User Manual

MM32F103xx

32-bit Microcontroller Based on ARM Cortex M3 Core

Version: 1.67_n

We reserve the right to change relevant information without notice.

Table of Contents

1 Memory and Bus Architecture 1

1.1 System architecture . . . 1

1.2 Memory organization . . . 2

1.2.1 Introduction . . . 2

1.2.2 Memory map and register addressing . . . 2

1.3 Embedded SRAM . . . 4

1.4 Overview of FLASH memory . . . 4

1.5 Boot configuration . . . 5

2 Embedded flash (FLASH) 7

2.1 Main features . . . 7

2.2 Functional description . . . 7

2.2.1 Structure . . . 7

2.2.2 Reading flash . . . 8

2.2.3 Programming and erasing flash . . . 9

2.3 Storage protection . . . 16

2.3.1 Write protection of main space . . . 16

2.3.2 Write protection of option bytes . . . 16

2.4 Flash interrupt . . . 16

2.5 Description of option bytes . . . 16

2.6 Description of Flash register . . . 18

2.6.1 Flash access control register(FLASH_ACR) . . . 19

2.6.2 Flash access control register(FLASH_KEYR) . . . 19

2.6.3 Flash OPTKEY register(FLASH_OPTKEYR) . . . 20

2.6.4 Flash status register(FLASH_SR) . . . 20

2.6.5 Flash control register(FLASH_CR) . . . 21

2.6.6 Flash address register(FLASH_AR) . . . 23

2.6.7 Option byte register(FLASH_OBR) . . . 23

2.6.8 Write protection register(FLASH_WRPR) . . . 24

3 Cyclic redundancy check calculation unit (CRC) 25

3.1 CRC introduction . . . 25

3.2 CRC main features . . . 25

3.3 CRC Functional description . . . 26

3.4 CRC register . . . 26

3.4.1 CRC data register(CRC_DR) . . . 26

3.4.2 CRC independent data register(CRC_IDR) . . . 27

3.4.3 CRC control register(CRC_CTRL) . . . 27

4 Power control (PWR) 28

4.1 Power supply . . . 28

4.1.1 Independent A/D converter supply and reference voltage . . . 28

4.1.2 Battery backup domain . . . 29

4.1.3 Voltage regulator . . . 29

4.2 Power supply supervisor . . . 29
4.2.1 Power on reset (POR)/power down reset (PDR) . . . 30
4.2.2 Programmable voltage detector (PVD) . . . 30
4.3 Low-power modes . . . 31
4.3.1 Slowing down system clocks . . . 32
4.3.2 Peripheral clock gating . . . 32
4.3.3 Sleep Mode . . . 32
4.3.4 Stop mode . . . 33
4.3.5 Standby mode . . . 34
4.3.6 Automatic wake-up (AWU) in low power mode . . . 36
4.4 Power control registers . . . 36
4.4.1 Power control registers(PWR_CR) . . . 36
4.4.2 Power control/status register(PWR_CSR) . . . 37

5 Backup registers (BKP) 40
5.1 BKP introduction . . . 40
5.2 BKP main features . . . 40
5.3 BKP functional description . . . 40
5.3.1 Tamper detection . . . 40
5.3.2 RTC calibration . . . 41
5.4 BKP description . . . 41
5.4.1 Backup data register n(BKP_DRn)(n = 1...10) . . . 41
5.4.2 RTC clock calibration register(BKP_RTCCR) . . . 42
5.4.3 Backup control register(BKP_CR) . . . 42
5.4.4 Backup control/status register(BKP_CSR) . . . 43

6 Reset and clock control (RCC) 45
6.1 Reset . . . 45
6.1.1 System reset . . . 45
6.1.2 Power reset . . . 45
6.1.3 Backup domain reset . . . 46
6.2 Clocks . . . 46
6.2.1 HSE clock . . . 48
6.2.2 HSI clock . . . 49
6.2.3 PLL . . . 49
6.2.4 LSE clock . . . 49
6.2.5 LSI clock . . . 50
6.2.6 System clock (SYSCLK) selection . . . 50
6.2.7 Clock security system (CSS) . . . 50
6.2.8 RTC clock . . . 51
6.2.9 Watchdog clock . . . 51
6.2.10 Clock-out capability . . . 51
6.3 RCC Register file and memory mapping description . . . 51
6.3.1 Clock control register(RCC_CR) . . . 52
6.3.2 Clock configuration register(RCC_CFGR) . . . 54

6.3.3 Clock interrupt register(RCC_CIR) . . . 56
6.3.4 APB2 peripheral reset register(RCC_APB2RSTR) . . . 59
6.3.5 APB1 peripheral reset register(RCC_APB1RSTR) . . . 61
6.3.6 AHB peripheral clock enable register(RCC_AHBENR) . . . 63
6.3.7 APB2 peripheral clock enable register(RCC_APB2ENR) . . . 63
6.3.8 APB1 peripheral clock enable register (RCC_APB1ENR) . . . 65
6.3.9 Backup domain control register(RCC_BDCR) . . . 67
6.3.10 Control status register(RCC_CSR) . . . 68
6.3.11 System configuration register(RCC_SYSCFG) . . . 70

7 General-Purpose and Alternate Functions I/O (GPIO and AFIO) 71

7.1 GPIO functional description . . . 71
7.1.1 General-Purpose I/O(GPIO) . . . 72
7.1.2 Atomic bits set or reset . . . 73
7.1.3 External interrupt/wakeup lines . . . 73
7.1.4 Alternate functions . . . 73
7.1.5 Software remapping of I/O alternate functions . . . 74
7.1.6 GPIO locking mechanism . . . 74
7.1.7 Input configuration . . . 74
7.1.8 Output configuration . . . 75
7.1.9 Alternate functions configuration . . . 76
7.1.10 Analog configuration . . . 77
7.1.11 GPIO configurations for device peripherals . . . 78
7.2 GPIO register description . . . 80
7.2.1 Port configuration low register(GPIOx_CRL)(x = A..D) . . . 80
7.2.2 Port configuration high register(GPIOx_CRH)(x = A..D) . . . 81
7.2.3 Port input data register(GPIOx_IDR)(x = A..D) . . . 82
7.2.4 Port output data register(GPIOx_ODR)(x = A..D) . . . 83
7.2.5 Port set/reset register(GPIOx_BSRR)(x = A..D) . . . 83
7.2.6 Port bits reset register(GPIOx_BRR)(x = A..D) . . . 84
7.2.7 Port configuration lock register(GPIOx_LCKR)(x = A..D) . . . 84
7.3 Multiplexed I/O and Debug Configuration (AFIO) . . . 85
7.3.1 Use OSC32_IN/OSC32_OUT as GPIO port PC14/PC15 . . . 85
7.3.2 Use the OSC_IN/OSC_OUT pin as GPIO PD0/PD1 . . . 85
7.3.3 CAN Multiplexing Function Remapping . . . 85
7.3.4 JTAG/SWD Multiplexing Feature Remapping . . . 86
7.3.5 Timer Multiplexing Function Remapping . . . 86
7.3.6 UART Reuse Function Remapping . . . 87
7.3.7 I2C1 Multiplexing Function Remapping . . . 88
7.3.8 SPI Multiplexing Function Remapping . . . 88
7.4 AFIO register description . . . 88
7.4.1 Multiplex Remapping and Debug I/O Configuration Register (AFIO_MAPR) . . . 89
7.4.2 External Interrupt Configuration Register 1 (AFIO_EXTICR1) . . . 91
7.4.3 External Interrupt Configuration Register 2 (AFIO_EXTICR2) . . . 92

7.4.4 External Interrupt Configuration Register 3 (AFIO_EXTICR3) . . . 92
7.4.5 External Interrupt Configuration Register 1 (AFIO_EXTICR4) . . . 93

8 Interrupts and events (EXTI) 94

8.1 Nested Vectored Interrupt Controller . . . 94
8.1.1 SysTick calibration value register . . . 94
8.1.2 Interrupt and exception vectors . . . 94
8.2 External interrupt/event controller (EXTI) . . . 96
8.2.1 Main features . . . 96
8.2.2 Block diagram . . . 97
8.2.3 Wakeup event management . . . 97
8.2.4 Functional description . . . 98
8.2.5 External interrupt/event line mapping . . . 98
8.3 EXTI register description . . . 99
8.3.1 Interrupt Mask Register(EXTI_IMR) . . . 100
8.3.2 Event Mask Register(EXTI_EMR) . . . 100
8.3.3 Rising Trigger Selection Register(EXTI_RTSR) . . . 101
8.3.4 Falling Trigger Selection Register(EXTI_FTSR) . . . 101
8.3.5 Software Interrupt Event Register(EXTI_SWIER) . . . 102
8.3.6 Pending register(EXTI_PR) . . . 102

9 Direct memory access controller (DMA) 104

9.1 DMA introduction . . . 104
9.2 DMA main features . . . 104
9.3 Functional description . . . 105
9.3.1 DMA transactions . . . 105
9.3.2 DMA arbiter . . . 106
9.3.3 DMA channels . . . 106
9.3.4 Programmable data width, data alignment and endians . . . 107
9.3.5 Error management . . . 110
9.3.6 Interrupts . . . 111
9.3.7 DMA request mapping . . . 111
9.4 DMA register description . . . 113
9.4.1 DMA interrupt status register(DMA_ISR) . . . 113
9.4.2 DMA interrupt flag clear register(DMA_IFCR) . . . 114
9.4.3 DMA channel x configuration register(DMA_CCRx) (x = 1…7) . . . 115
9.4.4 DMA channel x number of data register(DMA_CNDTRx) (x = 1…7) . . . 117
9.4.5 DMA channel x peripheral address register(DMA_CPARx) (x = 1…7) . . . 117
9.4.6 DMA channel x memory address register(DMA_CMARx) (x = 1…7) . . . 118

10 Analog-to-digital converter (ADC) 119

10.1 ADC introduction . . . 119
10.2 ADC main features . . . 119
10.3 ADC functional description . . . 120
10.3.1 ADC on-off control . . . 120
10.3.2 Channel selection . . . 121

10.4 ADC operating mode . . . 121
10.4.1 Single conversion mode . . . 121
10.4.2 Single-cycle scan mode . . . 121
10.4.3 Continuous scan mode . . . 122
10.4.4 DMA request . . . 123
10.5 Data alignment . . . 123
10.5.1 Programmable resolution . . . 123
10.5.2 Programmable sample time . . . 124
10.6 Conversion on external trigger . . . 124
10.7 Temperature sensor . . . 124
10.8 Internal reference voltage . . . 125
10.9 Monitoring of AD conversion results in window comparator mode . . . 125
10.10 ADC register description . . . 125
10.10.1 A/D data register(ADC_ADDATA) . . . 126
10.10.2 A/D configuration register(ADC_ADCFG) . . . 127
10.10.3 A/D control register(ADC_ADCR) . . . 128
10.10.4 A/D channel select register(ADC_ADCHS) . . . 130
10.10.5 A/D window compare register(ADC_ADCMPR) . . . 131
10.10.6 A/D status register(ADC_ADSTA) . . . 131
10.10.7 A/D data register(ADC_ADDR0 ~ 8) . . . 132

11 Advanced-Control Timer (TIM1) 134
11.1 TIM1 introduction . . . 134
11.2 Main features . . . 134
11.3 Functional description . . . 135
11.3.1 Time-base unit . . . 135
11.3.2 Counter modes . . . 137
11.3.3 Repetition counter . . . 146
11.3.4 Clock selection . . . 147
11.3.5 Capture/compare channels . . . 151
11.3.6 Input capture mode . . . 153
11.3.7 PWM input mode . . . 154
11.3.8 Forced output mode . . . 155
11.3.9 Output compare mode . . . 156
11.3.10 PWM mode . . . 157
11.3.11 Complementary outputs and dead-time insertion . . . 160
11.3.12 Using the break function . . . 161
11.3.13 Clearing the OCxREF signal on an external event . . . 164
11.3.14 Six-step PWM generation . . . 165
11.3.15 One-pulse mode . . . 166
11.3.16 Encoder interface mode . . . 168
11.3.17 Timer input XOR function . . . 170
11.3.18 Interfacing with Hall sensors . . . 170
11.3.19 TIMx and external trigger synchronization . . . 172

11.3.20 Timer synchronization . . . 175
11.3.21 Debug mode . . . 176
11.4 Register description . . . 176
11.4.1 Control register 1(TIMx_CR1) . . . 176
11.4.2 Control register 2(TIMx_CR2) . . . 178
11.4.3 Slave mode control register(TIMx_SMCR) . . . 181
11.4.4 DMA/interrupt enable register (TIMX_DIER) . . . 185
11.4.5 Status register(TIMx_SR) . . . 187
11.4.6 Event generation register (TIMx_EGR) . . . 189
11.4.7 Capture/compare mode register 1 (TIMx_CCMR1) . . . 190
11.4.8 Capture/compare mode register 2(TIMx_CCMR2) . . . 195
11.4.9 Capture/compare enable register(TIMx_CCER) . . . 197
11.4.10 Counter(TIMx_CNT) . . . 201
11.4.11 Prescaler(TIMx_PSC) . . . 201
11.4.12 Auto-reload register(TIMx_ARR) . . . 201
11.4.13 Repetition counter register(TIMx_RCR) . . . 202
11.4.14 Capture/compare register 1(TIMx_CCR1) . . . 202
11.4.15 Capture/compare register2(TIMx_CCR2) . . . 203
11.4.16 Capture/compare register 3(TIMx_CCR3) . . . 203
11.4.17 Capture/compare register 4(TIMx_CCR4) . . . 204
11.4.18 Break and dead-time register(TIMx_BDTR) . . . 205
11.4.19 DMA control register(TIMx_DCR) . . . 208
11.4.20 DMA address for full transfer(TIMx_DMAR) . . . 209

