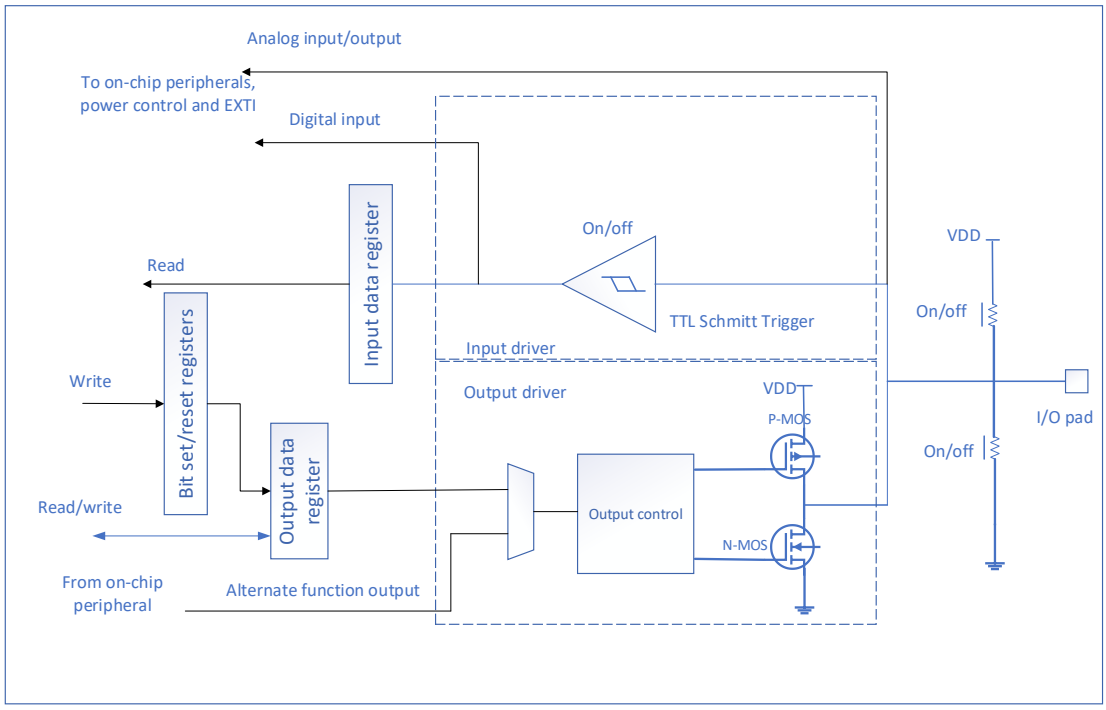


Figure 9-1 Basic structure of an I/O port bit

9.3.1. General-purpose I/O (GPIO)

During and after reset, the alternate functions are not active and most of the IOs are configured in analog mode.

The debug pins are in alternate function pull-up or pull-down after reset:

- PA14-SWCLK: in pull-down mode
- PA13-SWDIO: in pull-up mode

Boot pin is set to input pull-down mode after reset:

- PF4-Boot: in pull-down mode

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 drive 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 multiplexers 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 port has a multiplexer with up to 16 alternate function inputs (AF0 to AF7), which can be configured through the registers GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15).

- After reset, the multiplexer selection is AF0. The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The alternate function assignments for each pin are details in section 2.3.

In addition to this flexible multiplexer architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

The user configures IO as follows:

- Debug function: After each reset, these pins are assigned as alternate function pins immediately usable by the debugger host.
- GPIO: Configure the corresponding I/O port as output, input or analog mode in GPIOx_MODER register.
- Peripheral multiplexing function:
  - The I/O corresponding to the register GPIOx_AFRL or GPIOx_AFRH configuration is the alternate function x (x = 0… 15).
  - Registers GPIOx_OTYPER, GPIOx_PUPDR and GPIOX_OSPEEDER configure the type, pull-up/pull-down and output speed respectively.
  - Configure the corresponding I/O as an alternate function in the GPIOx_MODER register.
- Additional functions:
  - ADC and COMP functions are enabled in the registers of the ADC and COMP modules, in every I/O configuration. When the I/O is used as ADC or COMP, it is recommended to configure the port as analog mode through the register GPIOx_MODER
  - For additional functions of the crystal oscillator, configure the respective functions in the corresponding PWR and RCC module registers. These configurations have higher priority than standard GPIO configurations.

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 and GPIOx_PUPDR) to configure up to 16 I/Os. The register GPIOx_MODER 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 whateverr 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/written accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

9.3.5. I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bits 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 of GPIOx_BSRR: BS(i) and BR(i). When written bit BS(i) to 1 can set the corresponding bit of GPIOx_ODR to 1, and setting bit BR(i) to 1 can clear the corresponding bit of GPIOx_ODR to 0.

Write 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 operation has priority.

Using the GPIOx_BSRR register to change the values of individual bit 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 sinle atomic AHB write access.

9.3.6. GPIO locking mechanism

It is possible to freeze the IO control with GPIOx_LCKR registers through a series of special write timings, including GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

A special write/read sequence can manipulate the register GPIOx_LCKR. When the right lock sequence is applied to bit 16 in this register, the value of LCKR[15:0] can LOCK the I/O (during the write sequence, the value of LCKR[15:0] remains unchanged). When the LOCK sequence has been applied to a port bit, the value of the port bit cannot be modify 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 GPIOx_LCKR register with a word (32 bits) because the [15:0] bits are also set when the GPIOx_LCKR bit 16 is set.

9.3.7. I/O alternate function input/output

Two registers are provided to select one of the alternate function input/outputs available for each I/O. The user can connect an alternate function to the IO port according 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.

9.3.8. External interrupt/wakeup lines

All ports have external interrupt capability. To use the external interrupt lines, the given pin must be disabled in analog mode or as oscillator pin, so the input trigger is kepy enabled.

9.3.9. I/O input configuration

When the I/O port is configured as input:

- The output buffer is disabled
- The Schmitt trigger input is enable
- The pull-up and pull-down resistors can be enabled/disabled according to the configuration of the GPIOx_PUPDR register
- The data present on the I/O pins are sampled into the input data register on every AHB clock cycle
- A read access to the input data register provides the I/O status

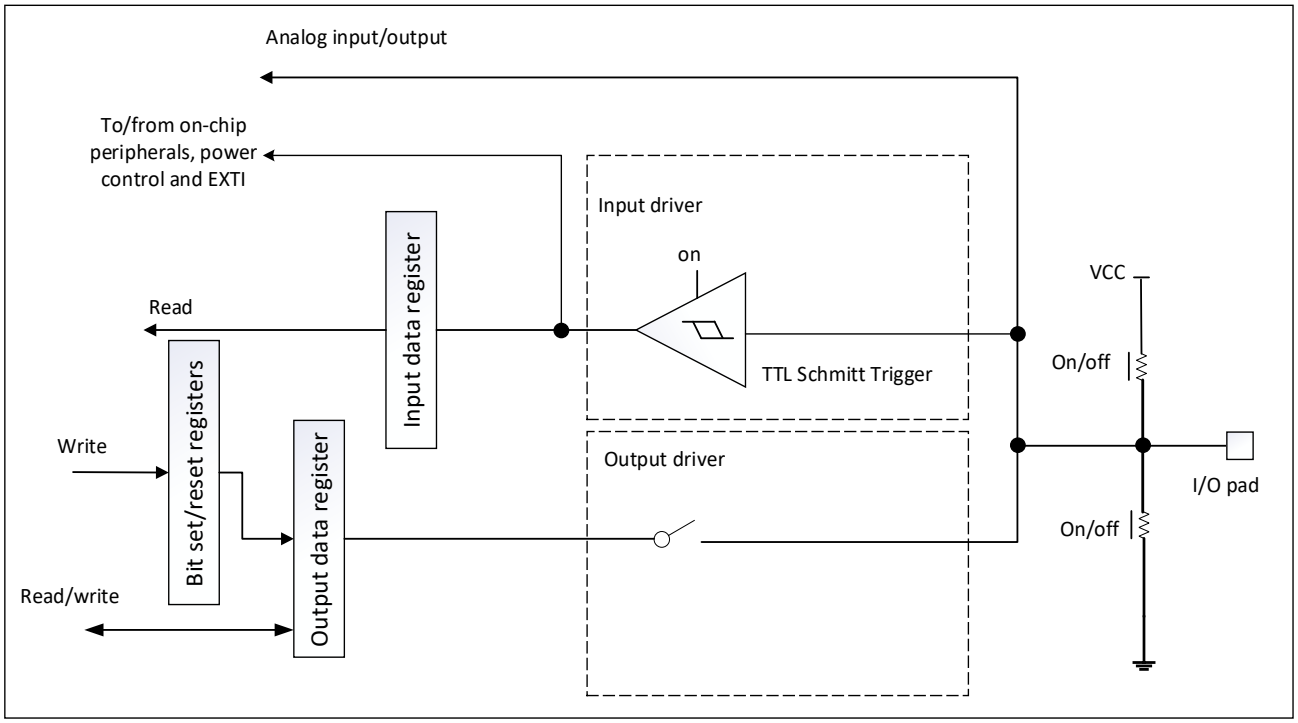


Figure 9-2Input floating/pull up/pull down configurations

9.3.10. I/O output configuration

When the I/O port is configured 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 a high-impedance state (the PMOS 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 can be enabled/disabled according to the configuration of the GPIOx_PUPDR register
- The data present on the I/O pins 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 value of the last write

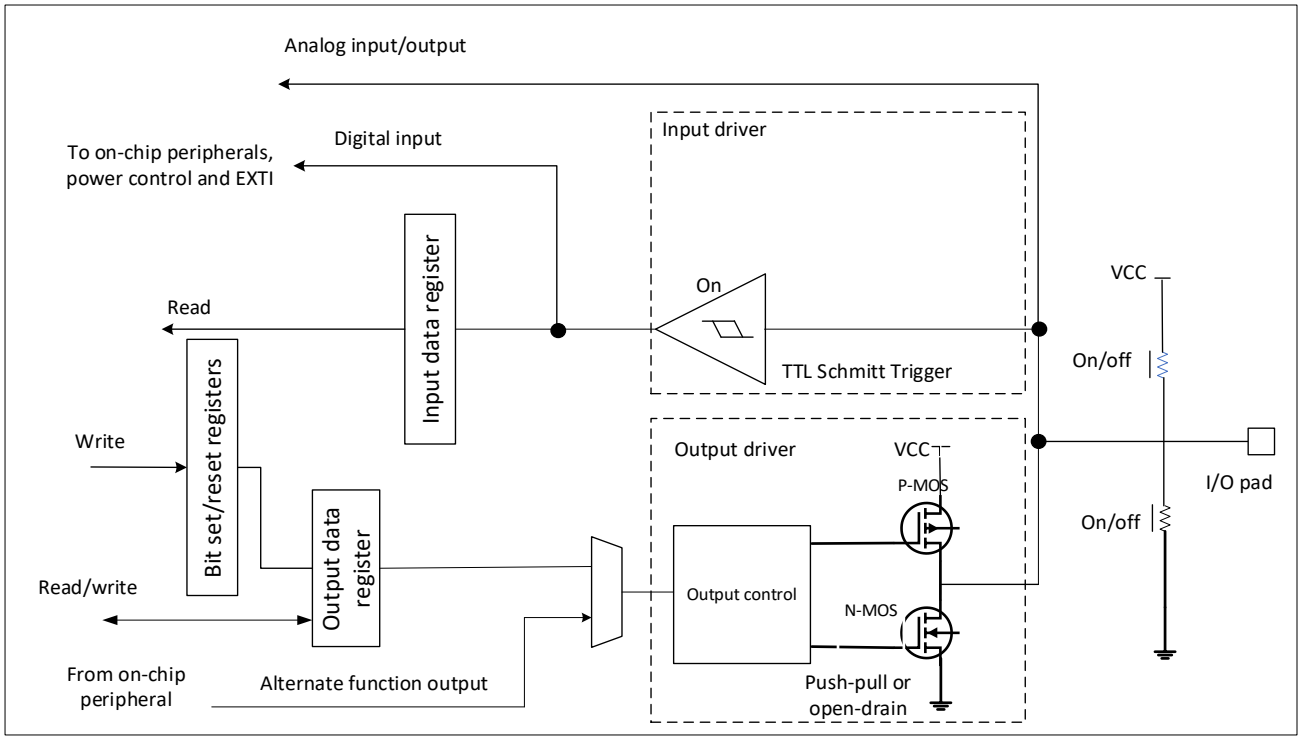


Figure 9-3 Output configuration

9.3.11. Alternate function configuration

When an I/O port is configured as alternate function:

- In an open-drain or push-pull configuration, the output buffer is turned on
- Built-in peripheral signal-driven output buffer (multiplexed function output)
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors can be enabled/disabled according to the configuration of the GPIOx_PUPDR register
- The data present on the I/O pins are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

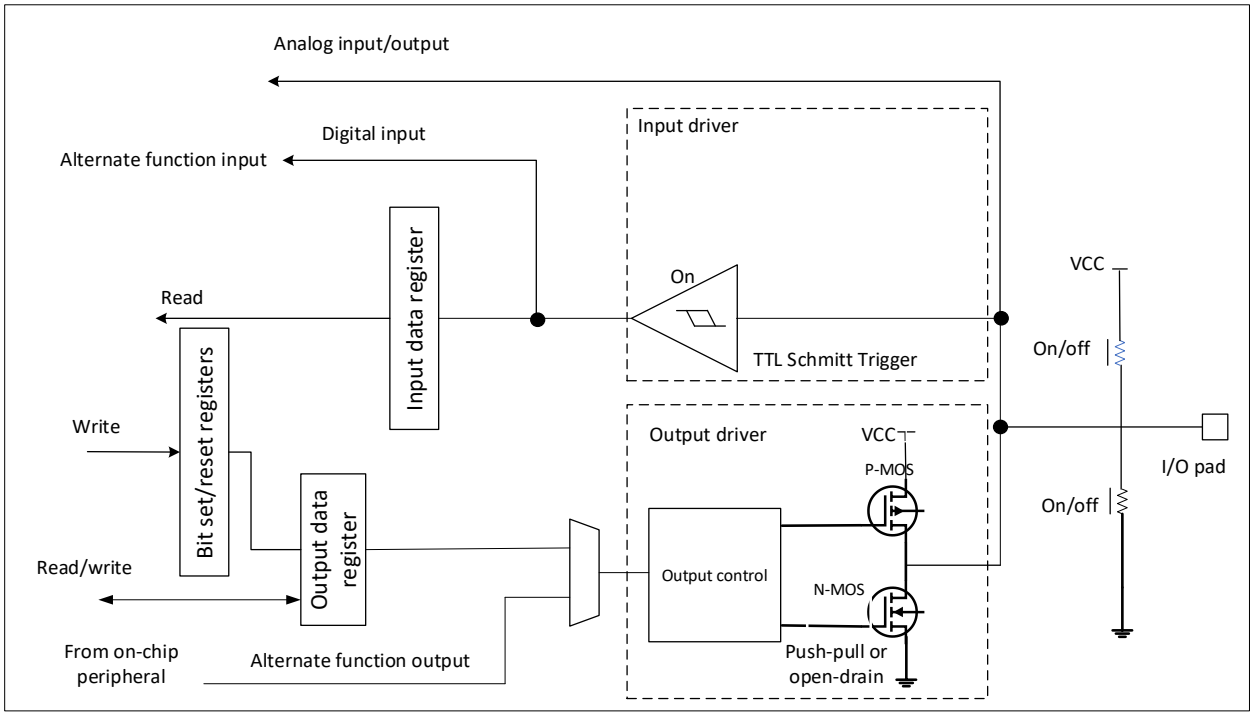


Figure 9-4 Alternate function configuration

9.3.12. Analog configuration

When an I/O port is configured 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 Schmitt trigger is forced to '0'
- The weak pull-up and pull-down resistors are disabled (software setting required)
- Read access to the input data register gets the value is '0'

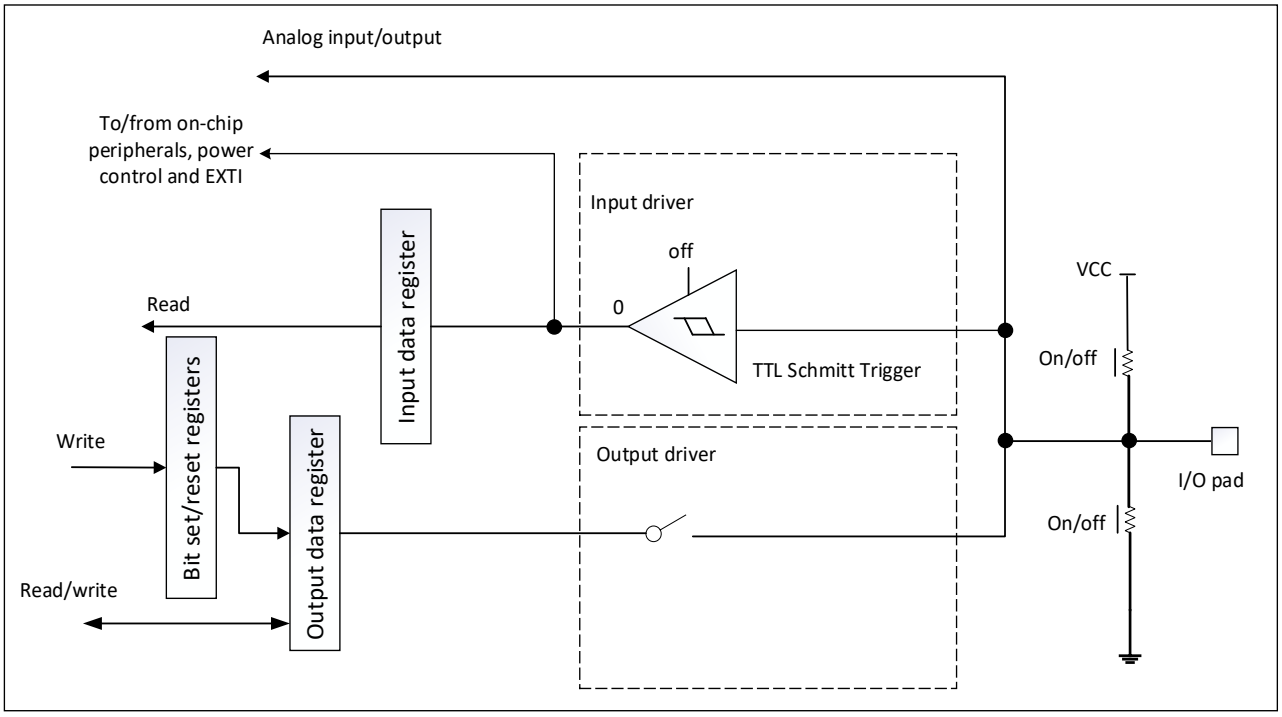


Figure 9-5 High impedance-analog configuration

9.3.13. Use the HSE oscillator pins as GPIOs

When the HSE oscillator is switch off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE oscillator is switch on (by setting the HSEON bit in the RCC_CSR register) the corresponding port needs to be configured as an analog port by software.

When the crystal oscillator is configured in a user external clock mode, only the pin is reserved for clock input and the OSC_IN or OSC32_IN pin can still be used as normal GPIO.

9.4. GPIO registers

The GPIO related registers can be written in word, half word and byte mode.

9.4.1. GPIO port mode register (GPIOx_MODER) (x = A, B, F)

Address offset: 0x00

Reset value:

- 0xEBFF FFFF for GPIOA
- 0xFFFF FFFF for GPIOB
- 0xFFFF FCFF for GPIOF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	MODEy[1:0]	RW		y = 15..0 These bits are written by software to configure the I/O mode 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode (reset state)

9.4.2. GPIO port output type register (GPIOx_OTYPER) (x = A, B, 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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	MODE[15:0]	RW		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, B, F)

Address offset: 0x08

Reset value:

- 0x0C00 0000 (for port A)
- 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15		OSPEED14		OSPEED13		OSPEED12		OSPEED11		OSPEED10		OSPEED9		OSPEED8	
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
OSPEED7		OSPEED6		OSPEED5		OSPEED4		OSPEED3		OSPEED2		OSPEED1		OSPEED0	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	OSPEEDy[1:0]	RW		y = 15..0 These bits are written by software to configure the I/O output speed 00:Very low speed 01:Low speed 10:High speed 11:Very high speed

9.4.4. GPIO port pull-up and pull-down register (GPIOx_PUPDR) (x = A, B, F)

Address offset: 0x0C

Reset value:

- 0x2400 0000 (for port A)
- 0x0000 0000 (for port B)
- 0x0000 0200 (for port F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31 :0	PUPDy[1:0]	RW		y = 15..0 These bits are written by software to configure the I/O pull-up or pull-down 00: No pull-up or pull-down 01: Pull-up 10: Pull-down 11: Reserved

9.4.5. GPIO port input data register (GPIOx_IDR) (x = A, B, F)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function

31:16	Reserved			
15:0	Idy	R		y = 15..0 This is read-only, it contain the input value of the corresponds I/O port

9.4.6. GPIO port output data register (GPIOx_ODR) (x = A, B, F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	Ody[1:0]	RW		y = 15..0 These bits are readable and writable by software. Note: For GPIOx_BSRR or GPIOx_BRR registers. (x = A,B,F), each ODR bit can be independently set/cleared.

9.4.7. GPIO port bit set/reset register (GPIOx_BSRR) (x = A, B, 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

Bit	Name	R/W	Reset Value	Function
31:16	BRy	W		y = 15..0 These bits are write-only. A read to these bits returns the value of 0. 0: No action on the corresponding ODRy bit 1: Clear the corresponding ODRy bit Note: If the corresponding bits of Bsy and Bry are set at the same time, the Bsy bit has priority.
15:0	BSy	W		y = 15..0 These bits are write-only. A read to these bits returns the value of 0. 0: No action on the corresponding ODRy bit 1: Set the corresponding ODRy bit

9.4.8. GPIO port configuration lock register (GPIOx_LCKR) (x = A, B, F)

This register is used to lock the configuration of the port bits when the correct write sequence is applied to bit 16 (LCKK) set. The value of bits [15:0] is used to lock the configuration of the GPIO, the value of LCKR [15:0] must not change. When the LOCK sequence has been applied on the a port bit, the configuration of the port bits cannot be changed until the next system reset.

Note: A special write sequence is used to write the GPIOx_LCKR register. Only word accesses can be performed during the lock 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

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	LCKK	RW		This bit can be read any time, it can only be modified by the lock key write sequence 0: The port configuration lock key not active 1: The port configuration lock key activated, and the GPIOx_LCKR register is locked until the next system reset LOCK key write sequence: The write sequence of the lock key: write 1- > write 0- > write 1- > read 0- > read 1. The last read can be ignored, but it can be used to confirm that the lock key has been activated. Note: During the LOCK key write sequence, the value of LCK[15:0] must not change. Any error in the lock sequence will stop the lock key from being activated. After the first lock sequence on any bit of the port, any read access on the LCKK will return 1 until the next MCU reset or peripheral reset.
15:0	LCKy	RW		y = 15..0 These bits are readable and writable 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 register (low) (GPIOx_AFRL) (x = A, B, F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] (y = 7 to 0)	RW		These bits are written by software to configure alternate function I/O. AFSELy selection: 0000: AF0 1000: AF8 0001: AF1 1001: AF9 0010: AF2 1010: AF10 0011: AF3 1011: AF11 0100: AF4 1100: AF12 0101: AF5 1101: AF13 0110: AF6 1110: AF14 0111: AF7 1111: AF15

9.4.10. GPIO alternate function register (high) (GPIOx_AFRH) (x = A, B, F)

Address offset: 0x24

Reset value: 0x0000 0000

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

AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
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**
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] (y = 8 to 15)	RW		These bits are written by software to configure alternate function I/O. AFSELy selection: 0000: AF0 1000: AF8 0001: AF1 1001: AF9 0010: AF2 1010: AF10 0011: AF3 1011: AF11 0100: AF4 1100: AF12 0101: AF5 1101: AF13 0110: AF6 1110: AF14 0111: AF7 1111: AF15

9.4.11. GPIO port bit reset register (GPIOx_BRR) (x = A, B, 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															
15	**14**	**13**	**2**	**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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	Bry	RW		y = 15..0 These bits are write-only. A read to these bots returns the value of 0. 0: No action on the corresponding Ody bit 1: Clear the corresponding Ody bit

9.4.12. GPIO register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	GPIOF_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	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	1	1	1	1	1	1

0x04		0x08		0x08		0x0C		0x0C		0x10		0x14		0x18	
GPIOF_OTYPER (x = A, B, F)		GPIOA_OSPEEDR		GPIOx_OSPEEDR (x = B, F)		GPIOA_PUPDR		GPIOB_PUPDR		GPIOx_IDR (x = A, B, F)		GPIOx_ODR (x = A, B, F)		GPIOx_BSRR (x = A, B, F)	
Reset value		Reset value		Reset value		Reset value		Reset value		Reset value		Reset value		Reset value	
	Res.	0	OSPEED15[1:0]	0	OSPEED15[1:0]	0	PUPD15[1:0]	0	PUPD15[1:0]		Res.		Res.	0	BR15
	Res.	0		0		0		0			Res.		Res.	0	BR14
	Res.	0	OSPEED14[1:0]	0	OSPEED14[1:0]	1	PUPD14[1:0]	0	PUPD14[1:0]		Res.		Res.	0	BR13
	Res.	0		0		0		0			Res.		Res.	0	BR12
	Res.	1	OSPEED13[1:0]	0	OSPEED13[1:0]	0	PUPD13[1:0]	0	PUPD13[1:0]		Res.		Res.	0	BR11
	Res.	1		0		1		0			Res.		Res.	0	BR10
	Res.	0	OSPEED12[1:0]	0	OSPEED12[1:0]	0	PUPD12[1:0]	0	PUPD12[1:0]		Res.		Res.	0	BR9
	Res.	0		0		0		0			Res.		Res.	0	BR8
	Res.	0	OSPEED11[1:0]	0	OSPEED11[1:0]	0	PUPD11[1:0]	0	PUPD11[1:0]		Res.		Res.	0	BR7
	Res.	0		0		0		0			Res.		Res.	0	BR6
	Res.	0	OSPEED10[1:0]	0	OSPEED10[1:0]	0	PUPD10[1:0]	0	PUPD10[1:0]		Res.		Res.	0	BR5
	Res.	0		0		0		0			Res.		Res.	0	BR4
	Res.	0	OSPEED9[1:0]	0	OSPEED9[1:0]	0	PUPD9[1:0]	0	PUPD9[1:0]		Res.		Res.	0	BR3
	Res.	0		0		0		0			Res.		Res.	0	BR2
	Res.	0	OSPEED8[1:0]	0	OSPEED8[1:0]	0	PUPD8[1:0]	0	PUPD8[1:0]		Res.		Res.	0	BR1
	Res.	0		0		0		0			Res.		Res.	0	BR0
0	OT15	0	OSPEED7[1:0]	0	OSPEED7[1:0]	0	PUPD7[1:0]	0	PUPD7[1:0]	x	ID15	0	OD15	0	BS15
0	OT14	0		0		0		0		x	ID14	0	OD14	0	BS14
0	OT13	0	OSPEED6[1:0]	0	OSPEED6[1:0]	0	PUPD6[1:0]	0	PUPD6[1:0]	x	ID13	0	OD13	0	BS13
0	OT12	0		0		0		0		x	ID12	0	OD12	0	BS12
0	OT11	0	OSPEED5[1:0]	0	OSPEED5[1:0]	0	PUPD5[1:0]	0	PUPD5[1:0]	x	ID11	0	OD11	0	BS11
0	OT10	0		0		0		0		x	ID10	0	OD10	0	BS10
0	OT9	0	OSPEED4[1:0]	0	OSPEED4[1:0]	0	PUPD4[1:0]	0	PUPD4[1:0]	x	ID9	0	OD9	0	BS9
0	OT8	0		0		0		0		x	ID8	0	OD8	0	BS8
0	OT7	0	OSPEED3[1:0]	0	OSPEED3[1:0]	0	PUPD3[1:0]	0	PUPD3[1:0]	x	ID7	0	OD7	0	BS7
0	OT6	0		0		0		0		x	ID6	0	OD6	0	BS6
0	OT5	0	OSPEED2[1:0]	0	OSPEED2[1:0]	0	PUPD2[1:0]	0	PUPD2[1:0]	x	ID5	0	OD5	0	BS5
0	OT4	0		0		0		0		x	ID4	0	OD4	0	BS4
0	OT3	0	OSPEED1[1:0]	0	OSPEED1[1:0]	0	PUPD1[1:0]	0	PUPD1[1:0]	x	ID3	0	OD3	0	BS3
0	OT2	0		0		0		0		x	ID2	0	OD2	0	BS2
0	OT1	0	OSPEED0[1:0]	0	OSPEED0[1:0]	0	PUPD0[1:0]	0	PUPD0[1:0]	x	ID1	0	OD1	0	BS1
0	OT0	0		0		0		0		x	ID0	0	OD0	0	BS0

0x1C	GPIOx_LCKR (x = A, B, 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 (x = A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (x = A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x = A, B, 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

10. System configuration controller (SYSCFG)

The devices feature a set of configuration registers. The main purpose of the system configuration controller are:

- Enable or disable I2C Fast Mode Plus on some IO pins
- Remap the memory located at the beginning of the code area
- Manage the external interrupts connected to GPIOs
- Manage robustness features

10.1. System configuration register

10.1.1. SYSCFG configuration register 1 (SYSCFG_CFGR1)

This register is used for specific configuration of memory request remap and control special I/O functions.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These two bits are used to select the physical remap by software, and bypass the hardware BOOT selection. After reset, these bits take the value configured by the actual boot mode.

Address offset: 0x00

Reset value: 0x0000 000x (x is the memory mode selected by the actual boot mode configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	I2C_PF1_ANF	I2C_PF0_ANF	I2C_PB8_ANF	I2C_PB7_ANF	I2C_PB6_ANF	I2C_PA12_ANF	I2C_PA11_ANF	I2C_PA10_ANF	I2C_PA9_ANF	I2C_PA8_ANF	I2C_PA7_ANF	I2C_PA3_ANF	I2C_PA2_ANF	Res	Res
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MEM_MODE[1:0]	
														RW	

Bit	Name	R/W	Reset Value	Function
31	Reserved	RW	-	Read and write
30:18	I2C_IOx_ANF	RW	0	Analog filter enable control of I2C related I/O 0: Analog filter disable 1: Analog filter enable
17:2	Reserved	RW	0	Read and write
1:0	MEM_MODE [1:0]			Memory mapping selection bit Set and clear by software. They control the mapping of memory at address 0x0000 0000. After reset, these bits take on the actual boot mode configuration values. X0: Main Flash, mapped at 0x0000 0000 01: System Flash, mapped at 0x0000 0000 11: SRAM, mapped at 0x0000 0000

10.1.2. SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x18

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	ETR_SRC_TIM1		Res	Res	COMP2_BRK_TIM16	COMP1_BRK_TIM16	COMP2_BRK_TIM1	COMP1_BRK_TIM1	Res	Res	LOCKUP_LOCK
					RW				RW	RW	RW	RW			RW

Bit	Name	R/W	Reset Value	Function
31:11	Reserved	-	-	-
10:9	ETR_SRC_TIM1[1:0]	RW	2'b00	TIMER1 ETR input source selection. 2 'b00: ETR source from GPIO 2 'b01: ETR source from COMP1 2 'b10: ETR source from COMP2 2 'b11: ETR source from ADC
8:7	Reserved	-	-	-
6	COMP2_BRK_TIM16	RW	0	COMP2 as TIMx break input enable. 0: COMP2 output is not used as TIM16 break input 1: COMP2 output as TIM16 break input
5	COMP1_BRK_TIM16	RW	0	COMP1 as TIMx break input enable. 0: COMP1 output is not used as TIM16 break input 1: COMP1 output as TIM16 break input
4	COMP2_BRK_TIM1	RW	0	COMP2 as TIMx break input enable. 0: COMP2 output is not used as TIM1 break input 1: COMP2 output as TIM1 break input
3	COMP1_BRK_TIM1	RW	0	COMP1 as TIMx break input enable. 0: COMP1 output is not used as TIM1 break input 1: COMP1 output as TIM1 break input
2:1	Reserved	-	-	-
0	LOCKUP_LOCK	RW		Cortex-M0+ LOCKUP enable bit Set by software and cleared by system reset. It can enable and lock the LOCKUP (hardfault) output of Cortex-M0+ to the brake input of TIM1/TIM16. 0: The LOCKUP output of Cortex-M0+ is not connected to the brake input of TIM1/TIM16 1: The LOCKUP output of Cortex-M0+ is connected to the brake input of TIM1/TIM16

10.1.3. SYSCFG register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_CFGR1	Res.	I2C_PF1_ANF	I2C_PF0_ANF	I2C_PB8_ANF	I2C_PB7_ANF	I2C_PB6_ANF	I2C_PA12_ANF	I2C_PA11_ANF	I2C_PA10_ANF	I2C_PA9_ANF	I2C_PA8_ANF	I2C_PA7_ANF	I2C_PA3_ANF	I2C_PA2_ANF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM_MODE[1:0]	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0																	X	X

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.	ETR_SRC_TIM1[1:0]		Res.	Res.	COMP2_BRK_TIM16	COMP1_BRK_TIM16	COMP2_BRK_TIM21	COMP2_BRK_TIM22	COMP2_BRK_TIM23	Res.	LOCKUP_LOCK
	Reset value																						0				0	0	0	0	0		0

11. Interrupts and events

11.1. Nested vectored interrupt controller (NVIC)

11.1.1. NVIC main features

- 32 maskable interrupt channels (not including the 16 ARM® Cortex®-M0 interrupt lines)
- 4 programmable priority levels (2 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.

11.1.2. SysTick calibration value register

The SysTick calibration value is set to 6000, which gives a reference time base of 1 ms with the SysTick clock set to 6 MHz (max fHCLK/8).

11.1.3. Interrupt and exception vectors

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-	fixed	Reset	Reset	0x0000_0004
-	-	fixed	NMI_Handler	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-	fixed	HardFualt_Handler	All class of fault	0x0000_000C
-	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
-	5	settable	PendSV	Pendable request for system service	0x0000_0038
-	6	settable	SysTick	System tick timer	0x0000_003C
-	7	-	Reserved	Reserved	0x0000_0040
-	8	-	Reserved	Reserved	0x0000_0044
-	9	-	Reserved	Reserved	0x0000_0048
3	10	settable	Flash	Flash global interrupt	0x0000_004C
4	11	settable	RCC	RCC global interrupt	0x0000_0050
5	12	settable	EXTI0_1	EXTI line[1:0] interrupt	0x0000_0054
6	13	settable	EXTI2_3	EXTI line[3:2] interrupt	0x0000_0058
7	14	settable	EXTI4_15	EXTI line[15:4] interrupt	0x0000_005C
-	15	-	Reserved	Reserved	0x0000_0060
-	16	-	Reserved	Reserved	0x0000_0064
-	17	-	Reserved	Reserved	0x0000_0068
-	18	-	Reserved	Reserved	0x0000_006C
12	19	settable	ADC_COMP	ADC and COMP interrupts (COMP combined with EXTI 17 & 18)	0x0000_0070
13	20	settable	TIM1_BRK_UP_TRG_COM	TIM1 break, update, trigger and commutation interrupt	0x0000_0074
14	21	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000_0078
15	22	-	Reserved	Reserved	0x0000_007C
-	23	-	Reserved	Reserved	0x0000_0080
17	24	settable	LPTIM1	LPTIM interrupt	0x0000_0084
18	25	-	Reserved	Reserved	0x0000_0088
-	26	-	Reserved	Reserved	0x0000_008C

20	27	-	Reserved	Reserved	0x0000_0090
21	28	settable	TIM16	TIM16 global interrupt	0x0000_0094
-	29	-	Reserved	Reserved	0x0000_0098
23	30	settable	I2C1	I2C1 global interrupt	0x0000_009C
24	31	-	Reserved	Reserved	0x0000_00A0
25	32	settable	SPI1	SPI1 global interrupt	0x0000_00A4
26	33	-	Reserved	Reserved	0x0000_00A8
27	34	settable	USART1	USART1 global interrupt	0x0000_00AC
-	35	-	Reserved	Reserved	0x0000_00B0
29	36	-	-	-	0x0000_00B4
30	37	-	Reserved	Reserved	0x0000_00B8
31	38	-	Reserved	Reserved	0x0000_00BC

- The grayed cells (the address less than 0x0000 0040) correspond to the Cortex®-M0+ interrupts.

11.2. Extended interrupts and events controller (EXTI)

The extended interrupt and event controller, through configurable (configurable) and direct (direct event) input (Lines), manages the CPU and system wake-up functions, and outputs the following request signals:

- Interrupt request, sent to the int_ctrl module to generate the IRQ of the CPU
- Event request, event input to CPU (RXEV)
- Wake-up request, sent to power management control module

EXTI wakeup request allows the system to wake up from stop mode, interrupt request and event request can also be used in run mode.

EXTI allows to manage up to 21 configurable/direct event lines (19 configurable event lines and 2 direct event lines).

11.2.1. EXTI main features

- The system can wake up through GPIO and specified module (COMP/LPTIM) input events
- Configurable events (from I/O, or peripherals with no state pending bits, peripherals that generate pulses)
  - ✓ Optional valid trigger edge (rising edge/falling edge)
  - ✓ Interrupt pending flag
  - ✓ Independent interrupt and event generation mask bit
  - ✓ Triggered by software
- Direct events (peripherals with associated flags and interrupt pending status bits)
  - ✓ Fixed rising edge trigger
  - ✓ No interrupt pending bit in EXTI module
  - ✓ Independent interrupt and event generation mask bit
  - ✓ No software trigger
- IO port selection

11.2.2. EXTI diagram

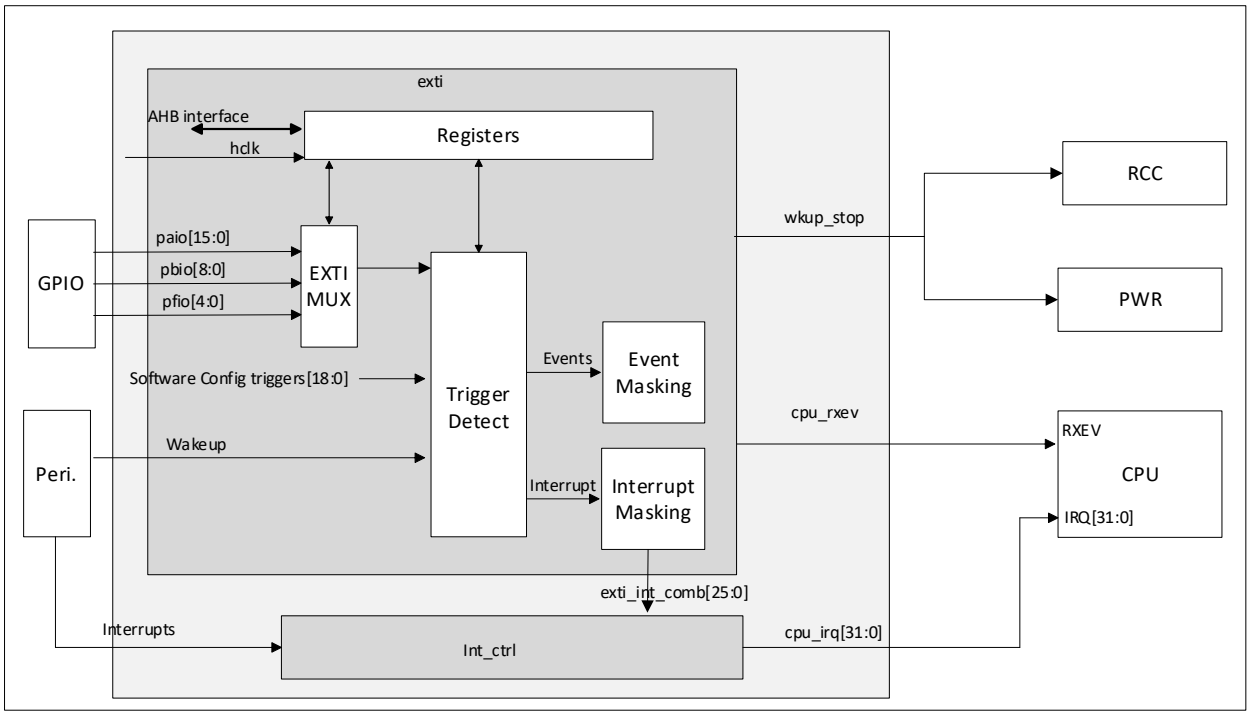


Figure 11-1 EXTI diagram

11.2.3. EXTI connection between peripherals and CPU

A peripheral that can generate a wake-up or interrupt event signal in stop mode is connected to the EXTI module.

- A wake-up signal that generates a pulse, or has no interrupt status bits inside the peripheral, is connected to the configurable line of the EXTI module. At this time, the EXTI module generates an interrupt pending bit (this bit needs to be cleared), and the EXTI interrupt will be used as the interrupt signal of the CPU.
- The interrupt and wake-up signal of the peripheral with the associated status bit (the bit is cleared in the peripheral) is connected to the wake-up trigger signal line of the EXTI module.
- All GPIO ports are input to the EXTI MUX module, and can be selected as a system wake-up signal through configurable configuration.

11.2.4. EXTI configurable event trigger wake-up

By configuring the EXTI_SWIER1 register, software can trigger the wake-up function.

There is a corresponding register configuration that triggers a rising edge or falling edge or a double edge to trigger a configurable type event. The hardware detects the input signal of the configurable type event according to the configuration, and generates a corresponding wake-up event or interrupt signal.

The CPU has dedicated interrupt mask registers and event mask registers. The event generated to the CPU after the event is enabled. The only event input signal rxev that is output to the CPU after all events to the CPU are OR'ed.

Configurable type events have a unique interrupt pending request register, which is shared with the CPU. The pending register is only set when the CPU Interrupt Mask Register (EXTI_IMR) is configured as unmasked. Each configurable type event corresponds to a CPU external interrupt signal (some will be multiplexed to the same CPU external interrupt signal). Configurable type event interrupt requires the CPU to confirm through the EXTI_PR register (write 1 to clear).

Note: When a bit of the interrupt pending register (EXTI_PR) remains valid (not cleared), the system cannot enter the low power consumption mode.

11.2.5. EXTI direct type event input wakeup

The direct type event will generate an interrupt in the EXTI module, and will generate an event signal to wake up the system and the CPU subsystem. When the CPU processes the interrupt generated by this type of trigger event, it needs to clear the interrupt status bit of the peripheral module.

11.2.6. External and internal interrupt/event line mapping

The GPIOs are connected to the 16 external interrupt/event lines in the following manner:

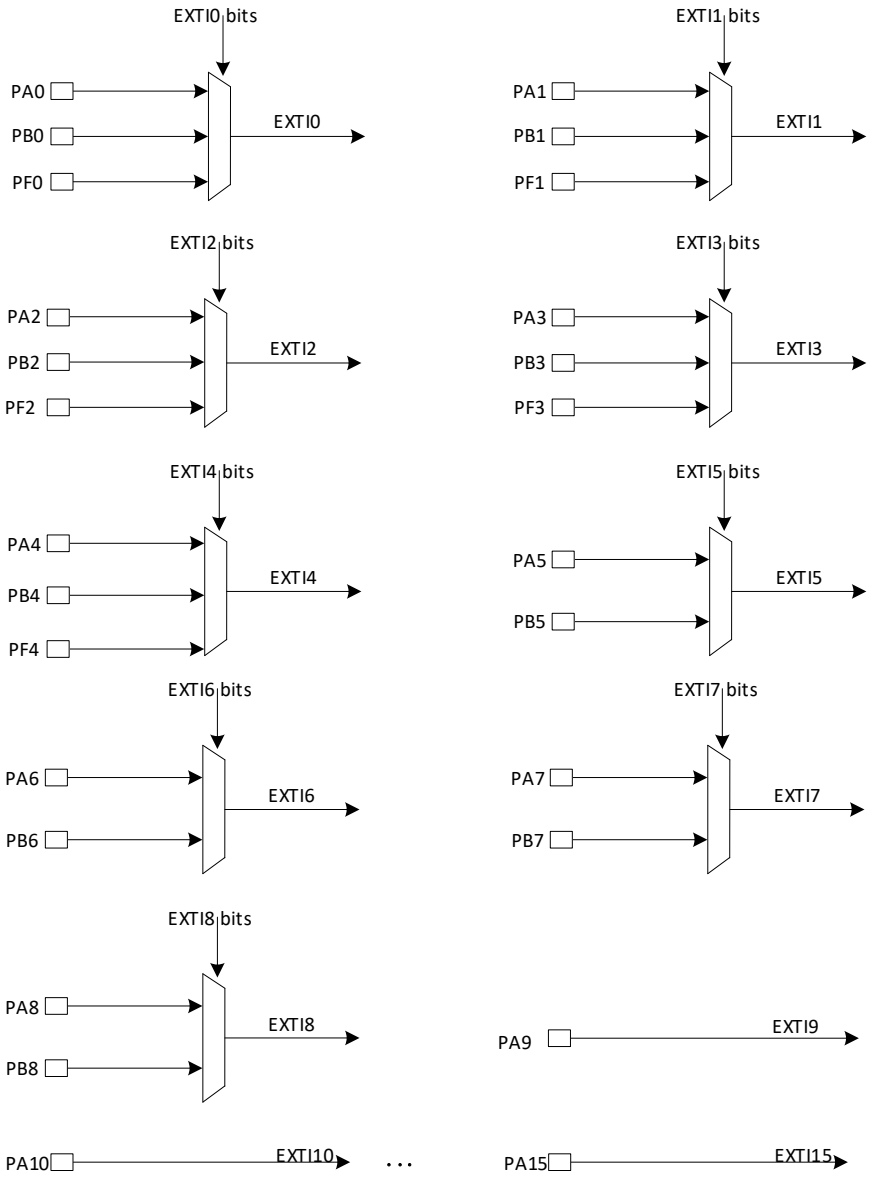


Figure 11-2 External interrupt/event GPIO mapping

The remaining lines are connected as follow:

EXTI line	Line source	Line type
Line 0-15	GPIO	configurable
Line 16	Reserved	
Line 17	COMP 1 output	Configurable
Line 18	COMP 2 output	Configurable
Line 19	Reserved	
Line 20	Reserved	

EXTI line	Line source	Line type
Line 21	Reserved	
Line 22	Reserved	
Line 23	Reserved	
Line 24	Reserved	
Line 25	Reserved	
Line 26	Reserved	
Line 27	Reserved	
Line 28	Reserved	
Line 29	LPTIM	Direct

11.3. EXTI registers

The registers of this peripheral can be accessed with word (32bit), half-word (16bit) and byte (8bit).

11.3.1. Rising trigger selection register (EXTI_RTSR)

Address offset: 0x00

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res		Res	Res	Res	Res	Res	Res	Res	Res	Res		RT18	RT17	RT16
													RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18	RT18	RW	0	Configurable type EXTI line18 rising edge trigger configuration. 0: Disable 1: enable
17	RT17	RW	0	Configurable type EXTI line17 rising edge trigger configuration. 0: Disable 1: enable
16	RT16	RW	0	Configurable type EXTI line16 rising edge trigger configuration. 0: Disable 1: enable
15	RT15	RW	0	Configurable type EXTI line15 rising edge trigger configuration. 0: Disable 1: enable
14	RT14	RW	0	Configurable type EXTI line14 rising edge trigger configuration. 0: Disable 1: enable
13	RT13	RW	0	Configurable type EXTI line13 rising edge trigger configuration. 0: Disable 1: enable
12	RT12	RW	0	Configurable type EXTI line12 rising edge trigger configuration. 0: Disable 1: enable
11	RT11	RW	0	Configurable type EXTI line11 rising edge trigger configuration. 0: Disable 1: enable
10	RT10	RW	0	Configurable type EXTI line10 rising edge trigger configuration. 0: Disable 1: enable
9	RT9	RW	0	Configurable type EXTI line9 rising edge trigger configuration. 0: Disable 1: enable
8	RT8	RW	0	Configurable type EXTI line8 rising edge trigger configuration. 0: Disable 1: enable

7	RT7	RW	0	Configurable type EXTI line7 rising edge trigger configuration. 0: Disable 1: enable
6	RT6	RW	0	Configurable type EXTI line6 rising edge trigger configuration. 0: Disable 1: enable
5	RT5	RW	0	Configurable type EXTI line5 rising edge trigger configuration. 0: Disable 1: enable
4	RT4	RW	0	Configurable type EXTI line4 rising edge trigger configuration. 0: Disable 1: enable
3	RT3	RW	0	Configurable type EXTI line3 rising edge trigger configuration. 0: Disable 1: enable
2	RT2	RW	0	Configurable type EXTI line2 rising edge trigger configuration. 0: Disable 1: enable
1	RT1	RW	0	Configurable type EXTI line1 rising edge trigger configuration. 0: Disable 1: enable
0	RT0	RW	0	Configurable type EXTI line0 rising edge trigger configuration. 0: Disable 1: enable

Configurable lines are edge-triggered, and glitches cannot be generated on these lines. If a rising edge occurs on the configurable interrupt line during a write to the EXTI_RTSR register, the associated Pending bit is not set. Both rising and falling edges can be set on the same line, in which case both edges will generate a trigger condition.

11.3.2. Falling trigger selection register (EXTI_FTSR)

Address offset: 0x04

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FT18	FT17	FT16
													RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:19	Reserved		-	
18	FT18	RW	0	Configurable type EXTI line18 falling edge trigger configuration. 0: Disable 1: enable
17	FT17	RW	0	Configurable type EXTI line17 falling edge trigger configuration. 0: Disable 1: enable
16	FT16	RW	0	Configurable type EXTI line16 falling edge trigger configuration. 0: Disable 1: enable
15	FT15	RW	0	Configurable type EXTI line15 falling edge trigger configuration. 0: Disable 1: enable
14	FT14	RW	0	Configurable type EXTI line14 falling edge trigger configuration. 0: Disable 1: enable
13	FT13	RW	0	Configurable type EXTI line13 falling edge trigger configuration. 0: Disable 1: enable
12	FT12	RW	0	Configurable type EXTI line12 falling edge trigger configuration. 0: Disable

				1: enable
11	FT11	RW	0	Configurable type EXTI line11 falling edge trigger configuration. 0: Disable 1: enable
10	FT10	RW	0	Configurable type EXTI line10 falling edge trigger configuration. 0: Disable 1: enable
9	FT9	RW	0	Configurable type EXTI line9 falling edge trigger configuration. 0: Disable 1: enable
8	FT8	RW	0	Configurable type EXTI line8 falling edge trigger configuration. 0: Disable 1: enable
7	FT7	RW	0	Configurable type EXTI line7 falling edge trigger configuration. 0: Disable 1: enable
6	FT6	RW	0	Configurable type EXTI line6 falling edge trigger configuration. 0: Disable 1: enable
5	FT5	RW	0	Configurable type EXTI line5 falling edge trigger configuration. 0: Disable 1: enable
4	FT4	RW	0	Configurable type EXTI line4 falling edge trigger configuration. 0: Disable 1: enable
3	FT3	RW	0	Configurable type EXTI line3 falling edge trigger configuration. 0: Disable 1: enable
2	FT2	RW	0	Configurable type EXTI line2 falling edge trigger configuration. 0: Disable 1: enable
1	FT1	RW	0	Configurable type EXTI line1 falling edge trigger configuration. 0: Disable 1: enable
0	FT0	RW	0	Configurable type EXTI line0 falling edge trigger configuration. 0: Disable 1: enable

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.