12 16-bit General-Purpose Timer (TIMx 16 Bit) 211
12.1 TIMx introduction . . . 211
12.2 TIMx Main features . . . 211
12.3 TIMx Functional description . . . 212
12.3.1 Time-base unit . . . 212
12.3.2 Counter modes . . . 214
12.3.3 Clock selection . . . 223
12.3.4 Capture/compare channels . . . 227
12.3.5 Input capture mode . . . 229
12.3.6 PWM input mode . . . 230
12.3.7 Forced output mode . . . 231
12.3.8 Output compare mode . . . 231
12.3.9 PWM mode . . . 233
12.3.10 One-pulse mode . . . 236
12.3.11 Clearing the OCxREF signal on an external event . . . 237
12.3.12 Encoder interface mode . . . 238
12.3.13 Timer input XOR function . . . 241
12.3.14 Timers and external trigger synchronization . . . 241
12.3.15 Timer synchronization . . . 244
12.3.16 Debug mode . . . 249

12.4 TIMx register description . . . 250
12.4.1 Control register 1(TIMx_CR1) . . . 250
12.4.2 Control register 2(TIMx_CR2) . . . 252
12.4.3 Slave mode control register(TIMx_SMCR) . . . 254
12.4.4 DMA/interrupt enable register(TIMx_DIER) . . . 257
12.4.5 Status register(TIMx_SR) . . . 259
12.4.6 Event generation register (TIMx_EGR) . . . 260
12.4.7 Capture/compare mode register 1(TIMx_CCMR1) . . . 261
12.4.8 Capture/compare mode register 2(TIMx_CCMR2) . . . 266
12.4.9 Capture/compare enable register(TIMx_CCER) . . . 268
12.4.10 Counter(TIMx_CNT) . . . 270
12.4.11 Prescaler(TIMx_PSC) . . . 270
12.4.12 Auto-reload register(TIMx_ARR) . . . 270
12.4.13 Capture/compare register 1(TIMx_CCR1) . . . 271
12.4.14 Capture/compare register2(TIMx_CCR2) . . . 271
12.4.15 Capture/compare register 3(TIMx_CCR3) . . . 272
12.4.16 Capture/compare register 4(TIMx_CCR4) . . . 272
12.4.17 DMA control register(TIMx_DCR) . . . 273
12.4.18 DMA address for full transfer(TIMx_DMAR) . . . 274

13 Real-time clock (RTC) 276

13.1 RTC introduction . . . 276
13.2 Main features . . . 276
13.3 Functional description . . . 277
13.3.1 Overview . . . 277
13.3.2 Resetting RTC registers . . . 277
13.3.3 Reading RTC registers . . . 278
13.3.4 Configuring RTC registers . . . 278
13.3.5 RTC flag assertion . . . 278
13.4 RTC registers . . . 279
13.4.1 RTC control register high(RTC_CRH) . . . 280
13.4.2 RTC control register low(RTC_CRL) . . . 280
13.4.3 RTC prescaler load register(RTC_PRLH/RTC_PRLL) . . . 282
13.4.4 RTC prescaler divider register(RTC_DIVH/RTC_DIVL) . . . 283
13.4.5 RTC counter register(RTC_CNTH/RTC_CNTL) . . . 284
13.4.6 RTC alarm register high(RTC_ALRH/RTC_ALRL) . . . 285

14 Independent watchdog (IWDG) 286

14.1 (IWDG introduction) . . . 286
14.2 IWDG main features . . . 286
14.3 Functional description . . . 286
14.3.1 Hardware watchdog . . . 287
14.3.2 Register access protection . . . 287
14.3.3 Debug mode . . . 288
14.4 IWDG register description . . . 288

14.4.1 Key register(IWDG_KR) . . . 288
14.4.2 Prescaler register(IWDG_PR) . . . 288
14.4.3 Reload register(IWDG_RLR) . . . 289
14.4.4 Status register(IWDG_SR) . . . 290

15 Window watchdog (WWDG) 292

15.1 WWDG introduction . . . 292
15.2 WWDG main features . . . 292
15.3 Functional description . . . 292
15.4 How to program the watchdog timeout . . . 294
15.5 Debug mode . . . 295
15.6 WWDG register description . . . 295
15.6.1 Control register(WWDG_CR) . . . 296
15.6.2 Configuration register(WWDG_CFGR) . . . 296
15.6.3 Status register(WWDG_SR) . . . 297

16 Universal serial bus full-speed device interface (USB) 298

16.1 USB introduction . . . 298
16.2 USB main features . . . 298
16.3 Functional description . . . 299
16.3.1 Description of USB blocks . . . 300
16.4 Programming considerations . . . 301
16.4.1 Overview of USB transmission . . . 301
16.4.2 USB enumeration . . . 303
16.4.3 USB transfer handling . . . 305
16.4.4 IN token packet . . . 306
16.4.5 OUT token packet . . . 307
16.5 USB register description . . . 307
16.5.1 USB TOP register(USB_TOP) . . . 308
16.5.2 USB interrupt status register(USB_INT_STATE) . . . 309
16.5.3 USB endpoint interrupt status register(EP_INT_STATE) . . . 309
16.5.4 USB endpoint 0 interrupt status register(EP0_INT_STATE) . . . 310
16.5.5 USB interrupt enable register (USB_INT_EN) . . . 311
16.5.6 USB endpoint interrupt enable register (EP_INT_EN) . . . 312
16.5.7 USB endpoint 0 interrupt enable register(EP0_INT_EN) . . . 312
16.5.8 USB endpoint X interrupt status register (EPX_INT_STATE) (X = 1 ~ 4) . . . 313
16.5.9 USB endpoint X interrupt status register(EPX_INT_EN) (X = 1 ~ 4) . . . 314
16.5.10 USB address register (USB_ADDR) . . . 315
16.5.11 USB endpoint enable register(EP_EN) . . . 315
16.5.12 USB data toggle control register(TOG_CTRL1_4) . . . 316
16.5.13 USB setup packet data register(SETUPX) (X = 0 ~ 7) . . . 316
16.5.14 USB transfer packet size register(PACKET_SIZEX)(X = 0 ~ 1) . . . 317
16.5.15 USB endpoint X valid data register(EPX_AVAIL) . . . 317
16.5.16 USB endpoint X control register (EPX_CTRL) . . . 317
16.5.17 USB endpoint X FIFO register (EPX_FIFO) . . . 318

16.5.18 USB endpoint X DMA enable register (EP_DMA) . . . 318
16.5.19 USB endpoint halt register(EP_HALT) . . . 319
16.5.20 USB power control register (USB_POWER) . . . 320

17 Controller area network (CAN) 321

17.1 CAN introduction . . . 321
17.2 CAN main features . . . 321
17.3 CAN general description . . . 321
17.3.1 CAN 2.0B active core . . . 322
17.3.2 CAN block diagram . . . 322
17.3.3 Interface management logic (IML) . . . 323
17.3.4 Transmit buffer (TXB) . . . 323
17.3.5 Receive buffer (RXB, RXFIFO) . . . 323
17.3.6 Acceptance filter (ACF) . . . 323
17.3.7 Bitstream processor (BSP) . . . 324
17.3.8 Bit timing logic (BTL) . . . 324
17.3.9 Error management logic (EML) . . . 324
17.4 CAN operating mode . . . 324
17.4.1 Difference between Basic CAN and Peli CAN modes . . . 324
17.5 CAN Functional description . . . 325
17.5.1 Basic CAN mode . . . 325
17.5.2 Peli CAN mode . . . 326
17.5.3 Transmission handling . . . 329
17.5.4 Reception handling . . . 330
17.5.5 Identifier filtering . . . 331
17.5.6 Message storage . . . 338
17.5.7 Error management . . . 341
17.5.8 Bit-time characteristics . . . 346
17.5.9 Arbitration lost . . . 347
17.5.10 CAN interrupts . . . 348
17.6 Description of CAN register . . . 349
17.6.1 CAN mode register(CAN_MOD) . . . 350
17.6.2 CAN control register(CAN_CR) . . . 351
17.6.3 CAN control register(CAN_CMR) . . . 352
17.6.4 CAN status register(CAN_SR) . . . 354
17.6.5 CAN interrupt register(CAN_IR) . . . 356
17.6.6 CAN interrupt enable register(CAN_IER) . . . 358
17.6.7 CAN acceptance code register(CAN_ACR) . . . 359
17.6.8 CAN acceptance mask register(CAN_AMR) . . . 360
17.6.9 CAN bus timing 0(CAN_BTR0) . . . 361
17.6.10 CAN bus timing 1(CAN_BTR1) . . . 362
17.6.11 CAN transmit identifier register (CAN_TXID0) . . . 363
17.6.12 CAN transmit identifier register 1(CAN_TXID1) . . . 363
17.6.13 CAN arbitration lost capture register(CAN_ALC) . . . 364

17.6.14 CAN error code capture register(CAN_ECC) . . . 367
17.6.15 CAN error warning limit register(CAN_EWLR) . . . 368
17.6.16 CAN RX error count register(CAN_RXERR) . . . 369
17.6.17 CAN TX error count register(CAN_TXERR) . . . 370
17.6.18 CAN transmit frame information register(CAN_SFF) . . . 371
17.6.19 CAN transmit identifier register 0 (CAN_TXID0) . . . 372
17.6.20 CAN transmit identifier register 1(CAN_TXID1) . . . 373
17.6.21 CAN transmit data register 0 (CAN_TXDATA0) . . . 373
17.6.22 CAN transmit data register 1(CAN_TXDATA1) . . . 374
17.6.23 CAN clock divider register (CAN_CDR) . . . 374

18 Serial peripheral interface (SPI) 376

18.1 SPI description . . . 376
18.2 Main features . . . 376
18.3 Functional description . . . 376
18.3.1 General . . . 376
18.3.2 SPI slave mode . . . 381
18.3.3 SPI master mode . . . 381
18.3.4 Status flags . . . 382
18.3.5 Baud rate setting . . . 383
18.3.6 SPI communication using DMA . . . 384
18.4 Register file and memory mapping description . . . 384
18.4.1 Transmit data register(SPI_TXREG) . . . 384
18.4.2 Receive data register(SPI_RXREG) . . . 385
18.4.3 Current status register(SPI_CSTAT) . . . 385
18.4.4 Interrupt status register(SPI_INTSTAT) . . . 386
18.4.5 Interrupt enable register(SPI_INTEN) . . . 388
18.4.6 Interrupt clear register . . . 389
18.4.7 Global control register(SPI_GCTL) . . . 390
18.4.8 General-purpose control register(SPI_CCTL) . . . 391
18.4.9 Baud rate generator(SPI_SPBRG) . . . 392
18.4.10 Receive data count register (SPI_RXDNR) . . . 393
18.4.11 Slave chip select register(SPI_NSSR) . . . 393
18.4.12 Data control register(SPI_EXTCTL) . . . 394

19 I2C interface (I2C) 395

19.1 I2C introduction . . . 395
19.2 I2C main features . . . 395
19.3 I2C protocol . . . 395
19.3.1 Start and Stop conditions . . . 395
19.3.2 Slave address protocol . . . 396
19.3.3 Transmission and reception protocols . . . 397
19.3.4 Transmitting buffer management and generation of Start, Stop, and repeated Start conditions 399
19.3.5 Multiple-master arbitration . . . 401
19.3.6 Clock synchronization . . . 401

19.4 I2C operating mode . . . 402
19.4.1 Slave mode . . . 403
19.4.2 Main mode . . . 405
19.4.3 I2C abort transmission . . . 406
19.5 Communication using DMA . . . 407
19.6 I2C interrupt . . . 407
19.7 I2C register description . . . 408
19.7.1 I2C control register(I2C_CR) . . . 409
19.7.2 I2C destination address register(I2C_TAR) . . . 411
19.7.3 I2C slave address register(I2C_SAR) . . . 411
19.7.4 I2C data command register(I2C_DR) . . . 412
19.7.5 Standard mode I2C clock high counter register(I2C_SSHR) . . . 413
19.7.6 Standard mode I2C clock low counter register(I2C_SSLR) . . . 413
19.7.7 Fast mode I2C clock high counter register(I2C_FSHR) . . . 414
19.7.8 Fast mode I2C clock low counter register(I2C_FSLR) . . . 414
19.7.9 I2C interrupt status register(I2C_ISR) . . . 414
19.7.10 I2C interrupt mask register(I2C_IMR) . . . 415
19.7.11 I2C RAW interrupt register(I2C_RAWISR) . . . 415
19.7.12 I2C reception threshold(I2C_RXTLR) . . . 417
19.7.13 I2C transmission threshold(I2C_TXTLR) . . . 418
19.7.14 I2C combined and independent interrupt clear register(I2C_ICR) . . . 418
19.7.15 I2C clear RX_UNDER interrupt register(I2C_RX_UNDER) . . . 418
19.7.16 I2C clears RX_OVER interrupt register(I2C_RX_OVER) . . . 419
19.7.17 I2C clear TX_OVER interrupt register(I2C_TX_OVER) . . . 419
19.7.18 I2C clear RD_REQ interrupt register(I2C_RD_REQ) . . . 419
19.7.19 I2C clear TX_ABRT interrupt register(I2C_TX_ABRT) . . . 420
19.7.20 I2C clear RX_DONE interrupt register(I2C_RX_DONE) . . . 420
19.7.21 I2C clear ACTIVITY interrupt register (I2C_ACTIV) . . . 421
19.7.22 I2C clear STOP_DET interrupt register(I2C_STOP) . . . 421
19.7.23 I2C clear START_DET interrupt register(I2C_START) . . . 421
19.7.24 I2C clear GEN_CALL interrupt register(I2C_GC) . . . 422
19.7.25 I2C enable register(I2C_ENR) . . . 422
19.7.26 I2C status register(I2C_SR) . . . 423
19.7.27 I2C transmitter buffer level register(I2C_TXFLR) . . . 424
19.7.28 I2C receiver buffer level register(I2C_RXFLR) . . . 424
19.7.29 I2C SDA hold time register(I2C_HOLD) . . . 424
19.7.30 I2C DMA control register(I2C_DMA) . . . 425
19.7.31 I2C SDA setup time register(I2C_SETUP) . . . 425
19.7.32 I2C general call ACK register(I2C_GCR) . . . 425