11.3.3. Software interrupt event register (EXTI_SWIER)

Address offset: 0x08

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SW18	SW17	SW16
													RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:19	Reserved		-	
18	SWI18	RW	0	Configurable type EXTI line18 software rising edge trigger configuration. 0: No effect

				1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
17	SWI17	RW	0	Configurable type EXTI line17 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
16	SWI16	RW	0	Configurable type EXTI line16 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
15	SWI15	RW	0	Configurable type EXTI line15 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
14	SWI14	RW	0	Configurable type EXTI line14 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
13	SWI13	RW	0	Configurable type EXTI line13 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
12	SWI12	RW	0	Configurable type EXTI line12 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
11	SWI11	RW	0	Configurable type EXTI line11 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
10	SWI10	RW	0	Configurable type EXTI line10 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
9	SWI9	RW	0	Configurable type EXTI line9 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
8	SWI8	RW	0	Configurable type EXTI line8 software rising edge trigger configuration. 0: No effect

				1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
7	SWI7	RW	0	Configurable type EXTI line7 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
6	SWI6	RW	0	Configurable type EXTI line6 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
5	SWI5	RW	0	Configurable type EXTI line5 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
4	SWI4	RW	0	Configurable type EXTI line4 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
3	SWI3	RW	0	Configurable type EXTI line3 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
2	SWI2	RW	0	Configurable type EXTI line2 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
1	SWI1	RW	0	Configurable type EXTI line1 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)
0	SWI0	RW	0	Configurable type EXTI line0 software rising edge trigger configuration. 0: No effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware, and a read returns 0 (after hardware clearing) or configuration value (before hardware clearing)

11.3.4. Pending register (EXTI_PR)

Address offset: 0x0C

Reset value: undefined

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR18	PR17	PR16
													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	Name	R/W	Reset Value	Function
31:19	Reserved	reserved	-	
18	PR18	RC_W1	0	Configurable type EXTI line18 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
17	PR17	RC_W1	0	Configurable type EXTI line17 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
16	PR16	RC_W1	0	Configurable type EXTI line16 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
15	PR15	RC_W1	0	Configurable type EXTI line15 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
14	PR14	RC_W1	0	Configurable type EXTI line14 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
13	PR13	RC_W1	0	Configurable type EXTI line13 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
12	PR12	RC_W1	0	Configurable type EXTI line12 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
11	PR11	RC_W1	0	Configurable type EXTI line11 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,

10	PR10	RC_W1	0	Configurable type EXTI line10 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
9	PR9	RC_W1	0	Configurable type EXTI line9 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
8	PR8	RC_W1	0	Configurable type EXTI line8 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
7	PR7	RC_W1	0	Configurable type EXTI line7 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
6	PR6	RC_W1	0	Configurable type EXTI line6 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
5	PR5	RC_W1	0	Configurable type EXTI line5 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
4	PR4	RC_W1	0	Configurable type EXTI line4 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
3	PR3	RC_W1	0	Configurable type EXTI line3 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
2	PR2	RC_W1	0	Configurable type EXTI line2 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,
1	PR1	RC_W1	0	Configurable type EXTI line1 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,

0	PR0	RC_W1	0	Configurable type EXTI line0 event pending flag. This bit is set when software or hardware generates a rising/falling edge trigger event. Software writes 1 to clear. 0: no event request is generated, 1: Generate rising edge/falling edge/software trigger event request,

11.3.5. External interrupt select register 1 (EXTI_EXTICR1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI3[1:0]		Res	Res	Res	Res	Res	Res	EXTI2[1:0]	
						RW	RW							RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	EXTI1[1:0]		Res	Res	Res	Res	Res	Res	EXTI0[1:0]	
						RW	RW							RW	RW

Bit	Name	R/W	Reset Value	Function
31:21	Reserved	-	-	Reserved
25:24	EXTI3[1:0]	RW	0	EXTI3 corresponds to GPIO port selection. 2'b00: PA[3] pin 2'b01: PB[3] pin 2'b10: PF[3] pin 2'b11: reserved
23:18	Reserved	-	-	Reserved
17:16	EXTI2[1:0]	RW	0	EXTI2 corresponds to GPIO port selection. 2'b00: PA[2] pin 2'b01: PB[2] pin 2'b10: PF[2] pin 2'b11: reserved
15:10	Reserved	-	-	Reserved
9:8	EXTI1[1:0]	RW	0	EXTI1 corresponds to GPIO port selection. 2'b00: PA[1] pin 2'b01: PB[1] pin 2'b10: PF[1] pin 2'b11: reserved
7:2	Reserved	-	-	Reserved
1:0	EXTI0[1:0]	RW	0	EXTI0 corresponds to GPIO port selection. 2'b00: PA[0] pin 2'b01: PB[0] pin 2'b10: PF[0] pin 2'b11: reserved

11.3.6. External interrupt select register 2 (EXTI_EXTICR2)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	EXTI7	Res	Res	Res	Res	Res	Res	Res	EXTI6
							RW								RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	RW	EXTI5	Res	Res	Res	Res	Res	Res	EXTI4[1:0]	
							RW							RW	RW

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	-	-	Reserved
24	EXTI7	RW	0	EXTI7 corresponds to GPIO port selection. 0: PA[7] pin 1: PB[7] pin
23:18	Reserved	-	-	Reserved
17:16	EXTI6	RW	0	EXTI6 corresponds to GPIO port selection. 0: PA[6] pin

				1: PB[6] pin
15:9	Reserved	-	-	Reserved
8	EXTI5	RW	0	EXTI5 corresponds to GPIO port selection. 0: PA[5] pin 1: PB[5] pin
7:2	Reserved	-	-	Reserved
1:0	EXTI4[1:0]	RW	0	EXTI4 corresponds to GPIO port selection. 2'b00: PA[4] pin 2'b01: PB[4] pin 2'b10: PF[4] pin 2'b11: reserved

11.3.7. External interrupt select register 3 (EXTI_EXTICR3)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXTI8
															RW

Bit	Name	R/W	Reset Value	Function
31:1	Reserved	-	-	Reserved
0	EXTI8	RW	0	EXTI8 corresponds to GPIO port selection. 0: PA[8] pin 1: PB[8] pin

11.3.8. Interrupt mask register (EXTI_IMR)

Address offset: 0x80

Reset value: 0x2008 0000

Note: The interrupt mask bit of the Direct type line is 1 by default, that is, the line is not masked, the mask bit of the configurable line, the default is 0, that is, the line is masked.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	IM29	Res	Res	Res	Res	Res	Res	Res	Res	Res	IM19	IM18	IM17	IM16
		RW										RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	IM29	RW	1	EXTI line29 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
28:20	Reserved			
19	IM19	RW	1	EXTI line19 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
18	IM18	RW	0	EXTI line18 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
17	IM17	RW	0	EXTI line17 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
16	IM16	RW	0	EXTI line16 is used as an interrupt to wake up the CPU mask control.

				0: interrupt wake-up mask 1: Interrupt wake-up is not masked
15	IM15	RW	0	EXTI line15 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
14	IM14	RW	0	EXTI line14 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
13	IM13	RW	0	EXTI line13 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
12	IM12	RW	0	EXTI line12 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
11	IM11	RW	0	EXTI line11 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
10	IM10	RW	0	EXTI line10 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
9	IM9	RW	0	EXTI line9 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
8	IM8	RW	0	EXTI line8 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
7	IM7	RW	0	EXTI line7 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
6	IM6	RW	0	EXTI line6 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
5	IM5	RW	0	EXTI line5 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
4	IM4	RW	0	EXTI line4 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
3	IM3	RW	0	EXTI line3 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
2	IM2	RW	0	EXTI line2 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
1	IM1	RW	0	EXTI line1 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked
0	IM0	RW	0	EXTI line0 is used as an interrupt to wake up the CPU mask control. 0: interrupt wake-up mask 1: Interrupt wake-up is not masked

11.3.9. Event mask register (EXTI_EMR)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	EM29	Res	Res	Res	Res	Res	Res	Res	Res	Res	EM19	EM18	EM17	EM16
		RW										RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29	EM29	RW	0	EXTI line29 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
28:20	Reserved			
19	EM19	RW	0	EXTI line19 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
18	EM18	RW	0	EXTI line18 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
17	EM17	RW	0	EXTI line17 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
16	EM16	RW	0	EXTI line16 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
15	EM15	RW	0	EXTI line15 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
14	EM14	RW	0	EXTI line14 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
13	EM13	RW	0	EXTI line13 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
12	EM12	RW	0	EXTI line12 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
11	EM11	RW	0	EXTI line11 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
10	EM10	RW	0	EXTI line10 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
9	EM9	RW	0	EXTI line9 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
8	EM8	RW	0	EXTI line8 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked

7	EM7	RW	0	EXTI line7 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
6	EM6	RW	0	EXTI line6 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
5	EM5	RW	0	EXTI line5 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
4	EM4	RW	0	EXTI line4 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
3	EM3	RW	0	EXTI line3 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
2	EM2	RW	0	EXTI line2 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
1	EM1	RW	0	EXTI line1 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked
0	EM0	RW	0	EXTI line0 wakes up the CPU mask control as an event. 0: Event wake-up mask 1: Event wakeup is not masked

11.3.10. EXTI register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT18	RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT18	FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI18	SWI17	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR18	PR17	PR16	PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x	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.

5C																																	
0x60	EXTI_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EXTI0[1:0]	
	Reset value							0	0							0	0							0	0							0	0
0x64	EXTI_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI5	Res.	Res.	Res.	Res.	Res.	Res.	EXTI4[1:0]	
	Reset value								0								0								0							0	0
0x68	EXTI_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI8
	Reset value																																0
0x6C-0x7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x80	EXTI_IMR	Res.	Res.	IM29	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value			1										1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	EXTI_EMR	Res.	Res.	EM29	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value			0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12. Cyclic redundancy check calculation unit (CRC)

12.1. Introduction

According to the generator polynomial, the CRC calculation unit will operate the input 32-bit data to generate a CRC result.

12.2. CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7

  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Support 32-bit data input
- A single input/output 32 data and result output share one register
- 8-bit register for general purpose (can be used as temporary storage)
- Computation time: 4 AHB clocks for 32bits data

12.3. CRC functional description

12.3.1. CRC block diagram

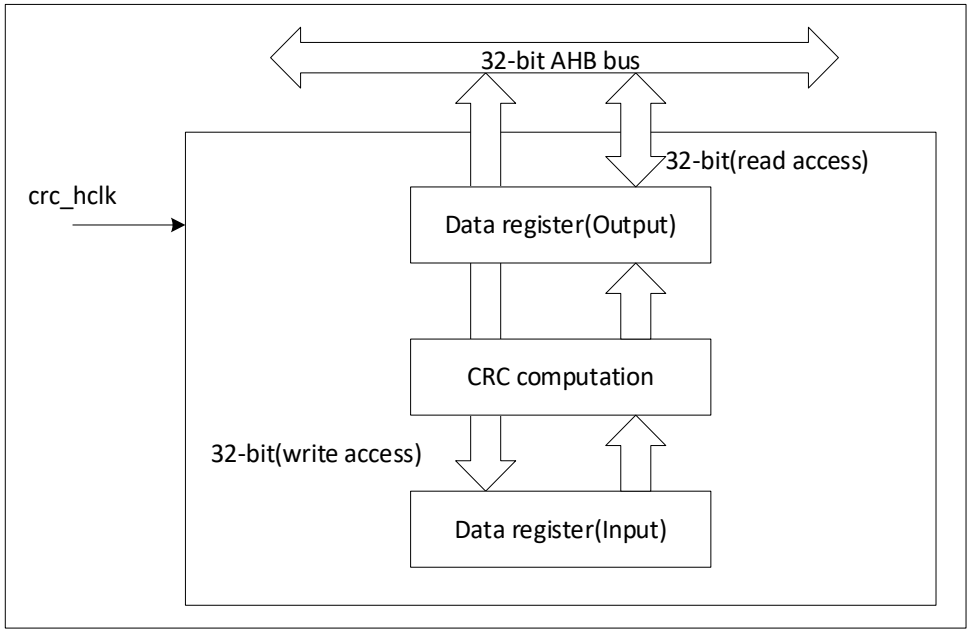


Figure 12-1 CRC calculation unit block diagram

The CRC calculation unit contains a 32-bit data register:

- When writing to this register, as an input register, new data to be calculated by CRC can be input.
- When the register is read, the result of the last CRC calculation is returned.

Each time a data register is written, the result of the calculation is the combination of the previous CRC calculation and the new calculation (CRC is calculated on the entire 32-bit word, not byte by byte).

While the CRC is being calculated, writes are blocked until the end of the CRC calculation.

The register CRC_DR can be reset to 0xFFFF FFFF by setting the RESET bit of the register CRC_CR. This operation does not affect the data in register CRC_IDR.

12.4. CRC registers

12.4.1. Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bit	Name	R/W	Reset Value	Function
31:0	DR	RW	32'hFFFFFFFF	data register. When writing new data, it is used as an input register. When read, the previous CRC calculation result is retained.

12.4.2. Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	Reserved		-	
7:0	IDR[7:0]	RW	0	General purpose 8bit data register These bits are used as temporary storage for one byte. This register is not reset by the RESET bit of the CRC_CR register.

12.4.3. Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	Reserved		-	
0	RESET		0	This bit is set by software to reset the CRC calculation unit. This bit can only be set and is automatically cleared by hardware.

12.4.4. CRC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	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
0x004	CRC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x008	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
	Reset value																																0

13. Analog-to-digital converter (ADC)

13.1. Introduction

The chip has a 12-bit SARADC (successive approximation analog-to-digital converter). The module has a total of 12 channels to be measured, including 10 external channels and 2 internal channels.

The conversion mode of each channel can be set to single, continuous, sweep, discontinuous mode.Conversion results are stored in left- or right-aligned 16-bit data registers.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

13.2. ADC main features

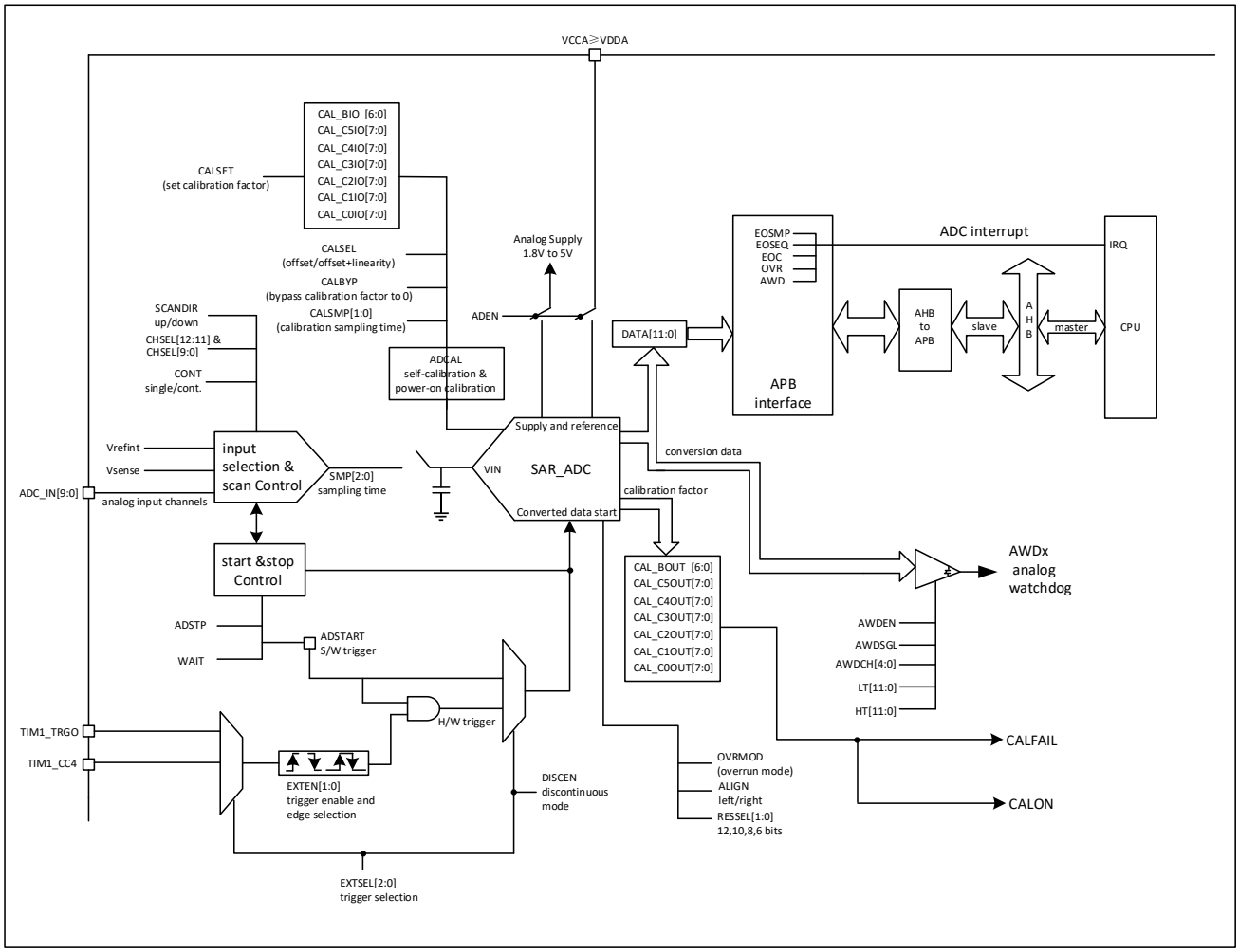
- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 1.0 μs for 12-bit resolution (1 MHz)
  - Self-calibration
  - Programmable sampling time
  - Programmable data alignment mode
- Low-power
  - Application can reduce PCLK frequency for low-power operation while still keeping optimum ADC performance.
  - Wait mode: prevents ADC overrun in applications with low frequency PCLK
- Analog input channels
  - 10 external analog inputs:PA[7:0] and PB[1:0]
  - 1 channel for internal temperature sensor (VSENSE)
  - 1 channel for internal reference voltage (VREFINT)
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (internal timer events from TIM1 and GPIO)
- Conversion modes
  - Single mode: Can convert a single channel or can scan a sequence of channels
  - Continuous mode: Continuous mode converts selected inputs continuously
  - Discontinuous mode: Convert selected channel once per trigger
- Interrupt generation
  - At the end of sampling
  - At the end of conversion
  - At the end of sequence conversion

- In case of analog watchdog
- Overrun events

■ Analog watchdog

13.3. ADC functional description

13.3.1. ADC diagram



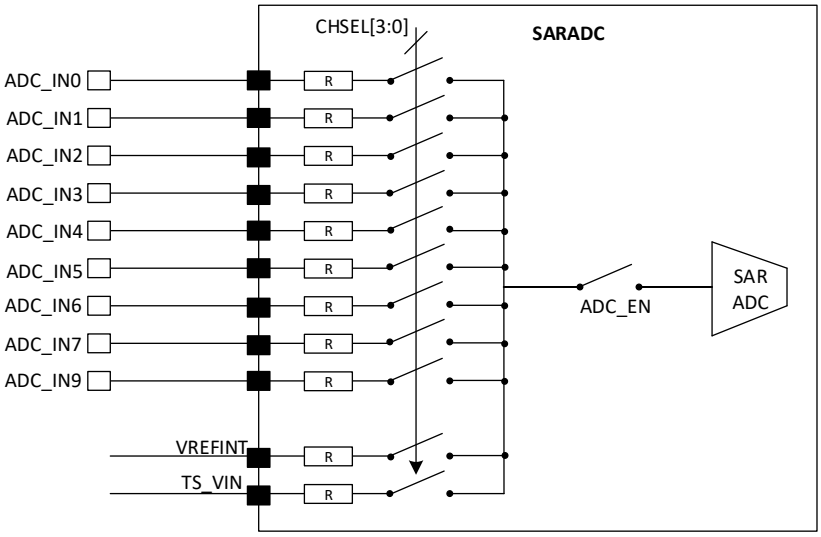


Figure 13-1 ADC channel with analog switch

13.3.2. Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration operations include power-on calibration and software calibration.

ADC power-on calibration

The hardware will automatically perform ADC calibration after power-up.

ADC software calibration

The software can set ADCAL = 1 to start the calibration. The calibration can only be started when the ADC is not enabled (ADEN = 0), and only the system clock can be selected as the ADC clock. ADCAL is cleared by hardware when calibration is complete.

When the working conditions of the ADC change (the change in VCC is the main factor for the offset of the ADC, followed by the change in temperature), it is recommended to perform a re-calibration operation.

Calibration software procedure:

- Ensure that ADEN = 0、CKMODE selects the system clock
- Set ADCAL = 1
- Wait until ADCAL = 0

13.3.3. ADC on-off control (ADEN)

At MCU power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

The following is the process to enable ADC:

1. Write 1 to clear the ADRDY bit in the ADC_ISR register
2. The ADEN bit of the ADC_CR register is set to 1

ADC conversions are also initiated by setting ADSRART or (if triggered) by an external trigger event.

The following is the procedure for disabling the ADC:

Check that ADSTART in the ADC_CR register is 0 to ensure the ADC is not in the process of converting. If necessary, set ADSTP in the ADC_CR register to 1 to stop the ongoing ADC conversion, and wait for ADSTP to be cleared by hardware (cleared to 0 means the conversion is stopped).

Warning: ADEN bit cannot be set to 1 during 4 ADC clocks after ADCAL is cleared by hardware and ADCAL = 1.

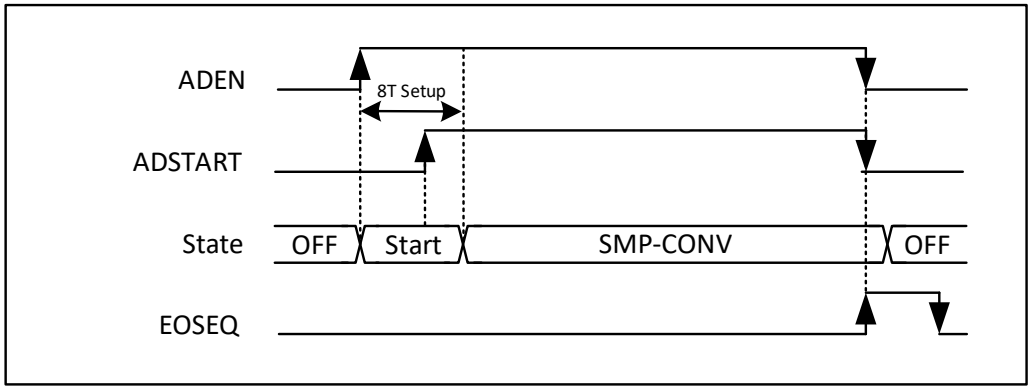


Figure 13-2 Enabling/disabling the ADC

13.3.4. ADC click

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC_CLK) independent from the APB clock (PCLK). ADC_CLK can be generated by two possible clock sources.

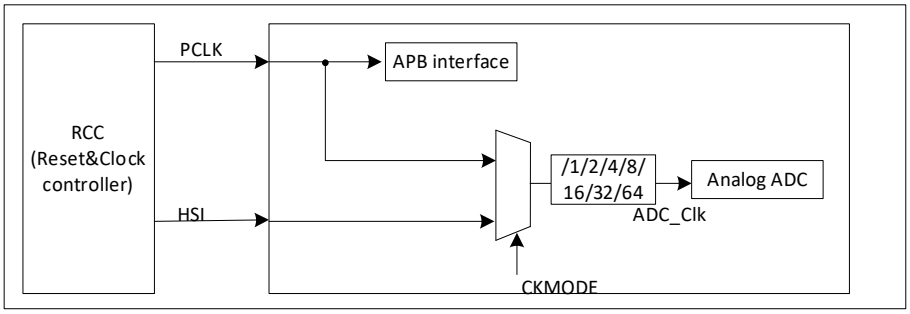


Figure 13-3 ADC clock scheme

Table 13-1 Delay between trigger and conversion start

ADC clock source	CKMODE[3:0]	Frequency division factor	Latency between the trigger event and the start of conversion (T is the clock period)
PCLK	0000	1	0
	0001	2	0
	0010	4	0
	0011	8	0
	0100	16	0
	0101	32	0
	0110	64	0
	0111	/	/
HSI	1000	1	0
	1001	2	0
	1010	4	0
	1011	8	0
	1100	16	0
	1101	32	0
	1110	64	0
	1111	/	/

13.3.5. Configuring the ADC

Software must write to the ADCAL and ADEN bits in the ADC_CR register if the ADC is disabled (ADEN must be 0).

Software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_CFGRi, ADC_SMPR, ADC_TR, ADC_CHSELR and ADC_CCR registers, software must only write to the configuration control bits if the ADC is enabled (ADEN = 1) and if there is no conversion ongoing (ADSTART = 0).

Software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

13.3.6. Channel selection (CHSEL, SCANDIR)

There are up to 12 multiplexed channels:

● 10 analog inputs from GPIO pins (ADC_IN0...ADC_IN9)

● 2 internal analog inputs (Temperature Sensor, Internal Reference Voltage)

It is possible to convert a single channel or to automatically scan a sequence of channels.

The sequence of the channels to be converted must be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSEL0...CHSEL11).

The order in which the channels will be scanned can be configured by programming the bit SCANDIR bit in the ADC_CFGR1 register:

● SCANDIR = 0: foRWard scan Channel 0 to Channel 11

● SCANDIR = 1: backward scan Channel 11 to Channel 0

Temperature sensor, VREFINT internal channels

The temperature sensor is connected to channel ADC_IN10. The internal voltage reference VREFINT is connected to channel ADC_IN11.

13.3.7. Programmable sampling time (SMP)

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows to trim the conversion speed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP[2:0] bits in the ADC_SMPR register.

This programmable sampling time is common to all channels. If required by the application, the software can change and adapt this sampling time between each conversions.

The total conversion time is calculated as follows:

$$t_{CONV}$$ = Sampling time + (Convert resolution +0.5) x ADC clock cycles

Example:

When ADC_CLK = 16MHz, the resolution is 12 bits, and the sampling time is 3.5 ADC clock cycles:

t_{CONV} = (3.5 + 12.5) x ADC clock period = 16 x ADC clock period = 1 μs

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

13.3.8. Single conversion mode (CONT = 0, DISCEN = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT = 0, DISCEN = 0 in the ADC_CFGR1 register.

ADC conversions can be initiated in two ways:

● Set ADSTART bit in ADC_CR register

● Hardware trigger events

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

The ADC cannot be in discontinuous conversion mode and continuous conversion mode at the same time, in this case (DISCEN = 1, CONT = 1), it behaves as single conversion mode.

13.3.9. Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT = 1 in the ADC_CFGR1 register.

Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR registe
- 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 EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE 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 bits DISCEN = 1 and CONT = 1.

13.3.10. Discontinuous conversion mode (DISCEN = 1)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to initiate each conversion defined in a sequence.

Conversely, when DISCEN = 0, a hardware or software trigger event can initiate all conversions defined in a sequence.

For example:

DISCEN = 1, the channels to be converted are: 0, 3, 7, 10

- 1st trigger: Channel 0 is converted and an EOC event occurs
- 2nd trigger: Channel 3 is converted and an EOC event occurs
- 3rd trigger: Channel 7 is converted and an EOC event occurs
- 4th trigger: Channel 10 is converted and EOC and EOSEQ events are generated
- 5th trigger: Channel 0 is converted and an EOC event occurs
- 6th trigger: Channel 3 is converted and an EOC event occurs
- ...

DISCEN = 0, the channels to be converted are: 0, 3, 7, 10

- 1st Trigger: The entire complete sequence of conversions, in turn, channels 0, 3, 7, and 10.

Each conversion is completed, an EOC event is generated, and the conversion to the last channel generates an EOSEQ event in addition to the EOC.

- Any trigger event restarts the complete sequence conversion.

Note: It is impossible to have the ADC in continuous mode and continuous conversion mode at the same time, in this case (DISCEN = 1, CONT = 1), it behaves as a single conversion mode.

13.3.11. Starting conversions (ADSTART)

Software starts ADC conversion with setting ADSTART = 1.

When ADSTART is set, the conversion:

- When EXTEN = 0x0 (software trigger), start immediately
- When if EXTEN ≠ 0x0, start at the next selected hardware trigger valid edge

The ADSTART bit is also used to indicate whether an ADC conversion operation is currently in progress. When ADSTART = 0, the ADC can be reconfigured, indicating that the ADC is idle at this time.

ADSTART bit can be cleared by hardware.

- One-shot conversion mode is triggered by software (CONT = 0, EXTSEL = 0x0)

- After sequence conversion is complete (EOSEQ = 1)

- Discontinuous conversion mode is triggered by software (CONT = 0, DISCEN = 1, EXTSEL = 0x0)

- After conversion is complete (EOC = 1)

- In all cases (CONT = X, EXTSEL = X)

- After the software calls and executes the ADSTP procedure

Note: In continuous mode (CONT = 1), the ADSTART bit cannot be cleared by hardware caused by EOSEQ because it automatically restarts the sequence conversion. When the hardware trigger is selected as single conversion mode (CONT = 0 and EXTSEL = 0x01), ADSTART will not be cleared by hardware after the EOSEQ

flag is set. This avoids the need for software to reset the ADSTART bit and ensures that no hardware trigger event is missed.

13.3.12. Timings

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:

$t_{ADC} = t_{SMPL} + t_{SAR} = [\ 3.5_{|min} + 12.5_{|12bit}] * t_{ADC\_CLK}$

$t_{ADC} = t_{SMPL} + t_{SAR} =\ 218.75ns_{|min} + 781.25\ ns_{|12bit} = 1\ \mu s_{|min}$ (for f_{ADC_CLK} = 16 MHz)

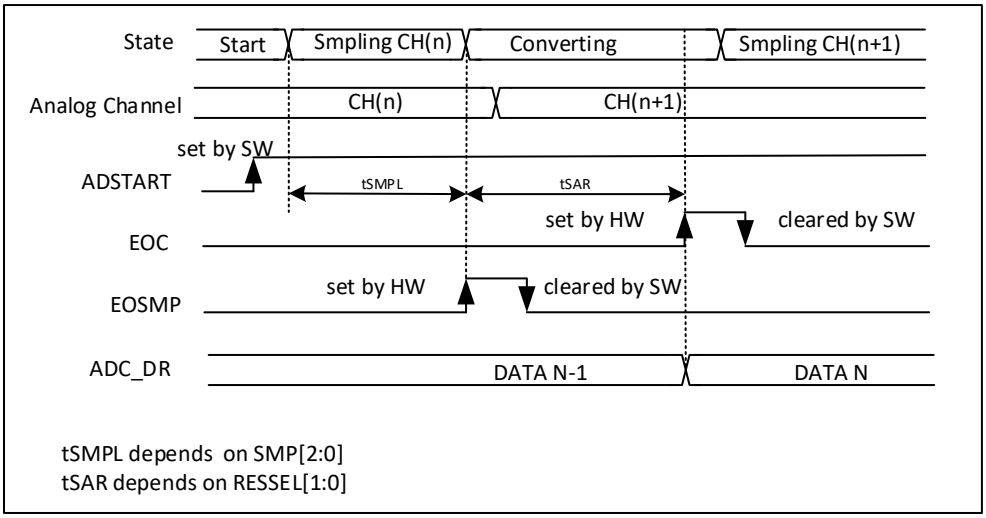


Figure 13-4 analog to digital conversion timing

13.3.13. Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP = 1 in the ADC_CR register.

This will reset the ADC operation and the ADC will be idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would restart a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware.

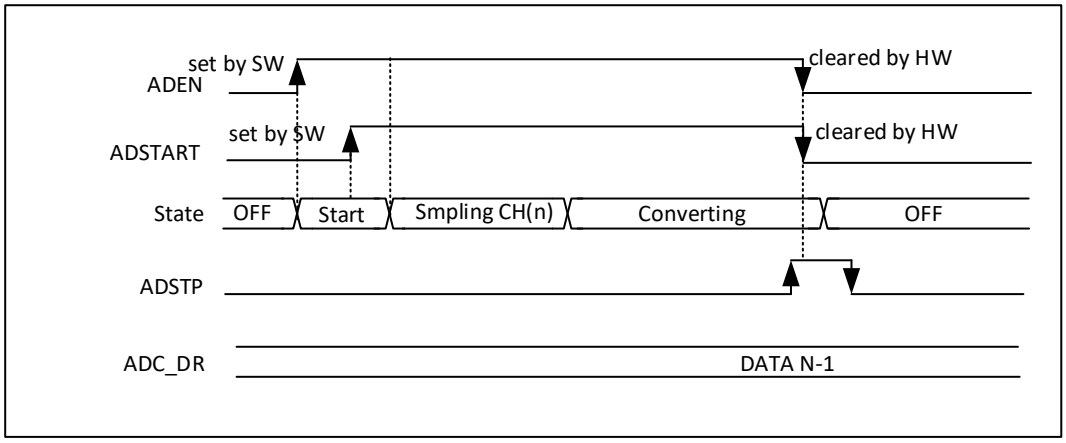


Figure 13-5 Stop timing

13.4. Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART = 0, any hardware triggers which occur are ignored.

Source	EXTEN[1:0]
Trigger detection disabled	00
Detect on rising edge	01
Detect on falling edge	10
Detects on rising and falling edges	11

Note: The polarity of the external trigger can be changed only when the ADC is not converting (ASTART = 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

The following table shows possible external triggers for rule transitions. A software source trigger event can be generated by setting the ADSTART bit in the ADC_CR register.

Table 13-2 External trigers

Name	Source	EXTSEL[2:0]
EXT0	TIM1_TRGO	000
EXT1	TIM1_CC4	001
EXT2	Reserved	010
EXT3	Reserved	011
EXT4	Reserved	100
EXT5	Reserved	101
EXT6	Reserved	110

Note: The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).

13.4.1. Programmable resolution (RES) - fast conversion mode

It is possible to obtain faster conversion times (tSAR) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Lower resolution mode reduces the conversion time of successive approximation as shown in the table below:

RESSEL [1:0]	t_{SAR} (ADC clock cycles)	t_{SAR}(ns) @ fADC = 24MHz	t_{SMP} (ADC clock cycles)	t_{ADC}(t_{SMP} = 3.5) (ADC clock cycles)	t_{CONV}(ns) @ f_{ADC} = 24MHz
12	12.5	521ns	3.5	16	667ns
10	10.5	438ns	3.5	14	583ns
8	8.5	396ns	3.5	12	500ns
6	6.5	271ns	3.5	10	417ns

13.4.2. End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

13.4.3. End of conversion sequence (EOSEQ flag)

The ADC notifies the application of each end of sequence (EOSEQ) event.

The ADC sets the EOSEQ flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSEQIE bit is set in the ADC_IER register. The EOSEQ flag is cleared by software by writing 1.

13.4.4. Example timing diagrams

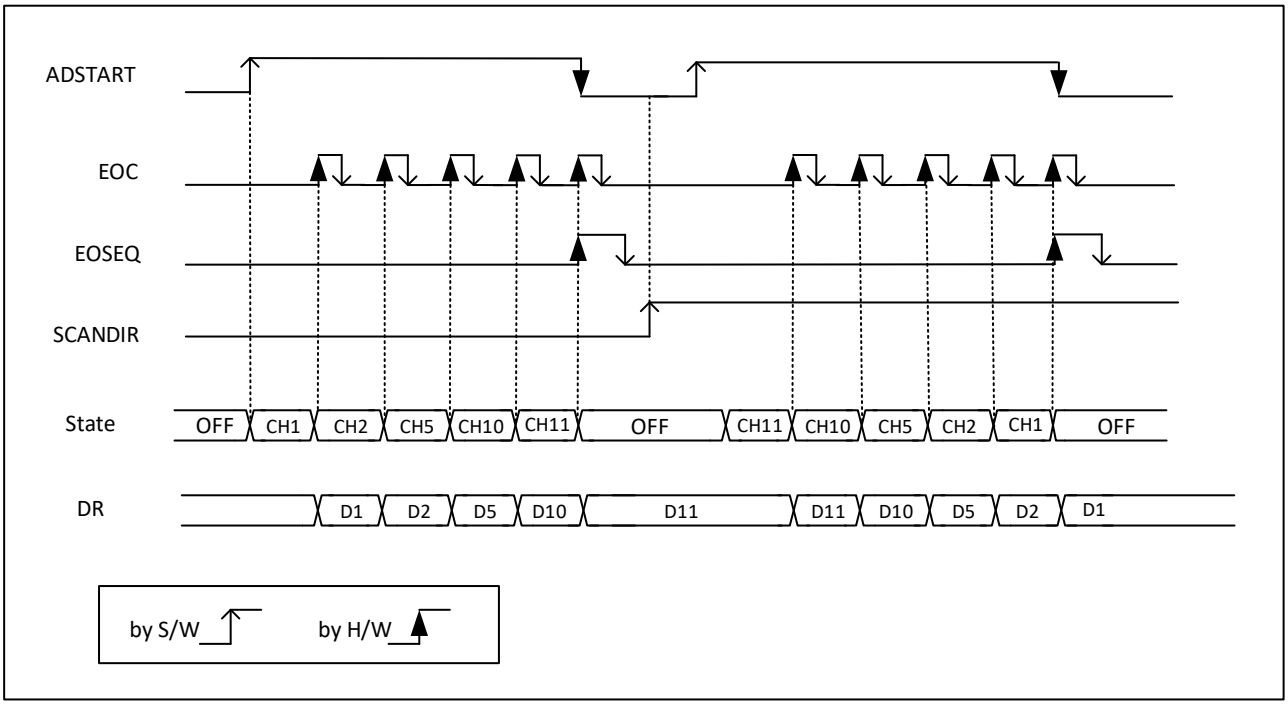


Figure 13-6 Single conversions of a sequence, software trigger

- EXTEN = 0x0, CONT = 0
- CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

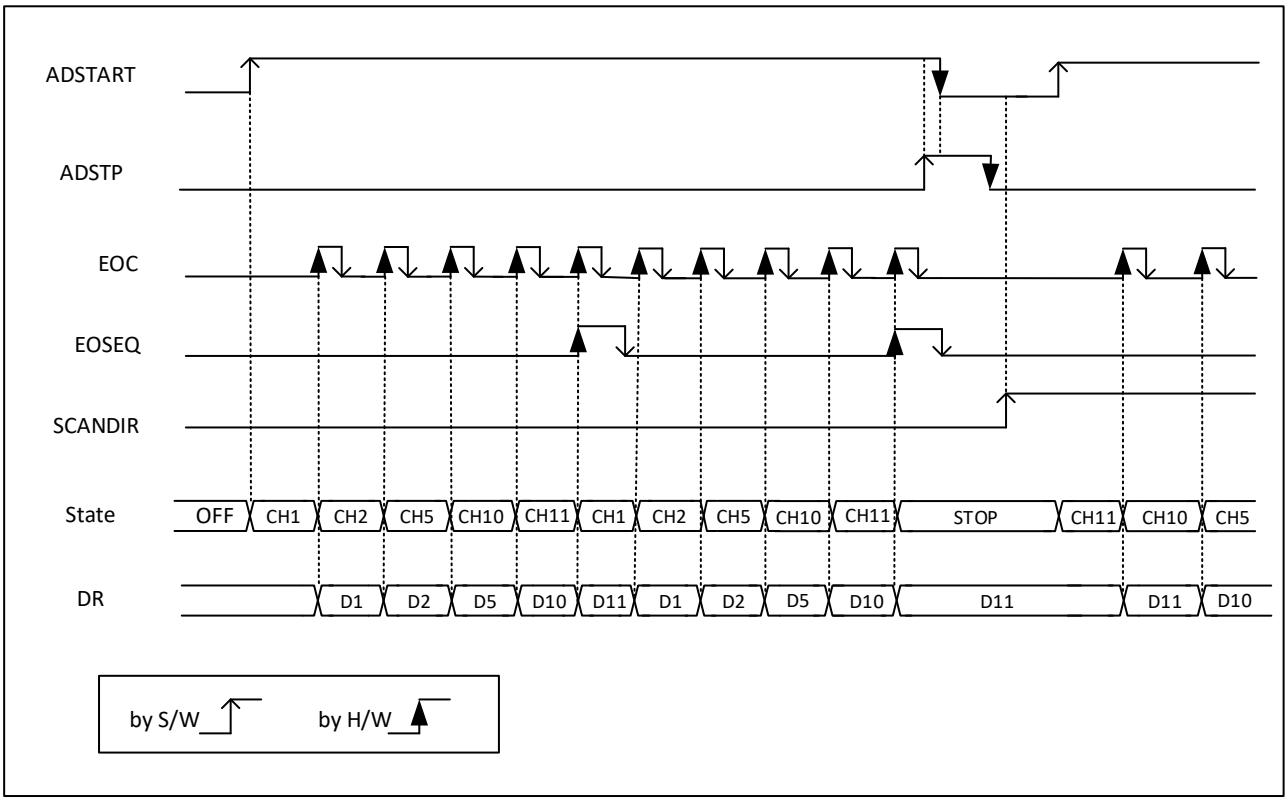


Figure 13-7 Continuous conversion of a sequence, software trigger

- EXTEN = 0x0, CONT = 1
- CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

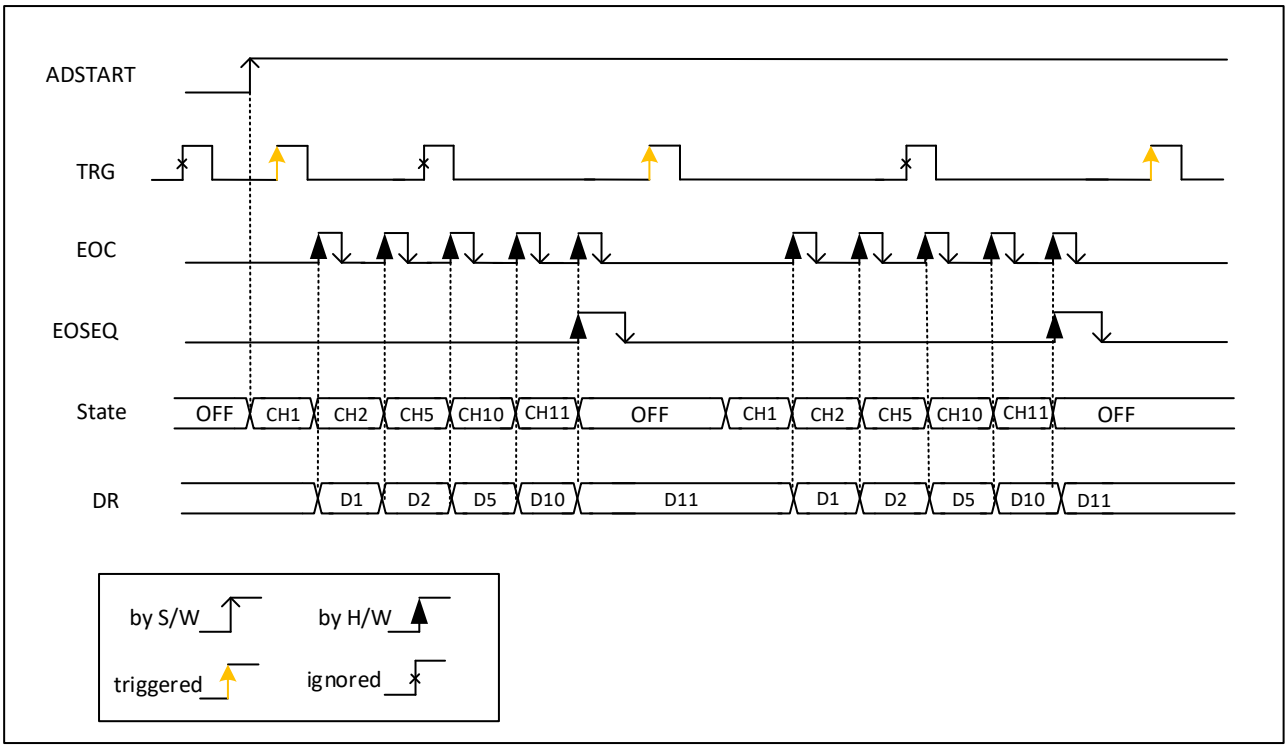


Figure 13-8 Single conversions of a sequence, hardware trigger

- EXTSEL = TRGx, EXTEN = 0x1 (rising edge), CONT = 0
- CHSEL = 0xF, SCANDIR = 0, AUTDLY = 0, AUTOFF = 0

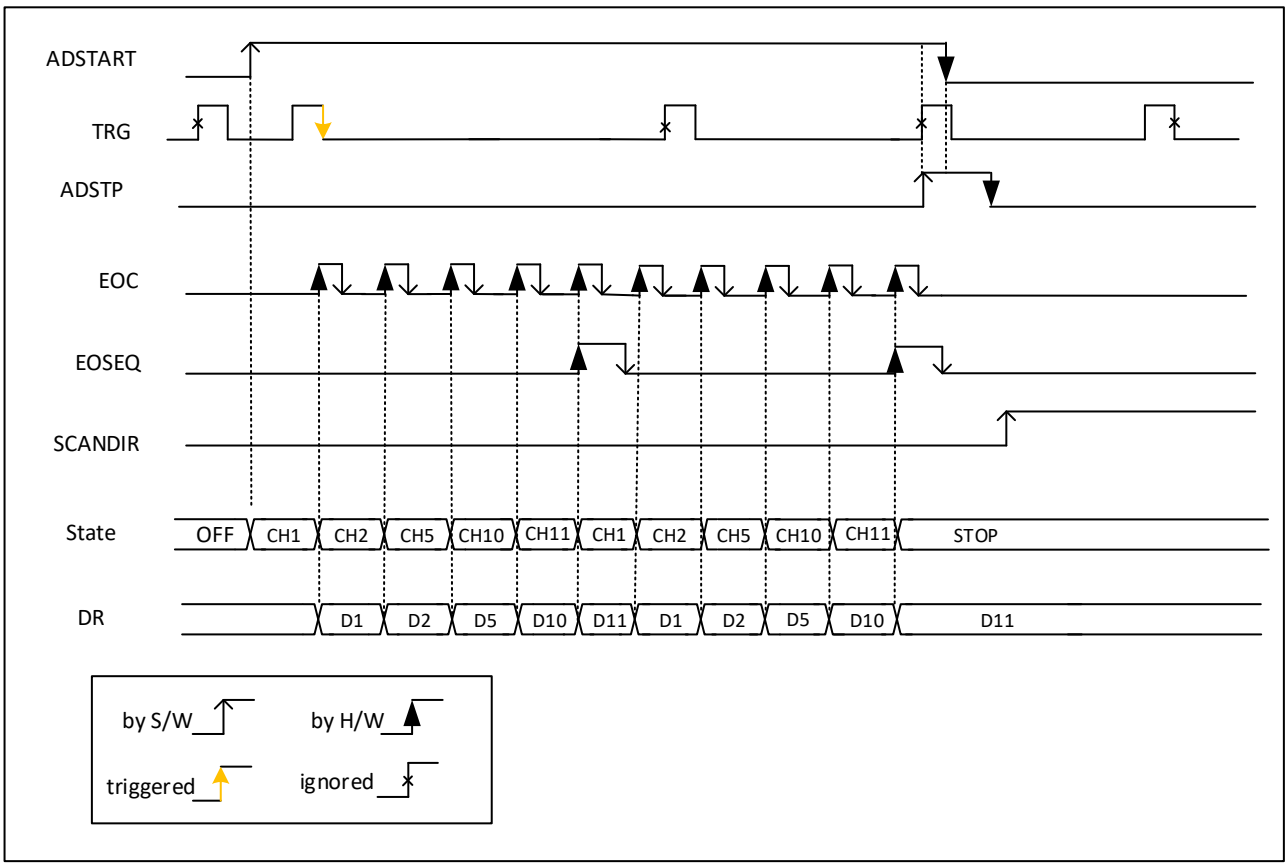


Figure 13-9 Continuous conversion of a sequence, software trigger

- EXTSEL = TRGx, EXTEN = 0x2 (falling edge), CONT = 1
- CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

13.5. Data management

13.5.1. Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution. The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) .

ALIGN	RESSEL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0X0	0X0				DATA[11:0]											
	0X1	0X0				DATA[9:0]										0X0	
	0X2	0X0				DATA[7:0]								0x0			
	0X3	0X0				DATA[6:0]						0X0					
1	0X0	DATA[11:0]												0X0			
	0X1	DATA[9:0]										0X0		0X0			
	0X2	DATA[7:0]								0x0				0X0			
	0X3	DATA[6:0]						0X0						0X0			

13.5.2. ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU, before the data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or oveRWritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- OVRMOD = 0

– An overrun event preserves the data register from being oveRWritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is 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, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.

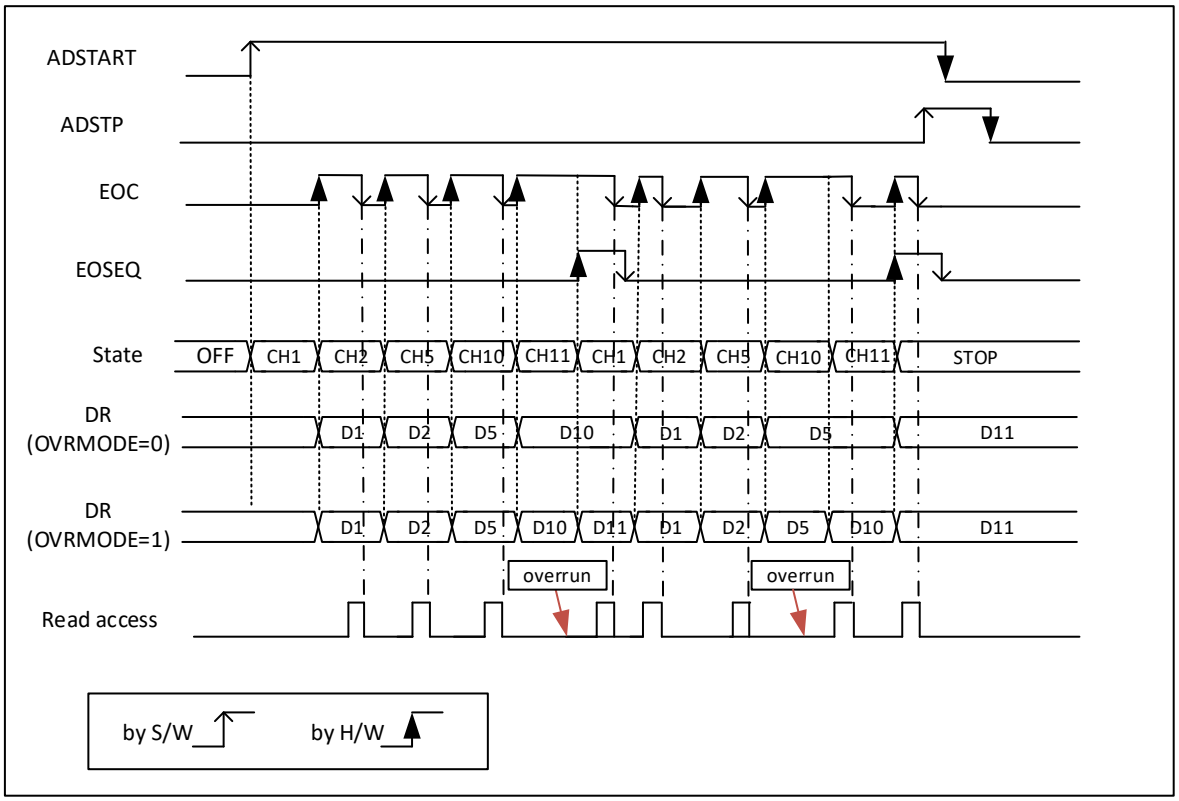


Figure 13-10 Overurn

13.5.3. Managing a sequence of data converted

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register should be configured to 0 to manage overrun events as an error.

13.5.4. Managing converted data

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD = 1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

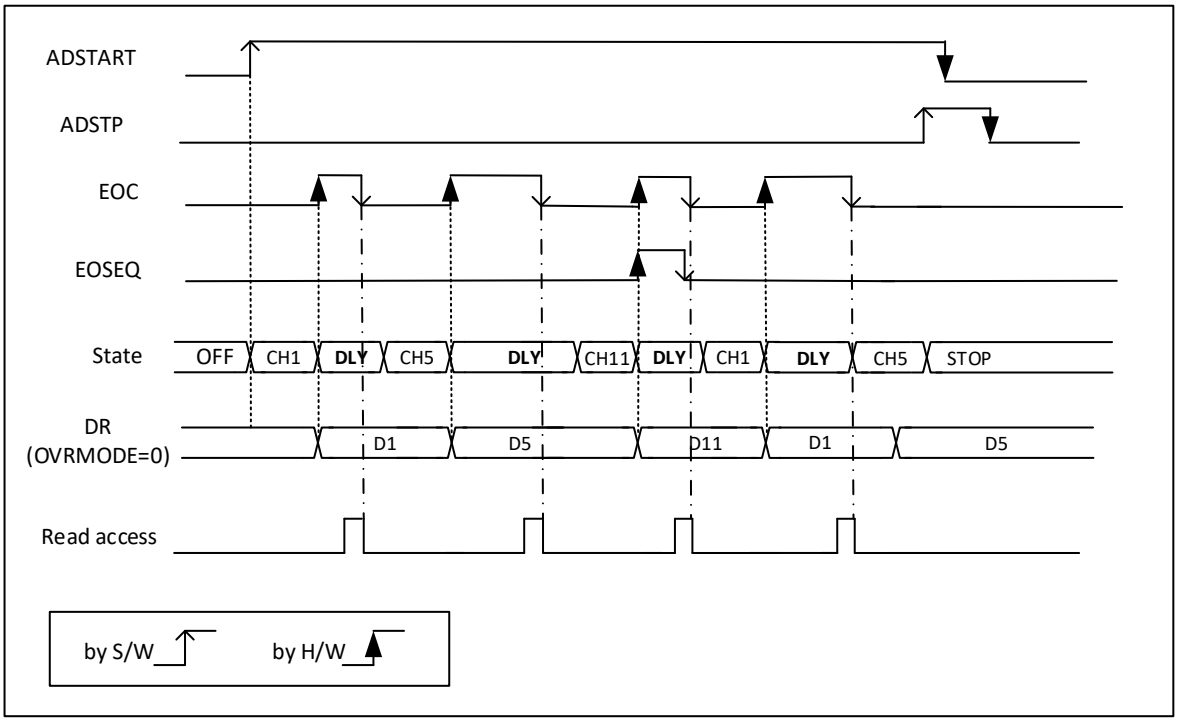
13.6. Low-power features

13.6.1. Wait mode conversion

Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set to 1 in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared. This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.



Figre 13-11 Wait mode conversion

- EXTEN = 0x0, CONT = 1
- CHSEL = 0x3, SCANDIR = 0, AUTDLY = 1, AUTOFF = 0

13.7. Analog window watchdog

The AWD analog watchdog feature is enabled by setting the AWDEN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels remain within a configured voltage range (window).

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by setting the AWDIE bit in the ADC_IER register. The AWD flag is cleared by software by writing 1 to it. When converting a data with a resolution of less than 12-bit (according to bits DRES[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 13-3 Analog watchdog comparison

Resolution bits RES[1:0]	Analog Watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:0]	LT[11:0] and HT[11:0]	-
01: 10-bit	DATA[11:2],00	LT[11:0] and HT[11:0]	The user must configure LT[1:0] and HT[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT[11:0] and HT[11:0]	The user must configure LT[3:0] and HT[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT[11:0] and HT[11:0]	The user must configure LT[5:0] and HT[5:0] to 00000

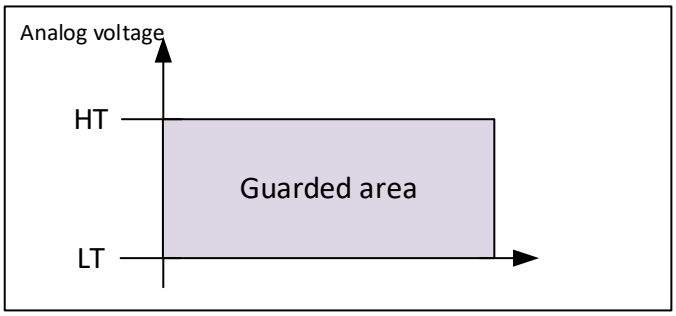


Figure 13-12 Analog watchdog guarded area

Table 13-4 Analog watchdog channel selection

Channels guarded by the analog watchdog	AWDSGL bit	AWDEN bit
None	x	0
All channels	0	1
Single channel	1	1

13.7.1. ADC_AWD_OUT signal output generation

The analog watchdog is associated with an internal hardware signal, ADC_AWD_OUT is directly connected to the ETR input (external trigger) of the on-chip timer TIM1.

When the analog watchdog is enabled, ADC_AWD_OUT is activated:

- When the conversion of the channel selected by AWDCH exceeds the programmed threshold, ADC_AWD_OUT will be set.
- After the conversion of the next channel selected by AWDCH, ADC_AWD_OUT is reset within the programmed threshold. It will remain at 1 if the next protected transition still exceeds the programmed threshold.
- ADC_AWD_OUT is also reset when ADC is disabled. Note that stopping conversion (ADSTP set to 1) may clear the ADC_AWDx_OUT state.
- Channels not selected as analog watchdog do not affect ADC_AWD_OUT status bits.

The AWD flag is set by hardware and reset by software: the AWD flag has no effect on the generation of ADC_AWD_OUT (eg, if the flag is not cleared by software, ADC_AWDx_OUT can toggle while the AWDx flag remains at 1).

The ADC_AWD_OUT signal is generated by the PCLK domain.

AWD comparison is performed at the end of each ADC conversion.

13.8. Temperature sensor and internal reference voltage

A temperature sensor can be used to measure the junction temperature (TJ) of the device.

The temperature sensor is internally connected to the ADC input channel, which can be used to convert the sensor's voltage value to a numerical value. The sampling time of the temperature sensor must be greater than the minimum value of Ts_temp given in the datasheet. When the temperature sensor is not in use, the sensor can be placed in a power-down mode.

The output voltage of the temperature sensor varies linearly with temperature, but each chip has subtle differences related to process variables. In order to improve this accuracy, the calibration value of each chip will be individually given by the product test and saved in the system storage area.

The internal voltage reference (VREFINT) provides a regulated voltage output to the ADC and comparator.

Note: The TSVREF bit must be set to activate two internal channels: temperature sensor, VREFINT.

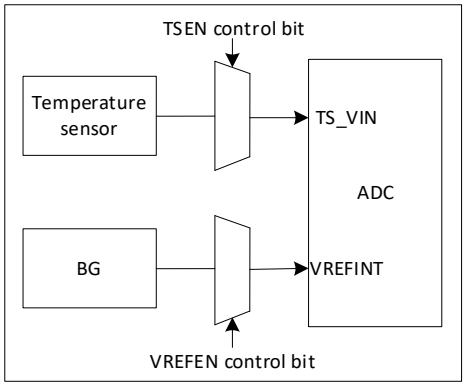


Figure 13-13 TS and VREFINT channel

How to use the temperature sensor to read the temperature:

1. Select ADC1_IN11 input channel
2. Select an appropriate sampling time according to the device specification
3. Set the TSEN bit in the ADC_CCR register to wake up the temperature sensor from power down mode
4. Start ADC conversion with ADSTART bit set in ADC_CR register (external trigger is also available)
5. Read VSENSE conversion data from ADC_DR register
6. Count the temperature using the following formula:

$$Temperature(in\ ℃) = \frac{85℃ - 30℃}{TS_{CAL2} - TS_{CAL1}} \times (TS_{DATA} - TS_{CAL1}) + 30℃$$

TSCAL2 represents the calibration value of the 85°C temperature sensor, the calibration value storage Address offset: 0x1FFF 0F18

TSCAL1 represents the calibration value of the 30°C temperature sensor, the calibration value storage Address offset: 0x1FFF 0F14

TSDATA is the actual output value converted by the ADC

Note: When the sensor wakes up from power-down mode, it needs a start-up time to correctly output VSENSE, and the ADC also has a start-up time after power-on. To reduce this delay, you need to set the ADEN and TSEN bits at the same time.

Calculating the actual Vcc voltage using the internal reference voltage

$$VREFINT = 1.2V = \frac{ADC\_DATAx}{4095} \times VCC$$

Calculating the Vchannnel voltage using the the actual Vcc

$$VCHANNEL = \frac{ADC\_DATAx}{4095} \times VCC$$

VREFINT is fixed at 1.2V,

VCHANNEL is the channel voltage,

ADC_DATA is the conversion data in ADC_DR,

4096 is represented as 12 bits.

13.9. ADC interrputs

ADC interrupts can be generated by any of the following events:

- End of any conversion (EOC flag)
- End of sequence conversion (EOS flag)
- When analog watchdog detection occurs (AWD flag)
- Occurs when the sampling phase ends (EOSMP flag)
- When data overshoot occurs (OVR flag)

Separate interrupt enable bit for flexible setting of ADC interrupts

Table 13-5 ADC interrput

Interrupt event	Event flag	Enable control bit
End of conversion	EOC	EOCIE
End of sequence of conversions	EOS	EOSIE
Analog watchdog status bit is set	AWD	AWDIE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

13.10. ADC registers

13.10.1. ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	AWD	Res	Res	OVR	EOSEQ	EOC	EOSMP	Res
								rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			
7	AWD	RC_W1	0	Analog watchdog flag This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software writing 1 to it. 0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software) 1: Analog watchdog event occurred
6:5	Reserved			
4	OVR	RC_W1	0	ADC overrun This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete 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
3	EOSEQ	RC_W1	0	End of sequence flag This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it. 0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software) 1: Conversion sequence complete
2	EOC	RC_W1	0	End of conversion flag This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register. 0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software) 1: Channel conversion complete
1	EOSMP	RC_W1	0	End of sampling flag This bit is set by hardware during the conversion, at the end of the sampling phase.It is cleared by software by programming it to ‘1’. 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
0	Reserved			

13.10.2. ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Re.s	AWDIE	Res	Res	OVRIE	EOSEQIE	EOCIE	EOSMPIE	Res
								RW			RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			
7	AWDIE	RW	0	Analog watchdog interrupt enable This bit is set and cleared by software to enable/disable the analog watchdog interrupt. 0: Analog watchdog interrupt disabled 1: Analog watchdog interrupt enabled
6:5	Reserved			
4	OVRIE	RW	0	Overrun interrupt enable

				This bit is set and cleared by software to enable/disable the overrun interrupt. 0: Overrun interrupt disabled 1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.
3	EOSEQIE	RW	0	End of conversion sequence interrupt enable This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt. 0: EOSEQ interrupt disabled 1: EOSEQ interrupt enabled. An interrupt is generated when the EOSEQ bit is set.
2	EOCIE	RW	0	End of conversion interrupt enable This bit is set and cleared by software to enable/disable the end of conversion interrupt. 0: EOC interrupt disabled 1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
1	EOSMPIE	RW	0	End of sampling flag interrupt enable This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt. 0: EOSMP interrupt disabled. 1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.
0	Reserved			

Description: Software can write these bits when ADSTART = 0 (to ensure that no conversion is in progress)

13.10.3. ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
RS															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ADSTP	Res	ADSTART	Res	ADEN
											RS		RS		RS

Bit	Name	R/W	Reset Value	Function
31	ADCAL	RS	0	ADC calibration This bit is set by software to start the calibration of the ADC. 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 is in progress.
30:5	Reserved			
4	ADSTP	RS	0	ADC stop conversion command This bit is set by software to stop and discard an ongoing conversion (ADSTP Command). It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command. 0: No ADC stop conversion command ongoing 1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.
3	Reserved			

2	ADSTART	RS	0	ADC start conversion command This bit is set by software to start ADC conversion. Depending on the EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration). It is cleared by hardware: – In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion Sequence (EOSEQ) flag. – In discontinuous conversion mode(CONT = 0, DISCEN = 1), when the software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion (EOC) flag. – In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware. Note: Software is allowed to set ADSTART only when ADEN = 1 (ADC is enabled and there is no pending request to disable the ADC)
1	Reserved			
0	ADEN	RS	0	ADC enable command Software setting this bit enables the ADC and the ADC will be ready to operate. 0: ADC disabled (OFF state) 1: enable ADC

13.10.4. ADC configuration register 1 (ADC_CFGR1)

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	AWDCH				Res	Res	AWDEN	AWDSGL	Res	Res	Re.	Res	Res	DISCEN
		RW	RW	RW	RW			RW	RW						RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	WAIT	CONT	OVRMOD	Res	Res	Res	EXTSEL			ALIGN	RES_SEL		SCADIR	Res	Res
	RW	RW	RW				RW	RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	Reserved			
29:26	AWDCH[3:0]	RW	0000	Analog watchdog channel selection, software can clear and set this bit. Analog Watchdog Monitors Selected Input Channels 0000: ADC analog input channel 0 0001: ADC analog input channel 1 0010: ADC analog input channel 2 …. 1011: Reserved 1011: ADC analog input channel 11 1100: ADC analog input channel 12 Other values: reserved bits Note: The channel configured by the AWDCH[3:0] bits also needs to be set to the CHSELR register Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)

25:24	Reserved			
23	AWDEN	RW	0	Analog Watchdog Enable Software can set and clear this bit 0: Disable analog watchdog 1: Enable watchdog Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
22	AWDSGL	RW	0	Enable analog watchdog on one channel or all channels Software can set and clear this bit to enable the analog watchdog on the channel or all channels set by the AWDCH[3:0] bits 0: Enable analog watchdog on all channels 1: Enable analog watchdog on one channel Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
21:17	Reserved			
16	DISCEN	RW	0	discontinuous mode Software can set and clear this bit to enable/disable discontinuous mode 0: Disable discontinuous mode 1: Enable discontinuous mode It is not possible to enable both discontinuous and continuous modes, setting DISCEN = 1 and CONT = 1 is prohibited. Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
15	Reserved			
14	WAIT	RW	0	wait for conversion mode Software can set and clear this bit to enable/disable wait for conversion mode 0: wait for conversion mode to close 1: Wait for conversion mode to open Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
13	CONT	RW	0	Single/Continuous Conversion Mode Software can set and clear this bit. If set to 1, the conversion will occur consistently until the bit is cleared It is not possible to enable both discontinuous and continuous modes, setting DISCEN = 1 and CONT = 1 is prohibited. Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
12	OVRMOD	RW	0	Overload Management Mode Software can set and clear this bit to configure how data overload is managed 0: ADC_DR register retains old value when overload occurs 1: When an overload occurs, the ADC_DR register will be oveRWritten by the last conversion result Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
11:10	EXTEN[1:0]	RW	00	External trigger enable and polarity selection Software can set and clear this bit, select drive polarity and enable drive 00: Hardware driver detection disabled (software boot transition) 01: Rising edge hardware drive detection 10: Falling edge hardware driver detection 11: Rising edge and falling edge hardware driver detection Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
9	Reserved			
8:6	EXTSEL[2:0]	RW	000	External trigger selection This bit selects the external event that triggers the start of a conversion 000: TRG0(TIM1_TRG0) 001: TRG1(TIM1_CC4) 010: TRG2(Reserved) 011: TRG3(Reserved) 100: TRG4(Reserved) 101: TRG5(Reserved) 110: TRG6(Reserved) 111: TRG7(Reserved)
5	ALIGN	RW	0	Data alignment Software sets and clears this bit to select right or left justification 0: right-aligned 1: Left-aligned

				Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
4:3	RESSEL[1:0]	RW	00	Data resolution Software sets this bit to select the conversion resolution 00: 12 bits 01: 10 bits 10: 8 bits 11: 6 bits These bits are software operable only when ADEN = 0
2	SCANDIR	RW	0	Scan sequence direction Software can set and clear this bit to select the scan sequence direction 0: Up (from channel 0 to channel 11) 1: Down (from channel 11 to channel 0) Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
1:0	Reserved			

13.10.5. ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:28	CKMODE [3:0]	RW	0	ADC clock mode, software can set and clear this bit to define the clock source of the analog ADC 0000: PCLK 0001: PCLK/2 0010: PCLK/4 0011: PCLK/8 0100: PCLK/16 0101: PCLK/32 0110: PCLK/64 1000: HIS 1001: HSI/2 1010: HSI/4 1011: HSI/8 1100: HSI/16 1101: HSI/32 1110: HSI/64 Note: ADCAL = 0, ADSTART = 0, ADSTP = 0 and ADEN = 0 only when ADC is not enabled). Software is allowed to manipulate these bits
27:0	Reserved			

13.10.6. ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

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

													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	Reserved			
2:0	SMP[2:0]	RW	000	Sampling time selection Software configurable bit selects the sampling time for all channels 000: 3.5 ADC clock cycles 001: 5.5 ADC clock cycles 010: 7.5 ADC clock cycles 011: 13.5 ADC clock cycles 100: 28.5 ADC clock cycles 101: 41.5 ADC clock cycles 110: 71.5 ADC clock cycles 111: 239.5 ADC clock cycles Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)

13.10.7. ADC watchdog threshold register (ADC_TR)

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	HT											
				RW	RW	RW	RW	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	LT											
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	Reserved			
27:16	HT[11:0]	RW	0xFFF	Analog Watchdog High Threshold Software configurable to define analog watchdog high threshold Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
15:12	Reserved			
11:0	LT[11:0]	RW	0x000	Analog Watchdog Low Threshold Software configurable to define analog watchdog low threshold Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)

13.10.8. ADC channel selection register (ADC_CHSELR)

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	CHSEL 12	CHSEL 11	Res	CHSEL 9	CHSEL 8	CHSEL 7	CHSEL 6	CHSEL 5	CHSEL 4	CHSEL 3	CHSEL 2	CHSEL 1	CHSEL 0
			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	Reserved		0	
13	Reserved	RW	0	
12	CHSEL12	RW	0	Channel 12 (VREFINT) select enable

				0: Channel12 is not selected for conversion 1: Channel12 is selected for conversion Software is allowed to write this bit only if ADSART = 0 (to ensure no conversions are in progress)
11	CHSEL11	RW	0	Channel 11 (TS) select enable 0: Channel11 is not selected for conversion 1: Channel11 is selected for conversion Software is allowed to write this bit only if ADSART = 0 (to ensure no conversions are in progress)
10	Reserved	RW	0	
9:0	CHSELx	RW	0x0000	Channel selection These bits are software configurable to define the sequence conversion channel 0: Input channel-x is not selected for conversion 1: Input channel-x is selected for conversion Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)

13.10.9. ADC data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved		0	
15:0	Reserved	R	0x0	Converted data This bit is read-only. The conversion result of the last converted channel is placed in this register. Data is left-aligned or right-aligned.

13.10.10. ADC calibration configuration and status registers (ADC_CCSR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON.	CALFAIL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
R	RC_W1														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	CALSMP[2:0]		CALSEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
		RW		RW											

Bit	Name	R/W	Reset Value	Function
31	CALON	R	0	Calibration flag, indicating that ADC calibration is in progress. 1: ADC calibration in progress 0: ADC calibration has ended or ADC calibration has not been started
30	CALFAIL	RC_W1	0	Calibration fail flag, which shows whether the ADC calibration is successful, used in conjunction with CALON. CALON = 0, CALFAIL = 1: ADC calibration failed CALON = 0, CALFAIL = 0: ADC calibration is successful CALON = 1, CALFAIL = 0: Calibrating

				CALON = 1, CALFAIL = 1: Invalid state Set by hardware, cleared by software writing 1 or cleared by software writing ADCAL = 1.
29:14	Reserved	-	0	-
13:12	CALSMP[2:0]	RW	0	Calibration sample time selection Configure the number of clock cycles for the sampling phase of calibration based on the following information: 00: 2 ADC clock cycles 01: 4 ADC clock cycles 10: 8 ADC clock cycles 11: 1 ADC clock cycle The longer the cycle of configuring SMP during calibration, the more accurate the calibration result, but this configuration will bring the problem of prolonged calibration cycle
11	CALSEL	RW	0	Calibration content selection bit, used to select the content that needs to be calibrated 1: Calibrate OFFSET and linearity 0: Only calibrate OFFSET
10:0	Reserved	-	0	-

13.10.11. ADC common configuration register (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	TSEN	VREFEN	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	Res	Res	Res	Res	Res	Res	Res	Res

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23	TSEN	RW	0	Temperature sensor enable bit, software can set and clear this bit, enable/disable temperature sensor 0: Disable 1: enable Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
22	VREFEN	RW	0	Reference Vrefint enable bit, software can set and clear this bit, enable/disable reference Vrefint 0: Disable 1: enable Software is allowed to write these bits only when ADSART = 0 (to ensure no conversions are in progress)
21:0	Reserved			

13.10.12. ADC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_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.	AWD	Res.	Res.	OVR	EOSEQ	EOC	EOSMP	Res.
	Reset value																									0			0	0	0	0	
0x04	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDIE	Res.	Res.	OVRIE	EOSE-	EOCIE	EOSMPI	Res.
	Reset																									0			0	0	0	0	

	0x0008		0x000C			0x0010		0x0014		0x0020		0x0028		0x0040		0x0044		0x3008	
	ADCC_CR	Reset value	ADCC_CFGR1	Reset value		ADCC_CFGR2	Reset value	ADCC_SMPR	Reset value	ADCC_TR	Reset value	ADCC_HSELR	Reset value	ADCC_DR	Reset value	ADCC_CSR	Reset value	ADCC_CCR	Reset value
	ADCAL	0	Res.			CKMODE[3:0]	0	Res.		Res.		Res.		Res.		CALON	0	Res.	
	Res.		Res.				0	Res.		Res.		Res.		Res.		CALFAIL	0	Res.	
	Res.		AWDCH[3:0]	0			0	Res.		Res.		Res.		Res.		Res.		Res.	
	Res.			0			0	Res.		Res.		Res.		Res.		Res.		Res.	
	Res.			0		Res.		Res.		HT[11:0]	1	Res.		Res.		Res.		Res.	
	Res.			0		Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		AWDEN	0		Res.		Res.			1	Res.		Res.		Res.		TSEN	0
	Res.		AWDSGL	0		Res.		Res.			1	Res.		Res.		Res.		VREFEN	0
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		DISCEN	0		Res.		Res.			1	Res.		Res.		Res.		Res.	
	Res.		Res.			Res.		Res.		Res.		Res.		DATA[15:0]	0	Res.		Res.	
	Res.		WAIT	0		Res.		Res.		Res.		Res.			0	Res.		Res.	
	Res.		CONT	0		Res.		Res.		Res.		Res.	0		0	CALSMP[1:0]	0	Res.	
	Res.		OVRMOD	0		Res.		Res.		Res.		CHSEL12	0		0		0	Res.	
	Res.		Res.			Res.		Res.		LT[11:0]	0	CHSEL11	0		0	CALSEL	0	Res.	
	Res.		Res.			Res.		Res.			0	Res.	0		0	Res.		Res.	
	Res.		Res.			Res.		Res.			0	CHSEL9	0		0	Res.		Res.	
	Res.		EXTSEL[2:0]	0		Res.		Res.			0	CHSEL8	0		0	Res.		Res.	
	Res.					Res.		Res.			0	CHSEL7	0		0	Res.		Res.	
	Res.					Res.		Res.			0	CHSEL6	0		0	Res.		Res.	
	Res.		ALIGN	0		Res.		Res.			0	CHSEL5	0		0	Res.		Res.	
	Res.		RES_SEL[1:0]	0		Res.		Res.			0	CHSEL4	0		0	Res.		Res.	
	Res.			0		Res.		Res.			0	CHSEL3	0		0	Res.		Res.	
	Res.		SCADIR	0		Res.		SMP[2:0]	0		0	CHSEL2	0		0	Res.		Res.	
	Res.		Res.			Res.					0	CHSEL1	0		0	Res.		Res.	
	Res.		ADEN	0		Res.		Res.			0	CHSEL0	0		0	Res.		Res.	

14. Advanced-control timer (TIM1)

14.1. TIM1 introduction

The advanced-control timers (TIM1) consist of a 16-bit auto-reload counter driven by a programmable prescaler. It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers. The advanced-control (TIM1) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together.

14.2. TIM1 main features

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’s output signals in reset state or in a known state.
- Interrupt generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

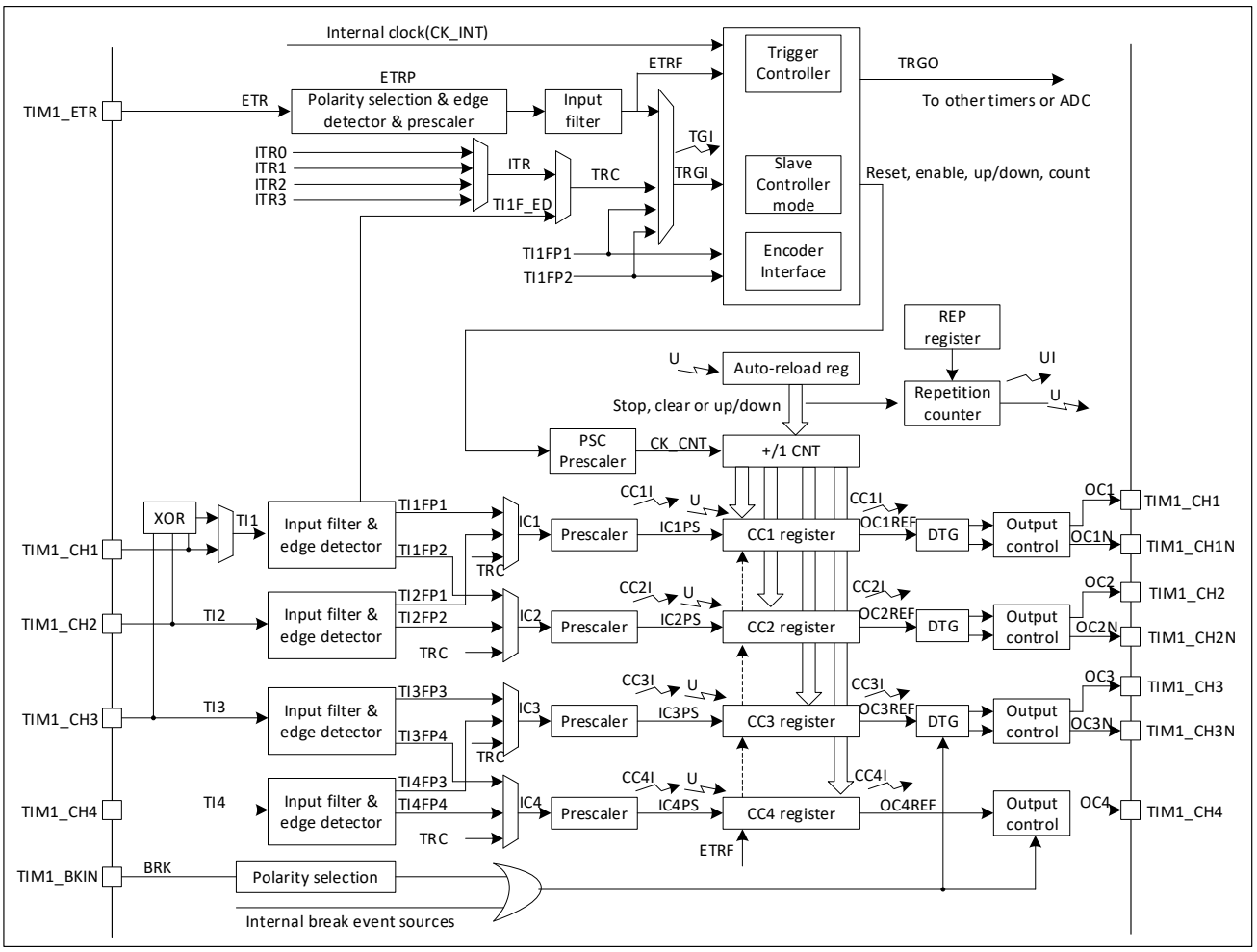


Figure 14-1 Advanced-control timer block diagram

Notes:

Reg — Preload registers transferred to active registers on U event according to control bit

Event

Interrupt & DMA output

14.3. TIM1 functional description

14.3.1. 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 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.

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 17-2 and Figure 17-3 give some examples of the counter behavior when the prescaler ratio is changed on the fly:

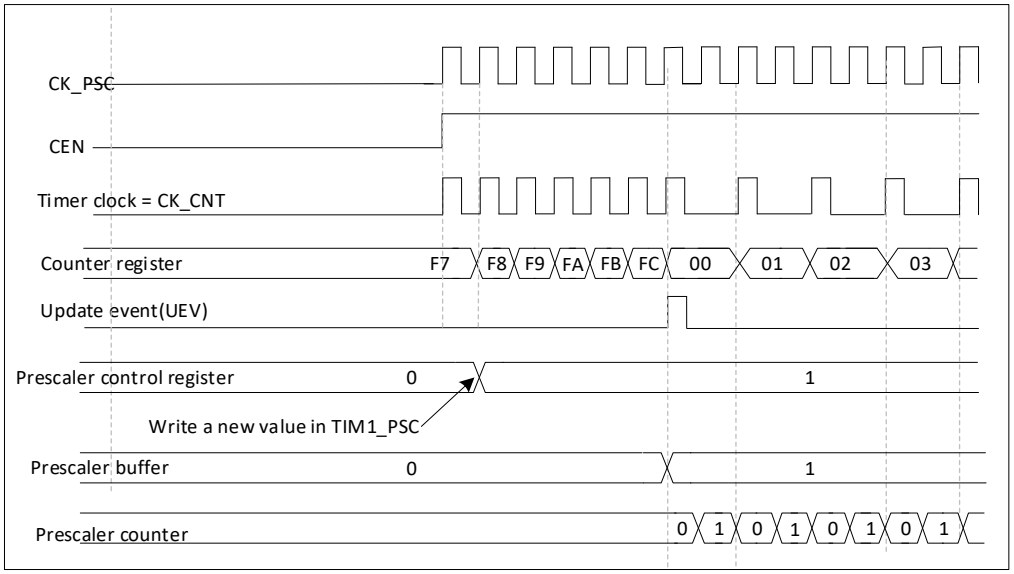


Figure 14-2 Counter timing diagram with prescaler division change from 1 to 2

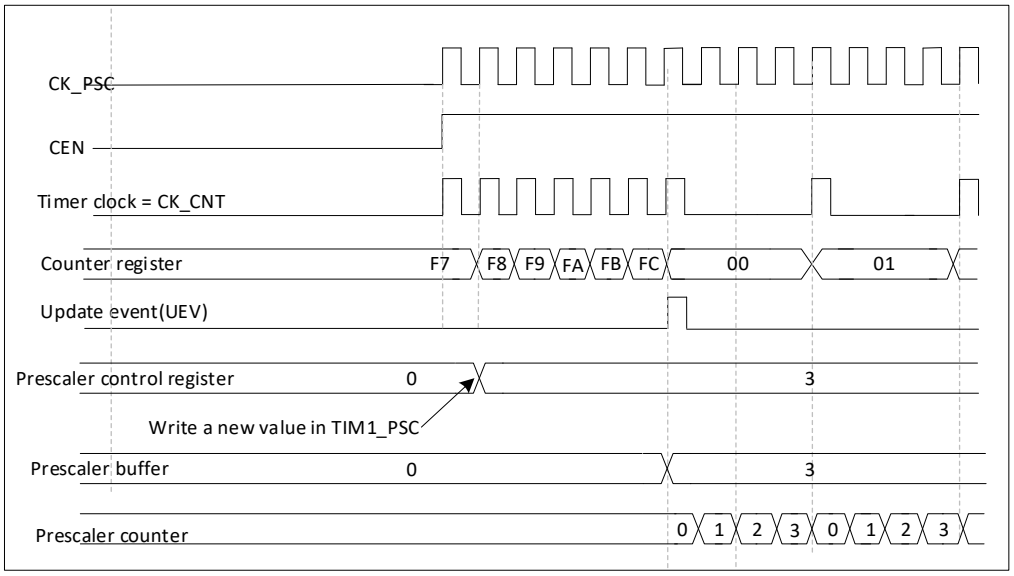


Figure 14-3 Counter timing diagram with prescaler division change from 1 to 4

14.3.2. Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register plus one (TIMx_RCR+1). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0.

However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt 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_PSCregister)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

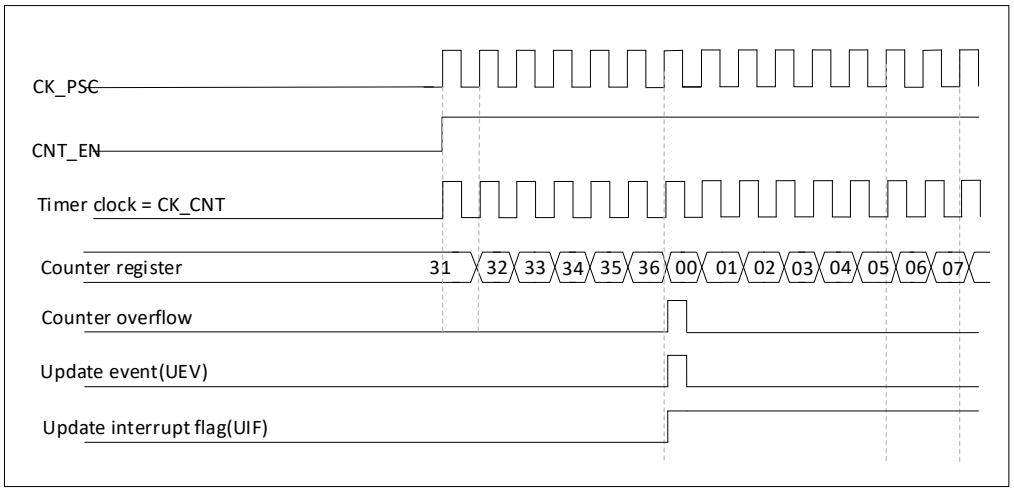


Figure 14-4 Counter timing diagram, internal clock divided by 1

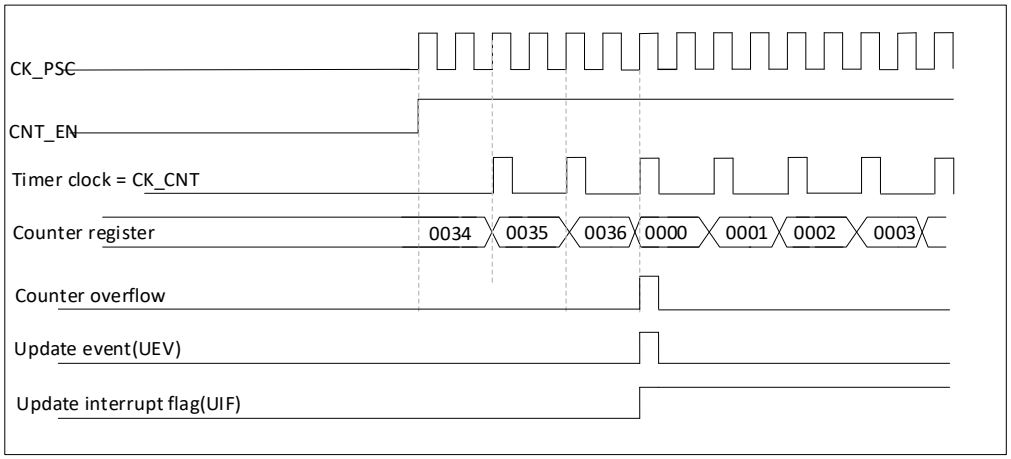


Figure 14-5 Counter timing diagram, internal clock divided by 2

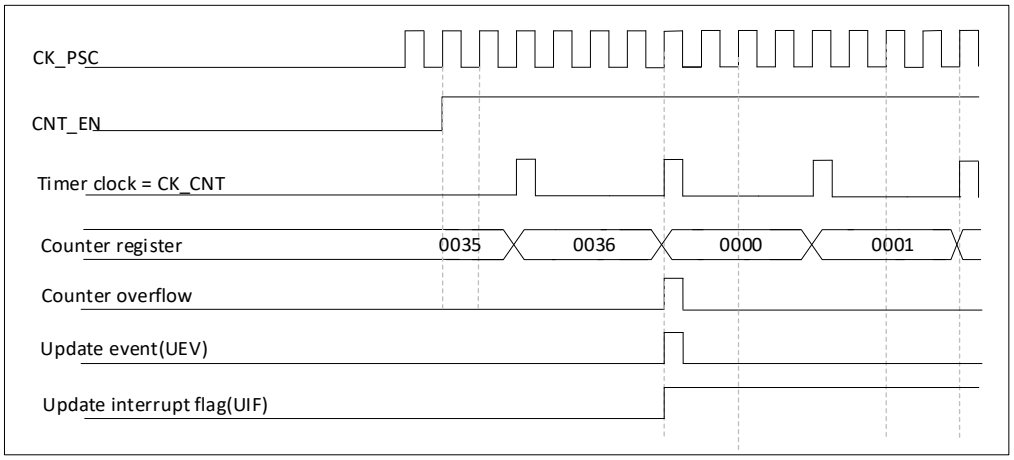


Figure 14-6 Counter timing diagram, internal clock divided by 4

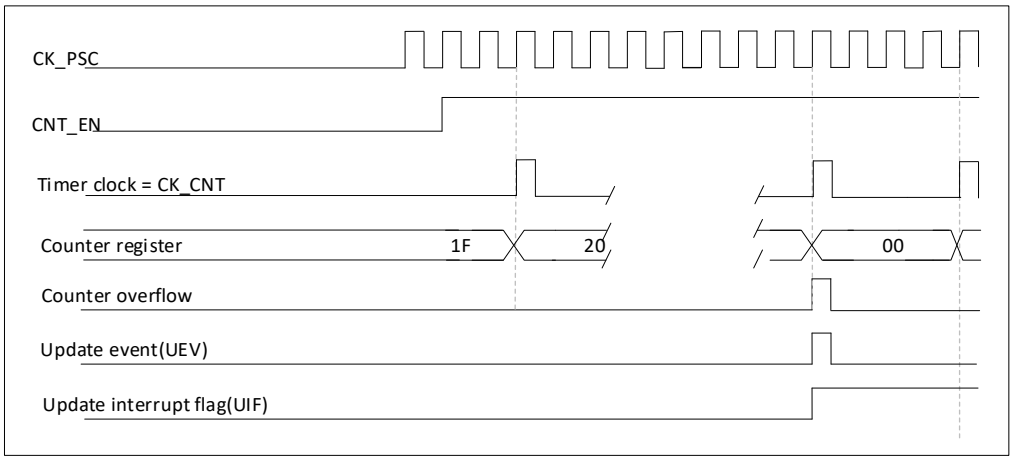


Figure 14-7 Counter timing diagram, internal clock divided by N

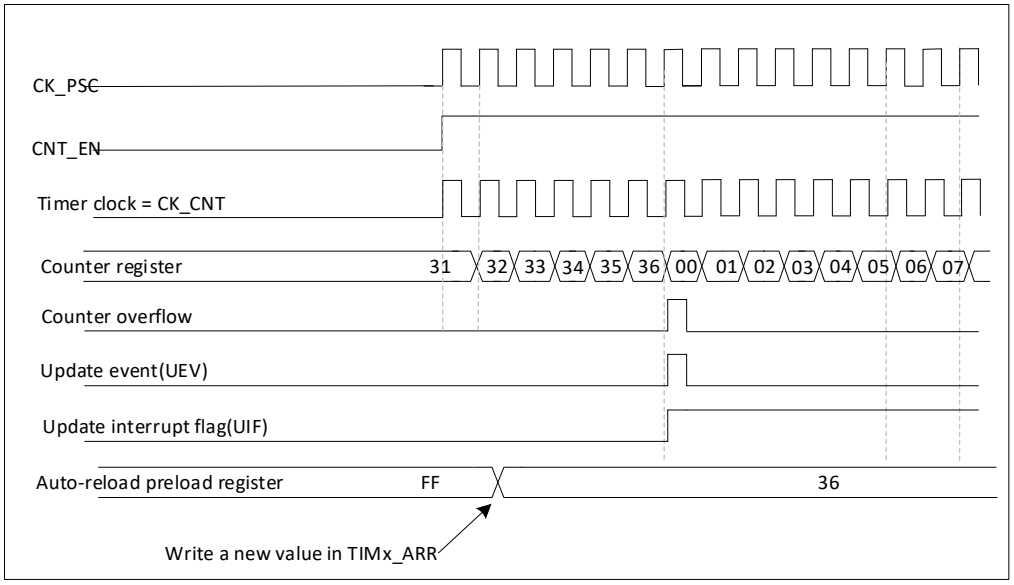


Figure 14-8 Counter timing diagram, update event when ARPE = 0 (TIMx_ARR no preloaded)

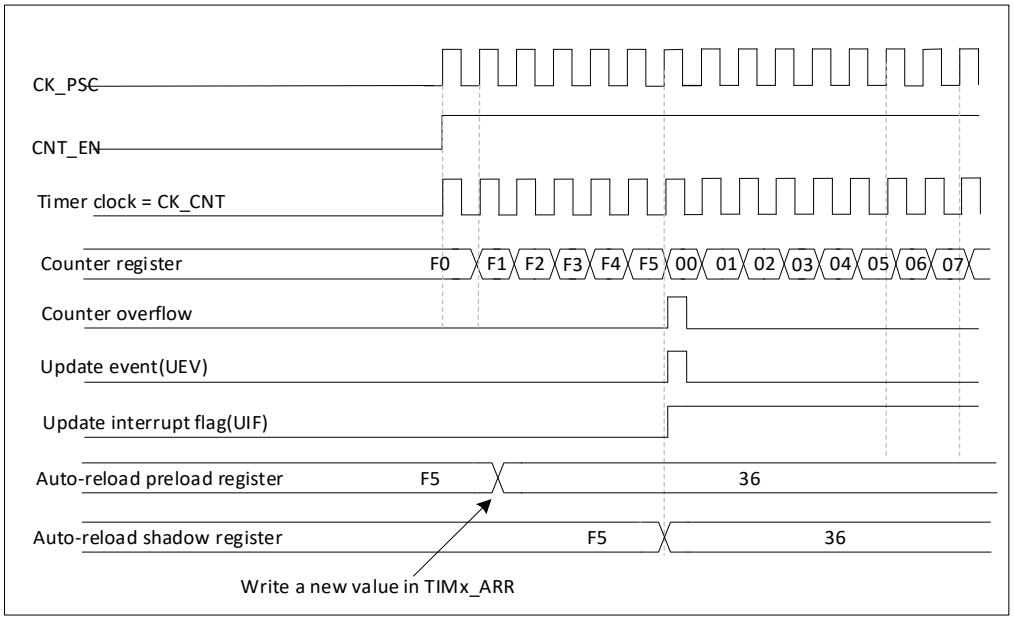


Figure 14-9 Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)

Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0.

However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt 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: the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

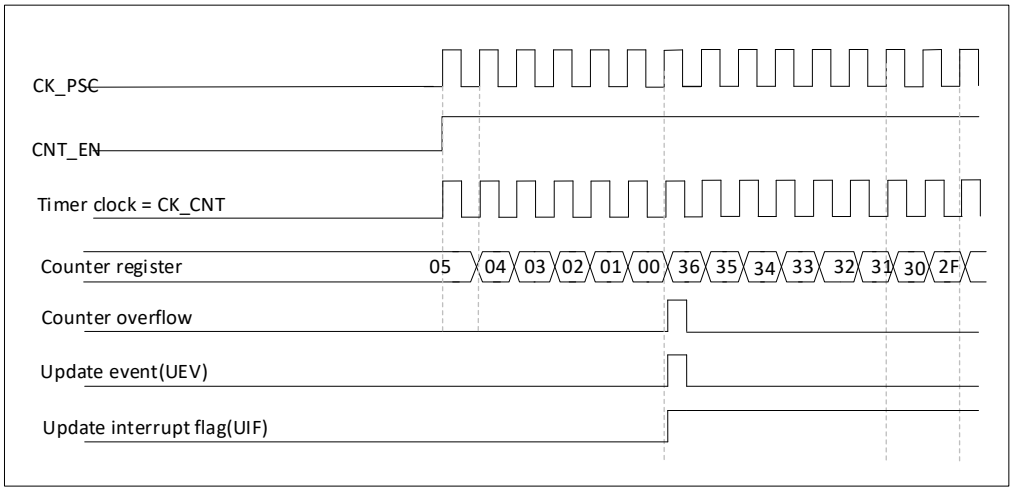


Figure 14-10 Counter timing diagram, internal clock divided by 1

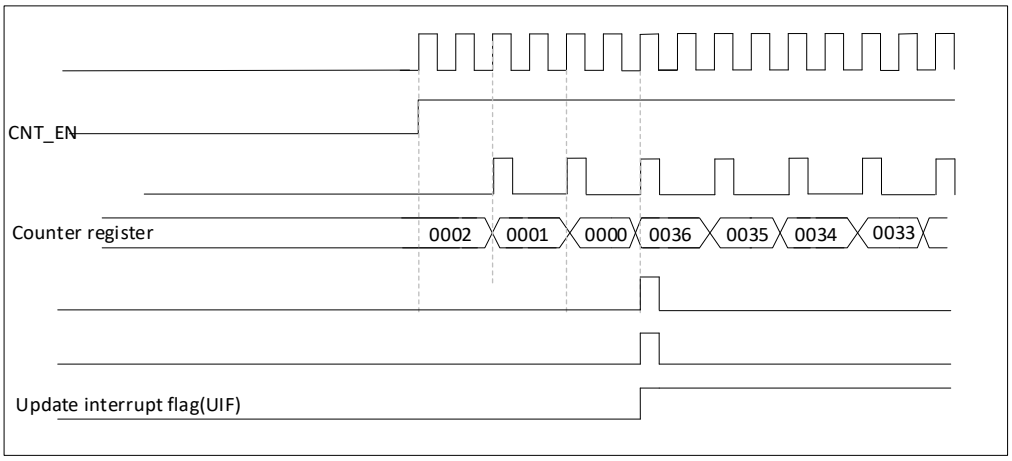


Figure 14-11 Counter timing diagram, internal clock divided by 2

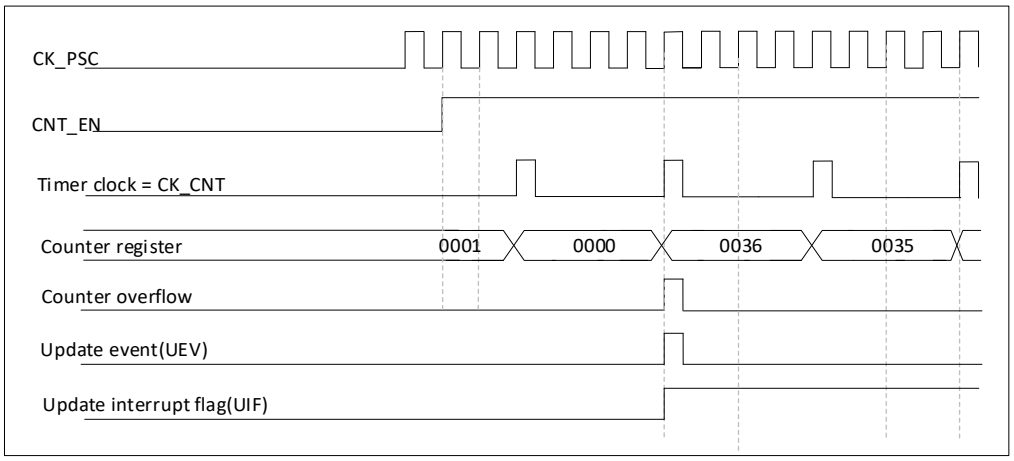


Figure 14-12 Counter timing diagram, internal clock divided by 4

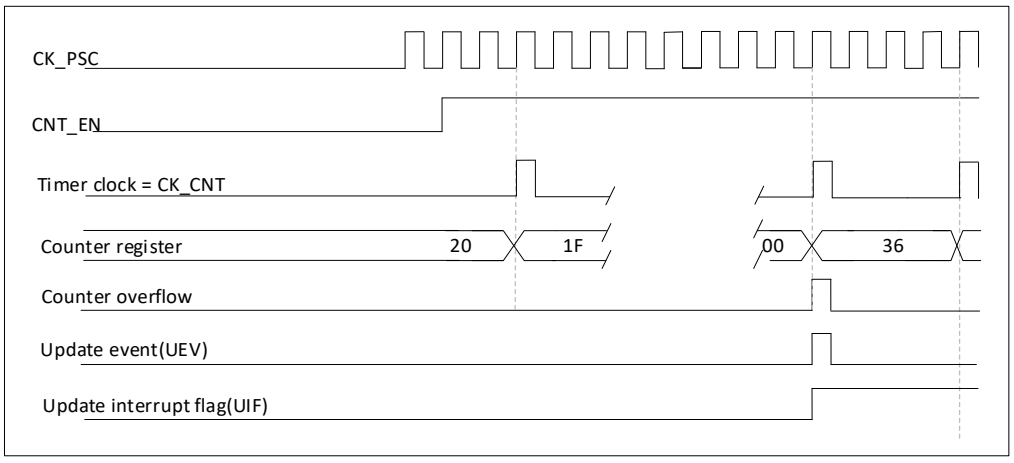


Figure 14-13 Counter timing diagram, internal clock divided by N

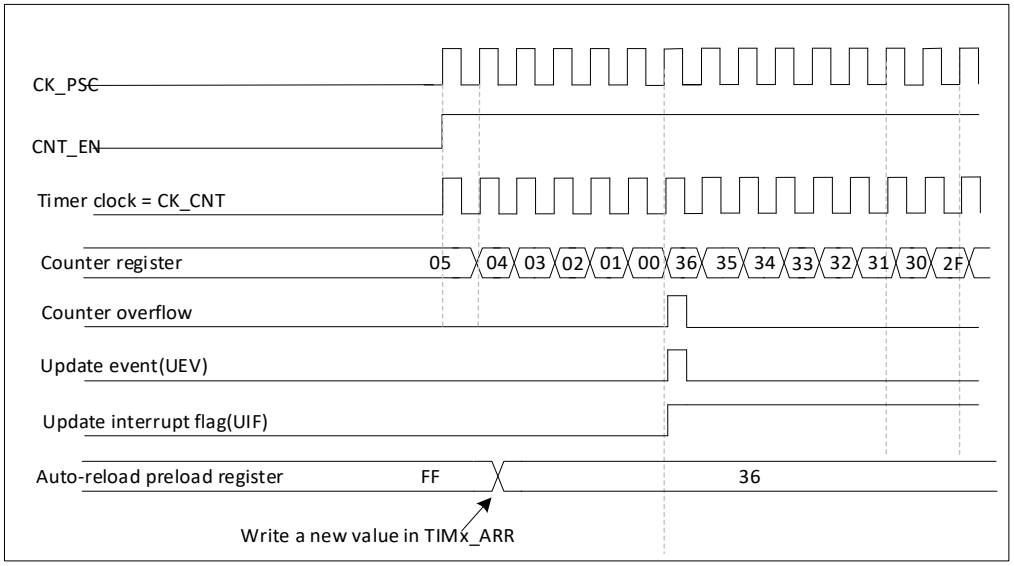


Figure 14-14 Counter timing diagram, update event when repetition counter is not used

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 autoreload 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 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: if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

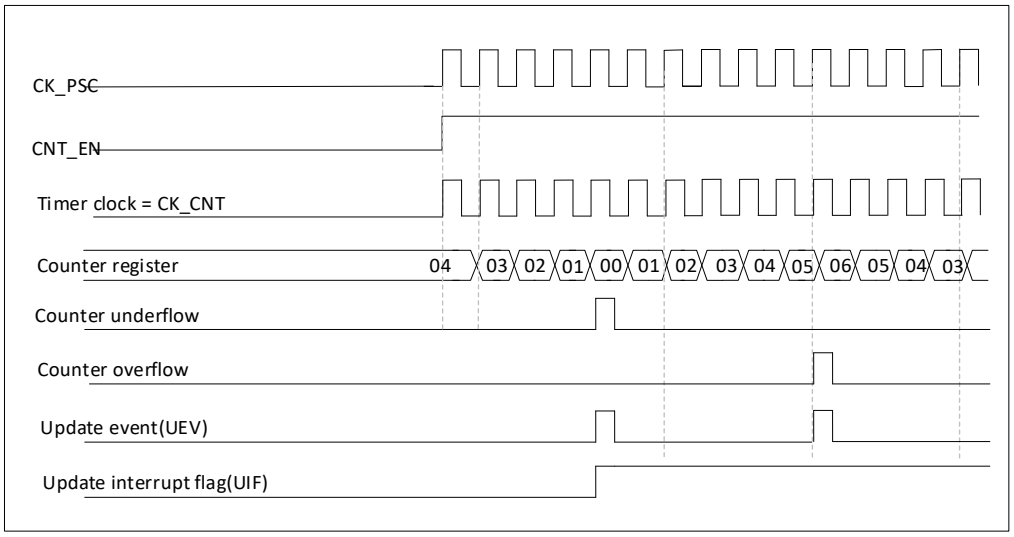


Figure 14-15 Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

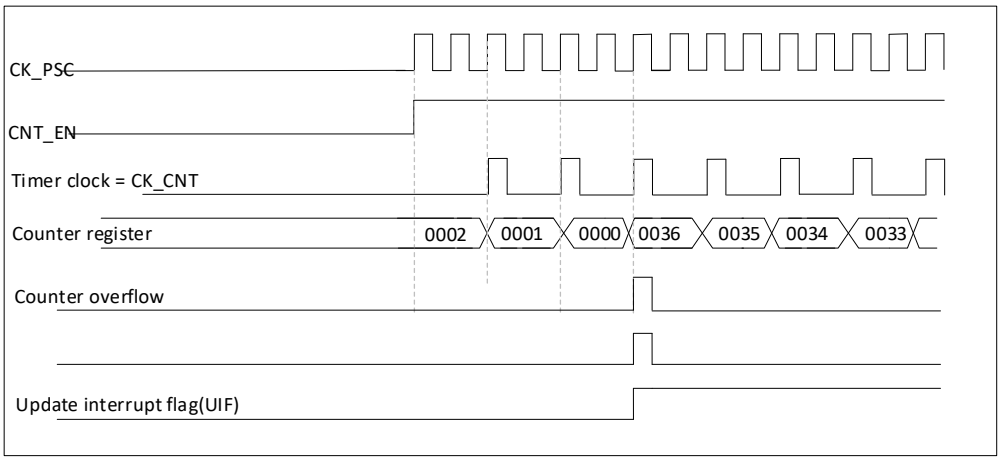


Figure 14-16 Counter timing diagram, internal clock divided by 2, TIMx_ARR = 0x36

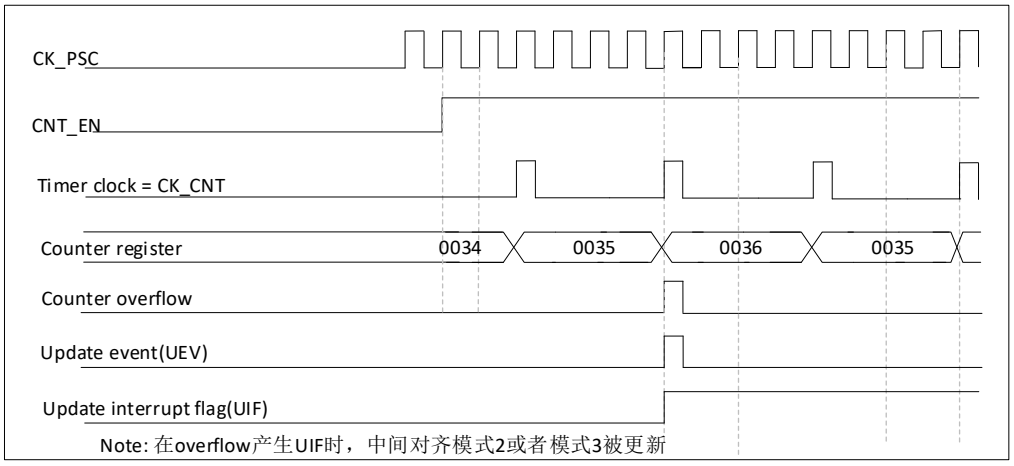


Figure 14-17 Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

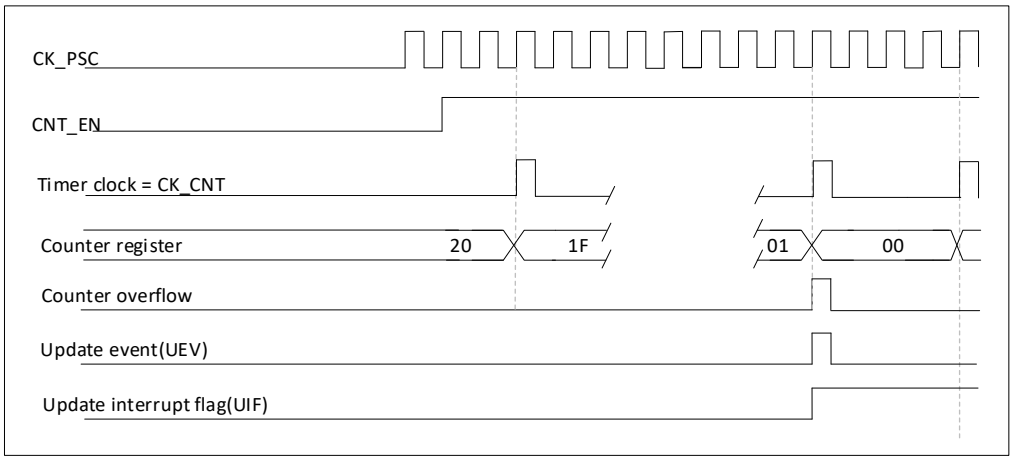


Figure 14-18 Counter timing diagram, internal clock divided by N

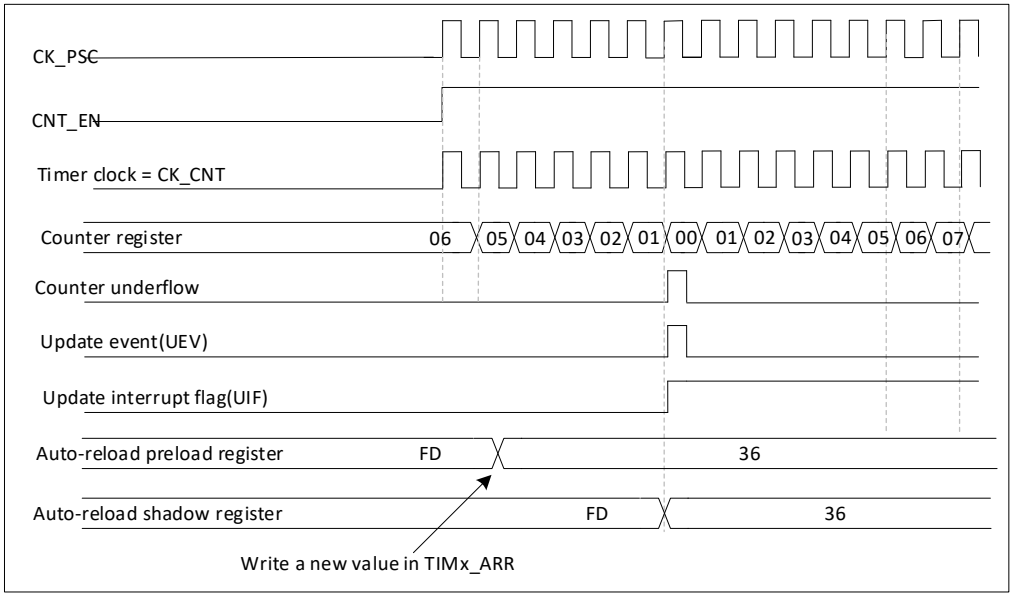


Figure 14-19 Counter timing diagram, update event with ARPE = 1 (counter underflow)

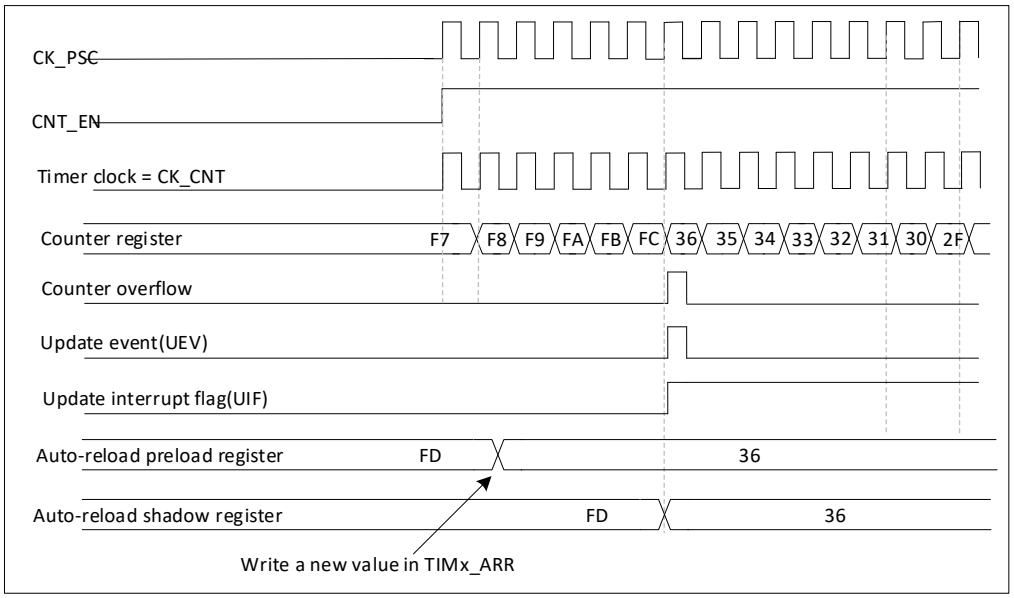


Figure 14-20 Counter timing diagram, Update event with ARPE = 1 (counter overflow)

14.3.3. Repetition counter

Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows,where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

- At each counter overflow in upcounting mode.
- At each counter underflow in downcounting mode.
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is 2xTck, 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. When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

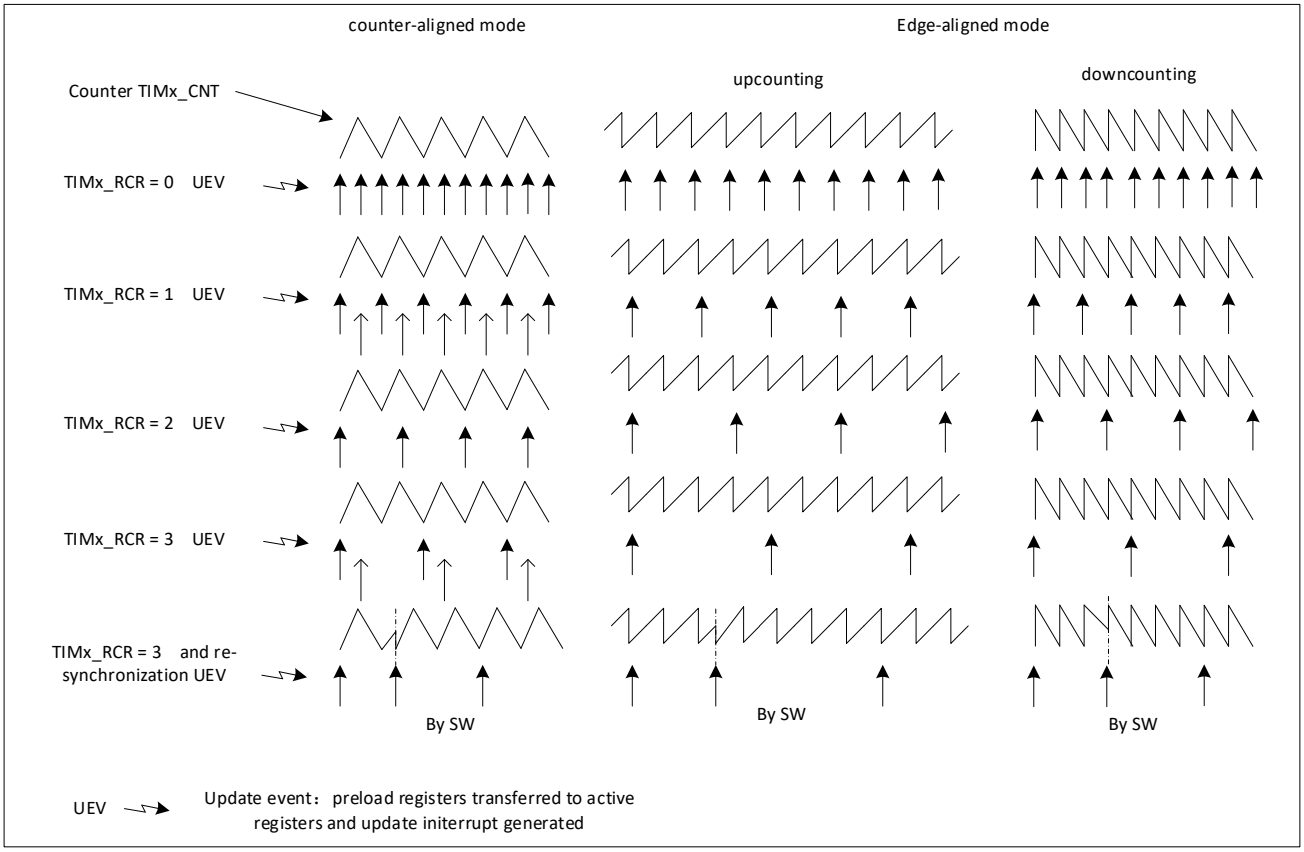


Figure 14-21 Update rate examples depending on mode and TIMx_RCR register settings

14.3.4. Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 3.