20 Universal asynchronous receiver transmitter (UART) 427
20.1 UART introduction . . . 427
20.2 UART main features . . . 427
20.3 UART functional description . . . 428

20.3.1 UART character description . . . 429
20.3.2 Transmitter . . . 429
20.3.3 Receiver . . . 431
20.3.4 Fractional baud rate generator . . . 432
20.3.5 Sampling . . . 432
20.3.6 Parity control . . . 433
20.3.7 Hardware flow control . . . 433
20.3.8 Communication using DMA . . . 435
20.4 UART interrupt requests . . . 435
20.5 UART registers . . . 436
20.5.1 UART transmit data register(UART_TDR) . . . 436
20.5.2 UART receive data register(UART_RDR) . . . 437
20.5.3 UART current status register(UART_CSR) . . . 437
20.5.4 UART interrupt status register . . . 438
20.5.5 UART interrupt enable register(UART_IER) . . . 439
20.5.6 UART interrupt clear register(UART_ICR) . . . 439
20.5.7 UART global control register(UART_GCR) . . . 440
20.5.8 UART general control register(UART_CCR) . . . 441
20.5.9 UART baud rate register(UART_BRR) . . . 442
20.5.10 UART fractional baud rate register(UART_FRA) . . . 442

21 Device electronic signature (Device) 444
21.1 Memory size registers . . . 444
21.1.1 Unique device ID register (96 bits) . . . 444
21.2 UID register description . . . 444
21.2.1 UID1 . . . 444
21.2.2 UID2 . . . 445
21.2.3 UID3 . . . 445
21.2.4 UID4 . . . 445

22 Debug Support (DBG) 447
22.1 Overview . . . 447
22.2 SWJ debug port (serial wire and JTAG) . . . 448
22.2.1 Mechanism to select the JTAG-DP or the SW-DP . . . 449
22.3 Pinout and debug port pins . . . 450
22.3.1 SWJ Debug Port Pins . . . 450
22.3.2 Flexible SWJ-DP pin assignment . . . 450
22.3.3 Internal pull-up and pull-down on JTAG pins . . . 451
22.3.4 Using the serial wire and releasing the unused debug pins as GPIOs . . . 452
22.4 JTAG TAP Connection . . . 452
22.5 ID codes and locking mechanism . . . 453
22.5.1 MCU device ID code . . . 453
22.5.2 Boundary scan TAP JTAG ID code . . . 454
22.5.3 CPU JTAG TAP . . . 454
22.5.4 Cortex JEDEC-106 ID Code . . . 454

22.6 JTAG Debug Port 454
22.7 SW debug port 456
22.7.1 SW protocol introduction 456
22.7.2 SW protocol sequence 456
22.7.3 SW-DP state machine (Reset, idle states, ID code) 457
22.7.4 DP and AP read/write accesses 457
22.7.5 SW-DP register 458
22.7.6 SW-AP register 458
22.8 MCU debug module (MCUDBG) 459
22.8.1 Debugg support in low power mode 459
22.8.2 Support timer, watchdog 459
22.8.3 Debug MCU configuration register 459
22.9 Description of DBG Register 460
22.9.1 DBG Control Register(DBG_CR) 460

23 Revision history **462**

Figures

1 System architecture . . . 1
2 Programming Flow . . . 10
3 Page Erasing Process of Flash Register . . . 12
4 Whole Erasing Process of Flash Register . . . 13
5 Option Byte Programming Process . . . 14
6 Option Byte Erasing Process . . . 15
7 CRC Calculation Unit Block Diagram . . . 25
8 Power Supply Overview . . . 28
9 Power on Reset/Power Down Reset Waveform . . . 30
10 PVD Thresholds . . . 31
11 Reset Circuit . . . 46
12 Clock Tree . . . 47
13 Clock Sources . . . 48
14 Basic Structure of I/O Port bits . . . 72
15 Input Floating/Pull Up/Pull Down Configurations . . . 75
16 Output Configuration . . . 76
17 Alternate functions configuration . . . 77
18 High Impedance-analog Configuration . . . 78
19 External Interrupt/Event Controller Block Diagram . . . 97
20 External Interrupt/Event GPIO Mapping . . . 99
21 DMA Block Diagram . . . 105
22 Peripheral DMA Request Mapping . . . 112
23 ADC block diagram . . . 120
24 Timing Diagram of Single Conversion Mode . . . 121
25 Timing Diagram of Enabled Channel During Conversion in Single-cycle Scan Mode . . . 122
26 Timing Diagram of Enabled Channel During Conversion in Continuous Scan Mode . . . 123
27 Data Alignment Modes . . . 123
28 Block Diagram of Advanced-Control Timer . . . 135
29 Counter Timing Diagram with Prescaler Division Change from 1 to 2 . . . 136
30 Counter Timing Diagram with Prescaler Division Change from 1 to 4 . . . 137
31 Counter Timing Diagram, Internal Clock Divided by 1 . . . 138
32 Counter Timing Diagram, Internal Clock Divided by 2 . . . 138
33 Counter Timing Diagram, Internal Clock Divided by 4 . . . 138
34 Counter Timing Diagram, Internal Clock Divided by N . . . 139
35 Counter Timing Diagram, Update Event When ARPE = 0 (TiMx_ARR Not Preloaded) . . . 139
36 Counter Timing Diagram, Update Event When ARPE = 1 (TiMx_ARR Preloaded) . . . 140
37 Counter Timing Diagram, Internal Clock Divided by 1 . . . 141
38 Counter Timing Diagram, Internal Clock Divided by 2 . . . 141
39 Counter Timing Diagram, Internal Clock Divided by 4 . . . 141
40 Counter Timing Diagram, Internal Clock Divided by N . . . 142
41 Counter Timing Diagram, Update Event when Repetition Counter is Not Used . . . 142
42 Counter Timing Diagram, Internal Clock Divided by 1, TIMx_ARR = 6 . . . 143

43 Counter Timing Diagram, Internal Clock Divided by 2 . . . 144
44 Counter Timing Diagram, Internal Clock Divided by 4, TIMx_ARR = 0×36 . . . 144
45 Counter Timing Diagram, Internal Clock Divided by N . . . 145
46 Counter Timing Diagram, Update Event with ARPE = 1(Counter Underflow) . . . 145
47 Counter Timing Diagram, Update Event with ARPE = 1(Counter Overflow)) . . . 146
48 Update Rate Examples Depending on Modes and TIMx_RCR Register Settings . . . 147
49 Control Circuit in Normal Mode, Internal Clock Divided By 1 . . . 148
50 TI2 External Clock Connection Example . . . 148
51 Control Circuit in External Clock Mode 1 . . . 149
52 External Trigger Input Block . . . 150
53 Control Circuit in External Clock Mode 2 . . . 151
54 Capture/Compare Channel (Example: Channel 1 Input Stage) . . . 152
55 Capture/Compare Channel 1 Main Circuit . . . 152
56 Output stage of Capture/Compare Channel (Channels 1 to 3) . . . 153
57 Output stage of Capture/Compare Channel (Channel 4) . . . 153
58 PWM Input Mode Timing . . . 155
59 Output Compare Mode, Toggle on OC1 . . . 157
60 Edge-aligned PWM Waveforms (ARR = 8) . . . 158
61 Center-aligned PWM Waveforms (ARR = 8) . . . 159
62 Complementary Output with Dead-time Insertion . . . 160
63 Dead-time Waveforms with Delay Greater Than the Negative Pulse . . . 161
64 Dead-time Waveforms with Delay Greater than the Positive Pulse . . . 161
65 Output Behavior in Response to A Break . . . 164
66 Clearing TIMx OCxREF . . . 165
67 Six-step PWM, COM Example (OSSR = 1) . . . 166
68 Example of One Pulse Mode . . . 167
69 Example of Counter Operation in Encoder Mode . . . 169
70 Example of Encoder Interface Mode with Inverted IC1FP1 . . . 170
71 Example of Hall Sensor Interface . . . 172
72 Control Circuit in Reset Mode . . . 173
73 Control Circuit in Gated Mode . . . 174
74 Control Circuit in Trigger Mode . . . 174
75 Control Circuit in External Clock Mode 2 + Trigger Mode . . . 175
76 Block Diagram of General-Purpose Timer . . . 212
77 Counter Timing Diagram with Prescaler Division Change from 1 to 2 . . . 213
78 Counter Timing Diagram with Prescaler Division Change from 1 to 4 . . . 214
79 Counter Timing Diagram, Internal Clock Divided by 1 . . . 215
80 Counter Timing Diagram, Internal Clock Divided by 2 . . . 215
81 Counter Timing Diagram, Internal Clock Divided by 4 . . . 215
82 Counter Timing Diagram, Internal Clock Divided by N . . . 216
83 Counter Timing Diagram, Update Event When ARPE = 0 (TiMx_ARR Not Preloaded) . . . 216
84 Counter Timing Diagram, Update Event When ARPE = 1 (TiMx_ARR Preloaded) . . . 217
85 Counter Timing Diagram, Internal Clock Divided by 1 . . . 218
86 Counter Timing Diagram, Internal Clock Divided by 2 . . . 218

87 Counter Timing Diagram, Internal Clock Divided by 4 . . . 218
88 Counter Timing Diagram, Internal Clock Divided by N . . . 219
89 Counter Timing Diagram, Update Event when Repetition Counter is Not Used . . . 219
90 Counter Timing Diagram, Internal Clock Divided by 1, TIMx_ARR = 6 . . . 220
91 Counter Timing Diagram, Internal Clock Divided by 2 . . . 221
92 Counter Timing Diagram, Internal Clock Divided by 4, TIMx_ARR = 0×36 . . . 221
93 Counter Timing Diagram, Internal Clock Divided by N . . . 222
94 Counter Timing Diagram, Update Event with ARPE = 1(Counter Underflow) . . . 222
95 Counter Timing Diagram, Update Event with ARPE = 1(Counter Overflow)) . . . 223
96 Control Circuit in Normal Mode, Internal Clock Divided By 1 . . . 224
97 TI2 External Clock Connection Example . . . 224
98 Control Circuit in External Clock Mode 1 . . . 225
99 External Trigger Input Block . . . 226
100 Control Circuit in External Clock Mode 2 . . . 227
101 Capture/Compare Channel (Example: Channel 1 Input Stage) . . . 228
102 Capture/Compare Channel 1 Main Circuit . . . 228
103 Output Stage of Capture/Compare Channel (Channel 1) . . . 229
104 Output Stage of Capture/Compare Channel (Channel 1) . . . 231
105 Output Compare Mode (Toggle OC1) . . . 233
106 Edge-aligned PWM Waveforms (ARR = 8) . . . 234
107 Center-aligned PWM Waveforms (ARR = 8) . . . 235
108 Example of One Pulse Mode . . . 236
109 Clearing TIMx OCxREF . . . 238
110 Example of Counter Operation in Encoder Mode . . . 240
111 Example of Encoder Interface Mode with Inverted Polarity IC1FP1 . . . 240
112 Control Circuit in Reset Mode . . . 242
113 Control Circuit in Gated Mode . . . 242
114 Control Circuit in Trigger Mode . . . 243
115 Control Circuit in External Clock Mode 2 + Trigger Mode . . . 244
116 Master/Slave Timer Example . . . 245
117 Gating Timer 2 with OC1REF of Timer 1 . . . 246
118 Gating Timer 2 with Enable of Timer 1 . . . 247
119 Triggering Timer 2 with Update of Timer 1 . . . 248
120 Triggering Timer 2 with Enable of Timer 1 . . . 248
121 Triggering Timer 1 and 2 with Timer 1 TI1 input . . . 249
122 RTC block diagram . . . 277
123 RTC second and alarm waveform example with PR=0003, ALARM=00004 . . . 279
124 RTC Overflow waveform example with PR=0003 . . . 279
125 Independent Watchdog Block Diagram . . . 287
126 Watchdog Block Diagram . . . 293
127 Window Watchdog Timing Diagram . . . 295
128 USB Block Diagram . . . 299
129 Basic Format of Packet . . . 301
130 USB Transaction . . . 302

131 USB Transfer . . . 302
132 Enumeration Process . . . 303
133 USB Transfer Flowchart . . . 306
134 CAN Network Topology . . . 322
135 CAN Block Diagram . . . 323
136 Example of CAN Identifier Reception . . . 331
137 Single Filter Configuration When Receiving Information of Standard Structure . . . 333
138 Single Filter Configuration for Receiving Extended Frame Information . . . 334
139 Dual-filter Configuration When Receiving Information of Standard Structure . . . 335
140 Dual-flter Configuration for Receiving Extended Frame Information . . . 336
141 List of Standard frame and Extended Frame Format Configured in Transmit Buffers . . . 340
142 List of Standard frame and Extended Frame Format Configured in Transmit Buffers . . . 341
143 Example of Error Code Capture Function . . . 343
144 Example of Arbitration lost Interpretation . . . 348
145 SPI Block Diagram . . . 377
146 Single Master/Single Slave Application . . . 378
147 Data Clock Timing Diagram . . . 380
148 Start and Stop Conditions . . . 396
149 7-bit Address Format . . . 396
150 10-bit Address Format . . . 397
151 Master Transmitting Protocol . . . 398
152 Master Receiving Protocol . . . 398
153 Start Byte Transmission . . . 399
154 DR Register . . . 400
155 Master Transmitting - Null Tx FIFO . . . 400
156 Master Receiving - Null Tx FIFO . . . 400
157 Multiple-master Arbitration . . . 401
158 Clock Synchronization of Multiple Masters . . . 402
159 I2C Functional Block Diagram . . . 403
160 Flow Chart of I2C Interface Master . . . 406
161 Interrupt Mechanism . . . 408
162 UART Block Diagram . . . 428
163 UART Timing . . . 429
164 Status Bit Change During Transmission . . . 431
165 RX Pin Sampling Plan . . . 433
166 Hardware Flow Control Between Two UARTs . . . 434
167 RTS Flow Control . . . 434
168 CTS Flow Control . . . 435
169 Block Diagram of MM32 Series Level and CPU Level Debug Support . . . 448
170 SWJ debug port . . . 449
171 JTAG TAP connection . . . 453