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.

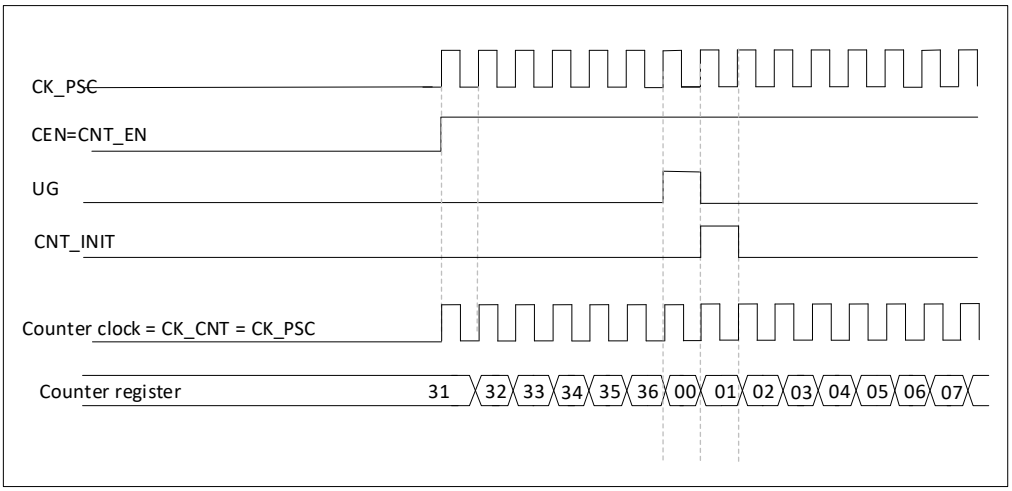


Figure 14-22 Control circuit in normal mode, internal clock divided by 1

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.

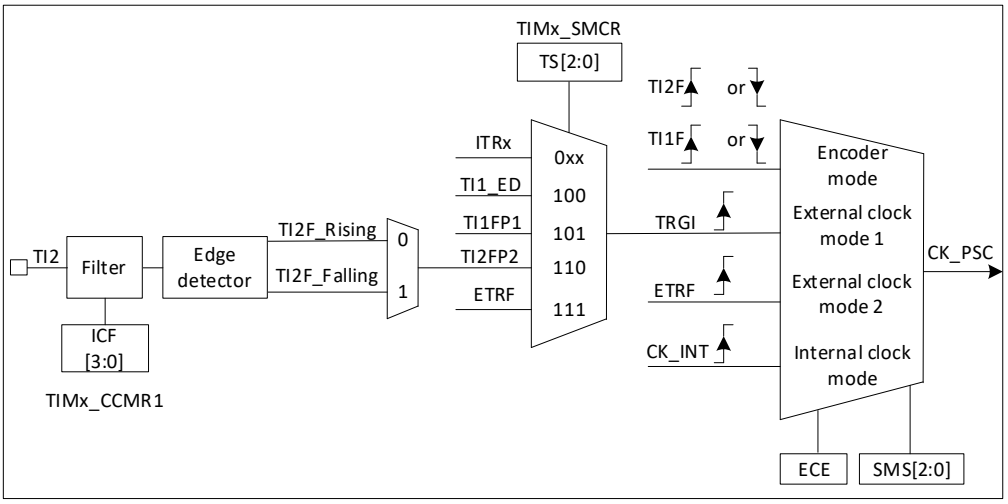


Figure 14-23 TI2 external clock connection example

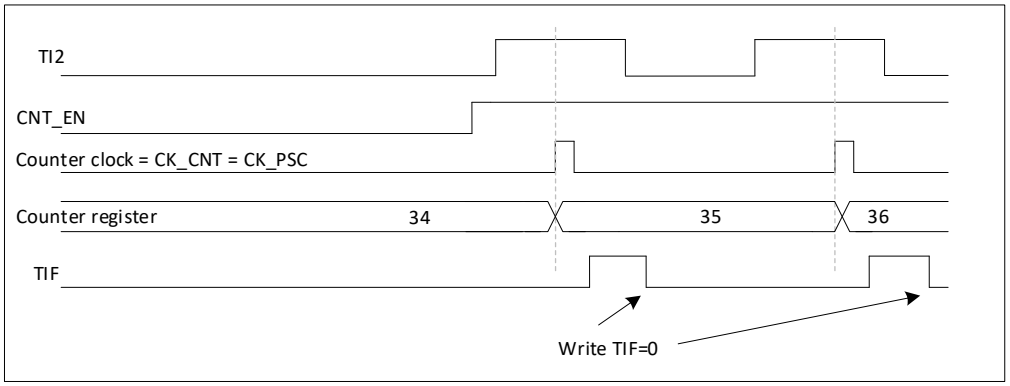


Figure 14-24 Control circuit in external clock mode 1

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.

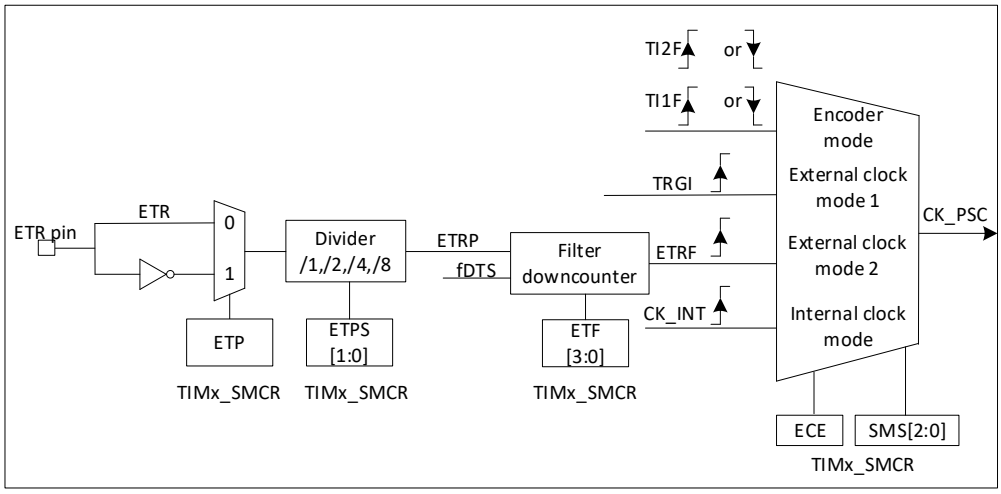


Figure 14-25 External trigger input block

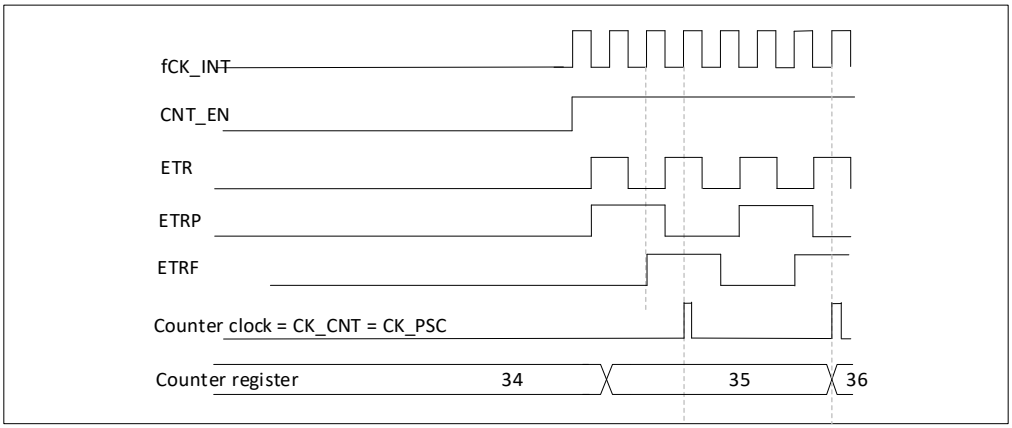


Figure 14-26 Control circuit in external clock mode 2

14.3.5. Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

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).

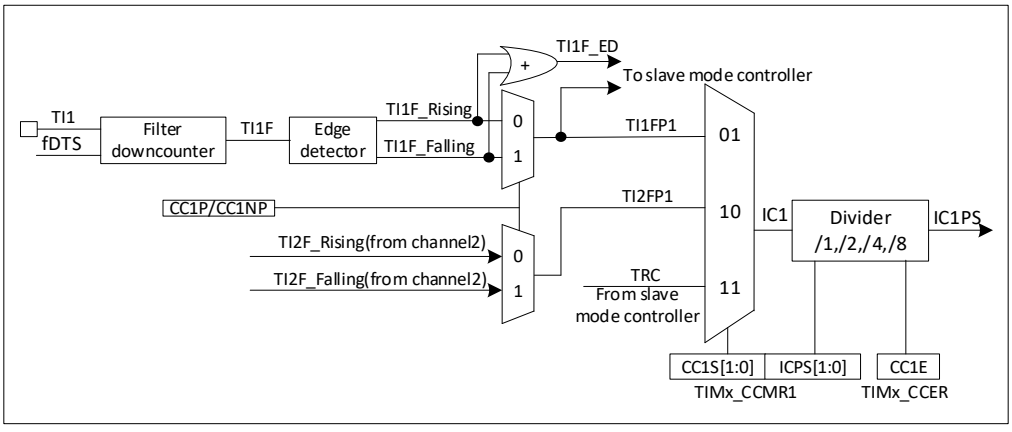


Figure 14-27 Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform that is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

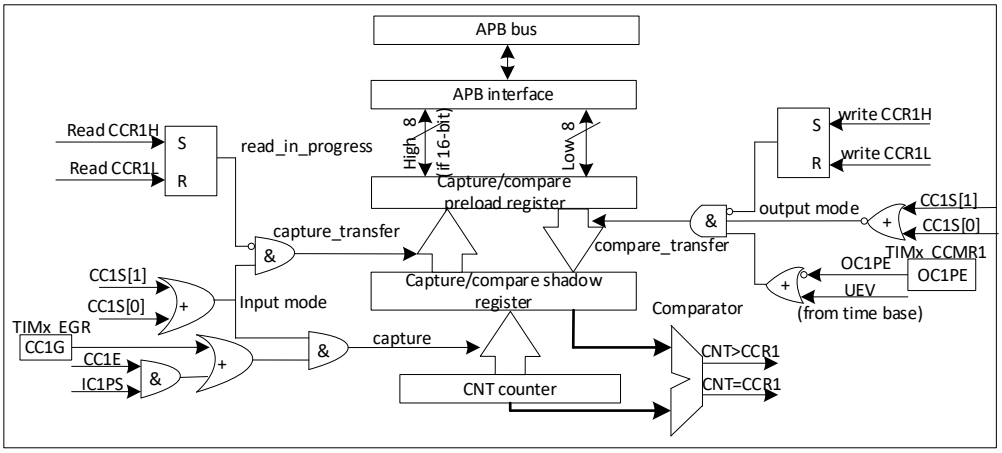


Figure 14-28 Capture/compare channel 1 main circuit

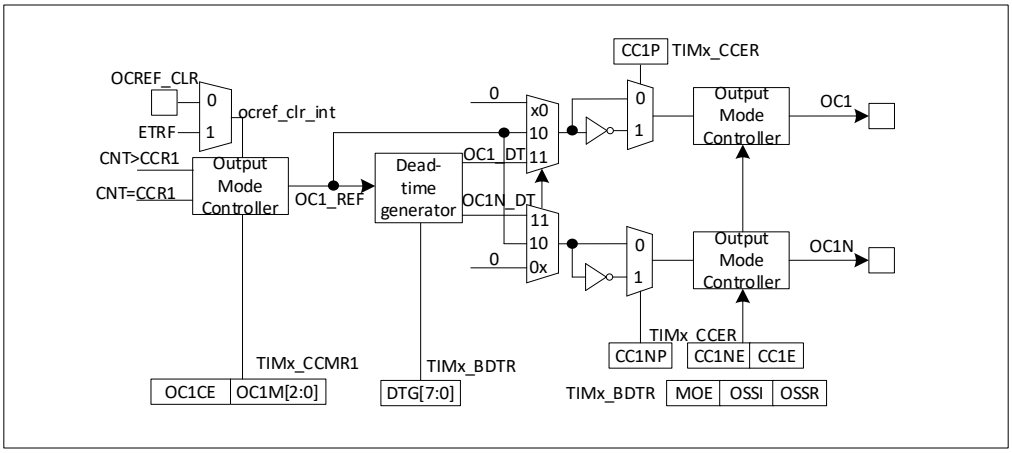


Figure 14-29 Output stage of capture/compare channel (channel 1 to 3)

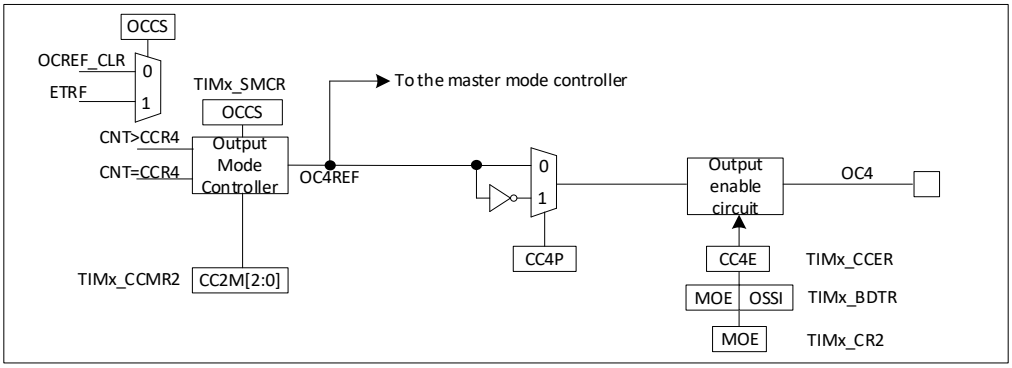


Figure 14-30 Output stage of capture/compare channel (channel 4)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

14.3.6. Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming ICxF bits in the TIMx_CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clockcycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

14.3.7. PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two Icx signals are mapped on the same Tix input.
- The 2 Icx signals are active on edges with opposite polarity.
- One of the two TixFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

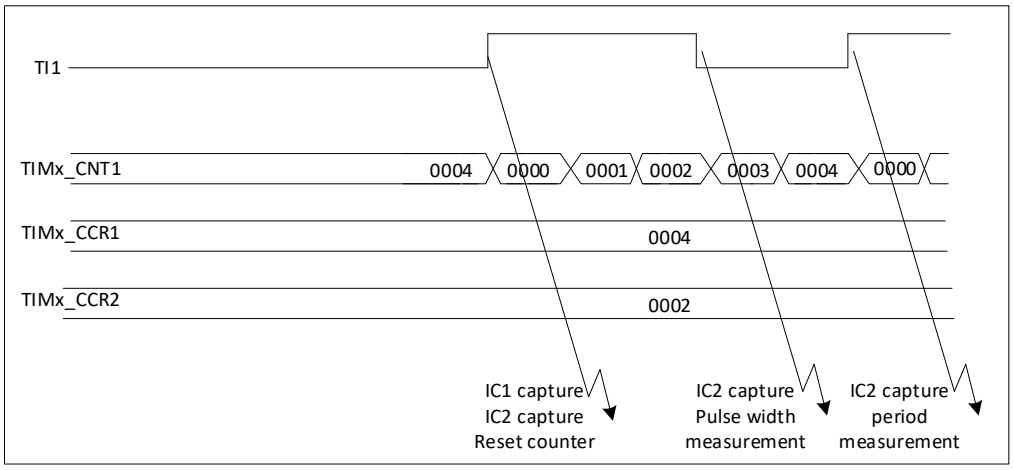


Figure 14-31 PWM input mode timing

14.3.8. Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (OCx active high) = > OCx is forced to high level. The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

The comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

14.3.9. Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed. When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output

pin can keep its level (OCXM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

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_CCRxshadow register is updated only at the next update event UEV). An example is given in Figure 14-32.

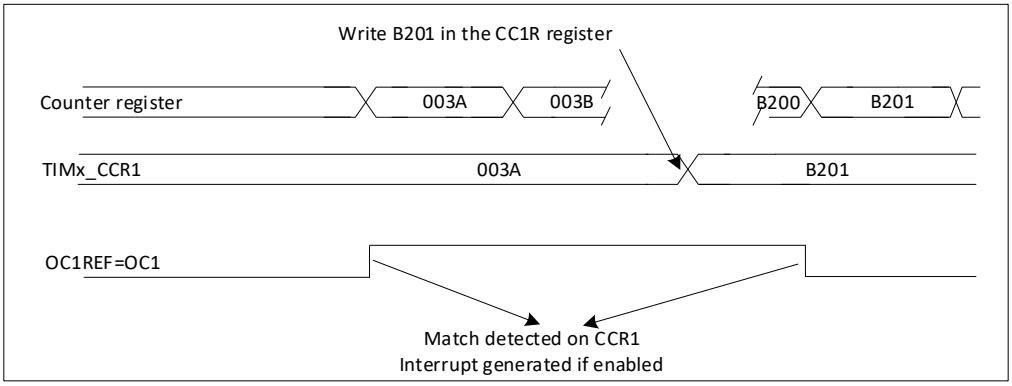


Figure 14-33 Output compare mode, toggle on OC1.

14.3.10. PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- **Upcounting configuration**

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to Upcounting mode. Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to Upcounting mode.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 14-34 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

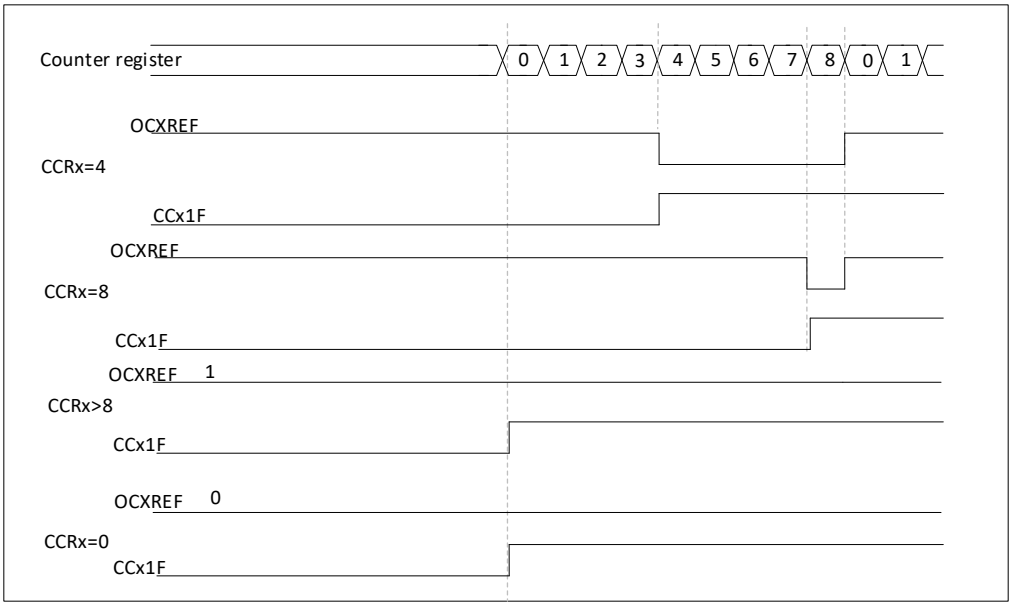


Figure 14-35 Edge-aligned PWM waveforms (ARR = 8)

- **Downcounting configuration**

Downcounting is active when DIR bit in TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals).

The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to Center-aligned mode (up/down counting).

- 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.

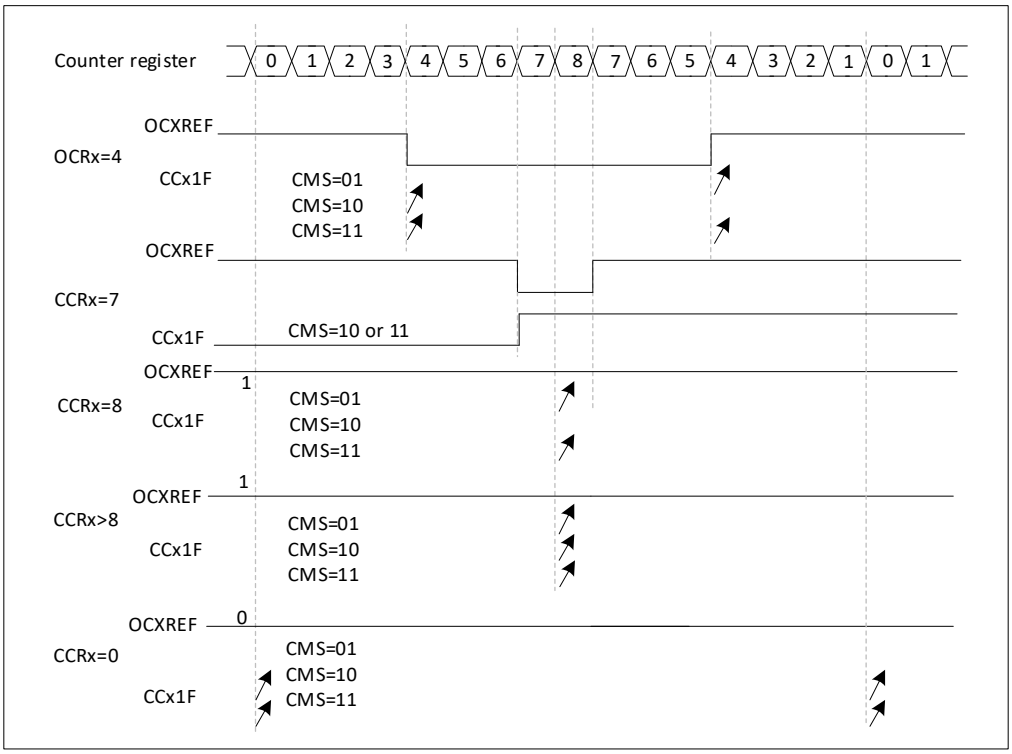


Figure 14-36 Center-aligned PWM waveforms (ARR = 8)

Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular: – The direction is not updated if the user writes a value in the counter greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it will continue to count up. – The direction is updated if the user writes 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

14.3.11. 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 adjust it

depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches).

User can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 17-3 for more details. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. DTG[7:0] bits of the TIMx_BDTR register are used to control the dead-time generation for all channels. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples).

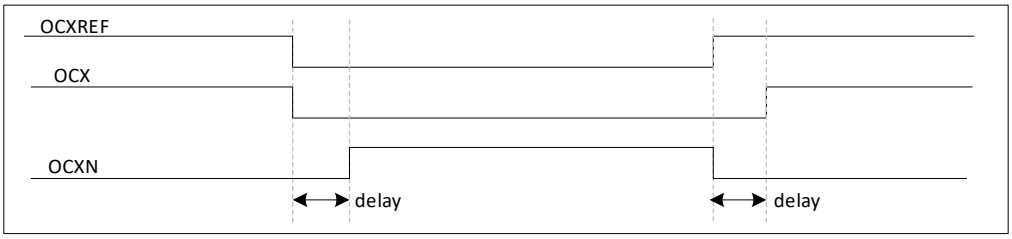


Figure 14-37 Complementary output with dead-time insertion

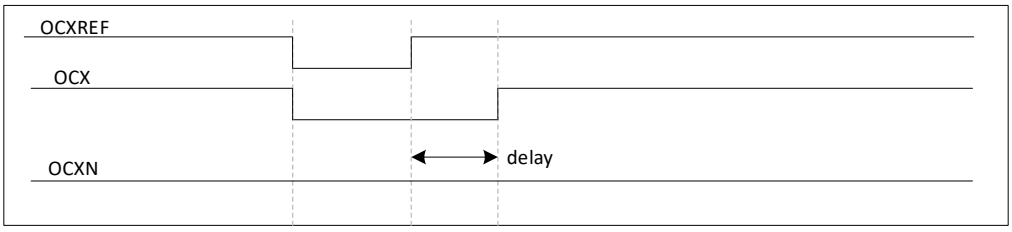


Figure 14-38 Dead-time waveforms with delay greater than the negative pulse.

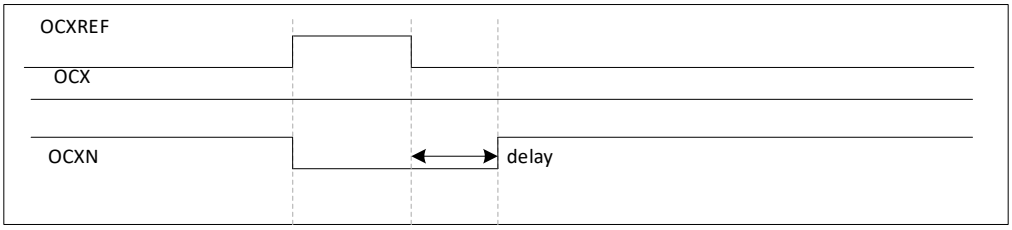


Figure 14-39 Dead-time waveforms with delay greater than the positive pulse.

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows the user to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP = 0 then OCxN = OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE = CCxNE = 1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

14.3.12. Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to Table 17-3 for more details.

The brake source can be either the brake input pin, or the following internal sources:

- Output from CPU LOCKUP
- Clock failure events generated by the Clock Security System (CSS)
- Output from comparator

After reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- he MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and

OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

- If OSSI = 0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

By using 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 freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. The LOCK bits can be written only once after an MCU reset.

The following figure shows the example of the output in response to a break.

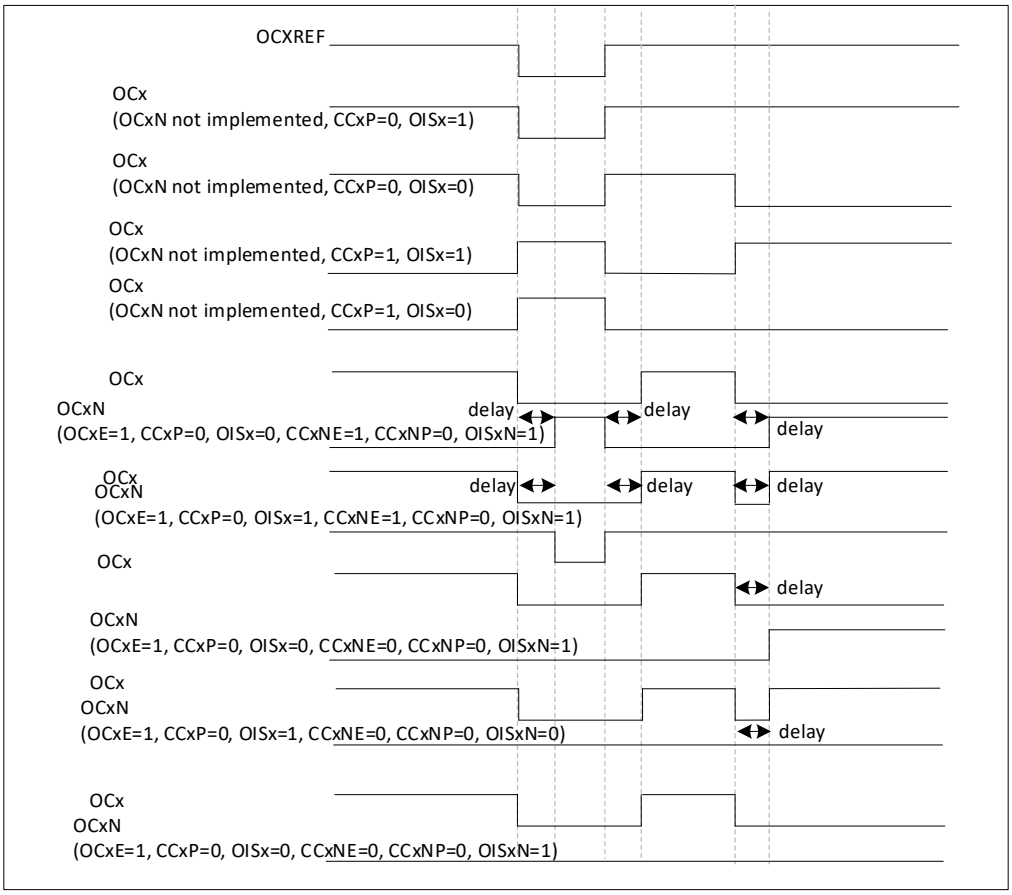


Figure 14-40 Output behavior in response to a break

14.3.13. Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to '1'). The OCxREF signal remains Low until the next update event of UEV occurs.

This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to '00'.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to '0'.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

The Figure below 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.

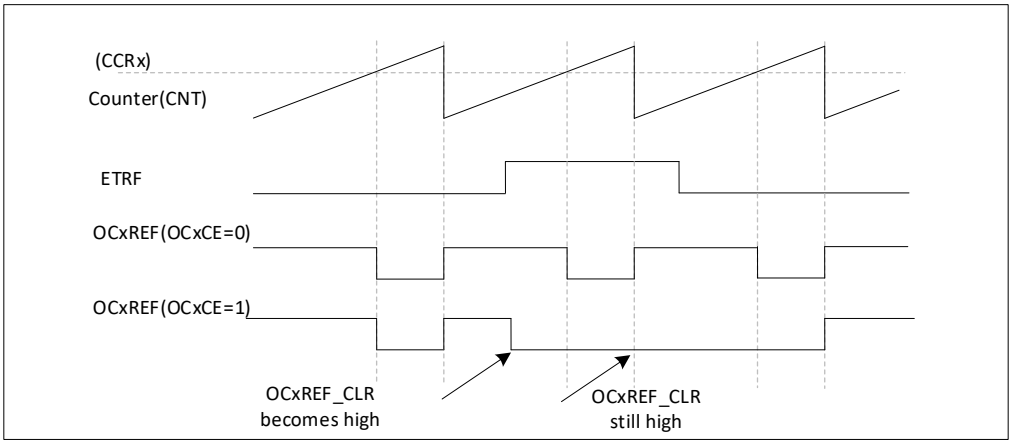


Figure 14-41 Clearing TIM1 OCxREF

14.3.14. 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. The user can thus program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) request (if the COMDE bit is set in the TIMx_DIER register). The figure blow describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

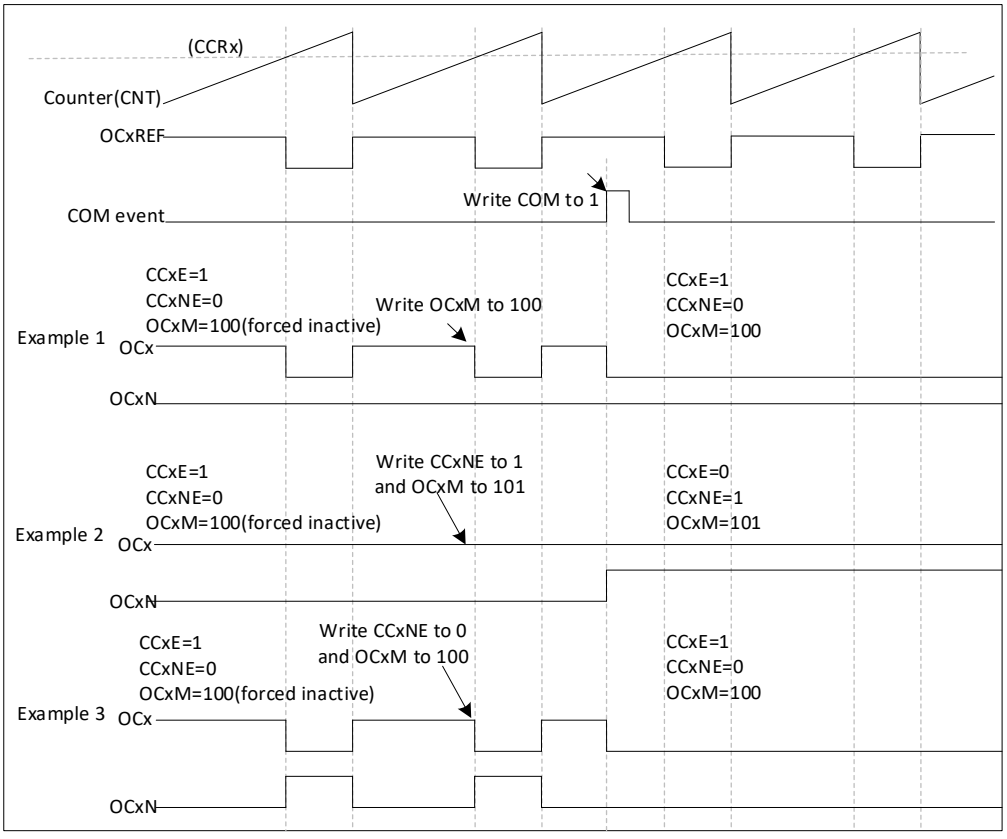


Figure 14-42 6-step generation, COM example (OSSR = 1)

14.3.15. One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value.

Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

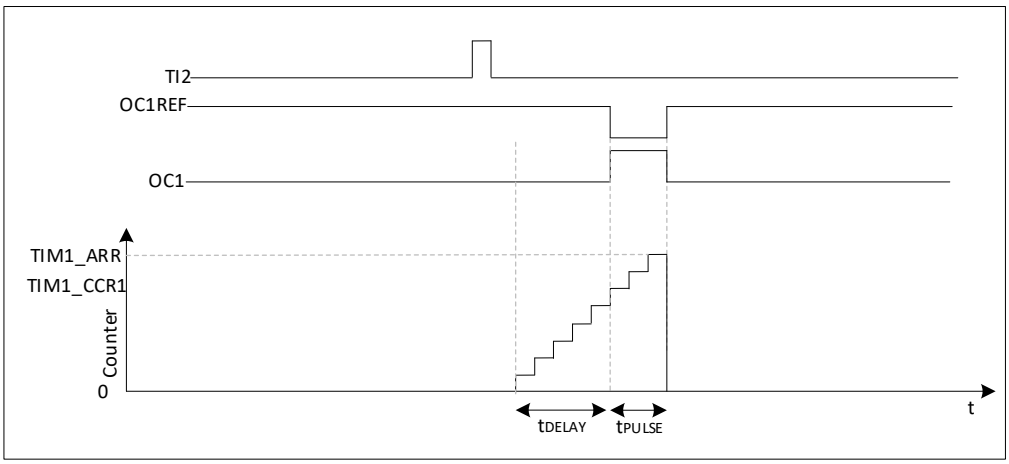


Figure 14-43 Example of one pulse mode

For example the user may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S = '01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P = '0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS = '110' in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR-TIMx_CCR1).
- When the user to build a waveform with a transition from '0' to '1' when a compare match oc-curs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this, enable PWM mode 2 by writing OC1M = 111 in the TIMx_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE = '1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case the compare value must be written in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register,

generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be low.

The user only wants one pulse (Single mode), so '1’ must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay tDELAY.

If the user wants to output a waveform with the minimum delay, the OCxFE bit in the TIMx_CCMRx register must be set. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

14.3.16. Encoder interface mode

To select Encoder Interface mode write SMS = ‘001’ in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS = ‘010’ if it is counting on TI1 edges only and SMS = ‘011’ if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the user can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to Table 17-1. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1 = TI1 if not filtered and not inverted,TI2FP2 = TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So user must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together. In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table below summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 14-1 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
	High	Down	Up	No count	No count

Counting on TI1 only	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.

The figure below 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 IC1F = '0000' (TIMx_CCER register, TI1FP1 non inverted, TI1FP1 = TI1).
- CC2P = '0', and IC2F = '0000' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2 = TI2).
- SMS = '011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = '1' (TIMx_CR1 register, counter enabled).

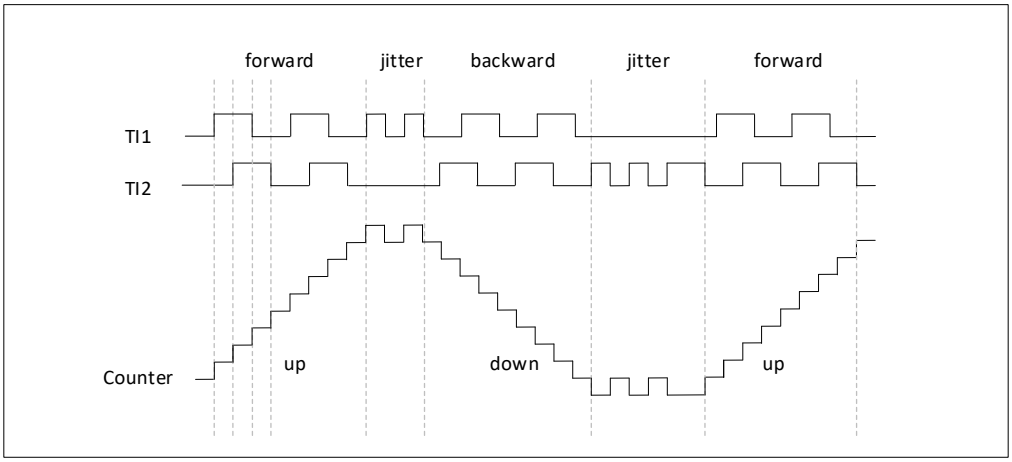


Figure 14-44 Example of counter operation in encoder interface mode.

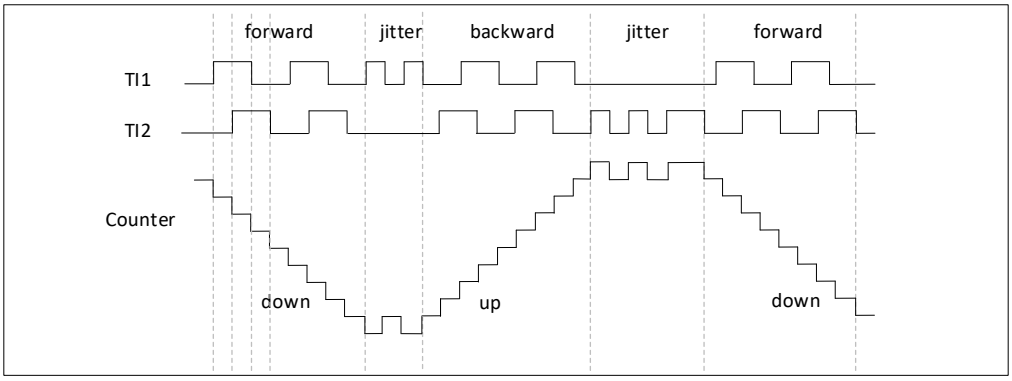


Figure 14-45 Example of encoder interface mode with TI1FP1 polarity inverted.

The timer, when configured in Encoder Interface mode provides information on the sensor's current position.The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates

the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer).

14.3.17. Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

14.3.18. Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1) to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in Figure 17-44. The “interfacing timer” captures the 3 timer input pins (TIMx_CH1, TIMx_CH2, and TIMx_CH3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode, the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (see Figure 17-27). 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: the user wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors.
- Program channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘11’. The user can also program the digital filter if needed.
- Program channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register.
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’.

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM

control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

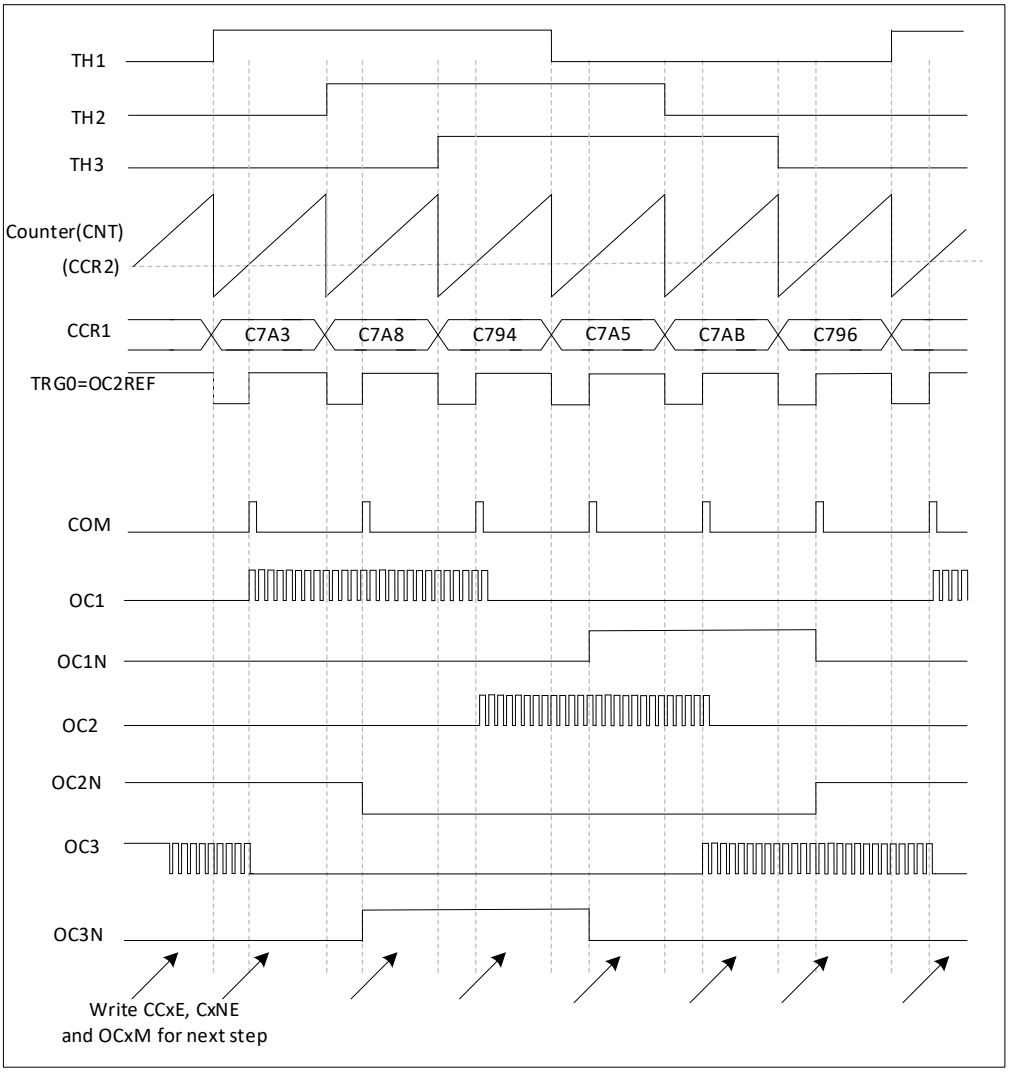


Figure 14-46 Example of Hall sensor interface

14.3.19. TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input.

Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 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 request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

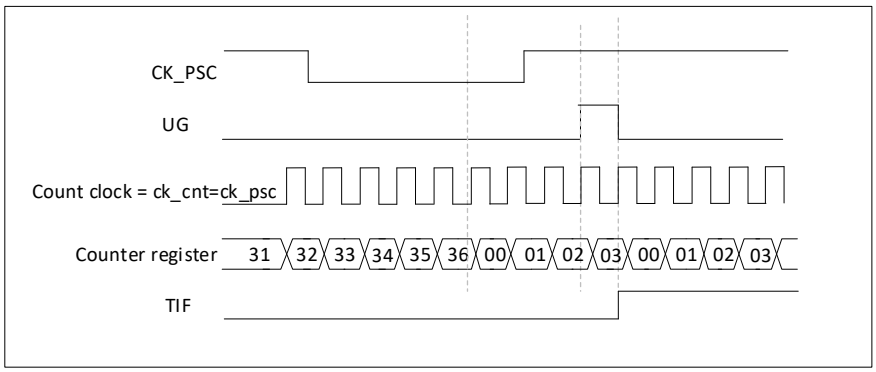


Figure 14-47 Control circuit in reset mode

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select TI1 as the input source by writing TS = 101 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter start 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.

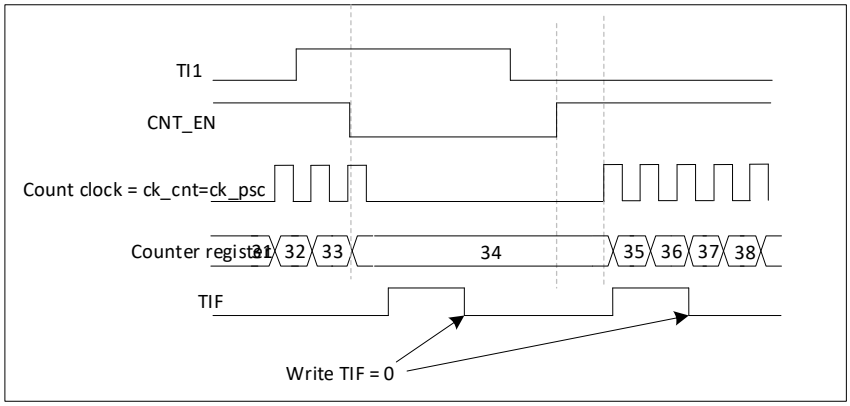


Figure 14-48 Control circuit in gated mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 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.

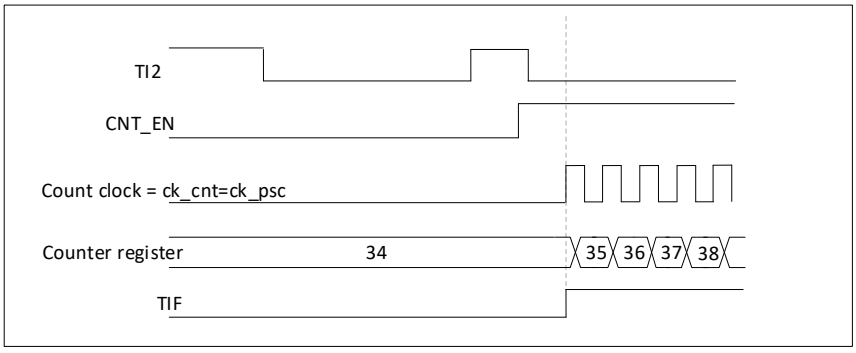


Figure 14-49 Control circuit in trigger mode

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   – ETF = 0000: no filter.

– ETPS = 00: prescaler disabled.