Table

1 Memory Map . . . 3
2 Boot Modes . . . 5
3 Flash Module Structure . . . 7
4 Flash Interrupt Request . . . 16
5 Option Byte Format . . . 17
6 Structure of Option Bytes . . . 17
7 Description of Option Bytes . . . 17
8 Overview of Flash Register . . . 19
9 CRC Register Overview . . . 26
10 Low-power Mode Summary . . . 32
11 Sleep-now . . . 33
12 Sleep-on-exit . . . 33
13 Stop Mode . . . 34
14 Standby Mode . . . 35
15 Overview of Power Control Registers . . . 36
16 Overview of BKP Registers . . . 41
17 Overview of RCC Registers . . . 52
18 Port bits Configuration Table . . . 72
19 Output MODE bits . . . 72
20 Advanced Timers TIM1 . . . 78
21 General-Purpose timers TIM2/3/4 . . . 78
22 UART . . . 78
23 SPI . . . 79
24 I2C . . . 79
25 CAN . . . 79
26 ADC . . . 79
27 Other I/Os . . . 79
28 Overview of GPIO Registers . . . 80
29 CAN reuse function remapping . . . 86
30 Debug interface signal . . . 86
31 Debug port image . . . 86
32 Timer 3 multiplexing function re-image . . . 87
33 Timer 2 multiplexing function re-image . . . 87
34 Timer 1 multiplexing function re-image . . . 87
35 UART3 re-image . . . 87
36 UART1 re-image . . . 88
37 I2C1 re-image . . . 88
38 SPI1 re-image . . . 88
39 AFIO Register Overview . . . 88
40 Vectors for the Product Series . . . 94
41 EXTI Register Overview . . . 100
43 Programmable Data Width and Endian Behavior (When Bits PINC = MINC = 1) . . . 108

44 DMA Interrupt Requests . . . 111
45 Summary of DMA Requests for Each Channel . . . 113
46 Summary of DMA Registers . . . 113
47 Summary of ADC Registers . . . 125
48 Counting Direction Versus Encoder Signals . . . 169
49 Summary of TIM1 Register . . . 176
50 TIMx internal trigger connection . . . 185
51 Output Control Bits for Complementary OCx and OCxN Channels with Break Feature . . . 200
52 Counting Direction Versus Encoder Signals . . . 239
53 Summary of TIMx Register . . . 250
54 TIMx Internal Trigger Connection . . . 257
55 Output Control Bit for Standard OCx Channels . . . 269
56 Overview of RTC Registers . . . 279
57 Min/max IWDG Timeout Period (in ms) at 40 kHz (LSI) . . . 287
58 Overview of IWDG Registers . . . 288
59 Overview of WWDG Registers . . . 296
60 USB Register Overview . . . 307
61 Permission Assignment for Basic CAN Mode Register . . . 325
62 Permission Assignment for Peli CAN Mode Register . . . 327
63 RX and TX Buffers in BasicCAN Mode . . . 339
64 Possible Errors During Reception . . . 344
65 Possible Errors During Reception . . . 344
66 Overview of CAN Registers . . . 349
67 Functions of Arbitration lost Capture Register's Bit 4 - Bit 0 . . . 365
68 SPI Status . . . 382
69 Baud Rate Formula . . . 383
70 Overview of SPI Register . . . 384
71 First Byte of I2C . . . 397
72 Set and Clear Interrupt Bits . . . 407
73 I2C Register Overview . . . 408
74 DISSLAVE (Bit 6) and MASTER (Bit 0) Configurations . . . 410
76 UART interrupt requests . . . 435
77 UART Register Overview . . . 436
78 Memory Capacity Register Description Overview . . . 444
79 SWJ debug port pin . . . 450
80 Flexible SWJ_DP pin assignment . . . 451
82 chip ID encoding . . . 454
83 JTAG debug port data register . . . 454
84 A[3: 2]Defined 32-bit debug interface register address . . . 455
85 Packet Request (8-bit) . . . 456
86 Packet Request (3-bit) . . . 457
87 Packet Request (33-bit) . . . 457
89 Summary of DBG Register . . . 460
90 Revision History . . . 462

1 Memory and Bus Architecture

Memory and Bus Architecture

1.1 System architecture

The main system consists of the following parts:

- Four drive units:
  - CPU ICode bus(I-bus), DCode bus(D-bus) and System bus(S-bus)
  - Generic DMA
- Three passive units:
  - Internal SRAM
  - Internal flash memory
  - AHB to APB bridges(APBx), connecting all AHB devices

They are interconnected through a multi-stage AHB architecture, as shown in Figure 1.

Figure 1. System architecture

ICode bus

The bus connects the instruction bus of the CPU core to the flash instruction interface. Instruction prefetching is done on this bus.

DCode bus

This bus connects the CPU core DCode bus to the data interface of the flash memory (constant loading and debug access).

System bus

The bus connects the system bus (peripheral bus) of the CPU core to the BusMatrix, which manages the access between the core and DMA.

DMA bus

The bus connects the AHB master interface of DMA with the BusMatrix, which manages the access of CPU and DMA to SRAM, flash memory and peripherals.

BusMatrix

The BusMatrix, composed of master module bus and slave module bus, manages the access arbitration between the core system bus and the DMA bus.

AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.

AHB2APB bridges - APB

The AHB-APB bridges provide the synchronous connections between the AHB and APB bus. After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash). Before using a peripheral, you have to enable its clock in the RCC_AHBENR, RCC_APB2ENR or RCC_APB1ENR register.

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

1.2 Memory organization

1.2.1 Introduction

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

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

The addressable memory space is divided into 8 main blocks, each of 512 MB. All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved". Refer to the section 1.2.2 and Peripherals.

1.2.2 Memory map and register addressing

See the memory map in Peripherals chapters.The following table gives the start addresses of the embedded peripherals.

Table 1. Memory Map

Bus	Addressing range	Size	Peripheral	Remarks
Flash	0x0000 0000 - 0x0001 FFFF	128KB	Main flash memory, system memory or SRAM, depending on BOOT configuration	
	0x000 20000 - 0x07FF FFFF	~128KB	Reserved	
	0x0800 0000 - 0x0801 FFFF	128KB	Main Flash memory	
Flash	0x0802 0000 - 0x0FFF FFFF	~128KB	Reserved	
	0x1000 0000 - 0x1000 1FFF	8KB	Reserved	
	0x1000 2000 - 0x1FFD FFFF	~256KB	Reserved	
	0x1FFE 0000 - 0x1FFE 01FF	0.5KB	Protect byte	
	0x1FFE 0200 - 0x1FFE 0FFF	3KB	Reserved	
	0x1FFE 1000 - 0x1FFE 1BFF	3KB	Security space	
	0x1FFE 1C00 - 0x1FFF F3FF	~256KB	Reserved	
	0x1FFF F400 - 0x1FFF F7FF	1KB	System memory	
	0x1FFF F800 - 0x1FFF F80F	16KB	Option bytes	
	0x1FFF F810 - 0x1FFF FFFF	~2KB	Reserved	
SRAM	0x2000 0000 - 0x2000 4FFF	20KB	SRAM	
	0x2000 5000 - 0x3FFF FFFF	~512MB	Reserved	
APB1	0x4000 0000 - 0x4000 03FF	1KB	TIM2	
	0x4000 0400 - 0x4000 07FF	1KB	TIM3	
	0x4000 0800 - 0x4000 0BFF	1KB	TIM4	
	0x4000 0C00 - 0x4000 27FF	7KB	Reserved	
	0x4000 2800 - 0x4000 2BFF	1KB	RTC	
	0x4000 2C00 - 0x4000 2FFF	1KB	WWDG	
	0x4000 3000 - 0x4000 33FF	1KB	IWWDG	
	0x4000 3400 - 0x4000 37FF	1KB	Reserved	
	0x4000 3800 - 0x4000 3BFF	1KB	SPI2	
	0x4000 3C00 - 0x4000 43FF	2KB	Reserved	
	0x4000 4400 - 0x4000 47FF	1KB	UART2	
	0x4000 4800 - 0x4000 4BFF	1KB	UART3	
	0x4000 4C00 - 0x4000 53FF	2KB	Reserved	
	0x4000 5400 - 0x4000 57FF	1KB	I2C1	
	0x4000 5800 - 0x4000 5BFF	1KB	I2C2	
	0x4000 5C00 - 0x4000 5FFF	1KB	USB	
	0x4000 6000 - 0x4000 63FF	1KB	Reserved	
	0x4000 6400 - 0x4000 67FF	1KB	CAN	
	0x4000 6800 - 0x4000 6BFF	1KB	Reserved	
	0x4000 6C00 - 0x4000 6FFF	1KB	BKP	
	0x4000 7000 - 0x4000 73FF	1KB	PWR	
	0x4000 7400 - 0x4000 77FF	34KB	Reserved	
	0x4000 7800 - 0x4000 FFFF	34KB	Reserved	

Bus	Addressing range	Size	Peripheral	Remarks
APB2	0x4001 0000 - 0x4001 03FF	1KB	AFIO	
	0x4001 0400 - 0x4001 07FF	1KB	EXTI	
	0x4001 0800 - 0x4001 0BFF	1KB	GPIOA	
	0x4001 0C00 - 0x4001 0FFF	1KB	GPIOB	
	0x4001 1000 - 0x4001 13FF	1KB	GPIOC	
	0x4001 1400 - 0x4001 17FF	1KB	GPIOD	
APB2	0x4001 1800 - 0x4001 1BFF	1KB	Reserved	
	0x4001 1C00 - 0x4001 23FF	2KB	Reserved	
	0x4001 2400 - 0x4001 27FF	1KB	ADC1	
	0x4001 2800 - 0x4001 2BFF	1KB	ADC2	
	0x4001 2C00 - 0x4001 2FFF	1KB	TIM1	
	0x4001 3000 - 0x4001 33FF	1KB	SPI1	
	0x4001 3400 - 0x4001 37FF	1KB	Reserved	
	0x4001 3800 - 0x4001 3BFF	1KB	UART1	
	0x4001 3C00 - 0x4001 3FFF	1KB	Reserved	
	0x4001 4000 - 0x4001 43FF	1KB	Reserved	
	0x4001 4400 - 0x4001 47FF	1KB	Reserved	
	0x4001 4800 - 0x4001 4BFF	1KB	Reserved	
	0x4001 4C00 - 0x4001 7FFF	13KB	Reserved	
	0x4001 8000 - 0x4001 FFFF	32KB	Reserved	
AHB	0x4002 0000 - 0x4002 03FF	1KB	DMA	
	0x4002 0400 - 0x4002 0FFF	3KB	Reserved	
	0x4002 1000 - 0x4002 13FF	1KB	RCC	
	0x4002 1400 - 0x4002 1FFF	3KB	Reserved	
	0x4002 2000 - 0x4002 23FF	1KB	Flash Interface	
	0x4002 2400 - 0x4002 2FFF	3KB	Reserved	
	0x4002 3000 - 0x4002 33FF	1KB	CRC	
	0x4002 3400 - 0x4002 43FF	4KB	Reserved	

1.3 Embedded SRAM

It features up to 20 K bytes of static SRAM.

It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

- SRAM up to 20K bytes on the data bus. Can be used by the CPU or DMA with the fastest system clock and without any waiting to access.

1.4 Overview of FLASH memory

The flash memory includes two different storage areas:

- The main block includes the program data area and the user data area (if necessary)

- The information block includes four parts:
  - Option bytes - Containing hardware and user storage protection configuration options.
  - System memory - Containing boot loader code. See Section "Embedded Flash Memory".

The flash memory interface, based on AHB protocol, executes instructions and accesses data. With the prefetch buffer function, it accelerates the code execution speed of CPU.

1.5 Boot configuration

3 different boot modes can be selected through BOOT0 pins and nBOOT1 bit, as shown in the following table.

Table 2. Boot Modes

Boot Mode selection		Boot Modes	Aliasing
BOOT1	**BOOT0**		
x	0	Main flash memory	Main flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

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

The BOOT pins are also re-sampled when waking up from Standby mode. Consequently, they must be kept in the required Boot mode configuration in Standby mode.

After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x00000000, then starts code execution from the boot memory starting from 0x00000004. Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF F400).
- Boot from the embedded SRAM: SRAM is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x2000 0000).

Embedded boot loader

The embedded boot loader is located in the System memory, programmed by the manufacturer during production. It is used to reprogram the Flash memory with UART1.

2 Embedded flash (FLASH)

Embedded flash (FLASH)

2.1 Main features

- Up to 128K bytes of flash memory

The flash memory interface features:

- Data interface with prefetch buffer (2x64-bit)
- Option byte Loader
- Flash program/erase operation
- Read / write protection
- Low power mode

2.2 Functional description

2.2.1 Structure

The flash space consists of 64-bit memory cells, in which both code and data can be saved. The main block is divided into 128 pages (1 K bytes per page) or 32 sectors (4 K bytes per sector), and the write protection is set in sectors (see related content of ”Storage Protection”).