– ETP = 0: detection of rising edges on ETR and ECE = 1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:

– IC1F = 0000: no filter.

– The capture prescaler is not used for triggering and does not need to be configured.

– CC1S = 01 in TIMx_CCMR1 register to select only the input capture source

– CC1P = 0 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.

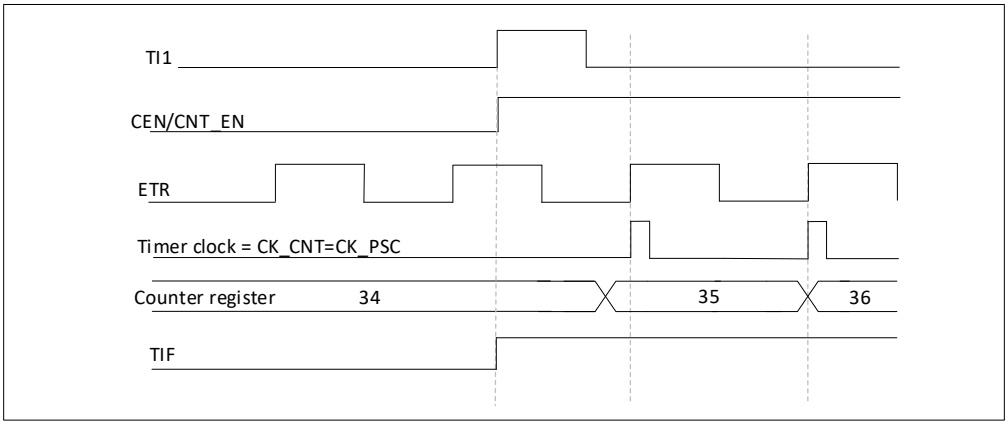


Figure 14-50 Control circuit in external clock mode 2 + trigger mode

14.3.20. Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. When a timer is in master mode, it can reset, start, stop or clock the counter of another timer in slave mode.

14.3.21. Debug mode

When the chip enters the debug mode, according to the setting of DBG_TIMx_STOP in the DBG module, the TIMx counter can continue to work normally or stop working.

14.4. TIM1 registers

14.4.1. TIM1 control register 1 (TIM1_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	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	Reserved			
9:8	CKD[1:0]	RW	00	Clock division This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS)used by the dead-time generators and the digital filters (ETR, TIx), 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not program this value
7	ARPE	RW	0	Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6:5	CMS[1:0]	RW	00	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).
4	DIR	RW	0	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.
3	OPM	RW	0	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)
2	URS	RW	0	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 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 request if enabled.
1	UDIS	RW	0	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.

0	CEN	RW	0	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.

14.4.2. 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	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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]			Res	CCUS	Res	CCPC
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
31:15	Reserved	-	0	Reserved, must be kept at reset value.
14	OIS4	RW		Output Idle state 4 (OC4 output) refer to OIS1 bit
13	OIS3N	RW	0	Output Idle state 3 (OC3N output) refer to OIS1N bit
12	OIS3	RW	0	Output Idle state 3 (OC3 output) refer to OIS1 bit
11	OIS2N	RW	0	Output Idle state 2 (OC2N output) refer to OIS1N bit
10	OIS2	RW	0	Output Idle state 2 (OC2 output) refer to OIS1 bit
9	OIS1N	RW	0	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 cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
8	OIS1	RW	0	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 cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register)
7	TI1S	RW	0	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)
6:4	MMS[2:0]	RW	000	Master mode selection These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows: 000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset. 001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

				010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer. 011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred(TRGO). 100: Compare - OC1REF signal is used as trigger output (TRGO) 101: Compare - OC2REF signal is used as trigger output (TRGO) 110: Compare - OC3REF signal is used as trigger output (TRGO) 111: Compare - OC4REF signal is used as trigger output (TRGO)
3	Res	-	0	Reserved, must be kept at reset value.
2	CCUS	RW	0	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.*
1	Res	-	0	Reserved, must be kept at reset value.
0	CCPC	RW	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 communication 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.

14.4.3. 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	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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		

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15	ETP	RW	0	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.
14	ECE	RW	0	External clock enable This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal. Note: 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111). 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

				3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.
13:12	ETPS[1:0]	RW	00	External trigger prescaler External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks. 00: Prescaler OFF 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8
11:8	ETF[3:0]	RW	0000	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 fDTS 0001: fSAMPLING = fCK_INT, N = 2 0010: fSAMPLING = fCK_INT, N = 4 0011: fSAMPLING = fCK_INT, N = 8 0100: fSAMPLING = fDTS / 2, N = 6 0101: fSAMPLING = fDTS / 2, N = 8 0110: fSAMPLING = fDTS / 4, N = 6 0111: fSAMPLING = fDTS / 4, N = 8 1000: fSAMPLING = fDTS / 8, N = 6 1001: fSAMPLING = fDTS / 8, N = 8 1010: fSAMPLING = fDTS / 16, N = 5 1011: fSAMPLING = fDTS / 16, N = 6 1100: fSAMPLING = fDTS / 16, N = 8 1101: fSAMPLING = fDTS / 32, N = 5 1110: fSAMPLING = fDTS / 32, N = 6 1111: fSAMPLING = fDTS / 32, N = 8 Note: Care must be taken that fDTS is replaced in the formula by CK_INT when ETF[3:0] = 1, 2 or 3.
7	MSM	RW	0	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.
6:4	TS[2:0]	RW	000	Trigger selection This bit-field selects the trigger input to be used to synchronize the counter. 000: Internal Trigger 0 (ITR0) 001: Reserved 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) 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.
3	OCCS	RW	0	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
2:0	SMS[2:0]	RW	000	Slave mode selection When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description. 000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock. 001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level. 010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1 level. 011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

				100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers. 101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter. Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS = '100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal. Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 14-2 TIM1 Internal trigger connection

Slave TIM	ITR0(TS = 000)	ITR1(TS = 001)	ITR2(TS = 010)	ITR3(TS = 011)
TIM1	reserved	reserved	reserved	reserved

14.4.4. TIM1 interrupt enable register (TIM1_DIER)

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	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			Reserved, must be kept at reset value.
7	BIE	RW	0	BIE: Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	TIE	RW	0	TIE: Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	COMIE	RW	0	COMIE: COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	CC4IE	RW	0	Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled 1: CC4 interrupt enabled
3	CC3IE	RW	0	CC3IE: Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled 1: CC3 interrupt enabled
2	CC2IE	RW	0	CC2IE: Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
0	UIE	RW	0	UIE: Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

14.4.5. TIM1 status register (TIM1_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	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

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	-	0	Reserved, must be kept at reset value.
12	CC4OF	Rc_w0	0	Capture/Compare 4 overcapture flag refer to CC1OF description
11	CC3OF	Rc_w0	0	Capture/Compare 3 overcapture flag refer to CC1OF description
10	CC2OF	Rc_w0	0	Capture/Compare 2 overcapture flag refer to CC1OF description
9	CC1OF	Rc_w0	0	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
8	Res	Rc_w0	0	Reserved, must be kept at reset value.
7	BIF	Rc_w0	0	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
6	TIF	Rc_w0	0	Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode.It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending
5	COMIF	Rc_w0	0	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 by writing it to '0'. 0: No COM event occurred. 1: COM interrupt pending.
4	CC4IF	Rc_w0	0	Capture/Compare 4 interrupt flag refer to CC1IF description
3	CC3IF	Rc_w0	0	Capture/Compare 3 interrupt flag refer to CC1IF description
2	CC2IF	Rc_w0	0	Capture/Compare 2 interrupt flag refer to CC1IF description
1	CC1IF	Rc_w0	0	Capture/Compare 1 interrupt flag **If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software. 0: No match. 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode) **If channel CC1 is configured as input:**

				This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred 1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)
0	UIF	Rc_w0	0	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 TIM1 slave mode control register (TIM1_SMCR)), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

14.4.6. TIM1 event generation register (TIM1_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	--	-	-	-	-	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:8	Reserved	-	0	Reserved, must be kept at reset value.
7	BG	W	0	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 transfer can occur if enabled.
6	TG	W	0	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 transfer can occur if enabled.
5	COMG	W	0	Capture/Compare control update generation This bit can be set by software, it is automatically cleared by hardware 0: No action 1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits Note: This bit acts only on channels having a complementary output.
4	CC4G	W	0	CC4G: Capture/Compare 4 generation Refer to CC1G description
3	CC3G	W	0	CC3G: Capture/Compare 3 generation Refer to CC1G description
2	CC2G	W	0	CC2G: Capture/Compare 2 generation Refer to CC1G description
1	CC1G	W	0	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 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 request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
0	UG	W	0	Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR = 1 (downcounting).

14.4.7. TIM1 capture/compare mode register 1 (TIM1_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

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	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2CE	OC2M[2:0]			OC2PE	CO2FE	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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-		Reserved, must be kept at reset value.
15	OC2CE	RW	0	Output Compare 2 clear enable
14:12	OC2M[2:0]	RW	000	Output Compare 2 mode
11	OC2PE	RW	0	Output Compare 2 preload enable
10	OC2FE	RW	0	Output Compare 2 fast enable
9:8	CC2S[1:0]	RW	00	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).
7	OC1CE	RW	0	Output Compare 1 clear enable OC1CE: Output Compare 1 Clear Enable 0: OC1Ref is not affected by the ETRF Input 1: OC1Ref is cleared as soon as a High level is detected on ETRF input
6:4	OC1M[2:0]	RW	00	Output Compare 1 mode These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits. 000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs (this mode is used to generate a timing base).

				001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1. 100: Force inactive level - OC1REF is forced low. 101: Force active level - OC1REF is forced high. 110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF = '0') as long as TIMx_CNT > TIMx_CCR1 else active (OC1REF = '1'). 111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT > TIMx_CCR1 else inactive. Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed(LOCK bits in TIMx_BDTR register) and CC1S = '00' (the channel is configured in output). 2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode. 3: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
3	OC1PE	RW	0	Output Compare 1 preload enable 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately. 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event. Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = '00' (the channel is configured in output). 2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.
2	OC1FE	RW	0	Output Compare 1 fast enable This bit is used to accelerate the effect of an event on the trigger in input on the CC output. 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.
1:0	CC1S[1:0]	RW	00	Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC1 channel is configured as output 01: CC1 channel is configured as input, IC1 is mapped on TI1 10: CC1 channel is configured as input, IC1 is mapped on TI2

				11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input Capture mode:

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-		Reserved, must be kept at reset value.
15:12	IF2F	RW	0000	Input capture 2 filter
11:10	IC2PSC[1:0]	RW	00	Input capture 2 prescaler
9:8	CC2S[1:0]	RW	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)
7:4	IC1F[3:0]	RW	0000	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 fDTS 0001: fSAMPLING = fCK_INT, N = 2 0010: fSAMPLING = fCK_INT, N = 4 0011: fSAMPLING = fCK_INT, N = 8 0100: fSAMPLING = fDTS / 2, N = 6 0101: fSAMPLING = fDTS / 2, N = 8 0110: fSAMPLING = fDTS / 4, N = 6 0111: fSAMPLING = fDTS / 4, N = 8 1000: fSAMPLING = fDTS / 8, N = 6 1001: fSAMPLING = fDTS / 8, N = 8 1010: fSAMPLING = fDTS / 16, N = 5 1011: fSAMPLING = fDTS / 16, N = 6 1100: fSAMPLING = fDTS / 16, N = 8 1101: fSAMPLING = fDTS / 32, N = 5 1110: fSAMPLING = fDTS / 32, N = 6 1111: fSAMPLING = fDTS / 32, N = 8 Note: Care must be taken that fDTS is replaced in the formula by CK_INT when ICxF[3:0] = 1, 2 or 3.
3:2	IC1PSC[1:0]	RW	00	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
1:0	CC1S[1:0]	RW	00	Capture/Compare 1 Selection This bit-field defines the direction of the channel (input/output) as well as the used input.

				00: CC1 channel is configured as output 01: CC1 channel is configured as input, IC1 is mapped on TI1 10: CC1 channel is configured as input, IC1 is mapped on TI2 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

14.4.8. TIM1 capture/compare mode register 2 (TIM1_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

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	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC4CE	OC4M[2:0]			OC4PE	CO4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-		Reserved, must be kept at reset value.
15	OC4CE	RW	0	Output compare 4 clear enable
14:12	OC4M[2:0]	RW	000	Output compare 4 mode
11	OC4PE	RW	0	Output compare 4 preload enable
10	OC4FE	RW	0	Output compare 4 fast enable
9:8	CC4S[1:0]	RW	00	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).
7	OC3CE	RW	0	Output compare 3 clear enable
6:4	OC3M[2:0]	RW	00	Output compare 3 mode
3	OC3PE	RW	0	Output compare 3 preload enable
2	OC3FE	RW	0	Output compare 3 fast enable
1:0	CC3S[1:0]	RW	00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER)

Input Capture mode:

Bit	Name	R/W	Reset Value	Function

31:16	Reserved	-		Reserved, must be kept at reset value.
15:12	IC4F	RW	0000	Input capture 4 filter
11:10	IC4PSC	RW	00	Input capture 4 prescaler
9:8	CC4S	RW	00	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)
7:4	IC3F	RW	0000	Input capture 3 filter
3:2	IC3PSC	RW	00	Input capture 3 prescaler
1:0	OC3S	RW	00	Capture/compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

14.4.9. 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	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	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

Bit	Name	R/W	Reset Value	Function
31:14	Reserved	-	0	Reserved, must be kept at reset value.
13	CC4P	RW	0	Capture/Compare 4 output polarity refer to CC1P description
12	CC4E	RW	0	Capture/Compare 4 output enable refer to CC1E description
11	CC3NP	RW	0	Capture/Compare 3 complementary output polarity refer to CC1NP description
10	CC3NE	RW	0	Capture/Compare 3 complementary output enable refer to CC1NE description
9	CC3P	RW	0	Capture/Compare 3 output polarity refer to CC1P description
8	CC3E	RW	0	Capture/Compare 3 output enable refer to CC1E description
7	CC2NP	RW	0	Capture/Compare 2 complementary output polarity refer to CC1NP description
6	CC2NE	RW	0	Capture/Compare 2 complementary output enable refer to CC1NE description
5	CC2P	RW	0	Capture/Compare 2 output polarity refer to CC1P description
4	CC2E	RW	0	Capture/Compare 2 output enable

				refer to CC1E description
3	CC1NP	RW	0	Capture/Compare 1 complementary output polarity **CC1 channel configuration as output:** 0: OC1N active high. 1: OC1N active low. **CC1 channel configuration as input:** This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description. Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bits only when a Commutation event is generated. Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = ”00” (the channel is configured in output).
2	CC1NE	RW	0	Capture/Compare 1 complementary output enable 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bits only when a Commutation event is generated.
1	CC1P	RW	0	Capture/Compare 1 output polarity **CC1 channel configured as output:** 0: OC1 active high 1: OC1 active low **CC1 channel configured as input:** CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations. 00: non-inverted/rising edge The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode). 01: inverted/falling edge The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode). 10: reserved, do not use this configuration. 11: non-inverted/both edges The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode. Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bits only when a Commutation event is generated. Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
0	CC1E	RW	0	Capture/Compare 1 output enable CC1 channel configured as output: 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits. 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits. CC1 channel configured as input:

				This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not. 0: Capture disabled. 1: Capture enabled. Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bits only when a Commutation event is generated.

Table 14-3 Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	**OSSI bit**	**OSSR bit**	**CCxE bit**	**CCxNE bit**	**OCx output state**	**OCxN output state**
1	x	0	0	1	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	1	0	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	OCxREF + Polarity OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	1	OCxREF + Polarity OCx = OCxREF xor CCxP, OCx_EN = 1	Output Disabled (not driven by the timer) OCxN = 0, OCxN_EN = 0
		1	0	0	OCREF + Polarity + dead-time OCx_EN = 1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN = 1
		1	0	1	Output Disabled (not driven by the timer) OCx = CCxP, OCx_EN = 0	Output Disabled (not driven by the timer) OCxN = CCxNP, OCxN_EN = 0
		1	1	0	Off-State (output enabled with inactive state) OCx = CCxP, OCx_EN = 1	OCxREF + Polarity OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + Polarity OCx = OCxREF xor CCxP, OCx_EN = 1	Off-State (output enabled with inactive state) OCxN = CCxNP, OCxN_EN = 1
		0	0	0	OCREF + Polarity + dead-time OCx_EN = 1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN = 1
0	0	X	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx = CCxP, OCx_EN = 0, OCxN = CCxNP, OCxN_EN = 0 Then if the clock is present: OCx = OISx and OCxN = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx = CCxP, OCx_EN = 1, OCxN = CCxNP, OCxN_EN = 1 Then if the clock is present: OCx = OISx and OCxN = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
	1		1	0		
	1		1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIOand AFIO registers.

14.4.10. 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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CNT[15:0]	RW	0	Counter value

14.4.11. TIM1 prescaler (TIM1_PSC)

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**
PSC[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	PSC[15:0]	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to fCK_PSC / (PSC[15:0] + 1). PSC contains the value to be loaded in the active pre-scaler 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”).

14.4.12. TIM1 auto-reload register (TIM1_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	ARR[15:0]	RW	0	Auto-reload value ARR is the value to be loaded in the actual auto-reload register. The counter is blocked while the auto-reload value is null.

14.4.13. TIM1 repetition counter register (TIM1_RCR)

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**	REP[7:0]							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			Reserved, must be kept at reset value.
7:0	REP[7:0]	RW	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.

14.4.14. TIM1 capture/compare register 1 (TIM1_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR1[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CCR1[15:0]	RW	0	Capture/Compare 1 value **If channel CC1 is configured as output:** CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. **If channel CC1 is configured as input:** CCR1 is the counter value transferred by the last input capture 1 event (IC1).

14.4.15. TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR2[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CCR2[15:0]	RW	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).

14.4.16. TIM1 capture/compare register 3 (TIM1_CCR3)

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**
CCR3[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CCR3[15:0]	RW	0	Capture/Compare value **If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output. **If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3).

14.4.17. TIM1 capture/compare register 4 (TIM1_CCR4)

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**
CCR4[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CCR4[15:0]	RW	0	Capture/Compare value **If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output. **If channel CC4 is configured as input:**

				CCR4 is the counter value transferred by the last input capture 4 event (IC4).

14.4.18. 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	Res	Res	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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RW	0	Reserved, must be kept at reset value.
15	MOE	RW	0	Main output enable This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output. 0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
14	AOE	RW	0	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 cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
13	BKP	RW	0	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.
12	BKE	RW	0	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.
11	OSSR	RW	0	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. 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0). 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1. Then, OC/OCN enable output signal = 1 Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
10	OSSI	RW	0	Off-state selection for Idle mode This bit is used when MOE = 0 on channels configured as outputs. 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).
9:8	LOCK[1:0]	RW	00	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 re-set. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.
7:0	DTG[7:0]	RW	0000 0000	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 tdtg with tdtg = tDTS. DTG[7:5] = 10x = > DT = (64+DTG[5:0])xtdtg with Tdtg = 2xtDTS. DTG[7:5] = 110 = > DT = (32+DTG[4:0])xtdtg with Tdtg = 8xtDTS. DTG[7:5] = 111 = > DT = (32+DTG[4:0])xtdtg with Tdtg = 16xtDTS. Example if TDTS = 125 ns (8 MHz), dead-time possible values are: 0 to 15875 ns by 125 ns steps, 16 us to 31750 ns by 250 ns steps, 32 us to 63 us by 1 us steps, 64 us to 126 us by 2 us steps Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

14.4.19. TIM1 register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM1_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x004	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			Res.	CCUS	Res.	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0		0		0
0x008	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

Offset	0x00C		0x010		0x014	0x018				0x01C				0x020		0x024
Register	TIM1_DIER	Reset value	TIM1_SR	Reset value	TIM1_EGR	TIM1_CCMR1(output compare mode)	Reset value	TIM1_CCMR1(Input Capture mode)	Reset value	TIM1_CCMR2(output capture mode)	Reset value	TIM1_CCMR2(Input Capture mode)	Reset value	TIM1_CCER	Reset value	TIM1_CNT
31	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
30	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
29	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
28	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
27	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
26	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
25	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
24	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
23	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
22	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
21	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
20	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
19	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
18	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
17	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
16	Res.		Res.		Res.	Res.		Res.		Res.		Res.		Res.		
15	Res.		Res.		Res.	OC2CE	0	IC2F[3:0]	0	OC4CE	0	IC4F[3:0]	0	Res.	0	CNT[15:0]
14	Res.		Res.		Res.	OC2M [2:0]	0 0 0		0 0 0	OC4M [2:0]	0 0 0		0 0 0	Res.	0	
13	Res.		Res.		Res.									CC4P	0	
12	Res.		Res.	0	CC4O									CC4E	0	
11	Res.		Res.	0 0 0	CC3O	OC2PE	0	IC2PSC [1:0]	0 0	OC4PE	0 0	IC4PSC [1:0]	0 0	CC3NP	0	
10	Res.		Res.	0 0 0	CC2O	CO2FE	0		0 0	CO4FE	0 0		0 0	CC3NE	0	
9	Res.		Res.	0	CC1O	CC2S [1:0]	0 0	CC2S [1:0]	0 0	CC4S [1:0]	0 0	CC4S [1:0]	0 0	CC3P	0	
8	Res.		Res.											CC3E	0	
7	BIF	0	BIF	0	BG	OC1CE	0	IC1F[3:0]	0	OC3CE	0	IC3F[3:0]	0	CC2NP	0	
6	TIF	0	TIF	0	TG	OC1M [2:0]	0 0 0		0 0 0	OC3M [2:0]	0 0 0		0 0 0	CC2NE	0	
5	COMIF	0	COMIF	0	COMG									CC2P	0	
4	CC4IF	0	CC4IF	0	CC4G									CC2E	0	
3	CC3IF	0	CC3IF	0	CC3G	OC1PE	0	IC1PSC [1:0]	0 0	OC3PE	0 0	IC3PSC [1:0]	0 0	CC1NP	0	
2	CC2IF	0	CC2IF	0	CC2G	OC1FE	0		0 0	OC3FE	0 0		0 0	CC1NE	0	
1	CC1IF	0	CC1IF	0	CC1G	CC1S [1:0]	0 0	CC1S [1:0]	0 0	CC3S [1:0]	0 0	CC3S [1:0]	0 0	CC1P	0	
0	UIE	0	UIF	0	UG		0		0		0		0	CC1E	0	

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

15. Comparator (COMP)

15.1. Introduction

Two general purpose comparators (general purpose comparators) COMP are integrated in the chip, namely COMP1 and COMP2. These two modules can be used as separate modules or combined with timer.

Comparators can be used as follows:

- Triggered by analog signal to generate low power mode wake-up function
- Analog signal conditioning
- Current control loop of Cycle by cycle when connected with PWM output from timer

15.2. COMP main features

- Each comparator has configurable positive or negative input for flexible voltage selection
  - Multiple I/O pins
  - VCC
  - Output of temperature sensor
  - Internal reference voltage and 3 fractional values (1/4, 1/2, 3/4) provided by voltage divider
- Configurable hysteresis function
- Programmable speed and power consumption
- Output can be connected to I/O or timer input as trigger
  - OCREF_CLR event (cycle by cycle current control)
  - Brake for fast PWM shutdown
- COMP1 and COMP2 can be combined into window COMP
- Each COMP has interrupt generation capability, which is used as wake-up (via EXTI) from low-power modes (sleep and stop modes)

15.3. COMP function description

15.3.1. COMP diagram

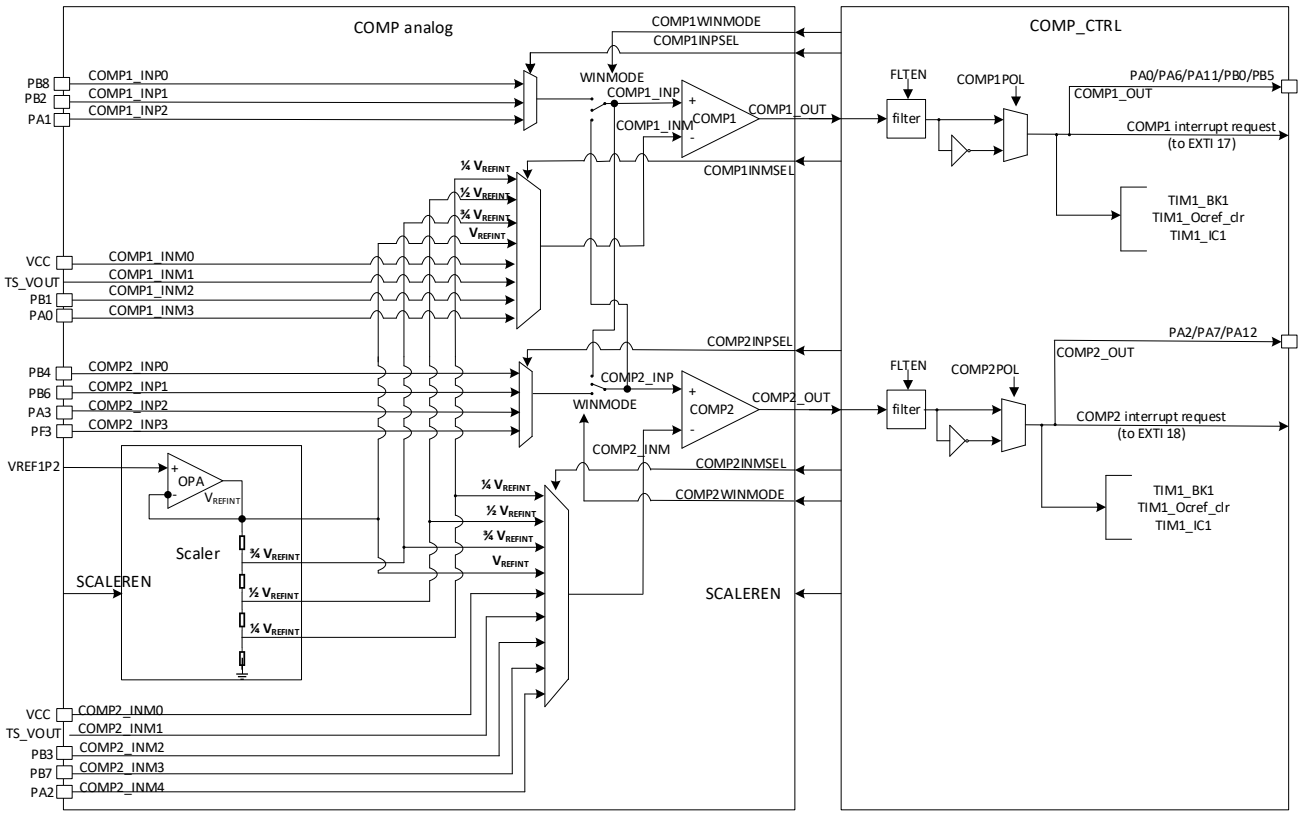


Figure 15-1 Comparator architecture block diagram

15.3.2. COMP pins and internal signals

The I/O used as comparator input must be configured in analog mode in the GPIO register.

The comparator output can be connected to the I/O pin through the alternate function channel (alternate function) on the GPIO.

The outputs can also be internally connected to the inputs of various timers for the following purposes:

- When the brake input is connected, the emergency shutdown of the PWM signal
- Cycle-by-cycle current control using OCREF_CLR input
- Input capture for timing measurements

15.3.3. COMP reset and clock

The COMP module has two clock sources:

1) PCLK (APB clock), used to provide the clock to the configuration register

2) COMP clock, used for the clock of the circuit after the analog comparator output (the latch circuit of the analog output, the glitch filter circuit, etc.), which can be selected as PCLK or LSI. When you need to work in stop mode, choose LSI.

The reset signal sources of the COMP module are:

1) The reset of the circuit after the analog comparator output (the latch circuit of the analog output, the glitch filter circuit, etc.), the reset signal includes the APB reset source and the COMP module software reset source (RCC_APBRSTR2.COMP1RST and RCC_APBRSTR2.COMP2RST)

15.3.4. COMP lock mechanism

Comparators can be used for safety purposes such as overcurrent and temperature protection. For applications with specific functional safety requirements, it is necessary to ensure that the comparator program cannot be rewritten in the event of register access errors and PC (program counter) confusion. Thus, the comparator control and status registers can be write protected (read only).

If the write to the register is complete, the COMPx Lock bit is set to 1, which makes the entire register read-only, including the COMPx Lock bit.

Write protection can only be reset by the chip's reset signal.

15.3.5. Window comparator

The role of the Window comparator is to monitor whether the analog voltage is within the low and high thresholds. A window comparator can be created using two comparators. The monitored analog voltage is connected to the non-inverting (+) inputs of both comparators at the same time, and the high and low thresholds are connected to the inverting inputs (-) of the two comparators, respectively.

By enabling the WINMODE bit, the non-inverting (+ input) of the two comparators can be connected together to save one I/O pin.

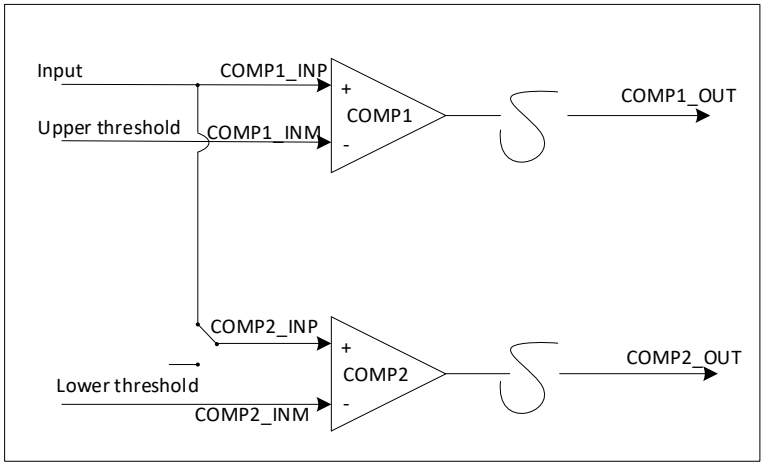


Figure 15-2 Window comparator

15.3.6. Hysteresis

To avoid spurious output transitions in noisy signal conditions, the comparator can be enabled with hysteresis (by enabling the HYST bit in COMP1_CSR, both COMP1 and COMP2 hysteresis can be turned on).

15.3.7. Power modes

The power consumption and propagation delay of the comparator can be selected from different modes by the PWRMODE[1:0] bits of the COMPx_CSR register to achieve the most suitable trade-off in a specific application. The optional modes include high speed and medium speed. Relatively speaking, the high speed mode consumes more power and has a smaller transmission delay. Note that before entering stop, if you select the PWR_CR2 register LPR = 1 (that is, choose to use the low power regulator to supply power), you need to first set COMP at Medium speed (PWRMODE = 01).

In addition, in order to reduce power consumption, APB clock and COMP clock are controlled by RCC_APBENR2.COMP1EN (and RCC_APBENR2.COMP2EN), software can only enable this register when using COMP module.

15.3.8. Comparator filtering

The output filter function of COMP and the corresponding filter width can be enabled by setting the COMP_FR register. Note that this setting should be done before COMP_EN is enabled.

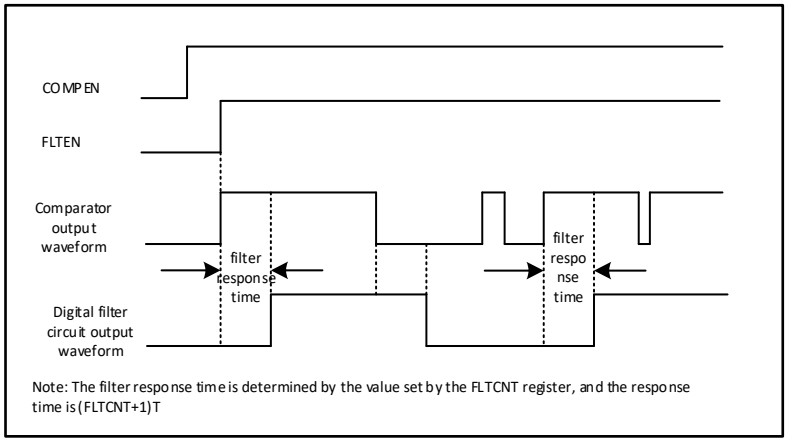


Figure 15-3 COMP filter

15.3.9. COMP interrupt

The comparator output is internally connected to the EXTI controller (extended interrupts and events). Each comparator has a separate EXTI line (17 and 18) and can generate interrupts or events. The same mechanism is used for wake-up from low power.

15.4. COMP registers

15.4.1. COMP1 control and status registers (COMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP_OUT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PWR-MODE[1:0]		Res	HYST
RW	R											RW	RW		RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO-LAR-ITY	Res	Res	Res	WINMODE	Res	INPSEL[1:0]		INMSEL[3:0]				Res	Res	SCALER_EN	COMP1_EN
RW		-	-	RW	-	RW	RW	RW	RW	RW	RW			RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP1_CSR register lock Set by software and cleared by system reset. When set, all 32 bits of the COMP1_CSR register are locked 0: Unlocked, the entire register can be read and written 1: Locked, the entire register is read-only
30	COMP_OUT	R		COMP1 output status This bit is read-only and reflects the polarity-selected output level of COMP1.
29:20	Reserved			
19:18	PWRMODE[1:0]	RW	0	COMP1 power mode selection

				Software is readable and writable, choosing the power consumption and thus the speed of the COMP1. In high speed mode, the power consumption is larger and the delay is smaller 00: High speed 01: Medium speed 10: High speed 11: High speed Note: This bit is not controlled by the LOCK function.
17	Reserved			
16	HYST	RW	0	COMP1 and COMP2 hysteresis function enable control 0: The hysteresis function is disabled 1: The Hysteresis function enabled
15	POLARITY	RW	0	COMP1 polarity selection Software readable and writable (if not locked) 0: do not reverse 1: Reverse
14:12	Reserved			
11	WINMODE	RW	0	COMP1 output selection (window mode) Software readable and writable (if not locked) 0: The signal is selected by INPSEL[1:0] 1: COMP2_INP signal of COMP2 Note that the WINMODE modes of the two COMPs cannot be enabled at the same time.
10	Reserved			
9:8	INPSEL[1:0]	RW	00	00: PB8 01: PB2 10: PA1 11: Reserved
7:4	INMSEL[3:0]	RW	0000	0000: 1/4 VREFINT 0001: 1/2 VREFINT 0010: 3/4 VREFINT 0011: VREFINT 0100: VCC 0101: TS 0110: PB1 0111: Reserved 1000: PA0 其他: 1/4 VREFINT
3:2	Reserved			
1	SCALER_EN	RW	0	The VREFINT related input is enabled. When any one of VREFINT, 3/4 VREFINT, 1/2 VREFINT, and 1/4 VREFINT is selected as the comparator input, this register bit must be turned on. 0: do not open SCALER 1: Enable SCALER
0	COMP1_EN	RW	0	COMP1 enable bit Software readable and writable (if not locked) 0: Disable 1: Enable

15.4.2. COMP1 filter register (COMP1_FR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1[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	Res	Res	Res	FLTEN1
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT1	RW	0x0	Comparator 1 Sample Filter Counter The sampling clock is APB or LSI. The filter count value is configurable. When the number of sampling times

				reaches the filter count value, the results are output consistently. Sampling count period = FLTCNT[15:0]
15:1	Reserved		0x0	
0	FLTEN1	RW	0x0	Comparator 1 digital filter function configuration 0: Disable digital filter function 1: Enable digital filter function Note: This bit must be set when COMP1_EN is 0

15.4.3. COMP2 control and status register (COMP2_CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP_OUT	Res	Res	Res	Res	Res	Res					PWR-MODE[1:0]		Res	
RW	R											RW	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PO-LAR-ITY	Res	Res	Res	WINMODE	Res	INPSEL[1:0]		INMSEL[3:0]				Res	Res	Res	COMP2_EN
RW		-	-	RW	-	RW	RW	RW	RW	RW	RW				RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP2_CSR register lock Set by software and cleared by system reset. When set, all 32 bits of the COMP2_CSR register are locked 0: Unlocked, the entire register can be read and written 1: Locked, the entire register is read-only
30	COMP_OUT	R		COMP2 output status This bit is read-only and reflects the polarity-selected output level of COMP2.
29:20	Reserved			
19:18	PWRMODE[1:0]	RW		COMP2 power mode selection Software readable and writable, power mode selected and the resulting speed of COMP2 00: High speed 01: Medium speed 10: High speed 11: High speed Note: This bit is not controlled by the LOCK function.
17:16	reserved			
15	POLARITY	RW		COMP2 polarity selection Software readable and writable (if not locked) 0: do not reverse 1: Reverse
14:12	Reserved			
11	WINMODE	RW		COMP2 non-inverting output selection (window mode) Software readable and writable (if not locked) 0: The signal is selected by INPSEL[1:0] 1: COMP1_INP signal of COMP1 Note that the WINMODE modes of the two COMPs cannot be enabled at the same time.
10	Reserved			
9:8	INPSEL[1:0]	RW		Signal selection of COMP2 non-inverting input Software readable and writable (if not locked) 00: PB4 01: PB6 10: PA3 11: PF3
7:4	INMSEL[3:0]	RW		0000: 1/4 VREFINT 0001: 3/4 VREFINT 0010: 1/2 VREFINT 0011: VREFINT

				0100: VCC 0101: TS 0110: PB3 0111: PB7 1000: PA2 > 1000: 1/4 VREFINT
3:1	Reserved			
0	COMP2_EN	RW		COMP2 enable bit Software readable and writable (if not locked) 0: Disable 1: Enable

15.4.4. COMP2 filter register (COMP2_FR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT2[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	Res	Res	Res	FLTEN2
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT2[15:0]	RW	0x0	Comparator 2 Sample Filter Counter The sampling clock is APB or LSI. The filter count value is configurable. When the number of sampling times reaches the filter count value, the results are output consistently. Sampling count period = FLTCNT[15:0]
15:1	Reserved		0x00	
0	FLTEN2	RW	0x0	Comparator 2 digital filter function configuration 0: Disable digital filter function 1: Enable digital filter function Note: This bit must be set when COMP2_EN is 0

15.4.5. COMP register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	COMP1_CSR	LOCK	COMP_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWR-MODE[1:0]		Res.	HYST	POLARITY	Res.	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	SCALER_EN	COMP1_EN
	Reset value	0	0											0	0		0	0				0		0	0	0	0	0	0			0	0
0x004	COMP1_FR	FLTCNT1[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLTEN1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																0
0x010	COMP2_CSR	LOCK	COMP_OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWR-MODE[1:0]		Res.	HYST	POLARITY	Res.	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	SCALER_EN	COMP2_EN
	Reset value	0	0											0	0		0	0				0		0	0	0	0	0	0			0	0

0x14	COMP2_FR	FLTCNT2[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLTEN2
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														0

16. General-purpose timers (TIM16)

TIM16 function exactly the same.

16.1. TIM16 main 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 65536
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Interrupt generation on the following events:
  - Update: counter overflow

16.2. TIM16 functional description

16.2.1. Time-base unit

The main block of the programmable general purpose 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 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.

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 65535. 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.

The following figures give some examples of the counter behavior when the prescaler ratio is changedy:

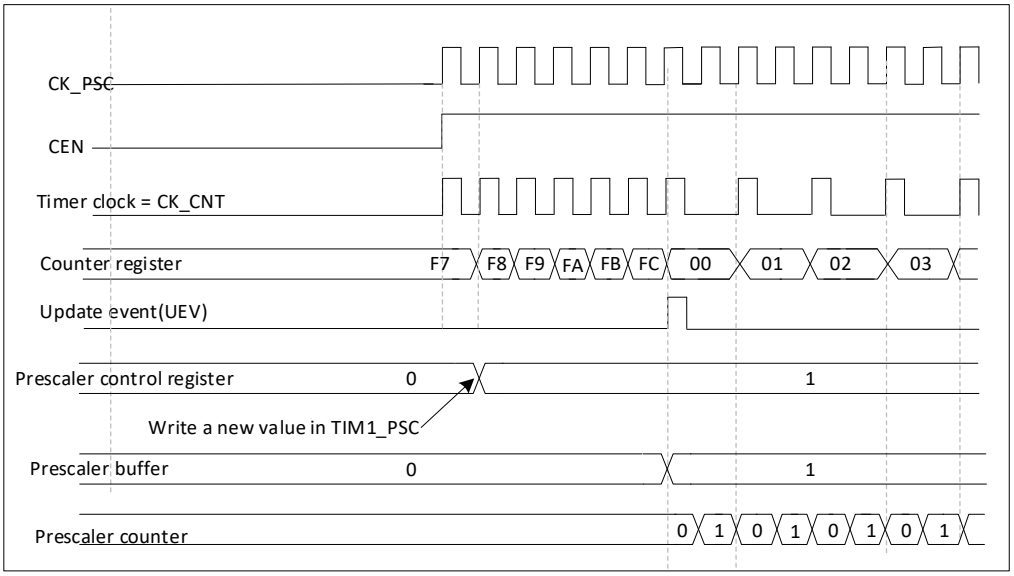


Figure 16-1 Counter timing diagram with prescaler division change from 1 to 2

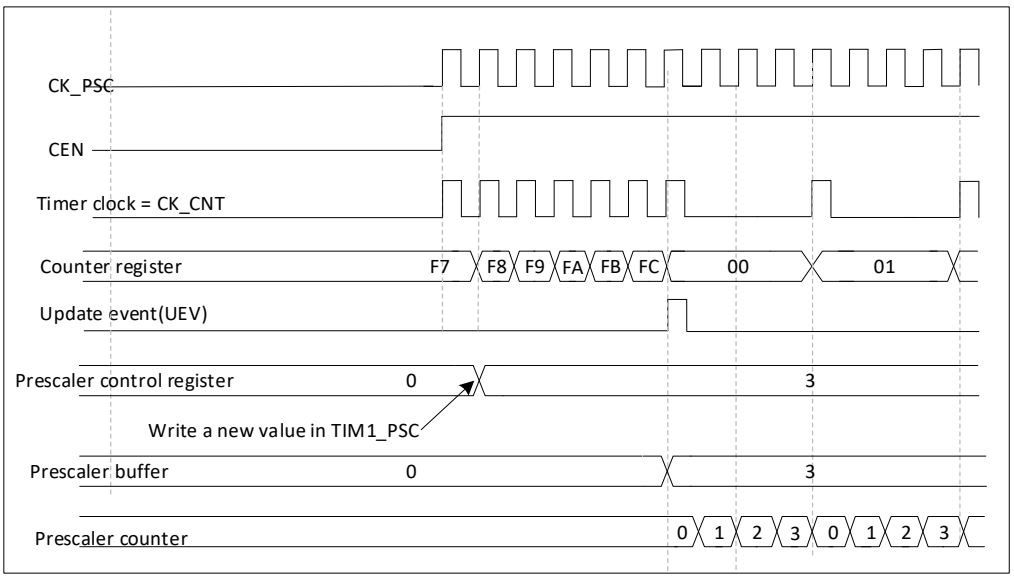


Figure 16-2 Counter timing diagram with prescaler division change from 1 to 4

16.2.2. Counter operation

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 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.

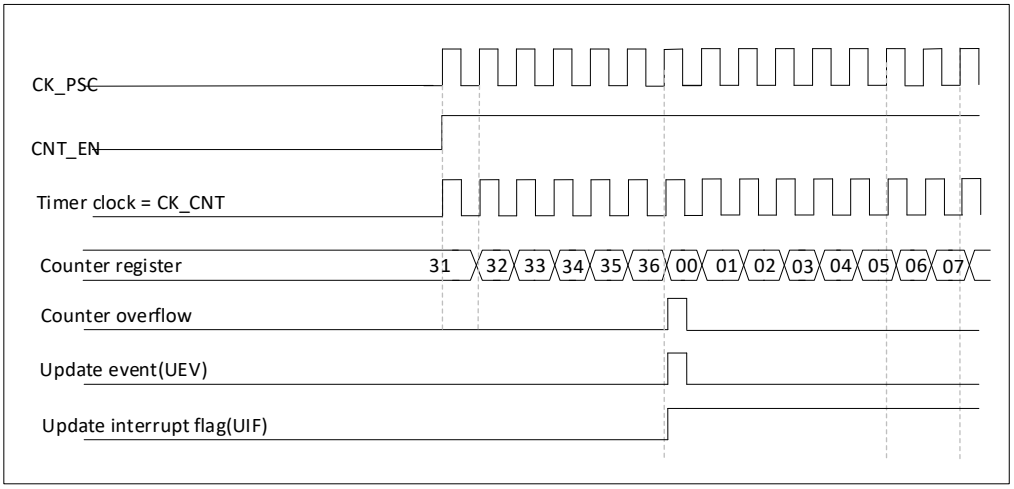


Figure 16-3 Counter timing diagram, internal clock divided by 1

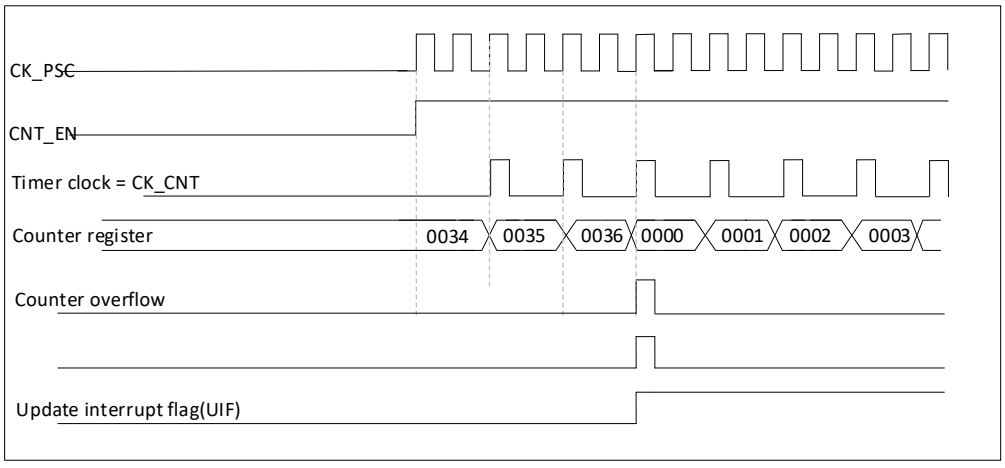


Figure 16-4 Counter timing diagram, internal clock divided by 2

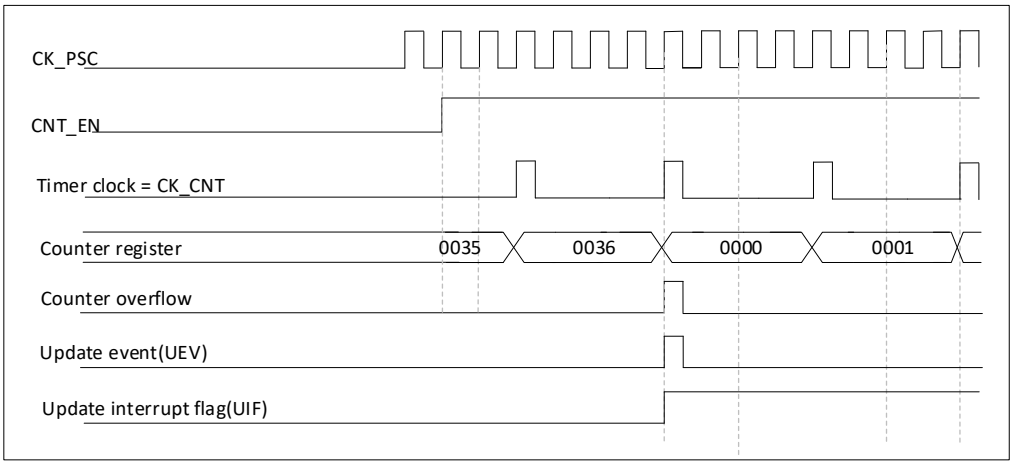


Figure 16-5 Counter timing diagram, internal clock divided by 4

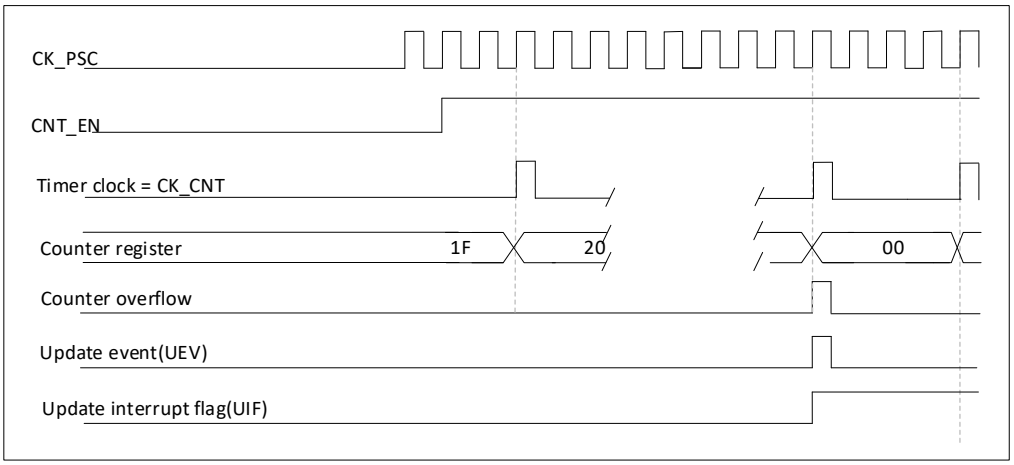


Figure 16-6 Counter timing diagram, internal clock divided by N

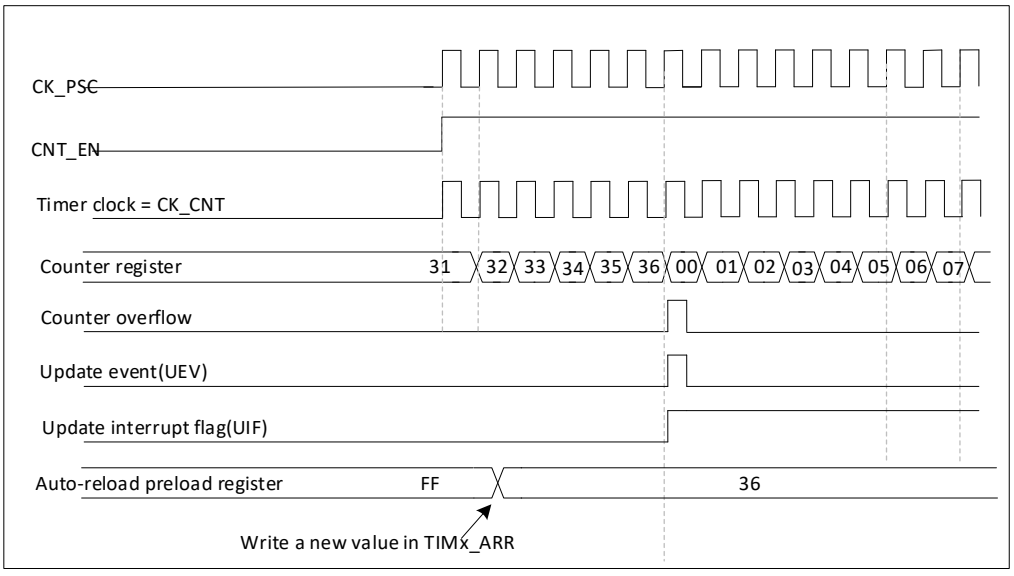


Figure 16-7 Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

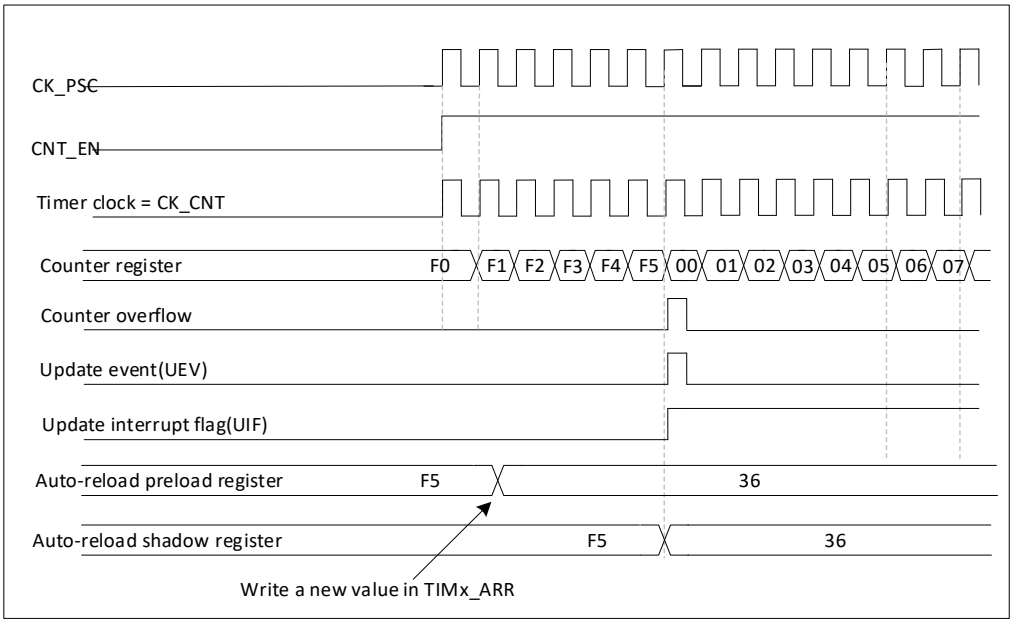


Figure 16-8 Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)

16.2.3. Repetition counter

Time-base unit describes how the update event (UEV) is generated withrespect to the counter overflows/underflows. It is actually generated only when the repetitioncounter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers(TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow in upcounting mode.

The repetition counter is an auto-reload type, the repetition rate is maintained as defined by the TIMx_RCR register value. 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.