Table 3. Flash Module Structure

Module	Name	Address	Size(bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1K
	Page 1	0x0800 0400 - 0x0800 07FF	1K
	Page 2	0x0800 0800 - 0x0800 0BFF	1K
	Page 3	0x0800 0C00 - 0x0800 0FFF	1K
	...	...	...
	...	...	...
	Page 28	0x0800 7000 - 0x0800 73FF	1K
	Page 29	0x0800 7400 - 0x0800 77FF	1K
	Page 30	0x0800 7800 - 0x0800 7BFF	1K
	Page 31	0x0800 7C00 - 0x0800 7FFF	1K
	...	...	...
	...	...	...
	Page 123	0x0801 EC00 - 0x0801 EFFF	1K
	Page 124	0x0801 F000 - 0x0801 F3FF	1K
	Page 125	0x0801 F400 - 0x0801 F7FF	1K

Module	Name	Address	Size(bytes)
Main memory	Page 126	0x0801 F800 - 0x0801 FBFF	1K
	Page 127	0x0801 FC00 - 0x0801 FFFF	1K
Information block	Guard bytes	0x1FFE 0000 - 0x1FFE 01FF	0.5K
	Secrecy space	0x1FFE 1000 - 0x1FFE 1BFF	3K
	System memory	0x1FFF F400 - 0x1FFF F7FF	1K
	Option bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

2.2.2 Reading flash

Embedded flash modules, like common storage space, can be directly addressed and accessed. The operation of reading flash module shall undergo a special judgment process.

Both instruction fetch and data fetch are performed through AHB bus and can be executed in the manner specified by the option in the flash access control register (FLASH_ACR):

- Instruction fetch: CPU running speed can be increased after the prefetch buffer is enabled
- Latency: number of wait states for a correct read operation

Instruction fetch

Instructions are fetched by CPU through AHB. With the prefetch module, the instruction fetch efficiency is improved.

Prefetch buffer

Prefetch buffer (2x64-bit): It is automatically opened after reset. Since the size of each buffer (64-bit) is the same as the bandwidth of the flash, the contents of the entire buffer can be updated only by reading flash memory once. Due to the existence of the prefetch buffer, the CPU can run at a higher dominant frequency. In each time, the CPU fetches a word up to 32 bits; the next instruction is waiting in the buffer while one instruction is being fetched.

Prefetch controller

The prefetch controller will timely access the flash according to the available space in the prefetch buffer. In case of at least one available space in the prefetch buffer, the prefetch controller will initiate a read request. After reset, the prefetch buffer is opened by default, and can only be enabled/disabled if SYSCLK is lower than 24 MHz and the AHB clock

has not undergone any frequency division (SYSCLK must be equal to HCLK). Usually, prefetch buffer has already determines its on/off state during the initialization process. At this time, MCU is running under the 8 MHz oscillator.

Note: When the prescaler of AHB clock is not equal to 1, the prefetch buffer shall initiate the access latency.

Access latency

In order to ensure the correct flash reading, the speed ratio of prefetch controller shall be specified in LATENCY 2: 0 in the flash access control register, and it is equal to the number of latent periods to be inserted between each flash accessing to the next access. After reset, this value defaults to zero, namely, there is no latent period to be inserted.

2.2.3 Programming and erasing flash

The embedded flash supports online programming and in-application programming.

ICP refers to rewriting flash online through SWD and burning the user code into the MCU. ICP provides a simple and efficient method, not requiring chip clamping during its writing.

Unlike ICP, IAP (in-application programming) enables downloading programs or data through any communication interface (I/Os, USB, UART, I2C, SPI, etc.) supported by MCU. IAP allows users to rewrite applications while running them, provided that some applications must be burned in advance by ICP/ISP.

The burn-in and erase operations can be completed within the whole operating voltage range of the product, and they are achieved with the following 7 registers:

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

As long as the CPU does not access the Flash space, the ongoing Flash programming will not hinder the operation of the CPU. That is to say, in the process of programming/erasing Flash, any access to Flash will disable the bus and it will not resume until the programming/erasing operation is completed, which means that instruction fetch and data access cannot be performed in case of Flash programming/erasing.

In the process of programming/erasing Flash space, the internal oscillator (HSI) must be on.

Unlocking Flash space

After reset, Flash memory is protected by default, preventing accidental erasing. The FLASH_CR is not allowed to be rewritten and the access to the FLASH_CR will not be authorized unless a series of unlocking operations for the FLASH_KEYR are performed. These operations include the following 2 write operations:

- Write key 1 = 0x45670123
- Write key 2 = 0xCDEF89AB

Any wrong sequence will lock FLASH_CR until the next reset.

In case of invalid keyword, a bus error will cause a hardware error interrupt; in case of KEY1 error, it will immediately interrupt, while a correct KEY1 will also lead to an interrupt when KEY2 error occurs.

Main flash programming

The 16-bit main flash can be programmed at a time. When PG bit in FLASH_CR is 1, writing a half words (16 bits) corresponding to the address is a programming operation. If you try to write another length instead of half words, the operation will cause a hardware error interrupt.

Figure 2. Programming Flow

The Flash memory interface will pre-read and judge whether the bytes behind those to be programmed are all 1s. If not, the programming operation will be automatically cancelled and an error warning will be prompted on the PGERR bit of the FLASH_SR.

If the write protection bit in FLASH_WRPR, corresponding to the address to be programmed, is valid, the programming will be disabled, and an error warning will be also generated. In addition, a prompt will be given at the EOP bit in FLASH_SR after the programming.

The programming process in the standard mode of the main Flash memory is as follows:

- Check BSY bit in FLASH_SR, to confirm that the previous operation has ended
- Set PG bit in FLASH_CR
- Write data to the target address in half words
- Wait for BSY in FLASH_SR to return to zero
- Read data for validation

Note: these registers cannot be written when the BSY bit in FLASH_SR is 1.

Erasing Flash memory

Flash memory can be erased by pages or whole pieces.

Page erasing

The specific procedures are as follows:

- Check BSY bit in FLASH_SR, to confirm that the previous operation has ended
- Set the PER bit in the FLASH_CR to 1
- Write the FLASH_AR, to select the page to be erased
- Set the STRT bit in the FLASH_CR to 1
- Wait for BSY in FLASH_SR to return to zero
- Read the erased pages for validation

Figure 3. Page Erasing Process of Flash Register

Whole erasing

The entire Flash user area can be erased at one time by using the whole erasing command, and the information block will not be affected by this command. The specific steps are as follows:

- Check BSY bit in FLASH_SR, to confirm that the previous operation has ended
- Set the MER bit in the FLASH_CR to 1
- Set the STRT bit in the FLASH_CR to 1
- Wait for BSY bit to return to zero
- Read and validate all pages

Figure 4. Whole Erasing Process of Flash Register

Option bytes programming

The programming of option bytes is different from that for the conventional user address, including 2 write protections and 1 hardware configuration. After the Flash access restriction is lifted, the FLASH_OPTKEYR needs to be written with keywords. After that, the OPTWRE bit in the FLASH_CR will be set to '1', then the OPTPG bit in the FLASH_CR can be set first, and then the target address can be written in half words. Similarly, it will automatically check if the option byte is 1. If not, the relevant operation will be cancelled and an error will be prompted at the WRPRTERR bit in FLASH_SR. After the programming, a prompt will be given at the EOP bit of FLASH_SR.

The option byte is 16-bit data, the valid data is the lower-8-bit, and the upper 8 bits are the inverse of the lower 8 bits. In the programming process, the hardware will automatically set the upper 8 bits to the inverse code of the lower 8 bits, to ensure that the write value of the option byte is always correct. The steps are as follows:

- Check BSY bit in FLASH_SR, to confirm that the previous operation has ended
- Unlock OPTWRE bit in FLASH_CR

- Set the OPTPG bit in the FLASH_CR to 1
- Write data (half word) to the target address
- Wait for BSY bit to return to zero
- Read and validate that a whole erasing will be automatically triggered when the protection option byte is changed from the protected state to the unprotected state. If the user only wants to rewrite other bytes, the whole erasing will not be activated. This mechanism is used to protect the Flash.

Figure 5. Option Byte Programming Process

Erasing process

The erasing process of option bytes is as follows:

- Check BSY bit in FLASH_SR, to confirm that the previous operation has ended
- Unlock OPTWRE bit in FLASH_CR

- Set OPTER bit in FLASH_CR to 1
- Set the STRT bit in the FLASH_CR to 1
- Wait for BSY bit to return to zero
- Read and validate

Figure 6. Option Byte Erasing Process

2.3 Storage protection

It is used to prevent the codes in the Flash area of the user area from being read by untrusted codes, and to prevent accidental erasure of the Flash in case of running-out. The minimum unit of write protection is one sector (4 pages).

2.3.1 Write protection of main space

The write protection is controlled by one sector (4 pages), to configure the WRP bit in the option byte, and the subsequent system reset will load the new option byte, to enable the protection. If an attempt is made to write or erase a protected sector, the WRPRTERR flag bit in FLASH_SR will be set.

Unclocking

It is applicable to the startup program realized and programmed by the user:

- Erase the entire option byte area by using the OPTER bit of the flash control register (FLASH_CR);
- Reset the system and reload option bytes (including new WRP bytes), to unlock the write protection.

With this method, unlock the write protection of the entire main flash module, excepting Pages 0-3 which are still protected.

2.3.2 Write protection of option bytes

By default, the option byte block is always readable and write protected. To write (programming/erasing) an option byte block, write the correct key sequence in the OPTKEYR (the same as the locking operation), then enable the write operation to the option byte block. Note that the OPTWRE bit in the FLASH_CR indicates the write permission, and disable the write operation by clearing this bit.

2.4 Flash interrupt

Table 4. Flash Interrupt Request

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

2.5 Description of option bytes

Option bytes are configured by the user according to the application requirements, for example, a hardware-based watchdog or a software-based watchdog can be selected.

Each 32-bit word in the option byte is classified into the following formats:

Table 5. Option Byte Format

Bits 31 ~ 24	Bits 23 ~ 16	Bits 15 ~ 8	Bits 7 ~ 0
Inversion of option byte 1	Option byte 1	Inversion of option byte 0	Option byte 0

Note: The inversed code is automatically achieved by hardware, and cannot be written through the software.

The organization of option bytes in the option byte block is as shown in the following table.

Option bytes can be read from the memory addresses listed in the following table or from the option byte register (FLASH_OBR).

Note: The newly-written option byte (user's or read/write-protected) will not take effect until the system is reset.

Table 6. Structure of Option Bytes

Address	[31: 24]	[23: 16]	[15: 8]	[7: 0]
0x1FFF F800	nUSER	USER		
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2

Table 7. Description of Option Bytes

Memory address	Option bytes
0x1FFF F800	Bit[31: 24] nUSER Bit[23: 16] USER: user option byte (saved in FLASH_OBR[9: 2]). It is used to configure the following functions: Select watchdog event: hardware or software Reset event when entering stop mode Reset event when entering standby mode Note: Only use Bits [18: 16], and do not use Bits [23: 19]. Bit 18: nRST_STDBY 0: Reset when entering standby mode 1: Do no reset when entering standby mode Bit 17: nRST_STOP 0: Reset when entering STOP mode 1: Do not reset when entering STOP mode Bit 16: WDG_SW 0: Hardware watchdog 1: Software watchdog

Memory address	Option bytes
0x1FFF F804	Datax: 2-byte user data This address can be programmed through the programming methods for option bytes. Bits [31: 24]: nData1 Bits [23: 16]: Data1(saved in FLASH_OBR[25: 18]) Bits [15: 8]: nData0 Bits [7: 0]: Data0 (saved in FLASH_OBR[17: 10])
0x1FFF F808	WRPx: Flash write protection for option bytes Bits [31: 24]: nWRP1 Bits [23: 16]: WRP1(saved in FLASH_WRPR[15: 8]) Bits [15: 8]: nWRP0 Bits [7: 0]: WRP0(saved in FLASH_WRPR[7: 0])
0x1FFF F80C	WRPx: Flash write protection for option bytes Bits [31: 24]: nWRP3 Bits [23: 16]: WRP3(saved in FLASH_WRPR[31: 24]) Bits [15: 8]: nWRP2 Bits [7: 0]: WRP2(saved in FLASH_WRPR[23: 16]) Each bit in the option byte WRPx is used to protect 4 memory pages in the main memory: 0: Write protection is enabled 1: Write protection is disabled Four user option bytes are used to protect the main memory of 128 K bytes. WRP0: write protection on Pages 0 ~ 31 WRP1: write protection on Pages 32 ~ 63 WRP2: write protection on Pages 64 ~ 95 WRP3: write protection on Pages 96 ~ 127

After each system reset, the option byte loader (OBL) reads the data of the information block and saves it in the option byte register (FLASH-OBR); each select bit has its inverse code bit in the information block. When the select bit is loaded, the inverse code bit is used to validate whether the select bit is correct. In case of any difference, an option byte error flag (OPTERR) will be generated. When an option byte error occurs, the corresponding option byte is set to 0xFF. When the option byte and its inverse code are 0xFF (erased state), the above verification function is disabled.

All the select bits (excluding their inversed code bits) are used to configure the microcontroller, and the option byte register is read by CPU.

2.6 Description of Flash register

Table 8. Overview of Flash Register

Offset	Acronym	Register Name	Reset	Section
0x00	FLASH_ACR	Flash access control register	0x00000038	section 2.6.1
0x04	FLASH_KEYR	FPEC key register	0xXXXXXXXX	section 2.6.2
0x08	FLASH_OPTKEYR	Flash OPTKEY register	0xXXXXXXXX	section 2.6.3
0x0C	FLASH_SR	Flash status register	0x00000000	section 2.6.4
0x10	FLASH_CR	Flash control register	0x00000080	section 2.6.5
0x14	FLASH_AR	Flash address register	0x00000000	section 2.6.6
0x1C	FLASH_OBR	Option byte register	0x03FFFC1C	section 2.6.7
0x20	FLASH_WRPR	Write protection register	0xFFFFFFFF	section 2.6.8

2.6.1 Flash access control register(FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0038

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PRFTBE	HLFCYA	LATENCY		
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 5	Reserved		0x0000 0001	Reserved, Bit 5 always reads as 1.
4	PRFTBE	rw	0x01	Prefetch buffer enable 0: Prefetch buffer is disabled 1: Prefetch buffer is enabled
3	HLFCYA	rw	0x01	Flash half cycle access enable 0: Half cycle is disabled 1: Half cycle is enabled
2 : 0	LATENCY	rw	0x00	Latency These bits represent the ratio of SYSCLK (system clock) period to Flash access time. 000: Zero wait state, if 0 < SYSCLK≤ 24 MHz 001: One wait state, if 24 MHz < SYSCLK ≤ 48 MHz 010: Two wait state, if 48 MHz < SYSCLK ≤ 72 MHz 011: Three wait state, if 72 MHz < SYSCLK ≤ 96 MHz

2.6.2 Flash access control register(FLASH_KEYR)

Address offest: 0x04

Reset value: 0xXXXX XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 0	FKEYR	w	0xXXXX XXXX	FPEC(Flash key) These bits are used to enter the unlock key of FPEC.

Note: All these bits are written only and 0 is returned when being read.

2.6.3 Flash OPTKEY register(FLASH_OPTKEYR)

Address offset: 0x08

Reset value: 0xXXXX XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 0	OPTKEYR	w	0xXXXX XXXX	Option byte key These bits are used to enter the key of the option byte, to disable OPTWRE.

Note: All these bits are written only and 0 is returned when being read.

2.6.4 Flash status register(FLASH_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EOP	WRPRTERR	Res.	PGERR	Res.	BSY
										rc_w1	rc_w1		rc_w1		r

Bit	Field	Type	Reset	Description
31 : 6	Reserved			Reserved, always read as 0.
5	EOP	rc_w1	0x00	End of operation When the flash operation (programming/erasing) is completed, the hardware sets this bit to ’1’, and eliminate this state by writing ’1’. Note: EOP status will be set for every successful programming or erasing.
4	WRPRTERR	rc_w1	0x00	Write protection error When the flash address for write protection is programmed, the hardware sets this bit to ’1’, and eliminate this state by writing ’1’.
3	Reserved			Reserved, always read as 0.
2	PGERR	rc_w1	0x00	Programming error Attempting to program an address that is not ’0xFFFF’, the hardware sets this bit to ’1’, and eliminate this state by writing ’1’. Note: The STRT bit in the FLASH_CR shall be cleared before the programming.
1	Reserved			Reserved, always read as 0.
0	BSY	r	0x00	Busy This bit indicates that the flash operation is in progress. This bit is set to ’1’ when the flash operation begins and written to ’0’ at the end of operation or in case of an error.

2.6.5 Flash control register(FLASH_CR)

Address offset: 0x10

Reset value: 0x0000 0080

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG
			rw		rw	rc_w0		rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 13	Reserved			Reserved, always read as 0.
12	EOPIE	rw	0x00	End of operation interrupt enable This bit leads to an interrupt when the EOP bit in the FLASH_SR register changes to '1'. 0: Interrupt is disabled 1: Interrupt is enabled
11	Reserved			Reserved, always read as 0.
10	ERRIE	rw	0x00	Error interrupt enable This bit enables an interrupt in case of an FPEC error (when PGERR/WRPRTERR in the Flash-SR is set to '1'). 0: Interrupt is disabled 1: Interrupt is enabled
9	OPTWRE	rc_w0	0x00	Option byte write enable When this bit is '1', the option byte is allowed to be programmed. This bit is set to '1' when the correct key sequence is written in the FLASH_OPTKEYR. This bit can be cleared by writing 0 through the software.
8	Reserved			Reserved, always read as 0.
7	LOCK	rw	0x01	Lock Only '1' can be written. When this bit is '1', FPEC and FLASH_CR are locked. After detecting the correct unlocking sequence, the hardware automatically clears and sets this bit to '0'. After an unsuccessful unlock operation, this bit cannot be changed again until the next system reset.
6	STRT	rw	0x00	Start When this bit is '1', an erasing operation will be enabled. This bit can only be set to '1' by software and cleared automatically when BSY changes to '1'.
5	OPTER	rw	0x00	Option byte erase Option bytes are erased.
4	OPTPG	rw	0x00	Option byte programming Option bytes are programmed
3	Reserved			Reserved, always read as 0.
2	MER	rw	0x00	Mass erase Select to erase all user pages.
1	PER	rw	0x00	Page erase Select to erase pages.
0	PG	rw	0x00	Programming Select to program.

2.6.6 Flash address register(FLASH_AR)

Address offset: 0x014

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 0	FAR	w	0x0000 0000	Flash Address Select the address to be programmed before programming, and the page to be erased when erasing. Note: it is not allowed to write the register when the BSY bit in FLASH_SR is 1.

Change to current/last used address from hardware. During the page erasing, modify the register, to specify the page to be erased.

2.6.7 Option byte register(FLASH_OBR)

Address offset: 0x1C

Reset value: 0x03FF FC1C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						Data1								Data0	
						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
Data0						Reserved					nRST_STDBY	nRST_STOP	WDG_SW	Res.	OPTERR
r	r	r	r	r	r						r	r	r		r

Bit	Field	Type	Reset	Description
31 : 26	Reserved			Reserved, always read as 0.
25 : 18	Data1	r	0xFF	Data1
17 : 10	Data0	r	0xFF	Data0
9 : 5	Reserved			Reserved, always read as 0.
4	nRST_STDBY	r	0x01	Reset event when entering standby mode 0: Reset when entering standby mode 1: Do no reset when entering standby mode
3	nRST_STOP	r	0x01	Reset event when entering stop mode 0: Reset when entering STOP mode 1: Do not reset when entering STOP mode

Bit	Field	Type	Reset	Description
2	WDG_SW	r	0x01	Select watchdog event 0: Hardware watchdog 1: Software watchdog
1	Reserved			Reserved, always read as 0.
0	OPTERR	r	0	Option byte error When this bit is '1', it means that the option byte does not match its inverse code. Note: This bit is read-only.

The reset value of this register is related to the value written in the option byte, and the reset value of the OPTERR bit is related to the result of comparing the option byte with its inverse code when the option bytes are loaded.

2.6.8 Write protection register(FLASH_WRPR)

Address offset: 0x20

Reset value: 0xFFFF FFFF

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

Bit	Field	Type	Reset	Description
31 : 0	WRP	r	0xFFFF FFFF	Write protect This register contains write protection option bytes loaded by OBL. 0: The write protection is enabled 1: The write protection is disabled Note: These bits are read-only.

3 Cyclic redundancy check calculation unit (CRC)

Cyclic redundancy check calculation unit (CRC)

3.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a 32-bit CRC result from a fixed generator polynomial. Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, which is compared with a reference signature generated at link time and then stored in a given memory space.

3.2 CRC main features

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

The CRC block diagram is shown in the following figure

Figure 7. CRC Calculation Unit Block Diagram

3.3 CRC Functional description

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

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

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

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

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

3.4 CRC register

The CRC calculation unit contains two data registers and a control register.

Table 9. CRC Register Overview

Offset	Acronym	Register Name	Reset	Section
0x00	CRC_DR	CRC data register	0xFFFFFFFF	section 3.4.1
0x04	CRC_IDR	CRC independent data register	0x00000000	section 3.4.2
0x08	CRC_CTRL	CRC control register	0x00000000	section 3.4.3

3.4.1 CRC data register(CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bit	Field	Type	Reset	Description
31 : 0	DR	rw	0xFFFF FFFF	DR: Data register bits Used as an input register when writing new data into the CRC calculator. The CRC results are returned when being read.

3.4.2 CRC independent data register(CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Always read as 0.
7 : 0	IDR	rw	0x00	IDR: General-purpose 8-bit data register bits Can be used as a temporary storage location for one byte. This register is not affected by CRC resets generated by the RESET bit in the CRC_CTRL register.

Note: This register is not involved in the CRC calculation and enables storing any data.

3.4.3 CRC control register(CRC_CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

Bit	Field	Type	Reset	Description
31 : 1	Reserved			Always read as 0.
0	RESET	w	0x00	RESET: CRC reset Sets the data register to 0xFFFF FFFF. This bit can only be set, it is automatically cleared by hardware.

4 Power control (PWR)

Power control (PWR)

4.1 Power supply

The chip requires a 2.0-to-5.5 V operating voltage supply (V_{DD}). An embedded regulator is used to supply the internal 1.5 V power.

When the main power supply V_{DD} is powered down, power is supplied to the real-time clock (RTC) and the backup register through the V_{BAT} pin.

Figure 8. Power Supply Overview

Note: V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}.

4.1.1 Independent A/D converter supply and reference voltage

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

- The ADC supply input is available on a V_{DDA} pin

- An isolated supply ground connection is provided on pin V_{SSA}

When available (according to package), pin V_{REF-} must be tied to V_{SSA}.

4.1.2 Battery backup domain

Use a battery or other power source to connect to the V_{BAT} pin. When V_{DD} is powered off, the contents of the backup register can be saved and the RTC function can be maintained.

The V_{BAT} pin powers the RTC, LSE oscillator, and PC13 to PC15 ports to ensure that the RTC continues to operate when the main power is turned off. Switch to V_{BAT} Powered switch, controlled by the Brown-Out Reset function in the Reset module.

Warning

The power switch between V_{BAT} and V_{DD} will remain connected to V_{BAT} after the V_{DD} rising phase ($t_{RSTTEMPO}$) or PDR (power-down reset) is detected.

If there is no external battery in the application, it is recommended that V_{BAT} be externally connected to V_{DD} and connect a 100nF ceramic filter capacitor.

When the backup area is powered by V_{DD}, the following features are available:

- PC14 and PC15 can be used for GPIO or LSE pins
- PC13 can be used as a general purpose I/O port, TAMPER pin, RTC calibration clock, RTC alarm or second output (see Backup Register (BKP))

Note: Because the analog switch can only pass a small amount of current (3mA), there are limits to using the I/O port function of PC13 to PC15 in output mode: the speed must be limited to 2MHz or less, the maximum load is 30pF, and these I/ Port O must not be used as a current source (such as driving LEDs).

When the backing area is powered by V_{BAT} (V_{DD} after the power is turned off, the analog switch is connected to V_{BAT}), the following functions can be used:

- PC14 and PC15 can only be used on the LSE pin
- PC13 can be used as a TAMPER pin, RTC alarm or second output (see: RTC Clock Calibration Register (BKP_RTCCR))

4.1.3 Voltage regulator

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

- In Run mode, the regulator supplies full power to the 1.5 V domain (core, memories and digital peripherals).
- In Stop mode, the regulator supplies low-power to the 1.5 V domain, preserving contents of registers and SRAM.
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

4.2 Power supply supervisor

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

The chip has a complete power-on reset (POR) and brown-out reset (PDR) circuit that works when the supply voltage reaches 2.0V.

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

Figure 9. Power on Reset/Power Down Reset Waveform

4.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS bits in the Power control register (PWR_CR).

The PVD is enabled by setting the PVDE bit.

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

Figure 10. PVD Thresholds

4.3 Low-power modes

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

The device features three low-power modes:

- Sleep mode (CPU clock off, all peripherals including CPU peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped, and contents of register and SRAM are retained)
- Standby mode (1.5V domain is powered off, and the contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain)

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

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

Table 10. Low-power Mode Summary

Mode	Entry	Wakeup	Effect on 1.5V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on-exit)	WFI (Wait for Interrupt)	Any interrupt	CPU clock OFF, no effect on other clocks or ADC clock	None	ON
	WFE (Wait for Event)	Wakeup event			
Stop	PDDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers)	All 1.5V domain clocks OFF	PLL, HSI and HSE oscillators OFF	ON
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm event, external reset in NRST pin, and IWDG reset			OFF

4.3.1 Slowing down system clocks

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

For more details refer to Section ”Clock Configuration Register (RCC_CFGR)

4.3.2 Peripheral clock gating

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

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

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

4.3.3 Sleep Mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the system control register of CPU:

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

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

Refer to Table 11 and Table 12 for details on how to enter Sleep mode.

Table 11. Sleep-now

Sleep-now mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - SLEEPDEEP = 0 - SLEEPONEXIT = 0 Refer to the CPU System Control register
Mode exit	If WFI was used for entry: Interrupt: Refer to Section ”Interrupt and Exception Vectors” If WFE was used for entry: Wakeup event: Refer to Section ”Wakeup Event Management”
Wakeup latency	None

Table 12. Sleep-on-exit

Sleep-on-exit mode	Description
Description	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - SLEEPDEEP = 0 - SLEEPONEXIT = 1 Refer to the CPU System Control register
Mode exit	If WFE was used for entry:Interrupt or clear Bit 1 of CPU control register If WFE was used for entry: Wakeup event: Refer to Section ”Wakeup Event Management”
Wakeup latency	None

4.3.4 Stop mode

The Stop mode is based on the CPU deepsleep mode combined with peripheral clock gating. And the voltage regulator can be configured in normal mode.In Stop mode, all clocks in the 1.5 V domain are stopped, the PLL, the HSI and the HSE oscillators are disabled. SRAM and register contents are preserved.

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

Entering Stop mode

Refer to Table 13 for details on how to enter the Stop mode.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option.
- Real Time Clock (RTC): Set by the RTCEN bit in the Backup Domain Control Register (RCC_BDCR).
- Internal oscillator (LSI): this is configured by the LSION bit in the Control/status register (RCC_CSR).

- External 32.768KHz oscillator (LSE): Set by the LSEON bit in the Backup Domain Control Register (RCC_BDCR).