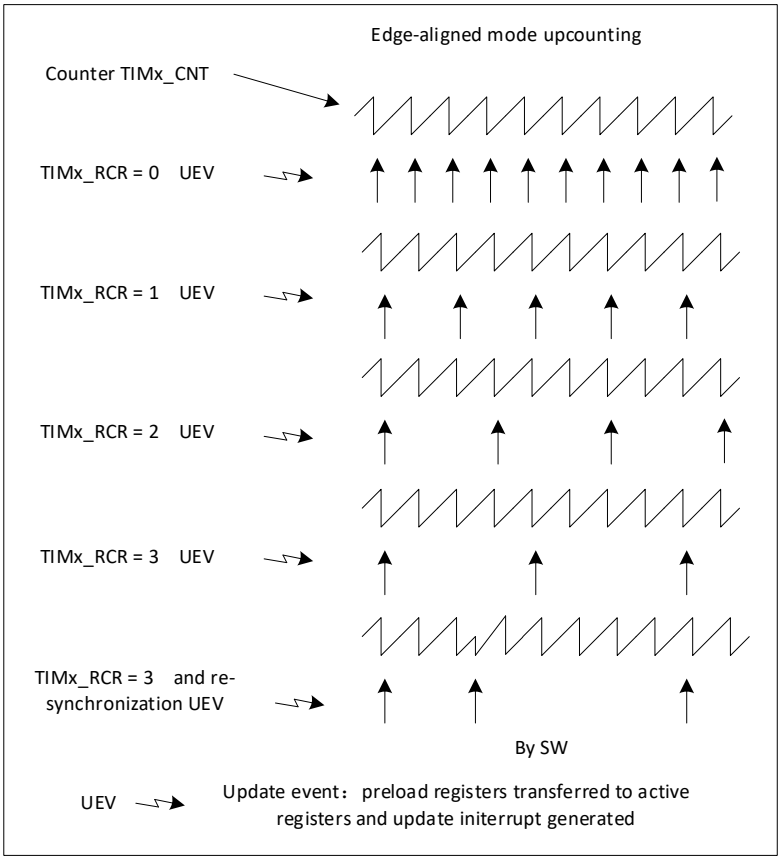


Figure 16-9 Update rate examples depending on mode and TIMx_RCR register settings

16.2.4. Clock sources

The counter clock can be provided Internal clock (CK_INT). the CEN 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.

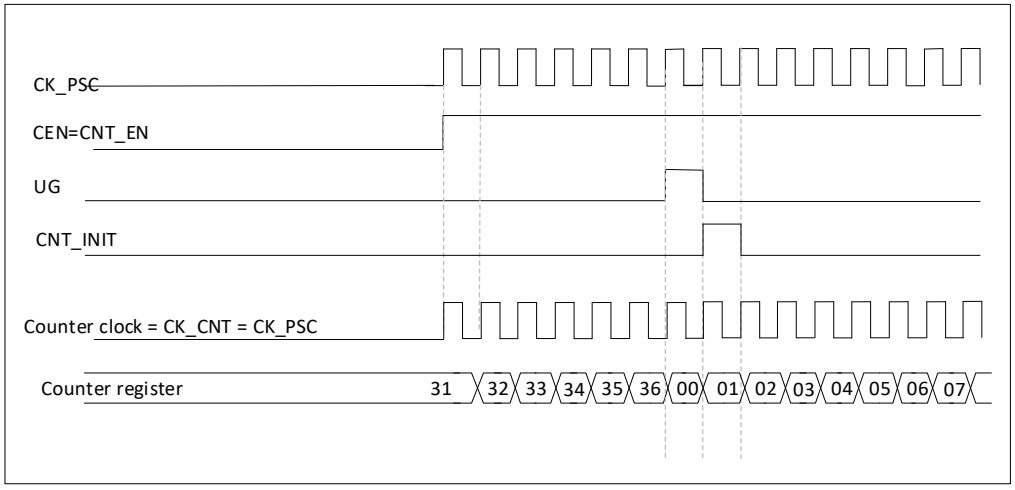


Figure 16-10 Control circuit in normal mode, internal clock divided by 1

16.3. TIM16 registers

16.3.1. TIM16 control register 1 (TIMx_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	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	CKD[1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
						RW		RW				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	-	0	Reserved, must be kept at reset value.
9:8	CKD[1:0]	RW	00	Clock division This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (tDTS)used by the dead-time generators and the digital filters (TIx), 00: tDTS = tCK_INT 01: tDTS = 2*tCK_INT 10: tDTS = 4*tCK_INT 11: Reserved, do not program this value
7	ARPE	RW	0	Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6:4	Reserved	-	0	Reserved, must be kept at reset value.
3	OPM	RW	0	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)
2	URS	RW	0	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 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 request if enabled.
1	UDIS	RW	0	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.
0	CEN	RW	0	Counter enable 0: Counter disabled 1: Counter enabled Note: External clock, gated mode and encoder mode can only work after the CEN bit is set by software. Trigger mode can automatically set the CEN bit by hardware.

16.3.2. TIM16 interrupt enable register (TIM16_DIER)

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	Res	UIE
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31:1	Reserved	-	0	Reserved, must be kept at reset value.
0	UIE	RW	0	UIE: Update interrupt enable 0: Update interrupt disabled. 1: Update interrupt enabled.

16.3.3. TIM16 status register (TIM16_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	UIF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RC_W0

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	-	0	Reserved, must be kept at reset value.
0	UIF	Rc_w0	0	Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update interrupt pending. This bit is set by hardware when the registers are updated: –At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register. –When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

16.3.4. TIM16 event generation register (TIM16_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	Reserved	-	0	Reserved, must be kept at reset value.
0	UG	W	0	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). If in center-align mode or DIR = 0 (upcounter), the counter will be cleared to 0, if DIR = 1 (downcounter), the counter will take the value of TIMx_ARR.

16.3.5. TIM16 counter (TIM16_CNT)

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**
CNT[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	CNT[15:0]	RW	0	Counter value

16.3.6. TIM16 prescaler (TIM16_PSC)

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**
PSC[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	PSC[15:0]	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to fCK_PSC / (PSC[15:0] + 1). PSC contains the value to be loaded in the active pre-scaler 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”).

16.3.7. TIM16 auto-reload register (TIM16_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			Reserved, must be kept at reset value.
15:0	ARR[15:0]	RW	0xFFFF	Auto-reload value ARR is the value to be loaded in the actual auto-reload register. The counter is blocked while the auto-reload value is null.

16.3.8. TIM16 repetition counter register (TIM16_RCR)

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	REP[7:0]							

-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	Reserved			Reserved, must be kept at reset value.
7:0	REP[7:0]	RW	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 edgealigned mode, the number of PWM half periods in center-aligned mode.

16.3.9. TIM16 register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x00C	TIM16_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																																0
0x010	TIM16_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
0x014	TIM16_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
0x024	TIM16_CNT	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
0x028	TIM16_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

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIM16_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	TIM16_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

17. Low power timer (LPTIM)

17.1. Introduction

LPTIM is a 16-bit timer. The ability of LPTIM to wake up the system from a low-power mode makes it suitable for implementing low-power applications.

LPTIM introduces a flexible clocking scheme that provides the required functionality and performance while minimizing power consumption.

17.2. LPTIM main features

- 16-bit up counter
- 3-bit prescaler with 8 possible division factors (1, 2, 4, 8, 16, 32, 64, 128)
- Optional clock
  - Internal clock source: LSI or APB clock
- 16-bit ARR reload register
- Single mode

17.3. LPTIM functional description

17.3.1. LPTIM block diagram

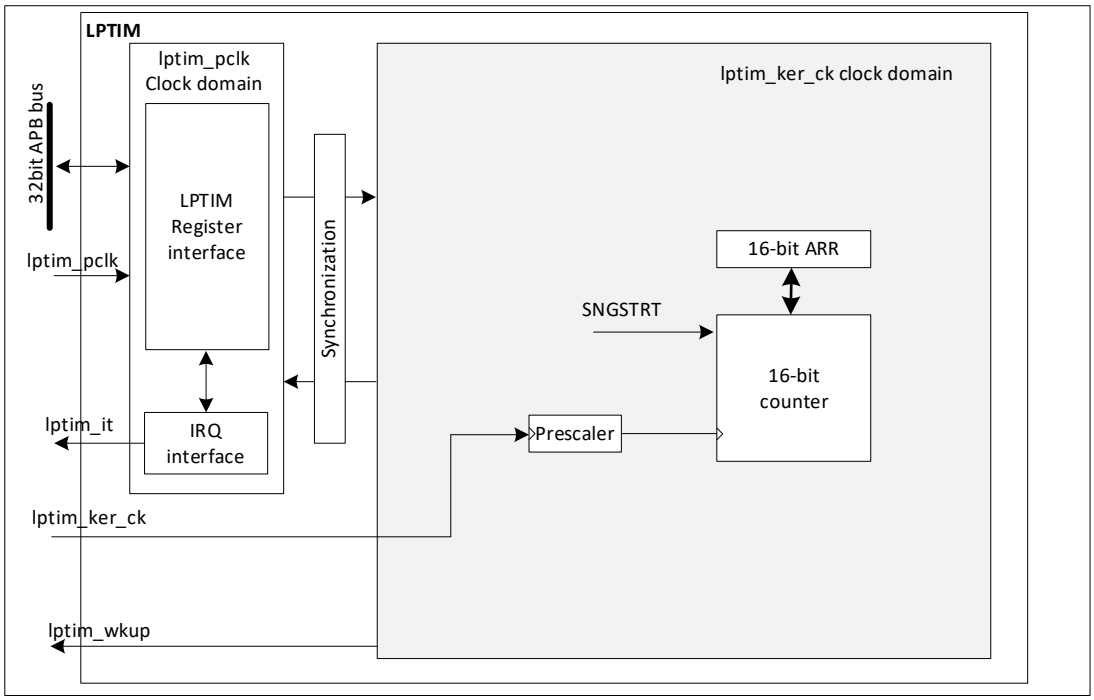


Figure 17-1 LPTIM block diagram

17.3.2. LPTIM reset and clock

LPTIM can use multiple clock sources for count.

Through the RCC module, it can be clocked with an internal clock signal (the clock signal can be selected among APB, LSI sources).

17.3.3. Prescaler

LPTIM 16-bit counter driven by a configurable 2 power prescaler control. The prescaler division ratio is controlled by PRESC[2:0].

The following table lists all cases:

Table 17-1 prescale factor

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

17.3.4. Operating mode

LPTIM has only one use timer mode.

- **Single mode**: The timer starts from a trigger event and stops when the ARR value is reached.

To enable single counting, the SNGSTRT bit must be set.

A new trigger event will restart the timer. Any trigger events after the counter has started and before the ARR is reached will be ignored.

17.3.5. Register update

The PRELOAD bit controls how the LPTIM_ARR register is updated:

- When the PRELOAD bit is reset to '0': The LPTIM_ARR register is updated immediately after any write access.
- When the PRELOAD bit is set to '1': If the timer has already started, LPTIM_ARR will be updated at the end of the current period.

The LPTIM APB interface and the LPTIM Kernel logic use different clocks, so there is a certain delay when the APB is written and the written value is applied to the counter comparator. During this delay period, any additional writes to these registers must be avoided.

17.3.6. Enable timer

The ENABLE bit in the LPTIM_CR register is used to enable/disable the LPTIM core logic. After setting the ENABLE bit, a delay of two counter clocks is required to enable LPTIM.

The LPTIM_CFGR and LPTIM_IER registers can only be modified when LPTIM is disabled.

17.3.7. Counter reset INDANG

In order to reset the contents of the LPTIM_CNT register, a reset mechanism is provided:

Asynchronous reset mechanism:

Asynchronous reset is controlled by the RSTARE bit in the LPTIM_CR register. When this bit is set to 1, any access to the LPTIM_CNT register resets its contents to zero.

Note that in order to reliably read the LPTIM_CNT register, two read accesses must be performed and the results are compared. If the results are consistent, the read value is considered to be reliable.

Notice:

- When asynchronous reset is enabled, the first read will reset the LPTIM_CNT, the second read can read the count result of the LPTIM_CNT register.
- When the LPTIM count clock selects PCLK/HSI, the read value cannot be guaranteed to be reliable even if it is accessed twice in a row.

17.3.8. Debug mode

When the microcontroller enters debug mode, the LPTIM counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

17.4. LPTIM low power mode

Table 17-2 The difference between different low power modes of LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	No effect when LPTIM is clocked by LSI. LPTIM interrupts cause the device to exit Stop.

17.5. LPTIM interrupt

The following events will generate interrupt/wake-up events if they are enabled in the LPTIM_IER register:

- Auto-reload match

Note: If the corresponding bit in the LPTIM_IER register (interrupt enable register) is set to 1 after the corresponding flag in the LPTIM_ISR register (status register) is set to 1, no interrupt will be generated.

Interrupt event	Description
Auto-reload match	When the content of the counter register (LPTIM_CNT) matches the content of the auto-reload register (LPTIM_ARR), the interrupt flag is set

17.6. LPTIM register

17.6.1. LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	-	0	
1	ARRM	R	0	Auto-reload match

				ARRM is set by hardware to inform the application that the LPTIM_CNT register value matches the LPTIM_ARR register value. Writing a 1 to the ARRMCF bit of the LPTIM_ICR register clears the ARRM flag.
0	Reserved			

17.6.2. LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	ARRMCF	Res
														w	

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	-	0	
1	ARRMCF	RW	0	Auto-reload match clear flag Writing a 1 to this bit clears the ARRM flag in the LPTIM_ISR register
0	Reserved			

17.6.3. LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	ARRMIE	Res
														RW	

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	-	0	
1	ARRMIE	RW	0	Auto-reload match interrupt enable 0: ARRM interrupt disabled 1: ARRM interrupt enabled
0	Reserved			

17.6.4. LPTIM configuration register (LPTIM_CFGR)

Address offset:0x00C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:23	Reserved	-	0	
22	PRELOAD	RW	0	Register update mode The preload bit controls the LPTIM_ARR register update mode 0: Update registers after each APB bus write access 1: Registers are updated at the end of the current LPTIM period
21:12	Reserved			
11:9	PRESC[2:0]	RW	0	clock prescaler The PRESC bits configure the prescaler division factor. It can be a factor in the following divisions: 000:/1 001:/2 010:/4 011:/8 100:/16 101:/32 110:/64 111:/128
8:0	Reserved	-	0	Reserved, must be kept at reset value.

17.6.5. LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	RSTARE	Res	Res	SNGSTRT	ENABLE
											RW			RW	RW

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	-	0	
4	RSTARE	RW	0	Reset enable after read This bit is set and cleared by software. When RSTARE is set to '1', any read access to the LPTIM_CNT register will asynchronously reset the LPTIM_CNT register contents.
3:2	Reserved	-	0	
1	SNGSTRT	RW	0	LPTIM starts single mode. This bit is set by software and cleared by hardware. Setting this bit will start LPTIM in single-pulse mode. Note: This bit can only be set when LPTIM is enabled. It will be reset automatically by hardware.
0	ENABLE	RW	0	LPTIM enable bit, set and cleared by software 0: LPTIM disabled 1: LPTIM enabled

17.6.6. LPTIM auto-reload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	0	Reserved, must be kept at reset value.
15:0	ARR	RW	0x0001	Auto-reload value

				ARR is the auto-reload value of LPTIM This register can only be updated when LPTIM is enabled

17.6.7. LPTIM counter (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	0	
15: 0	CNT	R	0	counter value When LPTIM is running on an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two values returned are the same. A read access can be considered reliable when the values of two consecutive read accesses are equal.

17.6.8. LPTIM register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRM	Res.
	Reset value																															0	
0x04	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRMC	Res.
	Reset value																															0	
0x08	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARRMI	Res.
	Reset value																															0	
0x0C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESC [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0											0	0	0									
0x10	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	Res.	Res.	SNGSTR	ENABLE
	Reset																												0			0	0

	value																														
0x18	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]													
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	LPTIM_CNT	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

18. Independent watchdog (IWDG)

18.1. Introduction

Independent watchdog is integrated in the chip, and this module has the characteristics of high-security level, accurate timing and flexible use. IWDG finds and resolves functional malfunctions due to software failure and triggers system reset when the counter reaches the specified timeout value.

The IWDG is clocked by LSI and thus stay active even if the main clock fails.

The IWDG is the best suited to applications that require the watchdog to run as a totally independent process outside the main application, without having high timing accuracy constraints.

18.2. IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset when the downcounter value of 0x000 is reached

18.3. IWDG functional description

18.3.1. IWDG block diagram

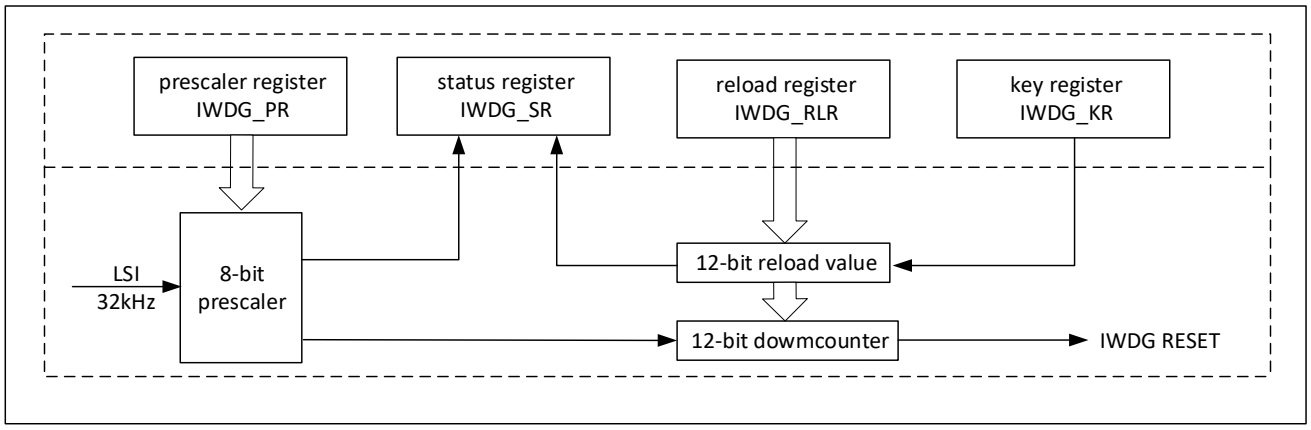


Figure 18-1 IWDG block diagram

When the independent watchdog is started by writing the value 0x0000 CCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 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_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once it starts running, IWDG cannot be stopped.

18.3.2. Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the Key register is written by the software before the counter reaches end of count.

18.3.3. Register access protection

Write access to the IWDG_PR and IWDG_RLR registers is protected. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

18.3.4. Debug mode

This function can only be used if the system supports DBG_MCU.

When the microcontroller enters debug mode, the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module.

18.4. IWDG registers

18.4.1. Key register (IWDG_KR)

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**
KEY[15:0]															
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	-	Reserved
15:0	KEY[15:0]	W	0x00	Key value. 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 and IWDG_RLR registers. Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

18.4.2. Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:3	Reserved	RES	-	Reserved
2:0	PR[2:0]	RW	0	Prescaler divider. They are select the prescaler divider feeding the counter clock by set this register. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

				000: divider /4 001: divider /8 010: divider /16 011: divider /32 100: divider /64 101: divider /128 110: divider /256 111: divider /256

18.4.3. Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	-	Reserved
11:0	RL[11:0]	RW	0	IWDG counter reload value. RL value will be loaded in the counter each time when the value 0xAAAA is writtern in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

18.4.4. Status register (IWDG_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	Res	Res	Res	Res	RVU	PVU
														R	R

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RES	-	Reserved
1	RVU	R	0	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.
0	PVU	R	0	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.

Note: It is mandatory to wait until RVU bit is reset before changing the IWDG_RLR, to wait until PVU bit is reset before changing the IWDG_PR. However, after updating the IWDG_PR and/or the IWDG_RLR, it is not necessary to wait until IWDG_SR.PVU or IWDG_SR.RVU is reset before continuing code execution

18.4.5. IWDG register map

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

0x00	IWDG_KR																																								
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]						
																																						0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]															
																										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVU	PVU
																																						0	0	0

19. Inter-integrated circuit (I2C) interface

19.1. Introduction

The I2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

19.2. I2C main features

- Slave and master modes
- Multimaster capability: Can be master or slave
- Support different communication speeds
  - Standard-mode: up to 100 kHz
  - Fast-mode: up to 400 kHz
- As Master
  - issue clock
  - issue Start & Stop clock
- As slave
  - Programmable I2C address detection
  - Stop bit discovery
- 7-bit addressing mode
- General call
- Status flag bit
  - Transmit/receive mode flag bit
  - Byte transfer completion flag bit
  - I2C busy flag bit
- error flag bit
  - Host Arbitration Lost
  - ACK failure after address/data transfer
  - Start/Stop error
  - Overrun/Underrun(Clock stretch function disabled)
- Optional clock stretching
- Software reset
- Analog noise filter function
- Configurable PEC (packet error checking) generation and verification
  - PEC value can be sent in the last bytes in Tx mode
  - The last byte does PEC error checking

19.3. I2C functional description

19.3.1. I2C block diagram

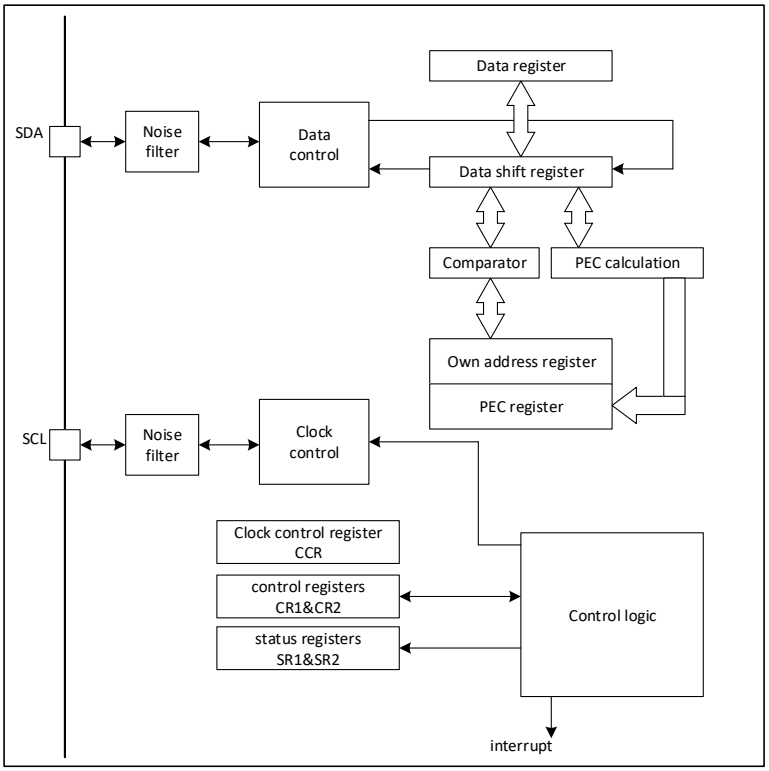


Figure 19-1 I2C block diagram

19.3.2. 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.

19.3.2.1. 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.

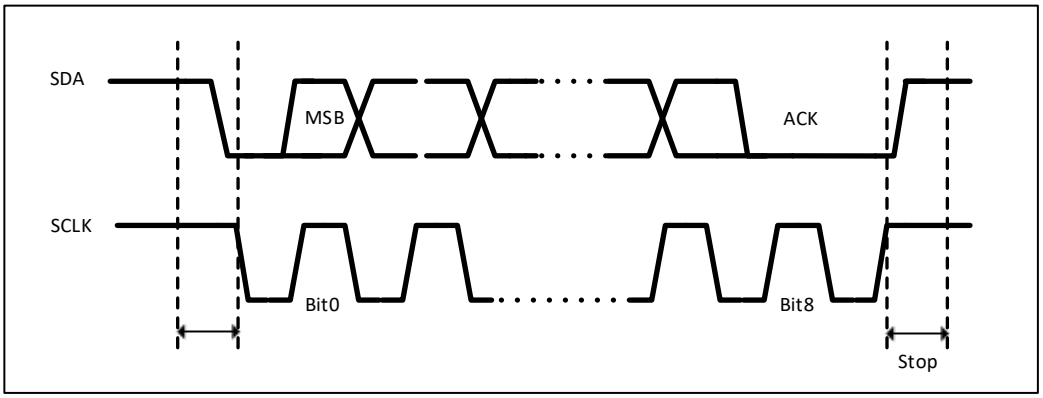


Figure 19-2 I2C bus protocol

Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

19.3.3. I2C initialization

19.3.3.1. Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled the bit of I2C_EN in the RCC_APBENR1 register. Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

19.3.3.2. 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 I2C_CCR and I2C_TRISE register.

19.3.4. I2C slave mode

By default, the I2C interface always works in slave mode. To switch from slave mode to master mode, a start condition needs to be generated.

In order to generate the correct timing, the input clock to this module must be programmed in the I2C_CR2 register. The frequency of the input clock must be at least:

Standard mode: 2 MHz

Fast mode: 4 MHz

If start condition is detected,the address received on the SDA line is sent to the shift register and is combined with the chip's address OAR1 or general The call address (if ENGC = 1) is compared.

Header or address mismatch:

The I2C interface ignores it and waits for another start condition.

Address match:

The I2C interface generates the following timings:

- ● If ACK is set to '1' by software, an acknowledge pulse is generated.
- ● The ADDR bit is set by hardware, and if the ITEVTEN bit is set, an interrupt is generated.

In slave mode, the TRA bit indicates that it is currently in receiver mode or transmitter mode.

19.3.4.1. Slave transmitter

After receiving the address and clearing the ADDR bit, (if the least significant bit of the address byte is 1) the Slave sends the data (byte) from the DR register to the SDA by the internal shift register.

Slave pulls SCL low until the ADDR bit is cleared and the data to be transmitted has been written to the DR register.

When an acknowledge pulse is received: The TxE bit is set by hardware and an interrupt is generated if the ITEVTEN and ITBUFEN bits are set.

If the TxE bit is set, but no new data is written to the I2C_DR register before the end of the next data transmission, the BTF bit is set. Slave pulls SCL low until the BTF bit is cleared by software (after reading _SR1, then writing to the I2C_DR register).

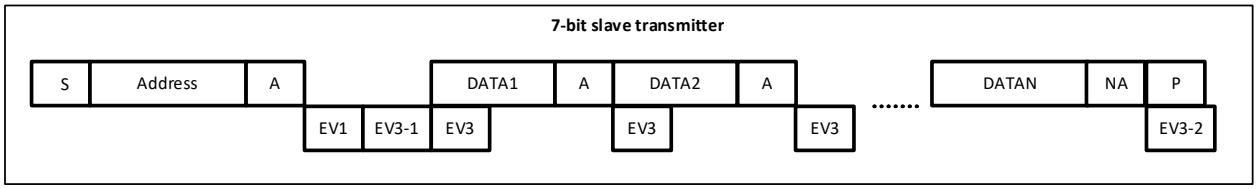


Figure 19-3 Transmission sequence diagram from the sender

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (interrupt when ITEVFEN = 1)

EV1: ADDR = 1, by first reading the SR1 register, then reading the SR2 register to clear the ADDR bit

EV3-1: TxE = 1, shift register empty, data register empty, write Data1 to DR register

EV3: TxE = 1, shift register is not empty, data register is empty, write to DR register (Data2) to clear TxE

EV3-2: AF = 1, software write 0 to AF bit to clear this bit

19.3.4.2. Slave receiver

After receiving the address and clearing ADDR, (if the least significant bit of the address byte is 0) the slave will store the byte received from the SDA line into the DR register by the internal shift register. The I2C interface performs the following actions after each byte is received:

- If the ACK bit is set, an acknowledge pulse is generated
- The hardware sets RxNE = 1. An interrupt is generated if the ITEVTEN and ITBUFEN bits are set.

If RxNE is set and the DR register is not read before the end of receiving new data, the BTF bit is set, and the slave keeps pulling SCL low until the BTF is cleared (the I2C_DR register is read after I2C_SR1). (See below).

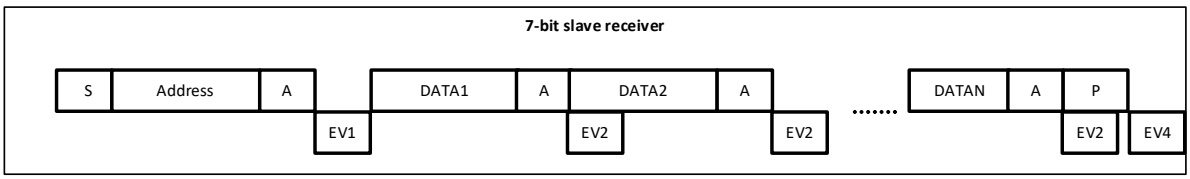


Figure 19-4 Slave receiver transmission sequence diagram

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledged, EVx = Event(with interrupt if ITEVFEN = 1)

EV1: ADDR = 1, by reading SR1 first, then SR2, the ADDR is cleared

EV2: RxNE = 1, read the DR register to clear this bit

EV4: STOPF = 1, clear this bit by first reading the SR1 register and then writing the CR1 register.

Note:

1) EV1 event pulls down SCL until the end of the corresponding software sequence.

2) EV2 software sequence must be completed before the current byte transfer is completed.

3) After checking the contents of the SR1 register, the user should perform a complete clearing sequence for each flag set that is found. For example, the ADDR and STOPF flags need to use the following sequence:
If ADDR = 1, read SR first, then read SR2,if STOPF = 1, read SR1 first, and then write CR1.
The purpose of this is to ensure that if both ADDR and STOPF are found to be set, they can both be cleared.

19.3.4.3. Close communication

After the last data byte is transmitted, the master generates a stop condition. When the slave detects this condition:

● The hardware sets STOPF, and if the ITEVTEN bit is set, an interrupt is generated.

By first reading SR1 and then writing CR1, the STOPF bit is cleared. (See EV4 in the figure above)

19.3.5. I2C master mode

In Master mode, the I2C interface starts data transfer and generates a clock signal. Serial data transfers always begin with a START condition and end with a STOP condition.
When a START condition is generated on the bus via the START bit, the device enters master mode.
The following is the sequence of operations required for master mode:

- Set the input clock to the module in the I2C_CR2 register to generate the correct timing
- Configure the clock control register
- Configure the rise time register
- Program the I2C_CR1 register to start the peripheral
- Set the START bit in the I2C_CR1 register to 1 to generate a start condition

  The input clock frequency to the I2C module must be at least:
- In standard mode: 2 MHz
- In fast mode: 4 MHz

19.3.5.1. The host generates clock

The CCR register generates high and low levels of SCL with rising or falling edges. Since the slave may stretch the SCL signal, after the rising edge of SCL occurs, the master checks the SCL signal from the bus when the time programmed in the TRISE register arrives.
— If SCL is low, it means that the slave is stretching the SCL bus, and the high-level counter stops counting until SCL is detected high. This is to ensure a minimum high time for the SCL parameter.
— If SCL is high, the high-level counter keeps counting.
In fact, even if the slave does not stretch SCK, it will take some time for such a feedback loop to occur from the rising edge of SCL to the detection of the rising edge of SCL. The time of this loop is related to the rise time of SCL (VIH data detection of SCL), plus the analog noise filtering of the SCL input path, and the SCL synchronization inside the chip due to the use of the APB clock. The maximum time for the feedback loop is programmed in the TRISE register, so the frequency of SCL remains stable regardless of the rise time of SCL.

19.3.5.2. Start condition

When BUSY = 0, set START = 1, the I2C interface will generate a Start condition and switch to master mode (MSL is set).
Note: Setting the START bit in master mode will generate a ReStart condition by hardware after the current byte is transmitted.

if Start condition is issued:

● The SB bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set.

The master reads the SR1 register, and then writes the slave address to the DR register. (Transfer sequence EV5)

19.3.5.3. Slave address sending

The slave address is sent to the SDA line through the internal shift register.

■ ● In 7-bit address mode, send out an address byte.

If the address byte is sent out

- The ADDR bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set.

Then the Master reads the SR1 register, followed by the SR2 register.

According to the lowest bit of the sent slave address, the master decides to enter the transmitter mode or the receiver mode.

■ ● In 7-bit address mode

- To enter transmitter mode, the master device sets the least significant bit to '0' when sending the slave address.
- To enter receiver mode, the master device sends the slave address with the least significant bit set to '1'.

The TRA bit indicates whether the master is in receiver mode or transmitter mode.

19.3.5.4. Master transmitter

After sending the address and clearing the ADDR bit, the master device sends the byte from the DR register to the SDA line through the internal shift register.

The Master waits until the first data byte is written to the DR register (see EV8_1).

When an ACK pulse is received, the TxE bit is set by hardware and an interrupt is generated if the INEVFEN and ITBUFEN bits are set.

If TxE is set and no new data byte is written to the DR register before the end of the last data transmission, BTF is set by hardware. The I2C interface will keep SCL low until the BTF is cleared (after reading I2C_SR1, then writing the I2C_DR register). Closing Communication.

After writing the last byte in the DR register, a STOP condition is generated by setting the STOP bit (see EV8_2 in Figure), then the I2C interface will automatically return to slave mode (MLS bit cleared).

Note: When the TxE or BTF bit is set, the stop condition should be scheduled on the EV8_2 event.

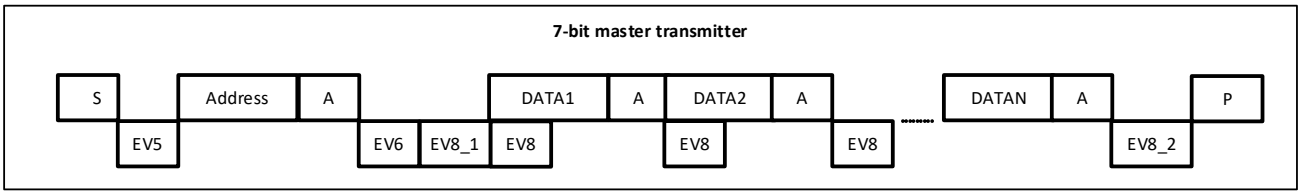


Figure 19-5 Master transmitter transmission sequence diagram

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, EVx = Event(with interrupt if ITEVFEN = 1)

EV5: SB = 1, by reading SR1, and then writing data to the DR register, the bit is cleared

EV6: ADDR = 1, by reading SR1, and then reading SR2, the bit is cleared

EV8_1: TxE = 1, shift register is empty, data register is empty, write Data1 to DR register

EV8: TxE = 1, shift register is not empty, data register is empty, write Data2 to DR register, this bit is cleared

EV8_2: TxE = 1, BTF = 1, Write the Stop bit register, when the hardware sends the Stop bit, TxE and BTF are cleared

Note:

1. EV5, EV6, EV8_1 and EV8_2 events, stretch the low level of SCL until the corresponding software sequence execution ends.
2. EV8 software The sequence must complete before the current byte is sent. If the EV8 software sequence cannot be completed before the end of the currently transmitted byte, it is recommended to use BTF instead of TxE, which has the disadvantage of slowing down the communication.

19.3.5.5. Master receiver

After sending the address and clearing the ADDR, the I2C interface enters the master receiver mode. In this mode, the I2C interface receives data bytes from the SDA line and sends them to the DR register through the internal shift register. After each byte, the I2C interface performs the following operations in sequence:

- ● If the ACK bit is set, issue an acknowledge pulse.
- ● The hardware sets RxNE = 1, if the INEVFEN and ITBUFEN bits are set, an interrupt will be generated

If the RxNE bit is set and the data in the DR register is not read before the end of receiving new data, the hardware will set BTF = 1, and the I2C interface will keep SCL low before clearing BTF, after reading I2C_SR1 Reading the I2C_DR register again will clear the BTF bit.

Close communication:

Method 1: The application scenario of this method is: when I2C is set as the highest priority interrupt in the application

The Master sends a NACK after receiving the last byte from the Slave. After receiving the NACK, the Slave releases control of the SCL and SDA lines. The Master can then send a Stop/Restart condition.

1) In order to generate a NACK pulse after the last byte is received, the ACK bit must be cleared after reading the second-to-last data byte (after the second-to-last RxNE event).

2) To generate a STOP/RESTART condition, software must set the STOP/START bit after reading the second-to-last data byte (after the second-to-last RxNE event).

3) When a single byte is received, the closing acknowledge and stop conditions should be generated just after EV6 (EV6_1, after clearing ADDR). After a stop condition is generated, the I2C interface automatically returns to slave mode (MSL bit is cleared).

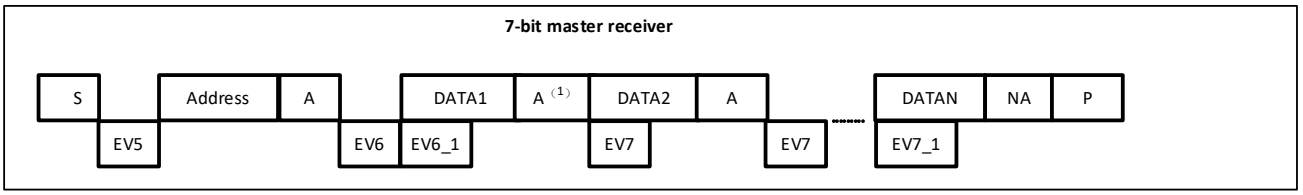


Figure 19-6 Method 1: Timing when master mode transmits

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, EVx = Event(with interrupt if ITEVFEN = 1)

EV5: SB = 1, read SR1, then write DR register, this bit is cleared

EV6:ADDR = 1, read SR1, then read SR2, this bit is cleared

EV6_1: No related flag event, only used for 1 byte reception.

EV7: RxNE = 1, read DR register, this bit is cleared

EV7_1: RxNE = 1, read DR register, write ACK = 0 and set STOP

1) If a single byte is received, the above marked as (1) The place will be NA

2) EV5, EV6 events, stretch the low level of SCL until the corresponding software sequence execution ends

3) EV7 software sequence must be executed before the current byte is sent. In EV7, the software sequence cannot be managed until the currently transferred byte has been transferred. It is recommended to use BTF instead of RXNE, which has the disadvantage of slowing down communication.

4) The software sequence of EV6_1 or EV7_1 must be completed before the ACK of the current byte transmission.

Method 2: The application scenario of this method is: the I2C interrupt is not the highest priority in the application, or the query method.

Use this method, DataN-2 is not read, so after DataN-1, the communication is stretched (RxNE and BTF is set). Then, before reading DataN-2 of the DR register, clear the ACK bit to ensure that it is cleared before DataN ACKs. After this, after reading DataN-2, set the STOP/START bit and read DataN-1. After RxNE is set, read DataN

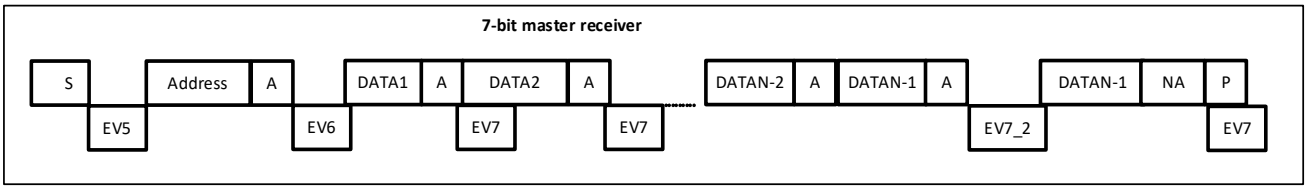


Figure 19-7 Method 2: Timing for master mode transmission when N > 2

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, EVx = Event(with interrupt if ITEVFEN = 1)

EV5: SB = 1, first read SR1 register, then write DR register, clear this bit

EV6: ADDR, first read SR1, then read SR2, clear this bit

EV7: RxNE = 1, read DR register to clear this bit

EV7_2: BTF = 1, DataN-2 is stored in DR register, DataN-1 is stored in shift register, write ACK = 0, read DataN-2 in the DR register. Set STOP, read DataN-1

Note:

1) EV5, EV6 events, stretch the low level of SCL until the corresponding software sequence execution ends

2) EV7 software sequence must be executed before the completion of the current byte transmission. In EV7, the software sequence cannot be managed until the currently transferred byte has been transferred. It is recommended to use BTF instead of RXNE, which has the disadvantage of slowing down communication.

- **When 3 bytes are to be read:**

— RxNE = 1 => Nothing (DataN-2 not read).

— DataN-1 received

— BTF = 1, shift and data registers are full: DR register stores DataN-2, shift register stores DataN-1 => SCL is pulled low: there is no other data to be received on the bus

— Clear the ACK bit

— Read DataN-2 in DR register => This will start the reception of DataN in shift register

— DataN reception complete (with a NACK)

— Write START or STOP bit

— Read DataN-1

— RxNE = 1

— Read DataN

The above process is a description for N > 2. The reception of 1 byte and 2 bytes uses different processing methods, see the following description:

■ ● **In the case of 2 bytes reception**

— Set the POS and ACK bits

— Wait for ADDR to be set

— Clear the ADDR bit

— Clear the ACK bit

— Wait for BTF to be set

— Write STOP bit

— Read DR twice

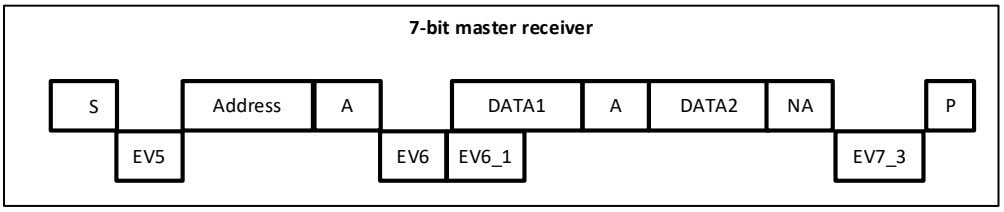


Figure 19-8 Method 2: timing when master mode transmits when N = 2

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, EVx = Event(with interrupt if ITEVFEN = 1)

EV5: SB = 1, first read SR1 register, then write DR register, clear this bit

EV6: ADDR = 1, first read the SR1 register, then read the SR2 register, clear the ADDR bit

EV6_1: no related flag events. After EV6, that is, after the address is cleared, ACK should be cleared

EV7_3: BTF = 1, write STOP = 1, then read DR twice (Data1 and Data2)

Note: 1) EV5, EV6 events, stretch SCL 2) The software sequence of EV6_1 must be completed before the ACK of the current byte transmission

■ ● **In the case of single byte reception**

— In ADDR event, clear the ACK bit

— Clear ADDR

— Write STOP or START bit

— Read data after RxNE flag is set

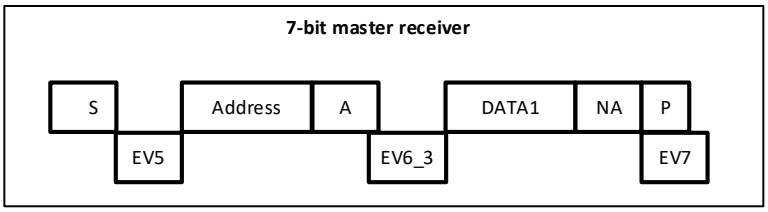


Figure 19-9 Method 2: timing when master mode transmits when N = 1

Legend: S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, EVx = Event(with interrupt if ITEVFEN = 1)

EV5: SB = 1, first read SR1 register, then write DR register, clear this bit

EV6_3: ADDR = 1, write ACK = 0. First read the SR1 register, then read the SR2 register, clear the ADDR bit. After ADDR is cleared, write STOP = 1

EV7: RxNE = 1, read DR register to clear this bit

Note:

EV5 event will stretch the low level of SCL until the corresponding software sequence execution ends.

19.3.6. Error stage

19.3.6.1. Bus error(BERR)

A bus error is generated when the I2C interface detects an external stop or start condition during an address or data byte transfer. at this time:

- ● The BERR bit is set to '1', if the ITERREN bit is set, an interrupt is generated
- ● In slave mode: the data is discarded, the hardware releases the bus:

— If it is a wrong Start condition, the slave considers a Restart and waits for an address or a stop condition

— If it is a wrong Stop condition, the slave operates according to the normal stop condition, and the hardware releases the bus at the same time

- ● In master mode: the hardware does not release the bus, and does not affect the current transmission status. At this point it is up to the software to decide whether to abort the current transfer.

19.3.6.2. ACK Failed(AF)

An acknowledge error occurs when the interface detects a no acknowledge bit. at this time:

- ● The AF bit is set, and an interrupt is generated if the ITERREN bit is set
- ● When the transmitter receives a NACK, the communication must be reset:

— If it is in slave mode, the hardware releases the bus.

— If in master mode, software must generate a stop condition or repeated start.

19.3.6.3. Arbitration loss (ARLO)

Arbitration loss error occurs when I2C interface detects arbitration loss, at this time:

- ● The ARLO bit is set by hardware and an interrupt is generated if the ITERREN bit is set
- ● The I2C interface automatically returns to slave mode (MSL bit is cleared). When the I2C interface loses arbitration, it cannot respond to its slave address in the same transfer, but it can respond after the master that wins the bus sends a repeated start condition

● Hardware release bus

19.3.6.4. Overrun/Underrun error (OVR)

In slave mode, if the clock extension is disabled and the I2C interface is receiving data, when it has received a byte (RxNE = 1), but the previous byte data in the DR register has not been read, an overrun occurs. mistake. at this time:

- ● The last received data is discarded
- ● On overrun error, software should clear the RxNE bit and the transmitter should resend the last transmitted byte

In slave mode, if the clock extension is disabled and the I2C interface is sending data, before the clock of the next byte arrives, the new data has not been written to the DR register (TxE = 1), an underrun error occurs. at this time:

- ● The previous byte in the DR register will be repeated
- ● The user should make sure that when an underrun error occurs, the receiver should discard the repeatedly received data. The transmitter should update the DR register at the specified time according to the I2C bus standard

When sending the first byte, the DR register must be written to after clearing ADDR and before the first rising edge of SCL, if this is not possible, the receiver should discard the first data.

19.3.7. SDA/SCL control

- ● If clock stretching is allowed:

— Transmitter mode: If TxE = 1 and BTF = 1: The I2C interface keeps the clock line low before transmission, waiting for software to read SR1, and then write the data into the data register (DR and shift registers are both empty).

— Receiver mode: if RxNE = 1 and BTF = 1: I2C interface keeps clock line low after receiving data byte, waiting for software to read SR1, then read data register DR (DR and shift registers are both full).

- ● If clock stretching is disabled in slave mode:

— If RxNE = 1, the DR has not been read before the next byte is received, an overun occurs. The last byte received is lost.

— If TxE = 1, an underrun occurs when no new data is written into the DR before the next byte must be sent. The same bytes will be sent repeatedly.

— Hardware does not implement control of write collisions.

19.4. I2C interrupts

Table 19-1 I2C interrupt requests

Interrupt event	Event flag	Interrupt enable control bit
START has been sent (Master)	SB	ITEVTEN
Address has been sent(Master) / Address matched(Slave)	ADDR	
Stop detection interrupt flag(Slave)	STOPF	
Transfer Complete Reload	BTF	
Receive buffer not empty	RxNE	ITEVTEN and ITBUFEN
Transmit buffer empty	TxE	
Bus error	BERR	ITERREN
Arbitration loss(Master)	ARLO	
ACK Failed	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	

19.5. I2C registers

Registers can be accessed half-word or word.

19.5.1. I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res	Res	Res	POS	ACK	STOP	START	NOSTRETCH	ENGC	Res	Res	Res	Res	Res	PE
RW				RW	RW	RW	RW	RW	RW						RW

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	Software reset. When set, I2C is in reset state. Before the reset release, make sure that the I2C pins are released and the bus is in an idle state. 0: I2C module is not in reset state 1: I2C module is in reset state Note: This bit can be used to reinitialize I2C in error or locked state. If the BUSY bit is 1, no stop condition is detected on the bus.
14:12	Reserved	RES	-	Reserved
11	POS	RW	0	ACK/PEC position (for data reception), this register can be set/cleared by software, or cleared by hardware when PE = 0. 0: The ACK bit controls the (N)ACK of the byte currently being received in the shift register. The PEC bit indicates that the byte in the current shift register is PEC 1: The ACK bit controls the (N)ACK of the next byte received in the shift register. The PEC bit indicates that the next byte received in the shift register is a PEC Note: The POS bit can only be used in a 2-byte receive configuration and must be configured before receiving data. In order to NACK the 2nd byte, the ACK bit must be cleared after clearing ADDR. In order to detect the PEC of the second byte, the PEC bit must be set during the ADDR stretch event after the POS bit is configured.
10	ACK	RW	0	acknowledgment enable. This register can be set/cleared by software, or cleared by hardware when PE = 0. 0: no response returned 1: Return an acknowledgment after a byte has been received. (matching address or data)
9	STOP	RW	0	When a stop condition occurs, software can set/clear this register, or when a stop condition is detected, it is cleared by hardware, when a timeout error is detected, hardware is set. In Master Mode: 0: No stop generation 1: Generate a stop condition after the current byte transfer or after the current start condition is issued In Slave Mode: 0: No stop condition is generated 1: Release the SCL and SDA lines after the current byte transfer
8	START	RW	0	The starting condition is generated. This register can be set/cleared by software, or cleared by hardware when a START condition is issued or when PE = 0. Master mode: 0: No start generation 1: Restart/Start condition Slave Mode: 0: No start generation

				1: When the bus is idle, generate a start generation (and automatically switch to Master Mode by hardware)
7	NOSTRETCH	RW	0	Clock stretching (Slave) is disabled. When the ADDR or BTF flag is set, this bit is used by the slave to disable clock stretching until reset by software. 0: allow clock stretching 1: Disable clock stretching
6	ENGC	RW	0	General call enabled. 0: Disable general call. Respond to address 00h with NACK 1: Allow general calls. Responds to address 00h with an ACK
5:1	Reserved	RES	-	Reserved
0	PE	RW	0	I2C module enabled. 0: Disable 1: I2C enable Note: If a communication is in progress when this bit is cleared, after the current communication ends, the I2C module is disabled and returns to the idle state. Since PE = 0 after the communication ends, all bits are cleared. In master mode, this bit must never be cleared until the communication has ended.

Note: When the STOP/START/PEC bit is set, software should not perform any write operations to I2C_CR1 until the hardware clears this bit, otheRWise, the STOP/START/PEC bit may be set a second time.

19.5.2. I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	IT-BUFEN	ITEV-TEN	ITER-REN	Res	Res	FREQ[5:0]					
					RW	RW	RW			RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:11	Reserved	RES	-	Reserved
10	ITBUFEN	RW	0	Buffer interrupt enable. 0: No interrupt is generated when TxE = 1 or RxNE = 1 1: When TxE = 1 or RxNE = 1, an event interrupt is generated
9	ITEVTEN	RW	0	Event interrupt enable. 0: Disable 1: Enable event interrupt This interrupt will be generated under thefollowing conditions: ● SB = 1 (main mode), ● ADDR = 1 (Master/Slave mode) ● STOPF = 1 (slave mode) ● BTF = 1, but no TxE or RxNE events ● If ITBUFFEN = 1, TxE event is 1 ● If ITBUFEN = 1, the RxNE event is 1
8	ITERREN	RW	0	Error interrupt enable. 0: Disable error interrupt, 1: Enable error interrupt, This interrupt will be generated under the following conditions: ● BERR = 1 ● ARLO = 1 ● AF = 1 ● OVR = 1 ● PECERR = 1

7:6	Reserved	RES	-	Reserved
5:0	FREQ	RW	0	I2C module clock frequency. This register must be configured with the value of the APB clock frequency to generate data setup and hold times that are compatible with the I2C protocol. The minimum allowable frequency that can be set is 4 MHz (standard mode, ie 100 k), 12 MHz (400 k), and the maximum frequency is the highest APB clock frequency of the chip. 000000: Forbidden 000001: Forbidden 000100: 4 MHz … 100100: 36 MHz … 110000: 48 MHz Greater than 100100: Forbidden.

19.5.3. I2C own address register 1 (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000

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

Bit	Name	R/W	Reset Value	Function
14:8	Reserved	RES	-	Reserved
7:1	ADD[7:1]	RW	0	Bit 7:1 of address
0	reserved			

19.5.4. I2C data register (I2C_DR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DR[7:0]							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
7:0	DR[7:0]	RW	0	8-bit data register, two independent buffers inside the chip share an address, which are used to store the received data (RX_DR) and place the data to be sent to the bus (TX_DR). Transmitter Mode: Data transfer is automatically started when a byte is written to the DR register (actually written to TX_DR). Once the transmission starts (TxE = 1), if the next data to be transmitted can be written into the DR register in time, the I2C module will maintain a continuous data flow. Receiver Mode: The received byte is copied to the DR register (actually RX_DR) (RxNE = 1). Read out the data register before receiving the next byte (RxNE = 1) to realize continuous data reception. Note: 1) In slave mode, the address will not be copied into the data register DR 2) The hardware does not handle write conflicts (if TxE = 0, the data register can still be written) 3) If the ARLO event occurs while processing the ACK pulse, the received byte will not be copied to the data register, so it cannot be read

19.5.5. I2C stage register 1 (I2C_SR1)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res	STOPF	Res	BTF	ADDR	SB
			RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	R	R		R		R	R	R

Bit	Name	R/W	Reset Value	Function
15:13	Reserved	RES	-	Reserved
12	PECERR	RC_W0	0	A PEC error occurred while receiving. 0: No PEC error, return ACK after receiving PEC (if ACK = 1) 1: There is a PEC error, and NACK is returned after receiving PEC (regardless of the value of ACK) This bit is cleared by software by writing 0, or by hardware when PE = 0.
11	OVR	RC_W0	0	Overload/Underload flag. 0: no overload/underload, 1: Overload/underload occurred. When NOSTRETCH = 1, this bit is set by hardware in slave mode, In the receive mode, when a new byte is received (including the ACK response pulse), and the contents of the data register have not been read out, the newly received byte will be lost. In transmit mode when a new byte is to be sent, but no new data is written to the data register, the same byte will be sent twice. This bit is cleared by software by writing 0, or by hardware when PE = 0. Note: If a write to the data register occurs very close to the rising edge of SCL, the transmitted data is indeterminate and a hold time error occurs.
10	AF	RC_W0	0	Reply failure flag. 0: no response failed, 1: Reply failed. Hardware will set this register when no acknowledgement is returned. This bit is cleared by software by writing 0, or by hardware when PE = 0.
9	ARLO	RC_W0	0	Arbitration lost (main mode). 0: no arbitration loss is detected. 1: Arbitration loss detected. This register is set by hardware when the interface loses control of the bus to another host. This bit is cleared by software by writing 0, or by hardware when PE = 0. After an ARLO event, the I2C interface automatically switches back to slave mode (M/SL = 0).
8	BERR	RC_W0	0	Bus error flag. 0: No start or stop condition error. 1: Error in start or stop condition. This bit is set by hardware when the interface detects a false start or stop condition. This bit is cleared by software by writing 0, or by hardware when PE = 0.
7	TxE	R	0	The data register is empty (when transmitting) flag. 0: The data register is not empty. 1: The data register is empty.