In Stop mode, these peripherals still consume current if the ADC and DAC are not turned off before entering this mode. These 2 peripherals can be turned off by setting the ADON bit of the ADC_CR2 register and the ENx bit of the DAC_CR register to '0'. Other unused GPIOs require an analog input to be set, otherwise there is current consumption.

Exiting Stop mode

Refer to Table 13 for details on how to exit the Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI oscillator is selected as system clock, with the frequency of HSI frequency divided by 6.

When the voltage regulator operates in normal-power mode, an additional startup delay is incurred when waking up from Stop mode.

Table 13. Stop Mode

Stop Mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - Set SLEEPDEEP bit in CPU System Control register - Clear PDDS bit in Power Control register (PWR_CR) Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupt pending bits, and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
must be reset	WFI (Wait for Interrupt) is executed in case of: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to Section "Interrupt and Exception Vectors". If WFE (Wait for Event) is executed in case of: Any EXTI Line configured in event mode. Refer to Section "Wakeup Event Management".
Wakeup latency	HSI wakeup time

4.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the CPU deepsleep mode, with the voltage regulator disabled. The 1.5 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry.

Entering Standby mode

Refer to Table 14 for details on how to enter the Standby mode.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option.
- Real Time Clock (RTC): Set by the RTCEN bit in the Standby Area Control Register (RCC_BDCR).
- Internal oscillator (LSI): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768KHz oscillator (LSE): Set by the LSEON bit in the spare area control register (RCC_BDCR).

Exiting Standby mode

When an external reset (NRST pin), IWDG reset, a rising edge on the WKUP pin, or a rising edge of an RTC alarm event (see RTC block diagram), the microcontroller exits from standby mode. After waking up from standby, all registers are reset except for the Power Control/Status Register (PWR_CSR).

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

Refer to Table 14 for details on how to exit the Standby mode.

Table 14. Standby Mode

Standby Mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - Set SLEEPDEEP bit in CPU System Control register - Set PDDS bit in Power Control register (PWR_CR) - Clear WUF bit in Power Control/Status register (PWR_CSR)
Exit	WKUP pin rising edge, RTC alarm event, external reset in NRST pin, and IWDG reset
Wakeup latency	Activating power regulator in the reset stage.

I/O states in Standby mode

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

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

Debug mode

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

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to Section ”Debug Support for Low-power Modes”.

4.3.6 Automatic wake-up (AWU) in low power mode

The RTC can wake up the microcontroller in low-power mode (auto-wake mode) without relying on external interrupts. The RTC provides a programmable time base for periodically waking up from stop or standby mode. Two of the three RTC clock sources can be selected to implement this function by programming the RTCSEL[1:0] bits of the Backup Area Control Register (RCC_BDCR).

- Low power 32.768KHz external crystal oscillator (LSE)
  - This clock source provides a low power and accurate time reference. (In typical cases, consume less than 1μA)
- Low-Power Internal Oscillator (LSI Oscillator)
  - Using this clock source saves the cost of a 32.768KHz crystal. But the oscillator will increase power consumption a little.

In order to wake the system from shutdown mode with an RTC alarm event, the following actions must be taken:

- Configure external interrupt line 17 to trigger on rising edge.
- Configure RTC to generate RTC alarm events.

4.4 Power control registers

Table 15. Overview of Power Control Registers

Offset	Acronym	Register Name	Reset	Section
0x00	PWR_CR	Power control register	0x00000000	section 4.4.1
0x04	PWR_CSR	Power control/status register	0x00000000	section 4.4.2

4.4.1 Power control registers(PWR_CR)

Address offset: 0x00

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLS				DBP	Reserved			PVDE	CSBF	CWUF	PDDS	Res.
			rw	rw	rw	rw	rw				rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
31 : 13	Reserved			always read as 0.

Bit	Field	Type	Reset	Description
12 : 9	PLS	rw	0x00	PVD level selection These bits are used to select the voltage threshold detected by the Power Voltage Detector 0000: 1.8V 0100: 3.0V 1000: 4.2V 0001: 2.1V 0101: 3.3V 1001: 4.5V 0010: 2.4V 0110: 3.6V 1010: 4.8V 0011: 2.7V 0111: 3.9V Other:reserved Note: Refer to the electrical characteristics of the datasheet for more details.
8	DBP	rw	0x00	domain write protection After reset, the RTC and the backup registers are protected from accidental writes. Set this to allow writing to these registers. 1: Allow writing to RTC and backing registers 0: Writing to RTC and backing registers is prohibited Note: If the clock of the RTC is HSE/128, this bit must remain at '1'.
7:5	Reserved			always read as 0.
4	PVDE	rw	0x00	Power voltage detector enable 1: PVD enabled 0: PVD disabled
3	CSBF	rw	0x00	Clear standby flag Always read as 0 1: Clear the SBF Standby Flag (write). 0: No effect
2	CWUF	rw	0x00	Clear wakeup flag Always read as 0 1: Clear the WUF Wakeup Flag after 2 System clock cycles (write) 0: No effect
1	PDDS	rw	0x00	Power down deepsleep 1: Enter Standby mode when the CPU enters Deepsleep. 0: Enter Stop mode when the CPU enters Deepsleep.
0	Reserved			always read as 0.

4.4.2 Power control/status register(PWR_CSR)

Address offset: 0x04

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

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EWUP	Reserved					PVDO	SBF	WUF
							rw						rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 9	Reserved			always read as 0.
8	EWUP	rw	0x00	Enable WKUP pin 1: WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes up the system from Standby mode). 0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup CPU from Standby mode. Note: This bit is reset by a system reset.
7:3	Reserved			always read as 0.
2	PVDO	rw	0x00	PVD output It is valid only if PVD is enabled by the PVDE bit. 1: V_{DD}/V_{DDA} is lower than the PVD threshold selected with the PLS bits. 0: V_{DD}/V_{DDA} is higher than the PVD threshold selected with the PLS bits. Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.
1	SBF	rw	0x00	Standby flag This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the Power control register (PWR_CR). 1: System has been in Standby mode 0: System has not been in Standby mode
0	WUF	rw	0x00	Wakeup flag This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CWUF bit in the Power control register (PWR_CR). 1: A wakeup event was received from the WKUP pin or from the RTC alarm 0: No wakeup event occurred Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.

5 Backup registers (BKP)

Backup registers (BKP)

5.1 BKP introduction

The backup registers are 10 16-bit registers for storing 20 bytes of user application data. They are implemented in the backup domain that remains powered on by V_{BAT} when the V_{DD} power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.

In addition, the BKP Control Register is used to manage intrusion detection and RTC calibration functions. After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access.

To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register.
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

5.2 BKP main features

- 20-byte backup data registers.
- a status/control register used to manage anti-intrusion detection and have a medium function.
- calibration register for storing the RTC calibration value.

5.3 BKP functional description

5.3.1 Tamper detection

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the Backup control register (BKP_CR). A tamper detection event resets all data backup registers.

However to avoid losing Tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a Tamper event in case it occurs before the TAMPER pin is enabled.

- when TPAL=0: If the TAMPER pin is already high before it is enabled (by setting TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set).
- When TPAL=1: If the TAMPER pin is already low before it is enabled (by setting the TPE

bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled(while there was no falling edge on the TAMPER pin after TPE was set).

By setting the TPIE bit in the BKP_CSR register, an interrupt is generated when a Tamper detection event occurs.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.

5.3.2 RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the RTC clock calibration register (BKP_RTCCR).

The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

5.4 BKP description

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

Table 16. Overview of BKP Registers

Offset	Acronym	Register Name	Reset	Section
0x04 + 4 × (n - 1)	BKP_DRn	Backup data register x	0x00000000	section 5.4.1
0x2C	BKP_RTCCR	RTC clock calibration register	0x00000000	section 5.4.2
0x30	BKP_CR	Backup control register	0x00000000	section 5.4.3
0x34	BKP_CSR	Backup control/status register	0x00000000	section 5.4.4

5.4.1 Backup data register n(BKP_DRn)(n = 1...10)

Address offset: 0x04 + 4 × (backup data register number - 1)

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	BKP	rw	0x0000	BKP: backup data These bits can be used to write user data. Note: The BKP_DRn registers are not reset by a System reset or Power reset or when the system wakes up from Standby mode. They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).

5.4.2 RTC clock calibration register(BKP_RTCCR)

Address offset:0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ASOS	ASOE	CCO	CAL						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 :10	Reserved			Always read as 0.
9	ASOS	rw	0x00	ASOS: Alarm or second output selection When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal. 0：RTC Alarm pulse output selected 1：RTC Second pulse output selected Note: This bit is reset only by a Backup domain reset.
8	ASOE	rw	0x00	ASOE: Alarm or second output enable Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit. The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set. Note: This bit is reset only by a Backup domain reset.
7	CCO	rw	0x00	CCO: Calibration clock output 0: No effect 1：Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection. Note: This bit is reset when the V_{DD} supply is powered off.
6 : 0	CAL	rw	0x00	CAL: Calibration value This value indicates the number of clock pulses that will be ignored every 2^{20} clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of $1000000/2^{20}$PPM. The clock of the RTC can be slowed down from 0 to 121PPM.

5.4.3 Backup control register(BKP_CR)

Address offset:0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TPAL	TPE
														rw	rw

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1	TPAL	rw	0x00	TPAL: TAMPER pin active level 0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set). 1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).
0	TPE	rw	0x00	TPE: TAMPER pin enable 0: The TAMPER pin is free for general purpose I/O. 1: Tamper alternate I/O function is activated.

Note: Setting the TPAL and TPE bits at the same time is always safe, however resetting both at the same time can generate a spurious Tamper event. For this reason it is recommended to change the TPAL bit only when the TPE bit is reset.

5.4.4 Backup control/status register(BKP_CSR)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved					TPIE	CTI	CTE
						r	r						rw	w	w

Bit	Field	Type	Reset	Description
15 : 10	Reserved			Always read as 0.
9	TIF	r	0x90	TIF: Tamper interrupt flag This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset. 0: No Tamper interrupt 1: A Tamper interrupt occurred Note: This bit is reset only by a system reset and wakeup from Standby mode.

Bit	Field	Type	Reset	Description
8	TEF	r	0x00	TEF: Tamper event flag This bit is set by hardware when a Tamper event is detected. It is cleared by writing 1 to the CTE bit. 0: No Tamper event 1: A Tamper event occurred Note: A Tamper event resets all the BKP_DRn registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRn registers is performed while this bit is set, the value will not be stored.
7 : 3	Reserved			Always read as 0.
2	TPIE	rw	0x00	TPIE: TAMPER pin interrupt enable 0: Tamper interrupt disabled 1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register Note1: A Tamper interrupt does not wake up the core from low-power modes. Note2: This bit is reset only by a system reset and wakeup from Standby mode.
1	CTI	w	0x00	CTI: Clear tamper interrupt This bit is write only, and is always read as 0. 0: No effect 1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.
0	CTE	w	0x00	CTE: Clear tamper event This bit is write only, and is always read as 0. 0: No effect 1: Reset the TEF Tamper event flag (and the Tamper detector)

6 Reset and clock control (RCC)

Reset and clock control (RCC)

6.1 Reset

There are three types of reset, defined as system reset ,power reset and Backup area reset.

6.1.1 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register, standby and wakeup flags in power control register (CSR) and the registers in the Backup domain.

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

1. A low level on the NRST pin (external reset)
2. Window watchdog end of count condition (WWDG reset)
3. Independent watchdog end of count condition (IWDG reset
4. A software reset (SW reset)

The reset source can be identified by checking the reset flags in the Control/Status register (RCC_CSR).

Software reset

The SYSRESETREQ bit in CPU Application Interrupt and Reset Control Register must be Set to ”1” to force a software reset.

6.1.2 Power reset

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

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

A power reset sets all registers to their reset values except the Backup domain.

These sources in Figure 11 act on the NRST pin and it is always kept low during Reset. The RESET service routine vector is fixed at address 0x0000_0004.

Figure 11. Reset Circuit

6.1.3 Backup domain reset

The backup area has a dedicated reset that only affects the backup area. The backup area reset can be generated by setting the BDRST bit in the backup area control register RCC_BDCR.

Note: After V_{BAT} is powered up and before BDRST is reset, the backup area is not reset. You need to set the BDRST bit to reset the backup area.

6.2 Clocks

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

- HSI oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following two secondary clock sources:

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

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

Figure 12. Clock Tree

Several prescaler factors allow the configuration of the frequency of AHB, the high-speed APB (APB2) and the low-speed APB (APB1) domains. The maximum frequency of the AHB, APB1 and the APB2 domains is 96MHzMHz.

The RCC feeds the CPU System Timer (SysTick) external clock with the AHB clock divided by 8. This clock can be used or AHB clock used as SysTick through the SysTick Control and by configuring Status Register. The ADC clock is generated by dividing the high-speed APB2 clock.

The timer clock frequencies are automatically fixed by hardware. There are two cases:

1. If the APB prescaler factor is 1, the timer clock frequencies are Set to the same frequency as that of the APB domain to which the timers are connected.
2. Otherwise, they are Set to twice (×2) the frequency of the APB domain to which the timers are connected.

FCLK is the free running clock.

6.2.1 HSE clock

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

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

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

Figure 13. Clock Sources

External clock source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 24 MHz. You can select this mode by setting the HSEBYP and HSEON bits in the Clock control register (RCC_CR). The external clock signal (square, sinus or triangle) with 50% dutycycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z.

External crystal/ceramic resonator (HSE crystal)

The external oscillator has the advantage of producing a very accurate rate on the main clock. The associated hardware configuration is shown in Figure 13. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag in the Clock control register (RCC_CR) indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is Set to '1' by hardware. An interrupt can be generated if enabled in the Clock interrupt register (RCC_CIR).

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

6.2.2 HSI clock

The HSI clock signal is generated from an internal HSI Oscillator and can be used directly as a system clock or as PLL input. The HSI oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator, however, even with calibration the frequency is less accurate.