				When sending data, this bit is set to 1 when the data register is empty, and this bit is not set during the sending address phase. This bit can be cleared by software writing data to the DR register, or automatically by hardware after a START or STOP condition occurs, or when PE = 0. This bit is not set if a NACK is received, or if the next byte to be sent is PEC (PEC = 1). Note: After writing the first data to be sent, or writing data when BTF is set, the TxE bit cannot be cleared, because the data register is empty at this time.
6	RxNE	R	0	Data register not empty (on reception) flag. 0: The data register is empty. 1: The data register is not empty. On reception, this register is set when the data register is not empty. During the receive address phase, this register is not set. This register is cleared by software reads and writes to the data register, or by hardware when PE = 0. Note: When BTF is set, reading data does not clear the RxNE bit because the data register is still full.
5	Reserved	RES	-	Reserved
4	STOPF	R	0	Stop condition detection bit (slave mode). 0: No stop bit detected. 1: Stop condition detected. After an acknowledgment (if ACK = 1), the hardware sets this bit to 1 when the slave device detects a stop condition on the bus. After the software reads the I2C_SR1 register, a write to the I2C_CR1 register will clear this bit, or when PE = 0, the hardware will clear this bit. Note: The STOPF bit is not set after a NACK is received.
3	Reserved			
2	BTF	R	0	End of byte transfer flag. 0: Byte transfer not complete 1: Byte transfer ended successfully The hardware will set this register in the following cases (when slave mode, NOSTRETCH = 0, master mode, regardless of NOSTRETCH): — On reception, when a new byte is received (including the ACK pulse) and the data register has not been read (RxNE = 1). — When transmitting, when a new data should be transmitted and the data register has not been written with new data (TxE = 1). This bit is cleared by a read or write to the data register after software reads the I2C_SR1 register, or by hardware after sending a start or stop condition, or when PE = 0. Note: After receiving a NACK, the BTF bit is not set.
1	ADDR	R	0	Address has been sent (Master mode)/Address matched (Slave mode). After the software reads the I2C_SR1 register, reading the I2C_SR2 register will clear this bit, when PE = 0, it will be cleared by hardware. Address matching (Slave): 0: The address does not match or the address was not received, 1: The received address matches. Hardware will set this bit when the received slave address matches the OAR register or general call address. Note: In slave mode, it is recommended to perform a complete clearing sequence, that is, after

				ADDR is set, read the SR1 register first, and then read the SR2 register. Address has been sent (Master): 0: Address sending has not ended, 1: Address sending ends. For a 7-bit address, it is set when the ACK byte is received. Note: This register will not be set after receiving a NACK.
0	SB	R	0	Start bit flag (master mode). 0: start condition not sent, 1: The start condition has been sent, — This register is set when a START condition is sent. — After the software reads the I2C_SR1 register, a write to the data register will clear this bit, or when PE = 0, it will be cleared by hardware.

19.5.6. I2C stage registor 2 (I2C_SR2)

Address offset: 0x18

Reset value: 0x0000

Note: Even if the ADDR flag is set after reading the I2C_SR1 register, reading the I2C_SR2 register after reading I2C_SR1 will clear the ADDR flag. Therefore, the I2C_SR2 register must be read only when the ADDR bit of the I2C_SR1 register is found to be set or the STOPF bit is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								Res	Res	Res	GEN-CALL	Res	TRA	BUSY	MSL
R	R	R	R	R	R	R	R				R		R	R	R

Bit	Name	R/W	Reset Value	Function
15:7	Reserved			
4	GENCALL	R	0	General call address (slave mode). 0: No broadcast call address received, 1: When ENGC = 1, the address of the general call is received. Hardware clears this register when a STOP condition or a repeated START condition occurs, or when PE = 0.
3	Reserved	RES	-	Reserved
2	TRA	R	0	send/receive flag. 0: data received 1: Data has been sent At the end of the entire address transfer phase, this register is set according to the R/W bit of the address byte. Hardware clears this register when a STOP condition is detected (STOPF = 1), or a repeated-START condition, or bus arbitration is lost (ARLO = 1), or when PE = 0.
1	BUSY	R	0	Bus busy flag. 0: No data communication on the bus 1: Polarity data communication is in progress on the bus Set by hardware when SDA or SCL is detected low. Hardware clears when a stop condition is detected. This register indicates the current bus communication in progress, this information is still updated when the interface is disabled (PE = 0).
0	MSL	R	0	Master-slave mode. 0: slave 1: master

				— Set by hardware when the interface is in master mode (SB = 1), — Hardware cleared when a stop condition is detected on the bus (STOPF = 1), arbitration lost (ARLO = 1), or when PE = 0.

19.5.7. I2C clock control register (I2C_CCR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Res	Res	CCR[11:0]											
RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	I2C Master mode selection. 0: Standard mode 1: Fast Mode
14	DUTY	RW	0	Duty cycle in fast mode. 0: In fast mode: $T_{low}/T_{high} = 2$ 1: In fast mode: $T_{low}/T_{high} = 16/9$
13:12	Reserved	RES	-	Reserved
11:0	CCR[11:0]	RW	0	Clock control division factor in fast/standard mode (master mode). This division factor is used to set the SCL clock in master mode. ● Standard Mode: ✓ $T_{high} = CCR \times Tpclk$ ✓ $T_{low} = CCR \times Tpclk$ ● Express Mode: ✓ DUTY = 0: $T_{high} = CCR \times Tpclk$ $Tlow = 2 \times CCR \times Tpclk$ ✓ DUTY = 1 (to reach 400KHz): $T_{high} = 9 \times CCR \times Tpclk$ $Tlow = 16 \times CCR \times Tpclk$ Note: 1. The minimum allowed setting is 0x04, and the minimum allowed in fast DUTY mode is 0x01 2. $T_{high} = t_{r(SCL)} + t_{w(SCLH)}$ 3. $T_{low} = t_{r(SCL)} + t_{w(SCLL)}$ 4. These delays have no filter 5. This register can only be configured when PE = 0, 6. f_{CK} should be an integer multiple of 10 MHz, so that the fast 400 kHz can be correctly generated at which

19.5.8. I2C TRISE register (I2C_TRISE)

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	TRISE[5:0]					
										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:6	Reserved	RES	-	Reserved
5:0	TRISE	RW	0	Maximum rise time in fast/standard mode (master mode). These bits should provide the maximum duration of the S CL feedback loop in master mode. The purpose of this is to maintain a stable frequency

				of SCL regardless of the duration of the rising edge of SCL. These bits must be set to the maximum SCL rise time given in the I2C bus specification in 1 increments. For example: the maximum allowable SCL rise time in standard mode is 1000ns. If the value in FREQ [5:0] in the I2C_CR2 register is equal to 0x08, Tpclk = 125ns, then TRISE is configured as 0x09 (1000ns/125ns = 8 + 1 = 9). Filter values can also be added to TRISE. If the result is not an integer, the integer part is written to TRISE to ensure the t_{HIGH} parameter. Note: This register can only be set when PE = 0.

19.5.9. I2C register map

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	SWRST	Res.	Res.	Res.	POS	ACK	STOP	START	NOS-TRETCH	ENGC	ENPEC	Res.	Res.	Res.	Res.	PE
	Reset value	0				0	0	0	0	0	0	0					0
0x04	I2C_CR2	Res.	Res.	Res.	LAST	Res.	IT-BUFEN	ITEV-TEN	ITER-REN	Res.	Res.	FREQ[5:0]					
	Reset value				0		0	0	0			0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD[7:1]							Res.
	Reset value									0	0	0	0	0	0	0	
0x10	I2C_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[7:0]							
	Reset value									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Res.	Res.	Res.	PECERR	OVR	AF	ARLO	BERR	TXE	RXNE	Res.	STOPF	Res.	BTF	ADDR	SB
	Reset value				0	0	0	0	0	0	0		0		0		0
0x18	I2C_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GEN-CALL	Res.	TRA	BUSY	MSL
	Reset value												0		0	0	0
0x1C	I2C_CCR	F/S	DUTY	Res.	Res.	CCR[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRISE[5:0]					
												0	0	0	0	1	0

20. Universal synchronous asynchronous receiver transmitter (USART)

20.1. 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 multi-processor communications.

20.2. USART main features

- Full-duplex asynchronous communications
- NRZ standard format
- 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 4.5Mbit/s.
- Programmable data word length (8 or 9 bits)
- Configurable stop bits (1 or 2 stop bits)
- Synchronous mode and clock output for synchronous communications
- Single-wire Half-duplex communications
- Separate enable bits for transmitter and receiver
- Hardware flow control
- Detection flag
  - Receive buffer full
  - Send buffer empty
  - End of transmission
- Parity Control
  - Send check digit
  - Check the received data
- Flagged interrupt sources
  - CTS change
  - Send register empty
  - Send completed
  - Receive data register full
  - Bus idle detected
  - Overflow error
  - Frame error
  - Noise operation
  - Error detection

- Multiprocessor communication
  - If the address does not match, enter silent mode
- Wake-up from silent mode: description by idle detection and address flag USART functional.

20.3. USART function description

USART interface is connected with other devices through three pins. 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.

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 1, 2 stop bits, thus indicating the end of the data frame
- The USART interface uses a baud rate generator: Representation of 12-bit Integers and 4-bit Fraction
- A status register (USART_SR)
- Data registers (USART_DR)
- A baud rate register (USART_BRR)

The following pins are required in synchronous mode:

CK: Transmitter clock output.

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 it, and a software option to end a clock pulse on the last data bit). In parallel, data an be received synchronously on RX. This can be used to control peripherals that ave shift registers (e.g. LCD drivers). The clock phase and polarity are software rogrammable.

The following pins are required in RS232 Hardware flow control mode:

- **nCTS**: Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS**: Request to send indicates that the USART is ready to receive data (when low).

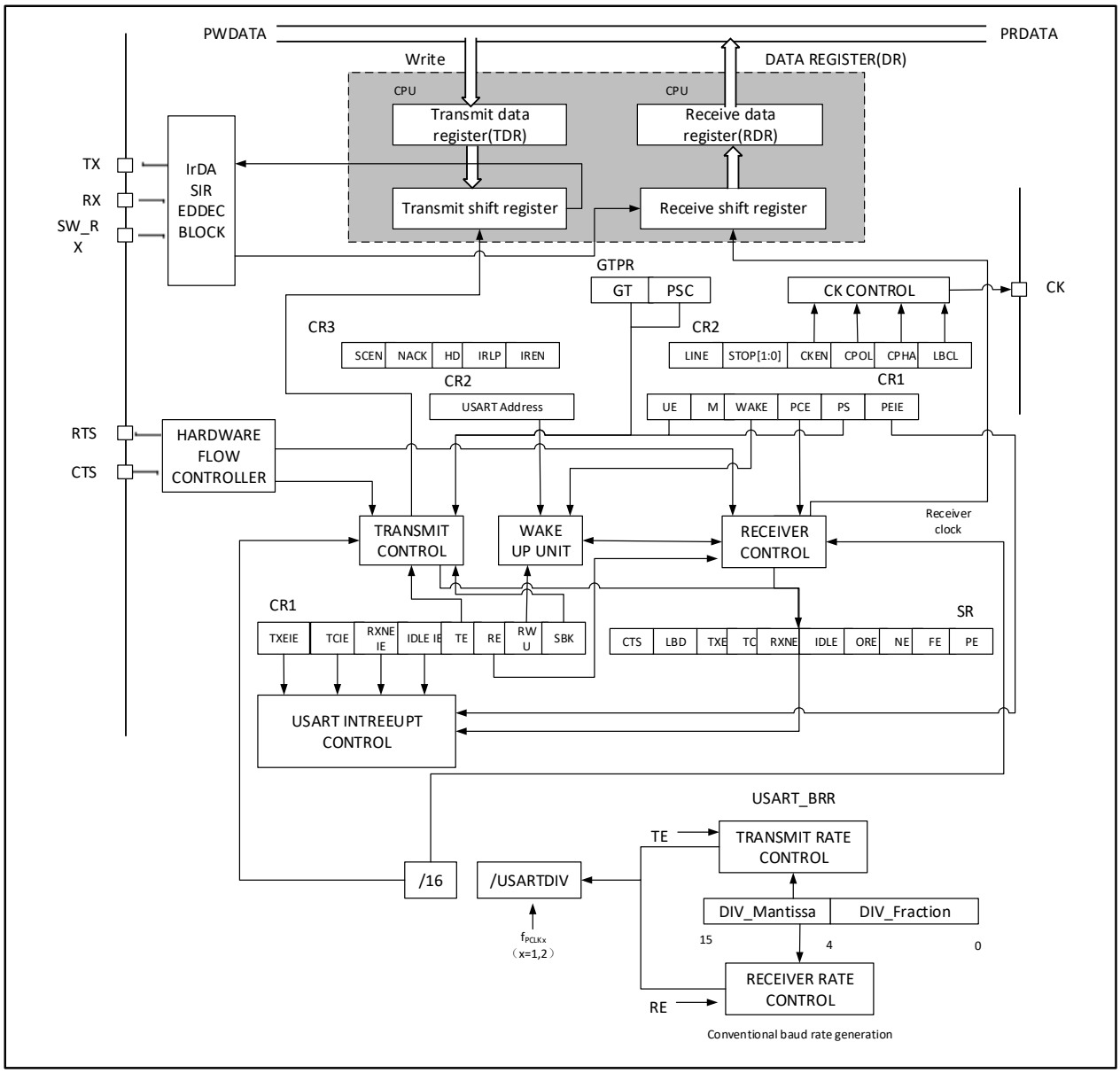


Figure 20-1 USART block diagram

20.3.1. USART character description

World length may be selected as being either 8 or 9 bits by programming the M bits in the USART_CR1 register. The TX pin is low during the start bit and high during the stop bit.

An **Idle character** is interpreted as an entire frame of “1”s followed by the start bit of the next frame containing the data(the number of “1”s includes the number of stop bits)

A **Break character** is interpreted on receiving “0”s for a frame period(including the stop bit period, which is also '0'). At the end of the break frame, the transmitter inserts 1 or 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.

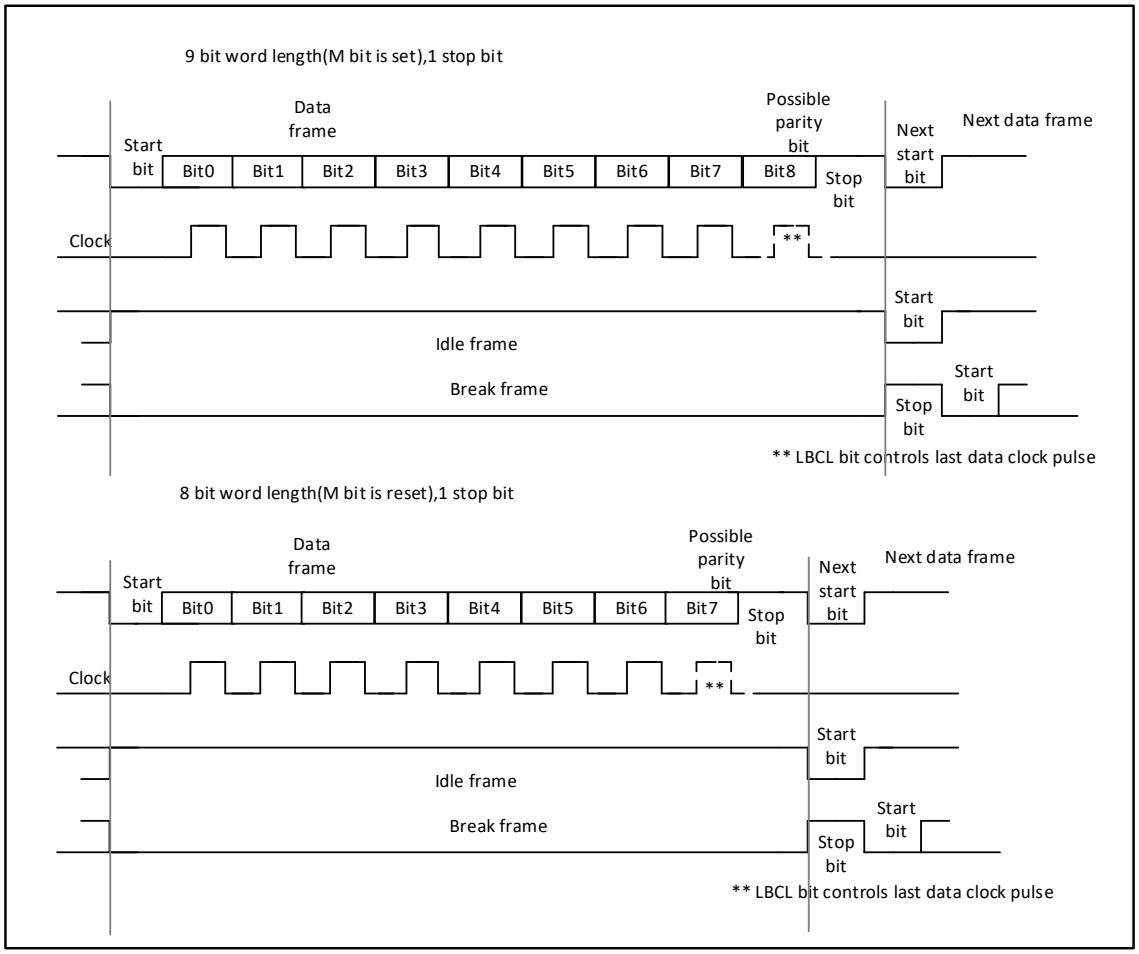


Figure 20-2 Word length programming

20.3.2. Transmitter

The transmitter can send data words of either 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.

20.3.2.1. Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USART_DR register consists of a buffer (DR) between the internal bus and the transmit shift register.

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 USART supports multiple stop bit configurations: 1 and 2 stop bits.

Note:

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.

20.3.2.2. 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) 1 stop bit: This is the default value of number of stop bits.

2) 2 stop bits: This will be supported by normal USART, Single-wire and Modem modes. An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when m = 0) or 11 low bits (when m = 1) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10 low bits).

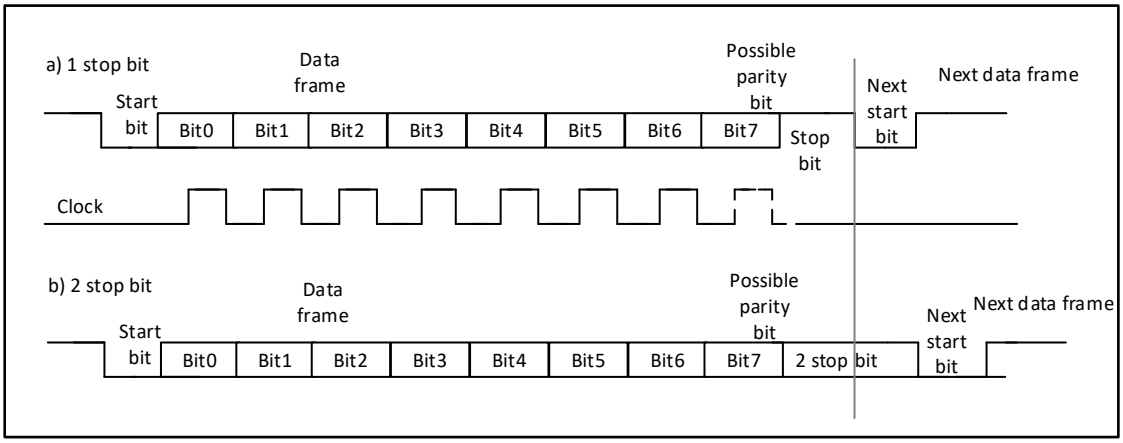


Figure 20-3 Configurable stop bits

Character transmission procedure

1) Enable the USART by writing the UE bit in USART_CR1 register to 1.

2) Program the M bits in USART_CR1 to define the word length.

3) Program the number of stop bits in USART_CR2.

4) Select the desired baud rate using the USART_BRR register.

5) Set the TE bit in USART_CR1 to send an idle frame as first transmission.

6) Write the data to send in the USART_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.

7) After writing the last data into the USART_DR register, wait until TC = 1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

20.3.2.3. 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_DR register stores the data in the DR 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_DR 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

Use the following software procedure to clear the TC bit:

1. Read the USART_SR register once,
2. Write the USART_DR register once.

Note: The TC bit can also be cleared by software by writing '0' to it. This clearing method is only recommended for use in multi-buffer communication mode.

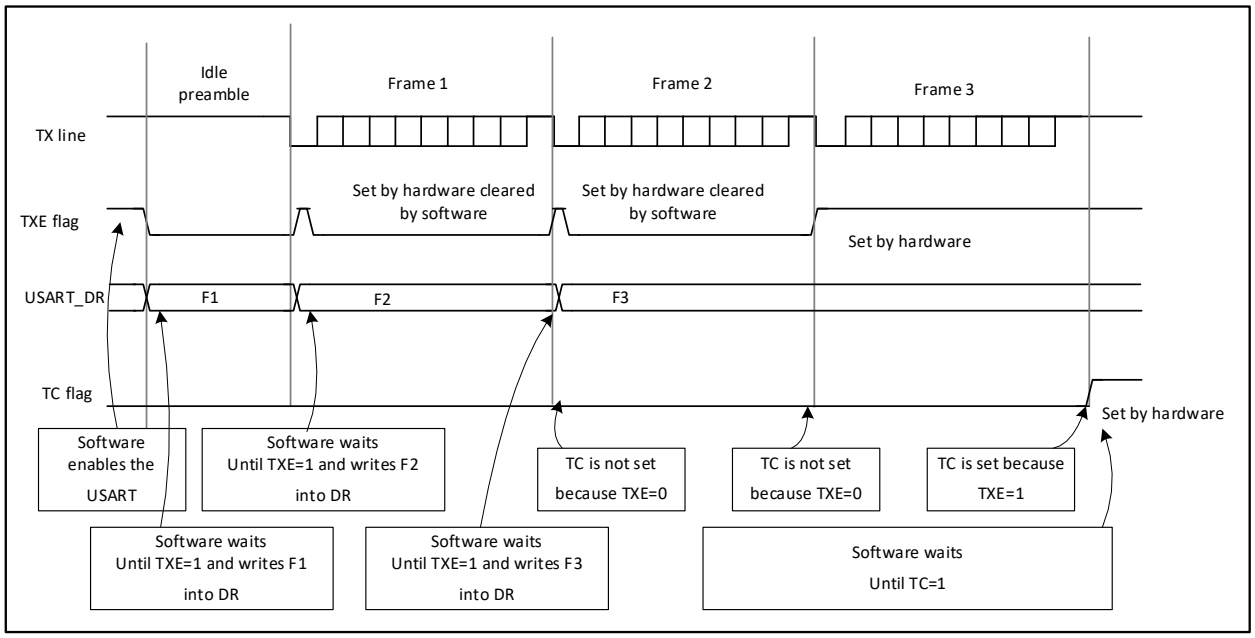


Figure 20-4 TC/TXE behavior when transmitting

20.3.2.4. Break characters

Setting the SBK bit transmits a break character. The break frame length depends on the M bits. If a ‘1’ is written to the SBK bit, a break character is sent on the TX line after completing the current character transmission. The SBK 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 at the end of the break frame to guarantee the recognition of the start bit of the next frame.

If software resets the SBK bit before starting to transmit the break frame, the break symbol will not be sent. If two consecutive break frames are to be sent, the SBK bit should be set after the stop bit of the previous break symbol.

20.3.2.5. Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

20.3.3. Recevier

The USART can receive data words of either 7, 8 or 9 bits depending on the M bits in the USART_CR1 register.

20.3.3.1. Start bit detection

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 0 0 0.

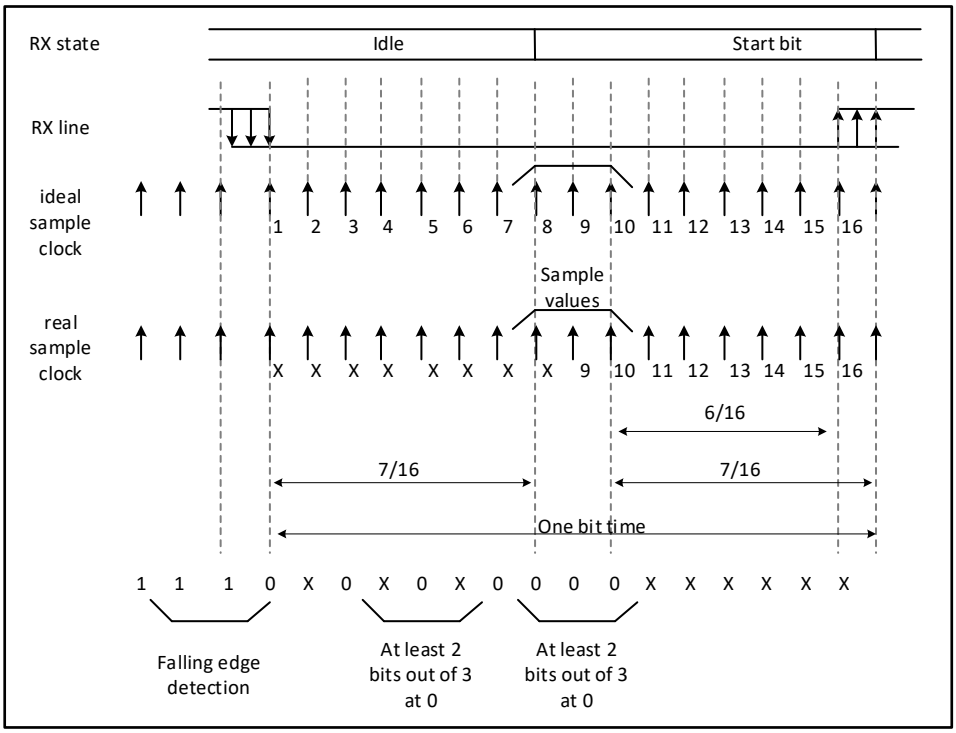


Figure 20-5 Start bit detection

If the sequence is incomplete, the receiver will exit the start bit detection and return to the idle state *(* without setting the flag *)* to wait for a falling edge. If all *3* sample points are *'0' (* the first sample at bits *3, 5,* and *7, and the second sample at bits 8*, *9*, and *10* are all *'0')*, then acknowledge receipt Start bit, then set *the RXNE* flag bit, if *RXNEIE = 1*, an interrupt will be generated.

If only *2 of the 3 sample points* are *'0' twice (* the *3rd, 5th, 7th* sample point and the *8th, 9th, 10th* sample point *),* then the start bit is still valid, but The *NE* noise flag is set. If this condition cannot be met, the detection process of the start bit is aborted, and the receiver will return to the idle state *(* the flag bit is not set *).*

If at one time only *2 of the 3 sample points* are *'0' (* the *3rd, 5th, 7th* sample point or the *8th, 9th, 10th* sample point *),* then the start bit is still valid, but the *NE* noise flag is set.

20.3.3.2. Character reception

During USART reception, the least significant bit of data is shifted in first from the RX pin. In this mode, the USART_DR register contains a buffer between the internal bus and the receive shift register.

Configuration steps:

1) Set UE in USART_CR1 register to 1 to activate USART.
2) Program the M bits of USART_CR1 to define the word length
3) Write the number of stop bits in USART_CR2
4) Select the desired baud rate using the baud rate register USART_BRR.

5) Set the RE bit of USART_CR1. Activate the receiver to start looking for the start bit.

When a character is received :

- The RXNE bit is set. It indicates that the contents of the shift register are transferred to the RDR. In other words, the data has been received and can be read (including error flags associated with it).
- If the RXNEIE bit is set, an interrupt is generated.
- If a frame error, noise or overflow error is detected during reception, the error flag will be set
- In multi-buffer communication, RXNE is set up after each byte is received.
- In single buffer mode, the RXNE bit is cleared by software reading the USART_DR register. The RXNE flag can also be cleared by writing 0 to it. The RXNE bit must be cleared before the end of the next character reception to avoid overrun errors. Note: The *RE* bit should not be reset while receiving data. If the *RE* bit is cleared on reception, the reception of the current byte is lost.

20.3.3.3. Break character

When a break character is received, the USART handles it as a framing error.

20.3.3.4. 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.

20.3.3.5. 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.

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 the EIE bit is set.
- Sequential read operations of USART_SR and USART_DR registers can reset the ORE bit

Note: When the *ORE* bit is set, it indicates that at least *1* data has been lost. There are two possibilities:

- If RXNE = 1, the last valid data is still in the receive register RDR and can be read.
- If RXNE = 0, it means that the last valid data has been read, and there is nothing to read in RDR. This can happen when new (ie lost) data is received while the last valid data is being read in the RDR. This can also happen when new data is received during the read sequence (between the USART_SR register read access and the USART_DR read access).

20.3.3.6. Noise error

Data recovery is performed by distinguishing between valid input data and noise using oversampling techniques (except synchronous mode).

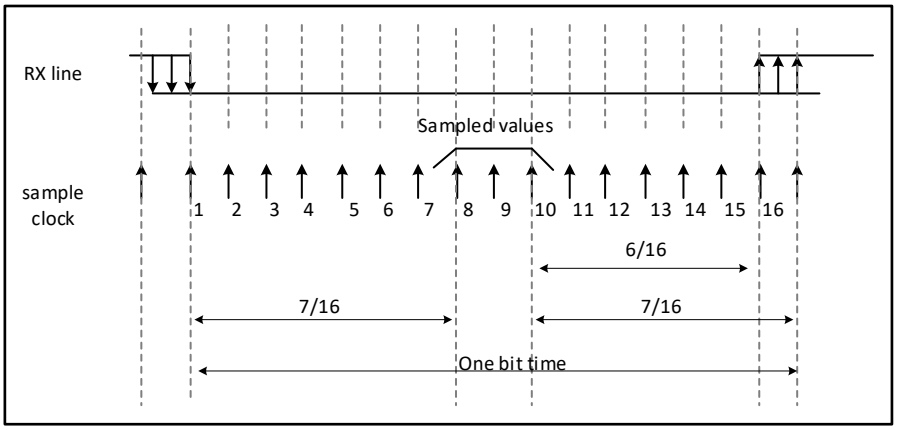


Figure 20-6 Data sampling for noise detection

Table 20-1 Noise detection from sampled data

Sample value	NE state	Bit value received	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

When noise is detected in the received frame:

- ● Set the NE flag on the rising edge of the RXNE bit.
- ● Invalid data is transferred from the shift register to the USART_DR register.
- ● In the case of single-byte communication, no interrupt is generated. However, since the NE flag and the RXNE flag are set at the same time, RXNE will generate an interrupt. In the case of multi-buffer communication, an interrupt will be generated if the EIE bit in the USART_CR3 register has been set.

First read USART_SR, then read USART_DR register, will clear the NE flag bit.

20.3.3.7. 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.

Sequential reads of the USART_SR and USART_DR registers reset the FE bit.

20.3.3.8. 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.

- 1 stop bit: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.

- 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.

20.3.4. 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.

$$\text{Tx / Rx baud} = fCK / (16 * USARTDIV)$$

Here *fCK* is the clock to the peripheral USARTDIV is an unsigned fixed-point number. The 12-bit value is set in the USART_BRR register.

Note: After writing to *USART_BRR,* the baud rate counter is replaced by the new value of the baud rate register. Therefore, do not change the value of the baud rate register while communication is in progress.

How to derive USARTDIV from USART_BRR register values

Example 1:

If DIV_Mantissa = 27, DIV_Fraction = 12 (USART_BRR = 0x1BC), then:

Mantissa (USARTDIV) = 27

Fraction (USARTDIV) = 12/16 = 0.75

Therefore USARTDIV = 27.75

Example 2:

To program USARTDIV = 25.62, then:

DIV_Fraction = 16*0.62 = 9.92

The nearest integer is: 10 = 0x0A

DIV_Mantissa = mantissa (25.620) = 25 = 0x19

Then, USART_BRR = 0x19A

Example 3:

To program USARTDIV = 50.99, then:

DIV_Fraction = 16*0.99 = 15.84

The nearest integer is: 16 = 0x10 = > DIV_frac[3:0] overflow = > carry must be added to the fractional part

DIV_Mantissa = mantissa (50.990 + carry) = 51 = 0x33

Then, USART_BRR = 0x330, USARTDIV = 51

Baud rate		F_{PCLK} = 36 MHz			F_{PCLK} = 72 MHz		
S.No	Kbps	Actual	BRR	Error(%)	Actual	BRR	Error(%)
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	N.A	N.A	N.A	4500	1	0%

Note: 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.

20.3.5. USART receiver's tolerance 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-tolow transition timing)

  DTRA + DQUANT + DREC + DTCL < USART receiver's tolerance.

For normal reception of data, the tolerance of the USART receiver is equal to the maximum tolerable variation, which depends on the following choices:

- 10- or 11-bit character length defined by the M bits of the USART_CR1 register
- whether to use fractional baud rate to generate

Table 20-2 USART receiver tolerance when DIV_Fraction is 0

M bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

Table 20-3 USART receiver tolerance when DIV_Franction is different from 0

M bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

20.3.6. 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:

1)The communication speed of the system is not known in advance

2)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 (oversampling by 16 must be selected and baudrate between fCK/65535 and fCK/16.

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_CR3 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.

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.

20.3.7. Multiprocessor communication using USART

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 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.

20.3.7.1. Idle line detection (WAKE = 0)

The USART enters mute mode when the RWU bit is written to 1. It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in Figure 27-7.

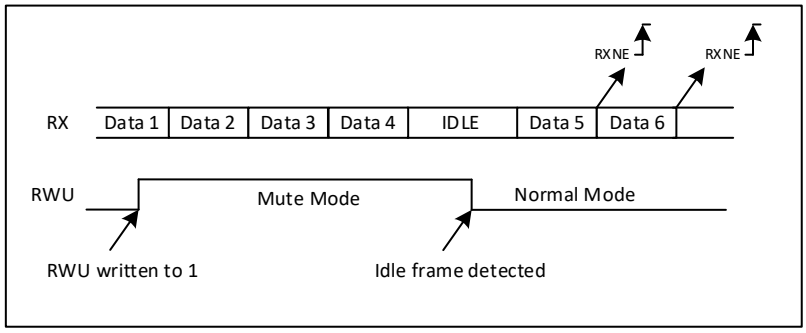


Figure 20-7 Mute mode using Idle line detection

20.3.7.2. Address mark check (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 LSBs. The choice of 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.

If the received byte does not match its programmed address, the USART enters silent mode. At this point, the hardware sets the RWU bit.

Receiving this byte will neither set the RXNE flag nor generate an interrupt because the USART is already in silent mode.

When the received byte matches the programmed address in the receiver, the USART exits silent mode. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit will be set when this matching address byte is received because the RWU bit has been cleared.

When the receive buffer contains no data (RXNE = 0 in USART_SR), the RWU bit can be written to 0 or 1. OtheRWise, the write operation is ignored. The figure below shows an example of using address mark detection to wake up and enter silent mode.

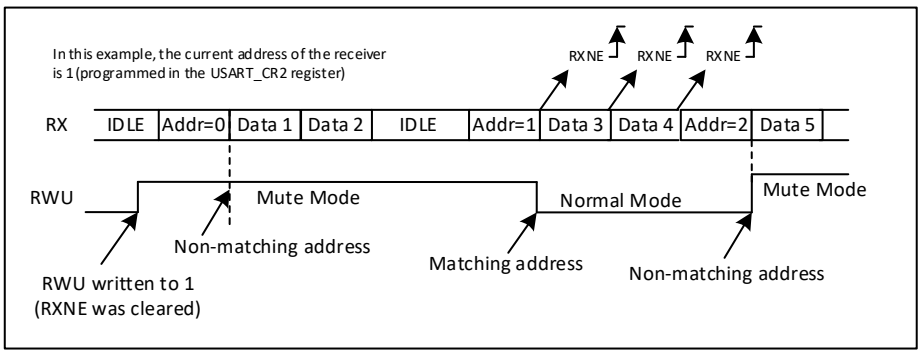


Figure 20-8 Slient mode with address tag detection

20.3.7.3. Check control

Setting the PCE bit on the USART_CR1 register enables parity control (generates a parity bit when transmitting, and performs parity checking when receiving). The possible USART frame formats are listed in the table below according to the frame length defined by the M bits.

Table 20-4 Frame format

M bit	PCE bit	USART fram
0	0	SB—8 bit data—STB
0	1	SB—7 bit data—PB—STB
1	0	SB—9 bit data—STB
1	1	SB—8 bit data—PB—STB

When waking up a device with an address tag, the address is matched only considering the *MSB* bits of the data, not the parity bits. *(MSB* is the last sent out of the data bits, followed by the parity bit or stop bit*)*

20.3.7.4. Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 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).

20.3.7.5. Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on 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).

20.3.7.6. Transfer mode

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 or an odd number of “1s” if odd parity is selected). 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 riting 1 to the PECF in the USART_ICR register.

20.3.8. 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:

- The HDSEL bit in USART_CR3 register

The USART allows the user to control bidirectional synchronous serial communications in a master mode. The CK pin is the output of the USART transmitter clock. During the start and stop bits, there is no clock pulse on the CK pin. Depending on the state of the LBCL bit in the USART_CR2 register, a clock pulse is generated or not generated during the last valid data bit. The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock.

The external CK clock is not activated during bus idle periods, before actual data arrives and when the disconnect symbol is sent.

In synchronous mode, the USART transmitter works exactly the same as in asynchronous mode. But because CK is synchronous with TX (according to CPOL and CPHA), data on TX is sent synchronously with CK.

The USART receiver in synchronous mode works differently than in asynchronous mode. If RE = 1, the data is sampled on CK (rising or falling according to CPOL and CPHA) without any oversampling. But setup time and duration (depending on baud rate, 1/16 bit time) must be considered.

Notice:

CK pin works together with the *TX* pin. Thus, the clock is only provided when the transmitter is enabled *(TE = 1)* and data is being sent *(* writing data to *the USART_DR* register *).* This means that it is impossible to receive a sync data without sending data.

LBCL, CPOL and *CPHA* bits should be when both the transmitter and receiver are disabled, these bits cannot be changed when the transmitter or receiver is enabled.

TE and *RE* in the same instruction to reduce receiver setup and hold time.

USART only supports master mode: it cannot receive or transmit data with an input clock from other devices *(CK* is always an output*).*

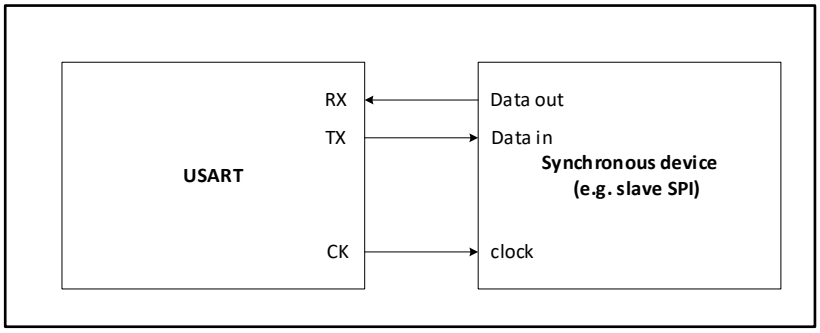


Figure 20-9 Example of USART isochronous transmission

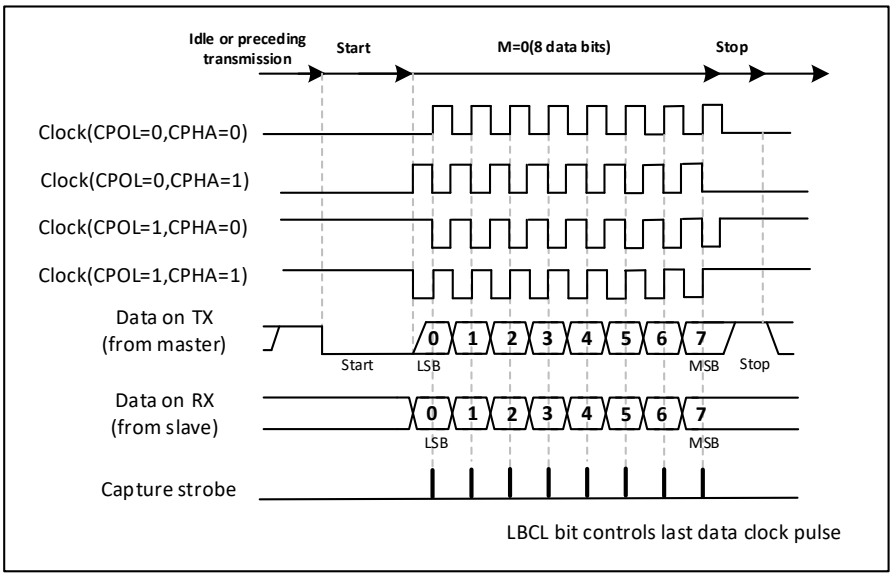


Figure 20-10 USART data clock timing example (M = 0)

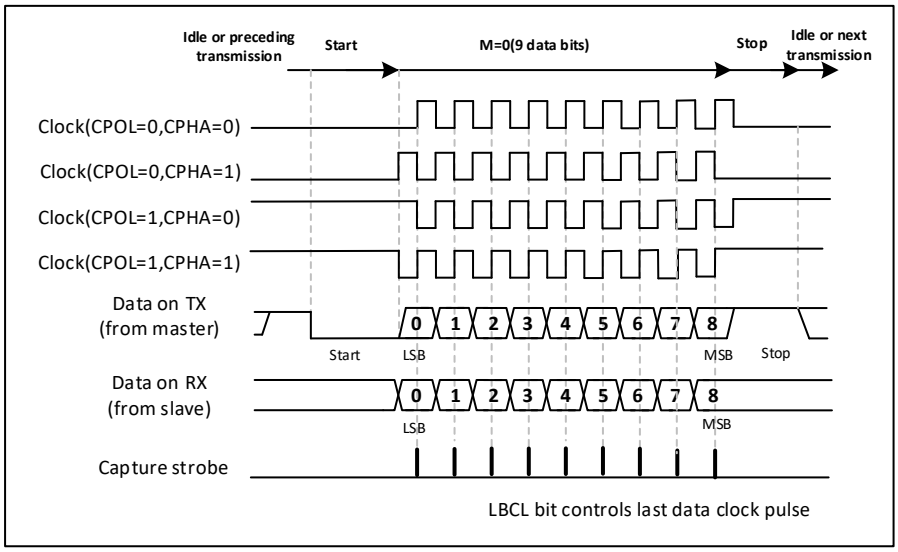


Figure 20-11 USART data clock timing example(M = 1)

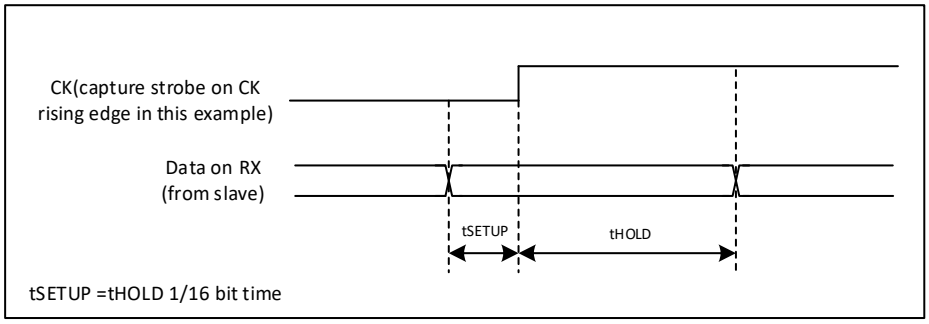


Figure 20-12 RX data sample/hold time

20.3.9. 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

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 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.

20.3.10. Hardware flow control

Using the nCTS input and nRTS output. The figure below shows how to connect 2 devices in this mode.

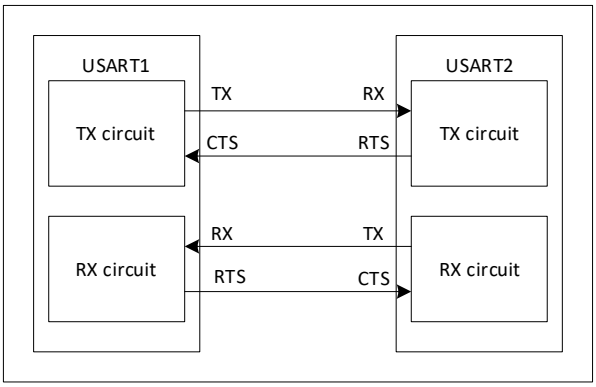


Figure 20-13 Hardware flow control between two USARTs

20.3.10.1. RTS flow control

If RTS flow control is enabled (RTSE = 1), nRTS becomes active (connected low) as soon as the USART receiver is ready to receive new data. When data arrives in the receive register, nRTS is released, thereby indicating that data transmission is to be stopped at the end of the current frame.

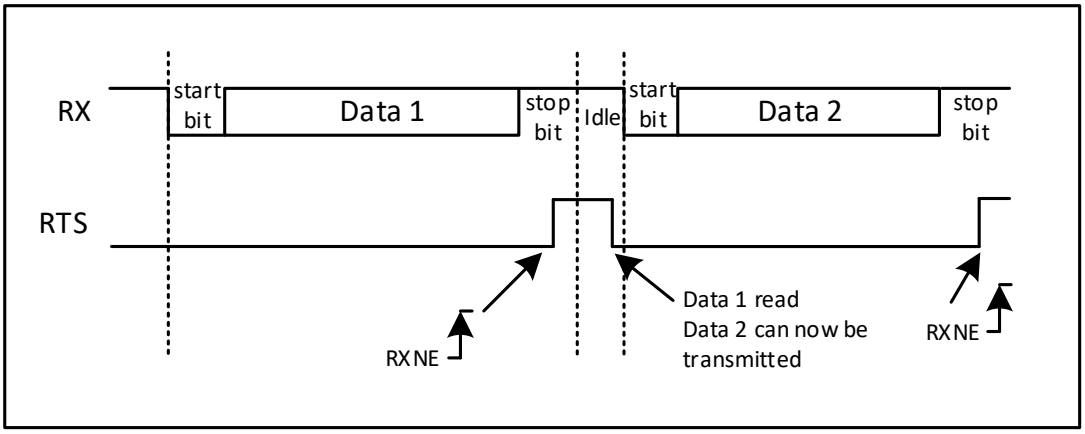


Figure 20-14 RTS flow control

20.3.10.2. CTS flow control

If CTS flow control is enabled (CTSE = 1), the transmitter checks the nCTS input before sending the next frame. If nCTS is valid (pulled to a low level), the next data is sent (assuming that data is ready to be sent, that is, TXE = 0), otheRWise the next frame of data is not sent. If the nCTS is invalidated during transmission, the transmission stops after the current transmission is completed.

When CTSE = 1, as long as the nCTS input changes state, the hardware automatically sets the CTSIF status bit. It indicates whether the receiver is ready to communicate. An interrupt is generated if the CTSIE bit in the USART_CT3 register is set.

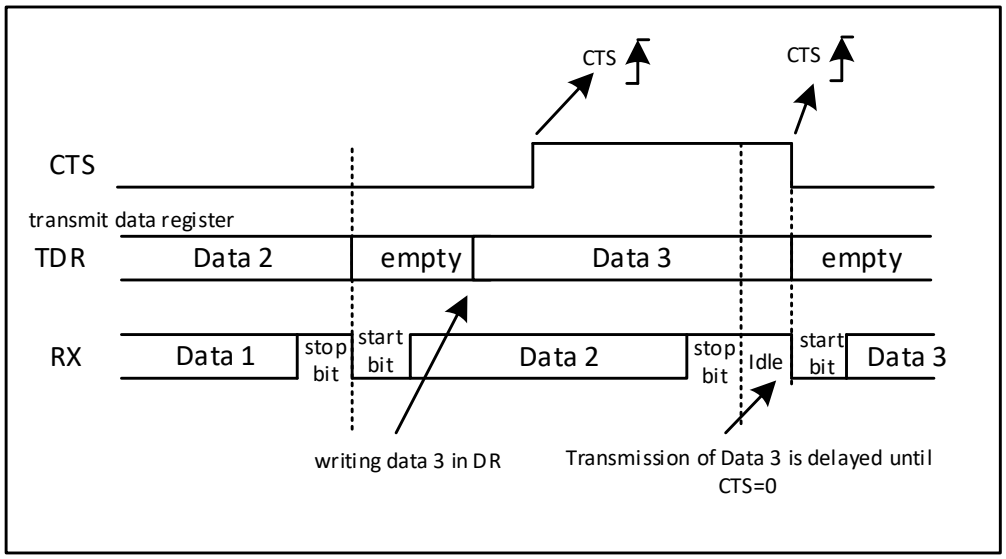


Figure 20-15 CTS flow control

20.4. USART interrupt request

Serial number	Interrupt event	Event flag	Enable bit	Send/receive
1	Send data register empty	TXE	TXEIE	Send
2	CTS (Clear to Send)interrupt	CTSIF	CTSIE	Send
3	Transmission completed	TC	TCIE	Send
4	The receive register is not empty (read data is ready)	RXNE	RXNEIE	Take over
5	Overrun error	ORE		Take over
6	Idle frame	IDLE	IDLEIE	Take over
7	Parity error	PE	PEIE	Take over
8	Noise, overrun and frame errors when communicating with multiple processors	NR/ORE/FE	EIE	Take over

All USART interrupts share the same interrupt vector.

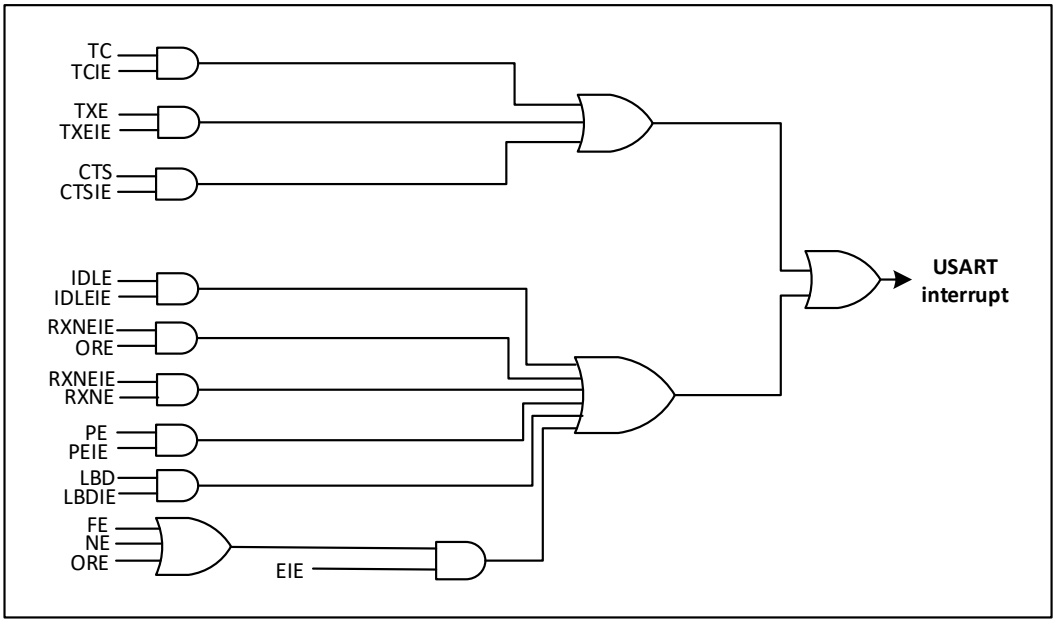


Figure 20-16 USART interrupt map

20.5. USART register

20.5.1. Status register (USART_SR)

Address offset: 0x00

Reset value: 0x0000 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	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	ABRRQ	ABRE	ABRF	CTS	Res	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
			W	R	R	RC_W0		R	RC_W0	RC_W0	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	RES	-	Reserved
12	ABRRQ	W	0	Automatic baud rate request Writing 1 to this bit resets the ABRF flag and requests automatic baud rate detection for the next frame.
11	ABRE	R	0	Autobaud error flag. This register is set by hardware when there is an error in automatic baud rate detection (baud rate out of range or character comparison error). Software clears this bit by writing a 1 to the ABRRQ register.
10	ABRF	R	0	Automatic baud rate detection flag. This bit is set to 1 by hardware when auto-baud rate is set (set RXNE = 1 at the same time, an interrupt will be generated when the interrupt is enabled), or when an error occurs in the auto-baud rate detection operation (ABRE = 1, RXNE = 1, FE = 1). Software clears this bit by writing a 1 to the ABRRQ bit in the USART_RQR register.
9	CTS	RC_W0	0	When CTS input toggle, do not CTSE = 1, this register is 1. Software writes 0 to clear. When CTSIE = 1, a CTS interrupt is generated. 0: CTS line value unchanged 1: CTS line value change
8	reserved			
7	TXE	R	1	Transfer register empty flag. This register is set by hardware when the USART_DR register data is transferred to the shift register. When TXEIE = 1, an interrupt is generated. Writing to the USART_DR register will clear this bit 0: Data is not transferred to the shift register 1: Data is transferred to the shift register
6	TC	RC_W0	1	Transmission complete flag. After the transmission of the data frame is completed, and TXE = 1, the hardware will set this register. An interrupt is generated when TCIE = 1. Software reading the USART_SR register first and then writing the USART_DR register will clear this bit (for multiprocessor communication). Software can also write 0 to clear. 0: Transmission not completed 1: Transmission completed
5	RXNE	RC_W0	0	The read data register is not empty flag.

				This register is set by hardware when the shift register value is transferred to the USART_DR register. Software reads the USART_DR register, or writes 0 to clear this bit. When RXNEIE = 1, an interrupt is generated. 0: No data received 1: Receive data ready
4	IDLE	R	0	Idle sign. Detect IDLE line, the hardware sets this register. An interrupt is generated when IDLEIE = 1. Software can clear this bit by reading the USART_SR register first and then the USART_DR register. 0: IDLE not detected line 1: IDLE detected line
3	ORE	R	0	Overrun error flag. When RXNE = 1, the hardware sets this bit when the data received in the shift register is about to be transferred to the RDR register. Software can clear this bit by reading the USART_SR register first and then the USART_DR register. When RXNEIE = 1, an interrupt is generated. 0: No Overrun error is generated 1: Generate Overrun error Note: When this register is set, the contents of the RDR register are not lost, but the contents of the shift register are oveRWritten. When EIE = 1, an ORE interrupt is generated.
2	NE	R	0	Noise error sign. This register is set by hardware when the data frame receives noise. Software can clear this bit by reading the USART_SR register first and then the USART_DR register. 0: No noise error detected 1: Noise error detected Note: When RXNE and NE are generated at the same time, no interrupt is generated when NE = 1, but an interrupt is generated when the RXNE flag is set. In multi-buffer communication mode, NE = 1 will generate an interrupt when EIE = 1.
1	FE	R	0	Framing error flag. This bit is set by hardware when out-of-sync, excessive noise, or abort characters are detected. Software can clear this bit by reading the USART_SR register first and then the USART_DR register. 0: no frame error detected 1: Framing error or break character detected Note: When RXNE and FE are generated at the same time, no interrupt is generated when FE = 1, but an interrupt is generated when the RXNE flag is set. If the currently transmitted data has both a frame error and an overload error, the hardware will continue to transmit the data and only set the ORE flag. In multi-buffer communication mode, FE = 1 will generate an interrupt when EIE = 1.
0	PE	R	0	Checksum error. This register is set by hardware when the parity value is incorrect during reception. Software can clear this bit by reading the USART_SR register first and then the USART_DR register. But software must wait for RXNE = 1 before clearing this bit. When PEIE, an interrupt is generated. 0: No parity error is generated 1: Generate parity error

20.5.2. UASRT data register (USART_DR)

Address offset: 0x04

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	DR[8:0]								
							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	Reserved	RES	-	Reserved
8:0	DR[8:0]	RW	undefined	Receive/transmit data register. Depending on the read or write operation, the former is the received data and the latter is the transmitted data. The DR register is physically composed of two registers (one is the transmitted T DR, the other is the received R DR), so the DR register implements two functions of reading and writing. T DR register provides a parallel interface between the internal bus and the output shift register, and the R DR register provides a parallel interface between the input shift register and the internal bus. When parity is enabled for transmit operation, writing the M SB bit (bit7 or bit8) has no effect because it has been replaced by the parity bit. When the parity enable is turned on for a receive operation, the read MSB bit is the received parity bit.

20.5.3. Baud rate register (USART_BRR)

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**
DIV_Mantissa[11:0]												DIV_Faction[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

In auto-baud detection mode, hardware updates this register.

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	-	Reserved
15:4	DIV_Mantissa[15:4]	RW	0	12bit integer
3:0	DIV_Fraction[3:0]	RW	0	4bit decimal

20.5.4. USART control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	Reserved	RES	-	Reserved
13	UE	RW	0	USART enabled. When this bit is cleared, the USART module will immediately stop the current operation. This bit is set and cleared by software. 0: USART prescaler and output disabled, low-power mode 1: USART enable The software needs to wait for USART_ISR.TC to be set before clearing the UE bit and entering the low power mode.
12	M	RW	0	0: 1 start bit, 8 data bits, n stop bit 1: 1 start bit, 9 data bit, n stop bit
11	WAKE	RW	0	Receive wakeup mode. How to wake up from mute mode. Set or cleared by software. 0: Idle line wake up 1: address wake-up
10	PCE	RW	0	Parity control. 0: Parity check disabled 1: Parity check enabled Parity bit: 9th bit of 9bit, 8th bit of 8bit.
9	PS	RW	0	Parity check selection. Set and cleared by software. 0: Even parity 1: odd parity
8	PEIE	RW	0	PE interrupt enable. Set and cleared by software. 0: Disable 1: PE interrupt enable
7	TXEIE	RW	0	TXE interrupt enable. Set and cleared by software. 0: Disable 1: TX E interrupt enable
6	TCIE	RW	0	End of transfer interrupt enable. Set and cleared by software. 0: Disable 1: TC interrupt enable
5	RXNEIE	RW	0	RXNE interrupt enable, set and cleared by software. 0: Disable 1: ORE or RXNE interrupt enable
4	IDLEIE	RW	0	IDLE interrupt enable. Set and cleared by software. 0: Disable 1: IDLE interrupt enable
3	TE	RW	0	Transmission enable. 0: Transmission prohibited 1: Transmission enable
2	RE	RW	0	Receive enable. 0: Reception prohibited 1:Receive enable, start to detect start bit
1	RWU	RW	0	Receive wakeup. This bit indicates whether the USART is in mute mode. This register is set when a mute mode sequence is received, this register is cleared when a wake-up sequence is received. The specific wake-up sequence (address or IDLE) is controlled by the register USART_CR1.WAKEbit. 0: The receiver is in working mode 1: The receiver is in silent mode Note 1: Before setting this bit to enter mute mode, the USART must have received a data byte, otheRWise in mute mode, it cannot be woken up by idle bus detection. Note 2: When configured as address mark detection wake-up (WAKE = 1), the RWU bit cannot be modified by software when RXNE is set.

0	SBK	RW	0	Send break frame. Software sets this register to send the break byte. This register is cleared by hardware after the stop bit of the break frame is sent. 0: do not send break bytes 1: send break byte

20.5.5. USART control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	STOP	Res	CLKEN	CPOL	CPHA	LBCL	Res	Res	Res	Res	ADD[3:0]			
		RW		RW	RW	RW	RW					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	Reserved	RES	-	Reserved
13	STOP	RW	0	Stop bit configuration. 0: 1 stop bit 1: 2 stop bit
12	Reserved			
11	CLKEN	RW	0	CK pin enable. 0: disable 1: CK pin enable This bit is reserved when synchronous mode is not supported.
10	CPOL	RW	0	Clock polarity. Sync mode, CK pin output clock polarity. 0: Outside the transmission window, CK pin is a stable low value, 1: Outside the transmission window, CK pin is a stable high value,
9	CPHA	RW	0	This bit is used to select the phase of the CK pin output clock in synchronous mode. It works with the CPOL bits to generate the desired clock/data relationship. 0: The first clock transfer is the first data capture edge, 1: The second clock transfer is the first data capture edge,
8	LBCL	RW	0	Whether the clock pulse of the last bit of data is in CK pin output. 0: The clock pulse of the last bit of data is not in CK pin output, 1: The clock pulse of the last bit of data is at CK pin output.
7:4	Reserved	RES	-	Reserved
3:0	ADD[3:0]	RW	4'b0	USART address. This register is used in the mute mode of the multiprocessor, and is used as the address when the 4- bit address wakes up.

20.5.6. USART control register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	ABR-MOD[1:0]		ABR EN	OVER8	CTSIE	CTSE	RTSE	Res	Res	Res	Res	HDSEL	Res	Res	EIE

	RW	RW	RW	RW	RW	RW					RW			RW

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	RES	-	Reserved
14:13	ABRMOD[1:0]	RW	2'b0	Automatic baud rate detection mode. 00: measure the baud rate from the start bit 01: Falling edge to falling edge measurement 10: Reserved 11: Reserved When ABREN = 0 or UE = 0, this register is write-only.
12	ABREN	RW	0	Auto-baud rate enabled. 0: Disable 1: Auto baud rate enabled
11	OVER8	RW	0	Oversampling mode. 0: Oversampling by 16 1: Oversampling by 8 can only be written when U E = 0.
10	CTSIE	RW	0	CTS interrupt enable. 0: Forbidden, 1: CTSIF interrupt enable,
9	CTSE	RW	0	CTS enabled. 0: CTS hardware flow control is disabled, 1: CTS mode enabled. Data is only transmitted when the CTS input is 0. At this point, after the data is written into the data register, the transmission will not be started until the CTS is valid.
8	RTSE	RW	0	RTS enabled. 0: RTS hardware flow control is disabled, 1: The RTS output is enabled, and the next data is requested only when the receive buffer is not full. After the current data is sent, the sending operation is suspended. If data can be received, set RTS to valid (0).
7:4	reserved			
3	HDSEL	RW	0	Half-duplex option. 0: Non-half duplex mode, 1: Half-duplex mode selection.
2:1				
0	EIE	RW	0	Error interrupt enable. 0: Forbidden, 1: Frame error FE, overrun error ORE, noise NF interrupt enable.

20.5.7. USART register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABRRQ	ABRE	ABRF	CTS	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																				0	0	0	0		1	1	0	0	0	0	0	0
0x04	USART_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

0x08		0x0C		0x10		0x14		0x18
USART_BRR	Reset value	USART_CR1	Reset value	USART_CR2	Reset value	USART_CR3	Reset value	Reserved
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
Res.		Res.		Res.		Res.		Res.
DIV_Mantissa[11:0]	0	Res.		Res.		Res.		Res.
	0	Res.		Res.		Res.	0	Res.
	0	UE	0	STOP	0	ABR-MOD[1:0]	0	Res.
	0	M	0	Res.		ABREN	0	Res.
	0	WAKE	0	CLKEN	0	OVER8	0	Res.
	0	PCE	0	CPOL	0	CTSIE	0	Res.
	0	PS	0	CPHA	0	CTSE	0	Res.
	0	PEIE	0	LBCL	0	RTSE		Res.
	0	TXEIE	0	Res.		Res.		Res.
	0	TCIE	0	Res.		Res.		Res.
	0	RXNEIE	0	Res.		Res.		Res.
	0	IDLEIE	0	Res.		Res.		Res.
DIV_Faction[3:0]	0	TE	0	ADD[3:0]	0	HDSEL	0	Res.
	0	RE	0		0	Res.		Res.
	0	RWU	0		0	Res.		Res.
	0	SBK	0		0	EIE	0	Res.

21. Serial peripheral interface (SPI)

This project designs and implements a SPI modules .

21.1. Introduction

Serial Peripheral Interface (SPI) allows the chip to communicate with external devices in half-duplex, full-duplex, and simplex synchronous serial communication. This interface can be configured in master mode and provides the communication clock (SCK) for external slave devices. The interface can also work in a multi-master configuration.

It can be used for a variety of purposes, including two-wire simplex simultaneous transmission using one bidirectional data line.

21.2. 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)
- 8-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to fPCLK/4.
- Slave mode frequency up to fPCLK/4.
- 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
- Master mode fault, overrun flags with interrupt capability

21.3. SPI function description

21.3.1. Overview

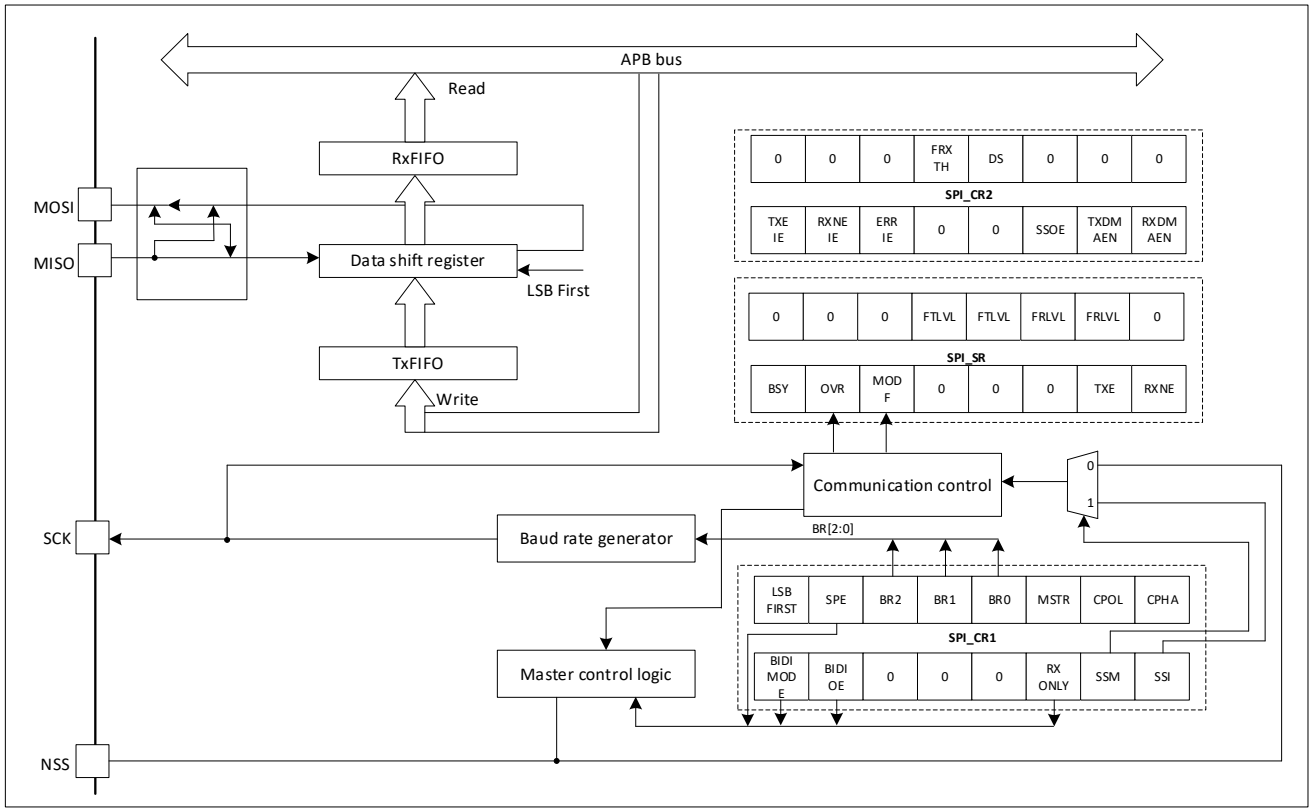


Figure 21-1 SPI block diagram

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

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.

21.3.2. 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.

21.3.2.1. 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.

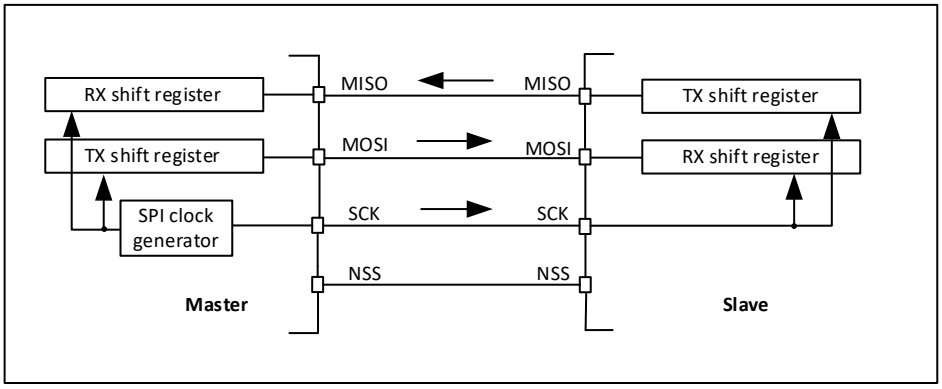


Figure 21-2 Full-duplex single master/slave application

21.3.2.2. 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.

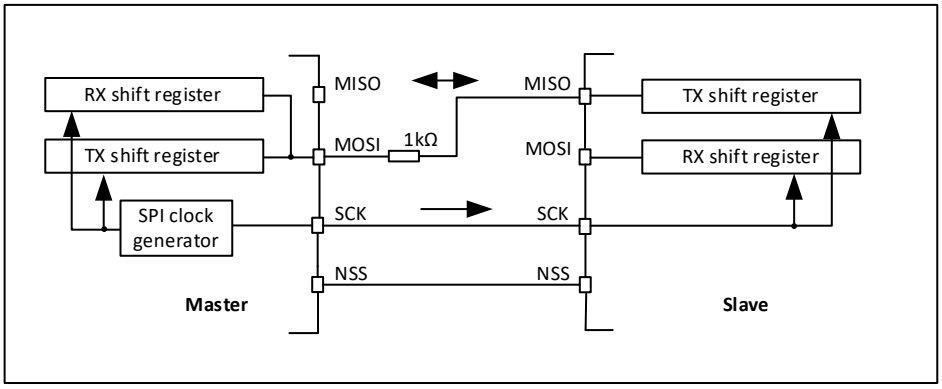


Figure 21-3 Half-duplex single master/slave application

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.

21.3.2.3. Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receiveonly 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 fullduplex. 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. 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

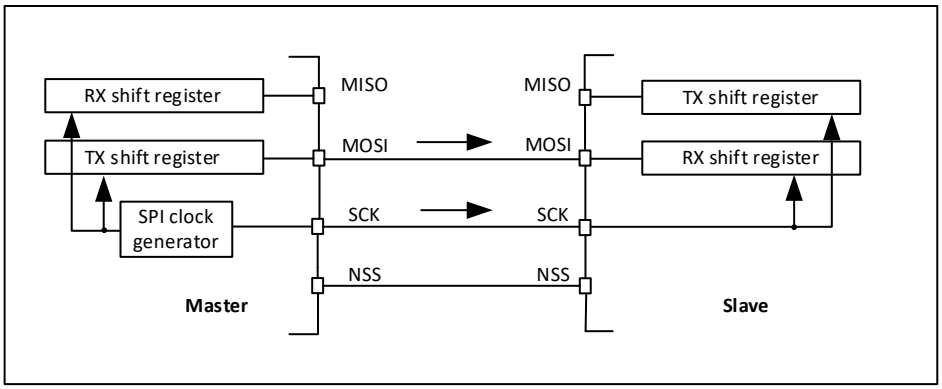


Figure 21-4 simplex single master/single slave application(master in transmit-only/slave in receive-only mode)

(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 28.5.5: 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.

(3) In this configuration, both the MISO pins can be used as GPIOs.

simplex communication can be replaced by half -duplex communication by setting the transfer direction (the bidirectional mode is enabled when the BI DIO E bit is not changed).

21.3.3. Multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO to manage NSS for each slave. The master must select a slave by pulling the connected slave NSS low. When this is done, standard master and dedicated slave communication is established.

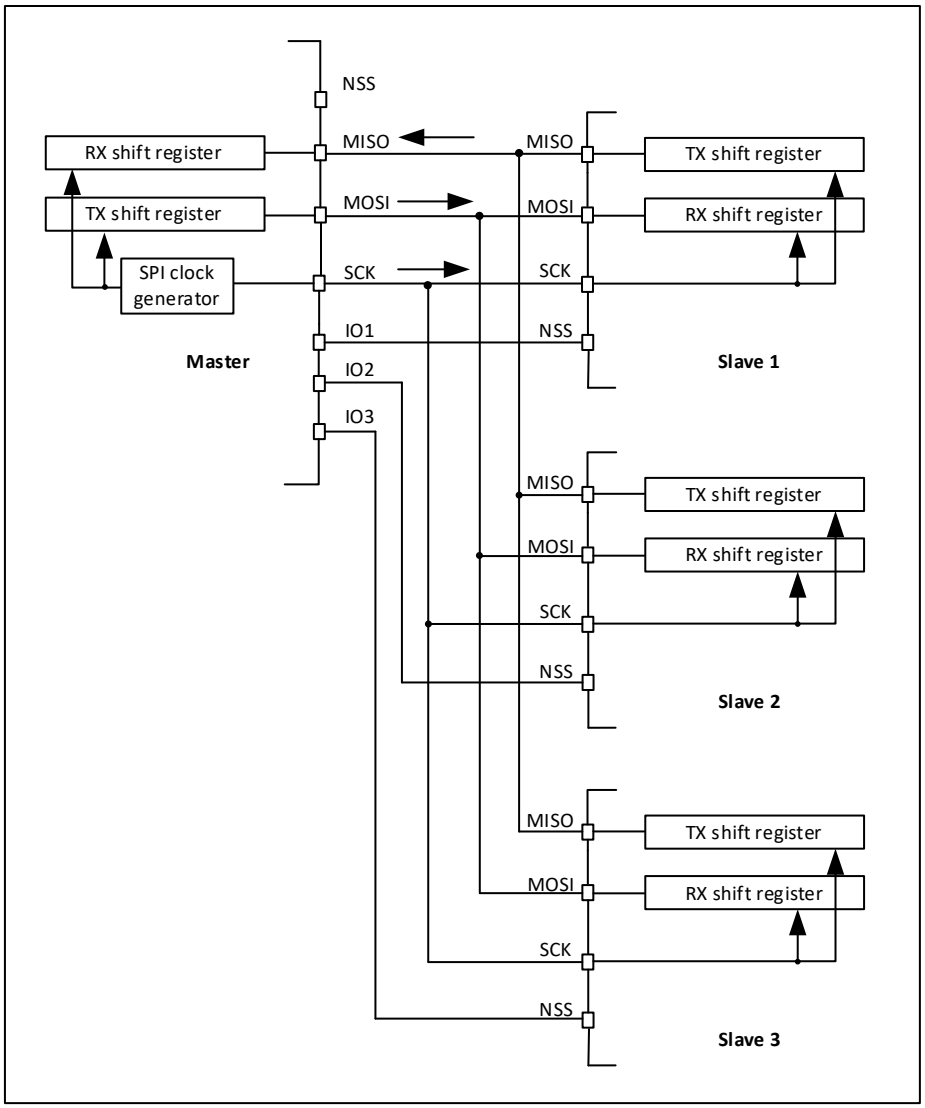


Figure 21-5 Master communicates with three independent slaves

NSS is not used on the host side in this configuration. Any MODF errors must be prevented by SSM = 1, SSI = 1.

Since the MISOs of the slaves are connected together, all slaves must configure their MISO's GPIO as AF open-drain.

21.3.4. 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).

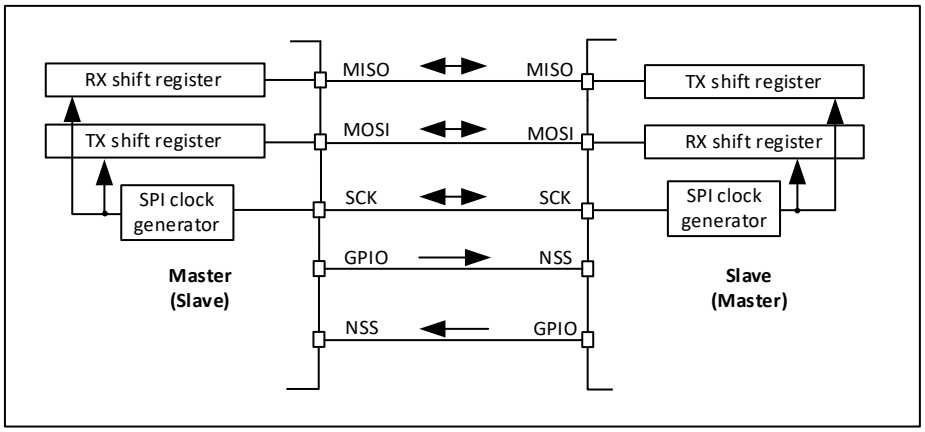


Figure 21-6 Multi-master application

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.

21.3.5. 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. s an input it can prevent multimaster bus collision, and as an output it can drive a slave elect 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
  1） – 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.
  2） – 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.

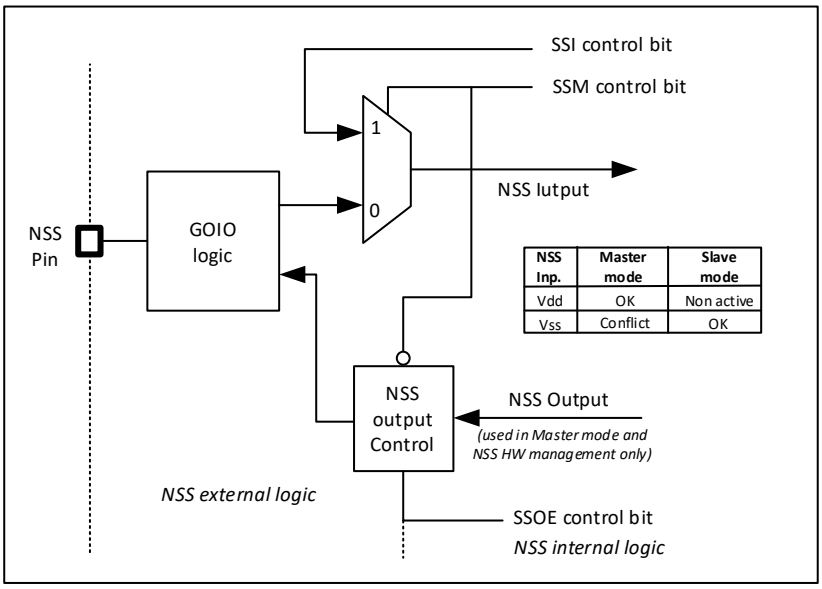


Figure 21-7 Hardware/software slave select management

21.3.6. 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.

21.3.6.1. Clock phase and polarity controls

There are 4 possible timings that can be configured by software through the CPOL and CPHA bits (SPI_CR1 register). CPOL (clock polarity) controls the IDLE state of the clock when no data is being transmitted. This bit affects both master and slave. If CPOL is reset, the SCK pin has a low state. If CPOL is set, the SCK pin has a high IDLE state.

If CPHA is set, the second edge of SCK captures the first data bit transmitted (falling edge if CPOL is reset, rising edge otheRWise). On the occurrence of clock change type, the data is latched. If CPHA is reset, the first edge of SCK captures the first transmitted data bit (falling edge if CPOL is set, rising edge otheRWise). Data is latched when this type of clock change occurs.

The combination of CPOL and CPHA selects the data capture clock edge.

SPI must be disabled (SPE = 0) before CPOL/CPHA is changed.

The IDLE state of SCK must correspond to the polarity selected by the SPI_CR1 register.

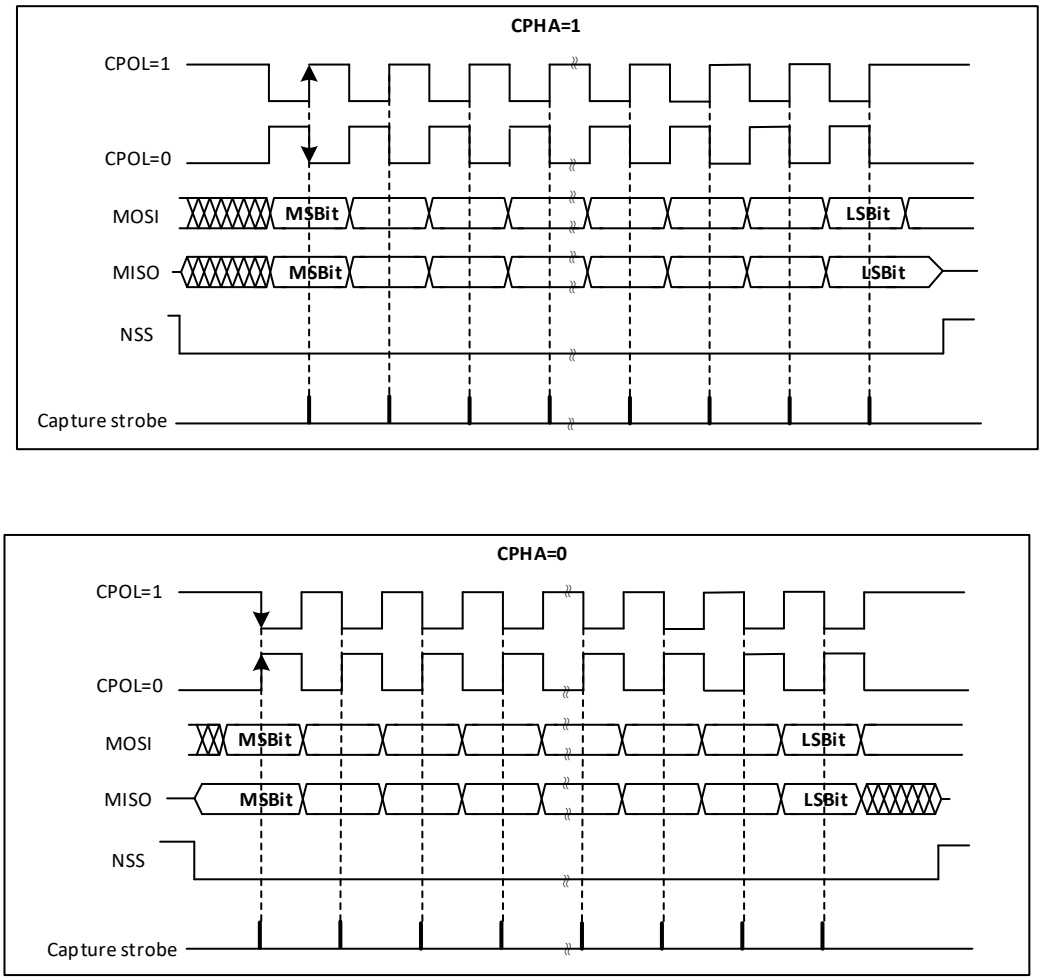


Figure 21-8 Data clock timing diagram

The order of data bits depends on LSBFIRST bit setting.

21.3.6.2. Data frame format

Through the LSBFIRST bit (SPI_CR 1 register), the SPI sh ift register can be set to MSB-FIRST or LSB-FIRST. Select the number of bits in the data frame by using the DS bit (SPI_CR2 register). It can be selected as 8-bit or 16-bit length, and this setting applies to both sending and receiving.

21.3.7. SPI configuration

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 related GPIO registers: configure MOSI, MISO and SCK pins
2. Write SPI_CR1 register
   1） Configure the clock baud rate via BR[2:0] (not required for slave mode)
   2） Configure CPOL and CPHA
   3） simplex or half -duplex mode by RXONLY or B IDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be active at the same time)
   4） Configure LSBFIRST
   5） Configure SSM and SSI
   6） Configure the MSTR bit (in multi-master In NSS configuration, if the host is configured to prevent MODF

errors, avoid NSS conflict state)

3. Write SPI_CR2 register
   1） Configure DS bit, select the number of data frame bits
   2） Configure SSOE (not required for slave mode)
   3） Configure the FRXTH bit. RXFIFO thresholds must be aligned with the number of bits accessed to the SPI_DR register

21.3.8. 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.

21.3.9. Data transmission and reception procedures

21.3.9.1. 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).

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).

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.

If the next data is received when the RXFIFO is full, an overrun event occurs. 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.

21.3.9.2. 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.

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 28.5.5: 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.

21.3.9.3. 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. 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.

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.
- When transactions' streams from 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).

21.3.9.4. Data packing

When the data frame size fits into one byte (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. First, SPI uses the pattern stored in the low-order bits of the word being accessed, followed by the high-order bits.

The figure below provides the data packing process. After a single 16- bit access by the sender, two data frames are sent. On the receiver side, if the RXFIFO threshold is set to 16 bits (FRXTH = 0), the sequence will be generated immediately on the RXNE event. In response to the RXNE event, the receiver accesses 2 data frames through a 16-bit read of the SPI_DR register. On the receiver side, the setting of the RxFIFO threshold and subsequent read accesses must remain aligned, otheRWise data will be lost.

On the sender side, it is sufficient to write the last data frame of the odd sequence with 8- bit access. In order to generate an RxNE event, for an odd number of data frames, the receiver must change the Rx_FIFO threshold for the last data frame received.

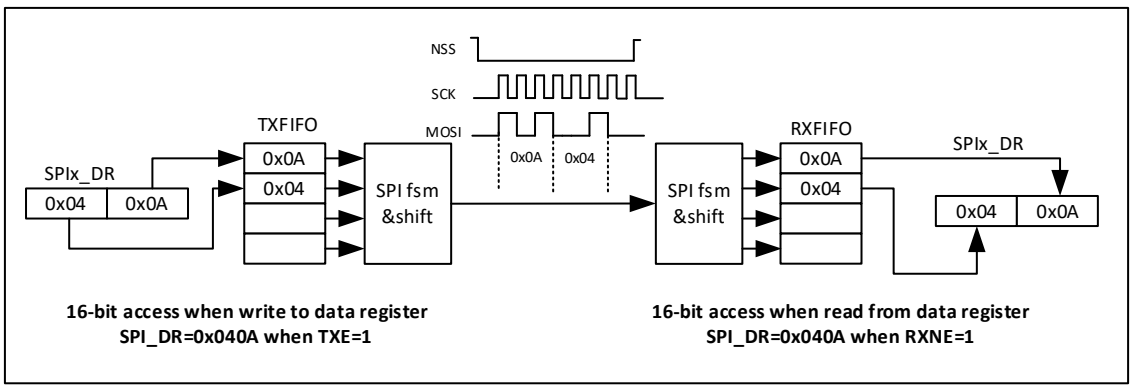


Figure 21-9 Packing data in FIFO for transmission and reception

21.3.9.5. 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. For simplicity, the LSBFIRST = 0, CPOL = 0 and CPHA = 1 setting is used as a common assumption here.

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 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.
5. In Data packed mode, TxE and RxNE events are paired, and each read/write FIFO access is 16 bits wide (until the number of data frames are even). If TxFIFO is 3/4 full, the FTLVL state stops at FIFO full level. That's why the last odd frame cannot be stored before the TxFIFO becomes 1/2 full. The data frame is stored in TxFIFO in 8 -bit access mode.
6. 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).

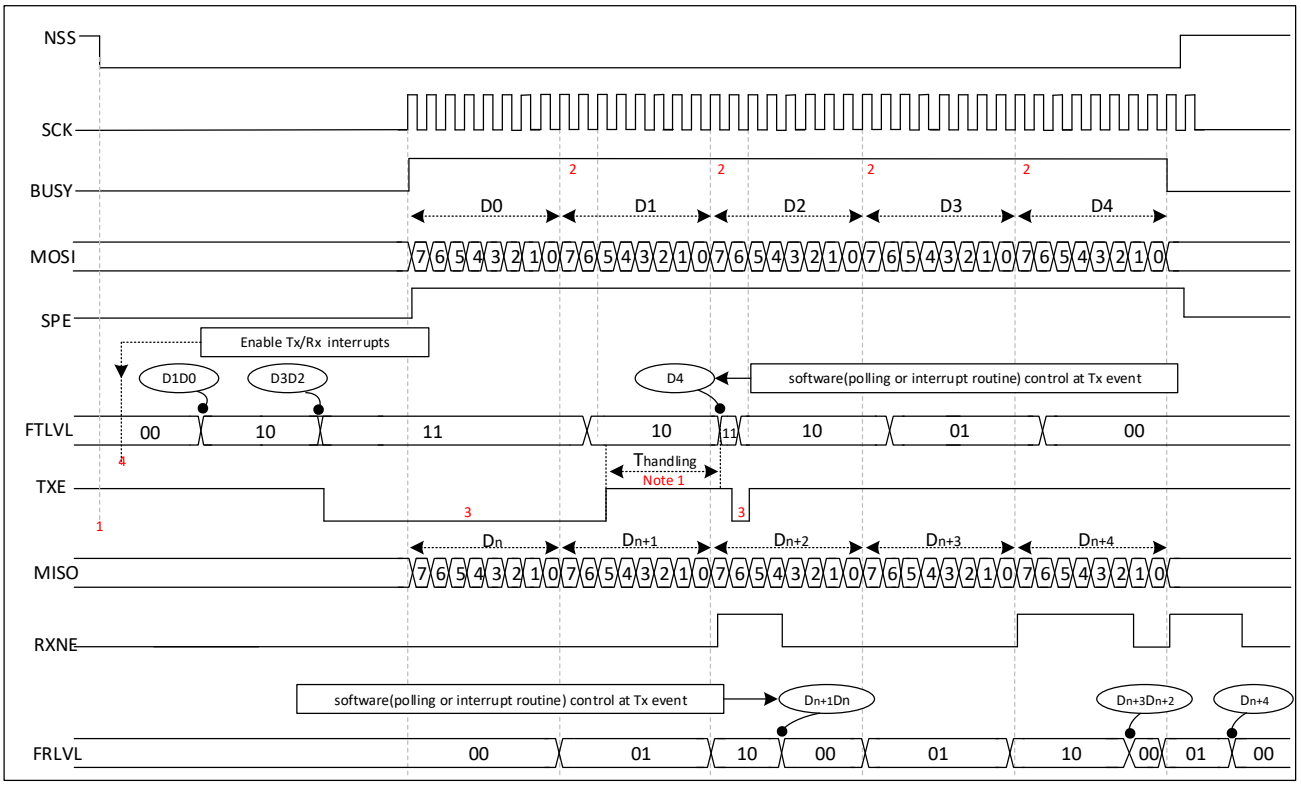


Figure 21-10 Master full-duplex communication diagram(bit frame = 8, FRXTH = 0)

Note1: $T_{handling}$ means cpu writes data to Tx fifo time spent.

21.3.10. Status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

21.3.10.1. 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.

21.3.10.2. 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.

21.3.10.3. Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). This flag indicates the state of the SPI communication layer.

When it is set to '1', it indicates that the SPI is busy communicating, with one exception: in the bidirectional receive mode of master mode (MSTR = 1, BDM = 1 and BDOE = 0), the BSY flag is held during reception to low.

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, so it needs to strictly follow the procedure below.

The BSY flag is also useful for preventing write collisions in a multimaster system.

Except for the bidirectional receive mode of the master mode (MSTR = 1, BDM = 1 and BDOE = 0), the BSY flag is set to '1' when the transmission starts.

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: it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

21.3.11. Error flags

21.3.11.1. 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.

21.3.11.2. 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 did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited.

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.

21.3.12. SPI interrupts

Table 21-1 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	ERRIE

21.4. SPI register

The SPI registers have to be accessed by 16 bits and 32 bits, and the DR registers support 32 bits, 16 bits and 8 bits.

21.4.1. SPI control register 1 (SPI_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BI-DIMODE	BIDIOE	Res	Res	Res	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
RW	RW				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	Bidirectional data mode enable 0: 2-line unidirectional data mode 1: 1-line bidirectional data mode
14	BIDIOE	RW	0	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)

				In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.
13	Reserved		0	
12	Reserved		0	
11	Reserved	RES	-	Reserved
10	RXONLY	RW	0	Receive control only. This bit, together with the BIDIMODE bit, determines the transfer direction in "2-wire unidirectional" mode. In the configuration of multiple slave devices, this bit is set to 1 on the slave device that is not accessed, so that only the slave device that is accessed has output, so there will be no data conflict on the data line. 0: Full-duplex (Transmit and receive) 1: Output disabled (Receive-only mode)
9	SSM	RW	0	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
8	SSI	RW	0	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.
7	LSBFIRST	RW	0	Frame format 0: Data is transmitted with the MSB first. 1: Data is transmitted with the LSB first. This bit should not be changed when communication is ongoing
6	SPE	RW	0	SPI enable 0: SPI disabled 1: SPI enable
5:3	BR[2:0]	RW	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: $f_{PCLK}/256$ These bits should not be changed when communication is ongoing. Note: In slave mode, the fastest baud rate only supports $f_{PCLK}/4$.
2	MSTR	RW	0	Master selection 0: Slave configuration 1: Master configuration Note: This bit should not be changed when communication is ongoing.
1	CPOL	RW	0	Clock polarity 0: CK to 0 when idle 1: CK to 1 when idle This bit should not be changed when communication is ongoing.
0	CPHA	RW	0	Clock phase 0: The first clock transition is the first data capture edge 1: The second clock transition is the first data capture edge This bit should not be changed when communication is ongoing.

21.4.2. SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SLVFM	Res	Res	FRXTH	DS	Res	Res	Res	TXEIE	RXNEIE	ERRIE	Res	Res	SSOE	Res	Res
RW			RW	RW				RW	RW	RW			RW		

Bit	Name	R/W	Reset Value	Function
31:16	Reserved		-	Reserved
15	SLVFM	RW	0	Slave fast mode enable 0: Slave normal mode, the speed of the slave mode supporting the fastest SPI clock is less than pclk/4 1: Slave fast mode, can support SPI clock speed in slave mode up to pclk/4 Note: When the speed of SPI clock is less than pclk/4, this register bit must not be set.
14:13	Reserved	-	-	-
12	FRXTH	RW	0	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)
11	DS	RW	0	SPI transmission data length 0: 8-bit data frame transmission 1: 16-bit data frame transmission
10:8	Reserved	-	-	-
7	TXEIE	RW	0	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
6	RXNEIE	RW	0	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
5	ERRIE	RW	0	Error interrupt enable 0: Error interrupt is masked 1: Error interrupt is enabled This bit controls the generation of an interrupt when an error condition occurs (CRCERR, VR, MODF in SPI mode).
4:3	Reserved	RES	-	Reserved
2	SSOE	RW	0	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.
1:0	Reserved	RES	-	Reserved

Note :

There are a total of 4 combinations of FRXTH and DS, but the process of restricting the use of the software is as follows:

1. This bit should be 0 if DS = F is configured (ie the transmission data length is 16)
2. If DS = 7 is configured (that is, the transmission data length is 8), the following two situations need to be distinguished:
   1） If the number of data frames of communication is 1 frame of data, you need to set this bit to 1
   2） If the number of communication data frames is greater than 1 frame data, clear this bit to 0

21.4.3. SPI status register (SPI_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	FTLVL [1:0]		FRLVL [1:0]		Res	BSY	OVR	MODF	Res	Res	Res	TXE	RXNE
			R	R	R	R		R	R	R				R	R

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	RES	-	Reserved
12:11	FTLVL	R	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)
10:9	FRLVL	R	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
7	BSY	R	0	Busy flag *0:* SPI (or I2S) not busy 1: SPI (or I2S) is busy in communication or Tx buffer is not empty
6	OVR	R	0	Overrun flag 0: No overrun occurred 1: Overrun occurred This flag is set by hardware and reset by a software sequence.
5	MODF	R	0	Mode fault 0: No mode fault occurred 1: Mode fault occurred This flag is set by hardware and reset by a software sequence.
4:2	Reserved		0	
1	TXE	R	1	Transmit buffer empty 0: Tx buffer not empty 1: Tx buffer empty
0	RXNE	R	0	Receive buffer not empty 0: Rx buffer empty 1: Rx buffer not empty

21.4.4. SPI data register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:0	DR[15:0]	RW	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. Note:Depending on the DS bit (data frame width selection), data transmission or reception is 8-bit or 16-bit. For 8-bit data frames, the data registers are sent and received based on right -aligned 8-bit data.

				When in receive mode, DR [15:8] is set to 0 by hardware. For 16-bit data frame, the data register is 16-bit, and the entire DR [15:0] is used for transmit and receive.

21.4.5. SPI register map

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	BIDIMODE	BIDIOE	Res.	Res.	Res.	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
0x04	SPI_CR2	SLVFM	Res.	Res.	FRXTH	DS	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	Res.	Res.	SSOE	Res.	Res.
	Reset value	0			0	0				0	0	0			0		
0x08	SPI_SR	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		Res.	BSY	OVR	MODEF	Res.	Res.	Res.	TXE	RXNE
	Reset value				0	0	0	0		0	0	0				1	0
0x0C	SPI_DR	DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22. Debug support

22.1. Overview

This devices are built around a Cortex-M0+ 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 MCUs.One interface for debug is available: serial wire. The debugging function in M0+ CPU Core is a set of ARM CoreSight Design kit.

The ARM Cortex®-M0 core provides integrated on-chip debug support. It is comprised of:

- SW-DP: serial wire
- BPU: Break point unit
- DWT: Data watchpoint trigger

It also includes debug features dedicated to this chip:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

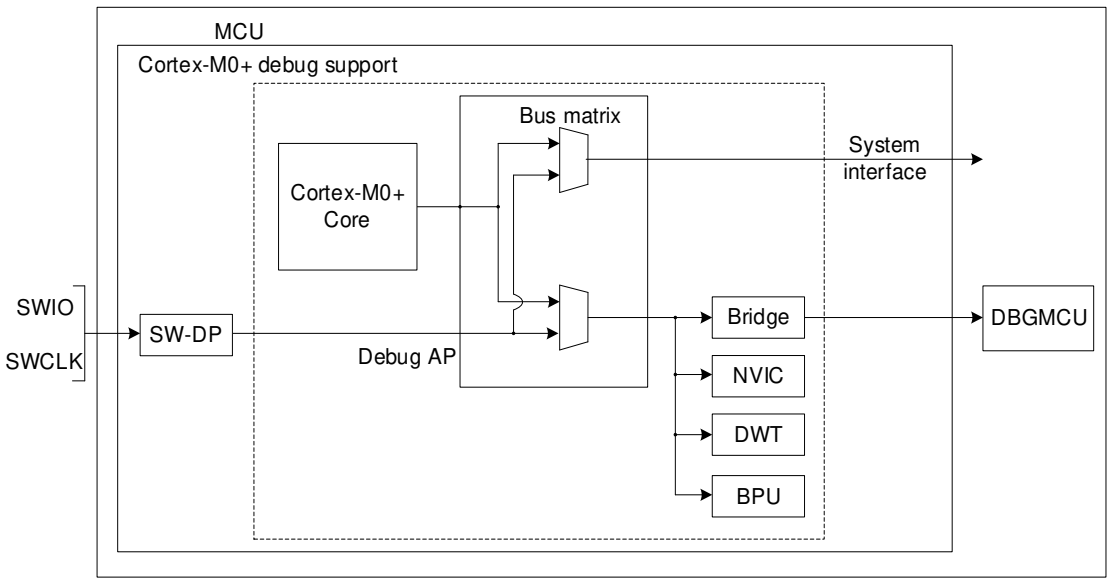


Figure 22-1 Block diagram of MCU debug support

22.2. Pinout and debug port pins

22.2.1. SWD port pins

Two pins are used to debug, these pins are available on all packages.

Table 22-1 SWD port pins

SW-DP Pin name	SW debug port		Pin assignment
	Type	Debug assignment	
SWDIO	I/O	Serial Wire Data Input/Output	PA13

SWDCLK	I	Serial Wire Clock	PA14

22.2.2. SW-DP pin assignment

After reset (SYSRESETn or PORESETn), the pins used for the SW-DP are assigned as dedicated pins which are immediately usable by the debugger host.

However, the MCU offers the possibility to disable the SWD port and can then release the associated pins for general-purpose I/O (GPIO) usage.

22.2.3. Internal pull-up & pull-down on SWD pins

Once the SW I/O is released by the user software, the GPIO controller takes control of these pins. The reset states of the GPIO control registers put the I/Os in the equivalent states:

- SWDIO: input pull-up
- SWCLK: input pull-down

Having embedded pull-up and pull-down resistors removes the need to add external resistors.

22.3. ID codes and locking mechanism

here are several ID codes inside the MCU. It is recommended that Keil, IAR and other tools use this ID Code (located at 0x4001 5800) locks debugging.

After the chip is powered on, the hardware reads the 0x1FFF 0FF8 address of the flash 's factory config.byte and loads it into the DBG _IDCODE register.

22.4. SWD debug port

22.4.1. SWD 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.

22.4.2. SWD protocol sequence

Each sequence consist of three phases:

- Packet request (8 bits) transmitted by the host
- Acknowledge response (3 bits) transmitted by the target
- Data transfer phase (33 bits) transmitted by the host or the target

Table 22-2 Packet request (8 bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request

		1: Read Request
4:3	A[3:2]	Address field of the DP or AP registers
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

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 22-3 ACK response (3 bits)

Bit	Name	Description
[2:0]	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 22-4 DATA transfer (33 bits)

Bit	Name	Description
[31:0]	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.

22.4.3. 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 0x0BB11477 (corresponding to Cortex®-M0).

22.4.4. 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 toaccept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state) This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

22.4.5. SW-DP registers

Access to these registers are initiated when APnDP = 0.

A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is set to the default ARM code for Cortex-M0: 0x0BB11477 (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.
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

22.4.6. SW-AP registers

Address	A[3:2] value	Description
0x0	00	Reserved
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)
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)

22.5. Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the debug access port. It consists of four registers:

Table 22-5 Core debug registers

Register	Description
DHCSR	32bit Debug halting control and status register
DCRSR	17bit Debug Core register selector register
DHCDR	32bit debug Core register Data register
DEMCR	32bit debug exception and monitor control register

These registers are not reset by a system reset. They are only reset by a power-on reset. Refer to the Cortex®-M0+ 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

22.6. Break point unit (BPU)

The Cortex-M0+ BPU implementation provides four breakpoint registers. The BPU is a subset of the Flash Patch and Breakpoint (FPB) block available in ARMv7-M (Cortex-M3 & Cortex-M4).

22.6.1. BPU functionality

The processor breakpoints implement PC based breakpoint functionality.
Refer the ARMv6-M ARM and the ARM CoreSight Components Technical Reference Manual for more information about the BPU CoreSight identification registers, and their addresses and access types.

22.7. Data watchpoint (DWT)

The Cortex-M0 DWT implementation provides two watchpoint register sets

22.7.1. DWT functionality

The processor watchpoints implement both data address and PC based watchpoint functionality.

22.7.2. DWT program counter sample register

A processor that implements the data watchpoint unit also implements the ARMv6-M optional DWT Program Counter Sample Register (DWT_PCSR). This register permits a debugger to periodically sample the PC without halting the processor. This provides coarse grained profiling. See the ARMv6-M ARM for more information.
The Cortex-M0 DWT_PCSR records both instructions that pass their condition codes and those that fail.

22.8. MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog and I2C during a breakpoint

22.8.1. 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: FCLK and HCLK are still active. Consequently, this mode does not impose any restrictions on the standard debug features.In stop mode: The DBG_STOP bit must be set in advance by the debugger.
- In Stop/Standby mode, the DBG_STOP bit must be previously set by the debugger. This enables the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.

22.8.2. Debug support for times, watchdog and IIC

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 I2C, the user can choose to block the SMBUS timeout during a breakpoint.

22.9. DBG register

22.9.1. DBG device ID code register (DBG_IDCODE)

Address offset: 0x00

Only supports 32-bit address access, read only.

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

Bit	Name	R/W	Reset Value	Function
31:0	Reserved	R		

22.9.2. Debug MCU configuration register (DBGMCU_CR)

This register configures the MCU low power mode in debug state.

This register is asynchronously reset by a power-on reset (not a system reset). It can be written by the debugger under system reset.

If the debugger host does not support this feature, it is still possible for the software user to write to these registers.

Address offset: 0x04

Reset value: 0x0000 0000 (will not be 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	Res	Res	DBG_ STOP	Res
														RW	

Bit	Name	R/W	Reset Value	Function
31:2	Reserved			
1	DBG_STOP	RW	0	Debug Stop mode 0: (FCLK = Off, HCLK = Off) In STOP mode, the clock controller disables HCLK and FCLK. When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the HSI). Consequently, the software must reprogram the clock controller to enable the clock configuration. 1: (FCLK = on, HCLK = on). When entering STOP mode, HSI will not be turned off, and FCLK and HCLK are generated by I. When exiting STOP mode, if the clock control needs to be changed, the software needs to be reconfigured.
0	Reserved			

22.9.3. DBG APB freeze register 1 (DBG_APB_FZ1)

This register is used to configure the clock of timer, IWDG, under debug. This register is asynchronously reset by a power-on reset (not a system reset). It can be written by the debugger under system reset.

Address offset: 0x08

Power on Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	DBG_LPTIM_STOP	RW	0	When the CPU is stopped, the counter clock control bit of the L PTIM 0: enable 1: Disable
30:13	Reserved			
12	DBG_IWDG_STOP	RW	0	When the CPU is stopped, the clock control bit of the IWDG counter 0: enable 1: Disable
11:0	Reserved	-	-	-

22.9.4. DBG APB freeze register 2 (DBG_APB_FZ2)

This register is used to configure the clock control of timer under debug. This register is asynchronously reset by a power-on reset (not a system reset). It can be written by the debugger under system reset.

Address offset: 0x0C

Power on Reset value: 0x0000 0000

Only supports 32-bit address access, read only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_ TIM16_STOP	Res
														RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DBG_ TIM1_STOP	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
				RW											

Bit	Name	R/W	Reset Value	Function
31:18	Reserved	-	-	-
17	DBG_TIM16_STOP			When the CPU is stopped, the clock control bit of the TIM16 counter 0: Enable 1: Disable
16:12	Reserved	-	-	-
11	DBG_TIM1_STOP			When the CPU is stopped, the clock control bit of the TIM1 counter 0: Enable 1: Disable
10:0	Reserved	-	-	-

22.9.5. DBG register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBG_ID-CODE	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DBG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STO	Res.
	Reset value																															0	
0x008	DBG_APB_FZ1	DBG_LPTIM_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0																			0												
0x00C	DBG_APB_FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM16_ST	Res.	Res.	Res.	Res.	Res.	DBG_TIM1_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0						0											

23. Version history

Version	Date	Description
V1.0	2022.10.20	Initial version