Calibration

The oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1%(25°C). After reset, the factory calibration value is loaded in the HSICAL bits in the Clock control register (RCC_CR).

The HSIRDY flag in the Clock control register (RCC_CR) indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is Set to '1' by hardware. The HSI RC can be switched on and off using the HSION bit in the Clock control register (RCC_CR).

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to Section section 6.2.7。

6.2.3 PLL

The internal PLL can be used to multiply the HSI RC output or HSE crystal output clock frequency. Refer to Figure 12 and Clock control register (RCC_CR). The PLL configuration (selection of HSI oscillator or HSE oscillator for PLL input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

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

If the USB interface is used in the application, the PLL must be programmed to output 48 or 96 MHz. This is needed to provide a 48 MHz USBCLK.

6.2.4 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit in Backup domain control register (RCC_BDCR). The LSERDY flag in the Backup domain control register (RCC_BDCR) indicates if the LSE crystal is stable or not. At startup, the LSE crystal output clock signal

is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt register (RCC_CIR).

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz.You select this mode by setting the LSEBYP and LSEON bits in the Backup domain control register (RCC_BDCR). The external clock signal (square, sinus or triangle) with 50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z.

6.2.5 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and Auto-wakeup unit (AWU). The clock frequency is around 40 kHz (between 30 kHz and 60 kHz). For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the Control/status register (RCC_CSR). The LSIRDY flag in the Control/status register (RCC_CSR) indicates if the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is Set to ’1’ by hardware. An LSI interrupt request can be generated if enabled in the Clock interrupt register (RCC_CIR).

6.2.6 System clock (SYSCLK) selection

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

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). The switch will occur only when the clock source is ready. Status bits in the Clock configuration register (RCC_CFGR). indicate which clock(s) is (are) ready and which clock is currently used as system clock.

6.2.7 Clock security system (CSS)

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

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1) and an interrupt is generated to inform the software about the failure (Clock Security SystemInterrupt CSSI), allowing the software to perform rescue operations. The CSSI is linked to the CPU 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. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the Clock interrupt register (RCC_CIR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure

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

6.2.8 RTC clock

The RTCCLK clock source can be either the HSE/128, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the Backup domain control register (RCC_BDCR). This selection cannot be modified without resetting the Backup domain.

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

- If LSE is selected as RTC clock:
  - The RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as Auto-Wakeup unit (AWU) clock:refer to section 6.2.5。
  - The AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock divided by 128 is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.5 V domain).
  - The DPB bit (disable backup domain write protection) in the Power controller register must be set to 1.

6.2.9 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator stabilization, the clock is provided to the IWDG.

6.2.10 Clock-out capability

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin.

The registers of the corresponding GPIO port shall be configured with functions.One of 6 clock signals can be selected as the MCO clock.

- SYSCLK
- HSI
- HSE
- LSI
- LSE
- PLLCLK/2
  - The selection is controlled by the MCO [2:0] bits of the Clock configuration register (RCC_CFGR).

6.3 RCC Register file and memory mapping description

Table 17. Overview of RCC Registers

Offset	Acronym	Register Name	Reset	Section
0x00	RCC_CR	Clock control register	0x0000FF01	section 6.3.1
0x04	RCC_CFGR	Clock configuration register	0x00000000	section 6.3.2
0x08	RCC_CIR	Clock interrupt register	0x00000000	section 6.3.3
0x0C	RCC_APB2RSTR	APB2 peripheral reset register	0x00000000	section 6.3.4
0x10	RCC_APB1RSTR	APB1 peripheral reset register	0x00000000	section 6.3.5
0x14	RCC_AHBENR	AHB peripheral clock enable register	0x00000014	section 6.3.6
0x18	RCC_APB2ENR	APB2 peripheral clock enable register	0x00000000	section 6.3.7
0x1C	RCC_APB1ENR	APB1 peripheral clock enable register	0x00000000	section 6.3.8
0x20	RCC_BDCR	Backup domain control register	0x00000000	section 6.3.9
0x24	RCC_CSR	Control status register	0xXC000000	section 6.3.10
0x40	RCC_SYSCFG	System configuration register	0x00000000	section 6.3.11

6.3.1 Clock control register(RCC_CR)

Address offset: 0x00

Reset value: 0x0000 FF01

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLMUL						PLLRDY	PLLON	Res.	PLLDIV			CSSON	HSEBYP	HSERDY	HSEON
rw	rw	rw	rw	rw	rw	r	rw		rw	rw	rw	rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL								Reserved						HSIRDY	HSION
r	r	r	r	r	r	r	r							r	rw

Bit	Field	Type	Reset	Description
31 : 26	PLLMUL	rw	0x00	PLLMUL: PLL multiplication factor
25	PLLRDY	r	0x00	PLLRDY: PLL clock ready flag Set to '1' by hardware to indicate that the PLL is locked. 0: PLL unlocked 1: PLL locked
24	PLLON	rw	0x00	PLLON: PLL enable Set to '1' or cleared by software. Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock. 0: PLL OFF 1: PLL ON
23	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
22 : 20	PLLDIV	rw	0x00	PLLDIV: PLL divider factor PLL configuration formula: $F_{CLKO} = F_{REFIN} * N / M$ Where: $F_{CLKO}$is the PLL output frequency and F_{REFIN} is the PLL input reference clock frequency M = PLLDIV + 1 N = PLLMUL + 1
19	CSSON	rw	0x00	CSSON: Clock security system enable Set to ’1’ or cleared by software to enable the clock detector . 0: Clock detector OFF 1: Clock detector ON if the external oscillator is ready
18	HSEBYP	rw	0x00	HSEBYP: External high-speed clock bypass Set to ’1’ or cleared by software to bypass the oscillator with an external clock. The HSEBYP bit can be written only if the external oscillator is disabled. 0: external oscillator not bypassed 1: external oscillator bypassed with external clock
17	HSERDY	r	0x00	HSERDY: External high-speed clock ready flag Set to ’1’ by hardware to indicate that the external clock is stable. 0: External clock not ready 1: External clock ready
16	HSEON	rw	0x00	HSEON: External high-speed clock enable Set to ’1’ or cleared by software. Cleared by hardware to stop the external clock when entering Stop or Standby mode. This bit cannot be reset if the external clock is used directly or indirectly as the system clock. 0: HSE oscillator OFF 1: HSE oscillator ON
15: 8	HSICAL	r	0xFF	HSICAL: Internal high-speed clock calibration These bits are initialized automatically at startup.
7: 2	Reserved			Always read as 0.
1	HSIRDY	r	0x00	HSIRDY: Internal high-speed clock ready flag A ’1’ is set by hardware to indicate that the internal 8 MHz clock h as stabilized. After the HSION bit is cleared, this bit requires six internal 8MHz clock cycles to be cleared. 0: Internal 8MHz clock is not ready 1: Internal 8MHz clock ready

Bit	Field	Type	Reset	Description
0	HSION	rw	0x01	HSION: Internal high-speed clock enable Set to '1' or cleared by software. Set to '1' by hardware to force the internal 8 MHz RC oscillator ON when leaving Stop or Standby mode or in case of failure of the external oscillator used as system clock. This bit cannot be reset if the internal 8 MHz RC is used directly or indirectly as system clock or is selected to become the system clock. 0: internal 8 MHz RC oscillator OFF 1: internal 8 MHz RC oscillator ON

6.3.2 Clock configuration register(RCC_CFGR)

Address offset：0x04

Reset value：0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO			USBPRE		Reserved				PLLXTPRE	PLLSRC
					rw	rw	rw	rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PPRE2			PPRE1			HPRE				SWS		SW	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit	Field	Type	Reset	Description
31 : 27	Reserved			Always read as 0.
26 : 24	MCO	rw	0x00	MCO: Microcontroller clock output Set to '1' or cleared by software. 00x: No clock; 010: LSI clock selected; 011 : LSE clock selected; 100: System clock (SYSCLK) selected; 101: HSI clock selected; 110: HSE clock selected; 111：PLL clock divided by 2 selected; Note: 1. This clock output may have some truncated cycles at startup or during MCO clock source switching. 2. When the System Clock is selected to output to the MCO pin, make sure that this clock does not exceed 50 MHz (the maximum I/O speed).

Bit	Field	Type	Reset	Description
23：22	USBPRE	rw	0x00	USBPRE: USB prescaler Set to '1' and cleared by software to generate 48 MHz USB clock. This bit must be valid before enabling the USB clock in the RCC_APB1ENR register. 00: PLL clock is directly used as the USB clock 01: PLL clock is divided by 2 and used as the USB clock 10: PLL clock is divided by 3 and used as the USB clock 11: PLL clock is divided by 4 and used as the USB clock
21：18	Reserved			Always read as 0.
17	PLLXTPRE	rw	0x00	PLLXTPRE: HSE divider for PLL entry Set to '1' and cleared by software to divide HSE before PLL entry. This bit can be written only when PLL is disabled. 0: HSE clock not divided 1: HSE clock divided by 2
16	PLLSRC	rw	0x00	PLLSRC: PLL entry clock source Set to '1' and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled. 0: HSI oscillator clock /4 selected as PLL input clock 1: HSE oscillator clock selected as PLL input clock
15: 14	Reserved			Always read as 0.
13: 11	PPRE2	rw	0x00	PPRE2: APB high-speed prescaler(APB2) Set to '1' and cleared by software to control the prescaler factor of the APB high-speed clock (PCLK2). 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
10: 8	PPRE1	rw	0x00	PPRE1: APB low-speed prescaler(APB1) Set to '1' and cleared by software to control the prescaler factor of the APB1 low-speed clock (PCLK1). 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16

Bit	Field	Type	Reset	Description
7: 4	HPRE	rw	0x00	HPRE: AHB Prescaler Set to '1' and cleared by software to control the prescaler factor of the AHB clock. 0xxx: SYSCLK not divided 1000: SYSCLK divided by 2 1001: SYSCLK divided by 4 1010: SYSCLK divided by 8 1011: SYSCLK divided by 16 1100: SYSCLK divided by 64 1101: SYSCLK divided by 128 1110: SYSCLK divided by 256 1111: SYSCLK divided by 512 Note: 1. The prefetch buffer must be kept on when using a prescaler factor greater than 1 on theAHB clock. Refer to Section "Reading the Flash Memory" for more details.
3: 2	SWS	r	0x00	SWS: System clock switch status Set to '1' and cleared by hardware to indicate which clock source is used as system clock. 00: HSI oscillator divided by 6 and used as system clock 01: HSE oscillator used as system clock 10: PLL used as system clock 11: Not available.
1: 0	SW	rw	0x00	SW: System clock switch Set to '1' and cleared by software to select SYSCLK source. Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock. 00: HSI oscillator divided by 6 and used as system clock 01: HSE oscillator selected as system clock 10: PLL used as system clock 11: Not available.

6.3.3 Clock interrupt register(RCC_CIR)

Address offset：0x08

Reset value：0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved		PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 24	Reserved			Always read as 0.
23	CSSC	rc_w1	0x00	CSSC: Clock security system interrupt clear This bit is set to ’1’ by software to clear the CSSF flag. 0: No effect 1: Clear CSSF flag
22 : 21	Reserved			Always read as 0.
20	PLLRDYC	rc_w1	0x00	PLLRDYC: PLL ready interrupt clear This bit is set to ’1’ by software to clear the PLLRDYF flag. 0: No effect 1: PLLRDYF cleared
19	HSERDYC	rc_w1	0x00	HSERDYC: HSE ready interrupt clear This bit is set to ’1’ by software to clear the HSERDYF flag. 0: No effect 1: HSERDYF cleared
18	HSIRDYC	rc_w1	0x00	HSIRDYC: HSI ready interrupt clear This bit is set to ’1’ by software, to clear the HSIRDYF flag. 0: No effect 1: HSIRDYF cleared
17	LSERDYC	rc_w1	0x00	LSERDYC: LSE ready interrupt clear This bit is set by software to clear the LSERDYF flag. 0: No effect 1：LSERDYF cleared
16	LSIRDYC	rc_w1	0x00	LSIRDYC: LSI ready interrupt clear This bit is set to ’1’ by software to clear the LSIRDYF flag. 0: No effect 1: LSIRDYF cleared
15: 13	Reserved			Always read as 0.
12	PLLRDYIE	rw	0x00	PLLRDYIE: PLL ready interrupt enable Set to ’1’ and cleared by software to enable/disable interrupt caused by PLL lock. 0: PLL lock interrupt disabled 1: PLL lock interrupt enabled

Bit	Field	Type	Reset	Description
11	HSERDYIE	rw	0x00	HSERDYIE: HSE ready interrupt enable Set to '1' and cleared by software to enable/disable interrupt caused by the external oscillator stabilization. 0: HSE ready interrupt disabled 1: HSE ready interrupt enabled
10	HSIRDYIE	rw	0x00	HSIRDYIE: HSI ready interrupt enable Set to '1' and cleared by software to enable/disable interrupt caused by the internal 8 MHz RC oscillator stabilization. 0: HSI ready interrupt disabled 1: HSI ready interrupt enabled
9	LSERDYIE	rw	0x00	LSERDYIE: LSE ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the external 32 kHz oscillator stabilization. 0: LSE ready interrupt disabled 1: LSE ready interrupt enabled
8	LSIRDYIE	rw	0x00	LSIRDYIE: LSI ready interrupt enable Set to '1' and cleared by software to enable/disable interrupt caused by the internal 40 kHz RC oscillator stabilization. 0: LSI ready interrupt disabled 1: LSI ready interrupt enabled
7	CSSF	r	0x00	CSSF: Clock security system interrupt flag Set to '1' by hardware when a failure is detected in the external oscillator. Cleared by software through setting the CSSC bit to '1'. 0: No clock security interrupt caused by HSE clock failure 1: Clock security interrupt caused by HSE clock failure
6: 5	Reserved			Always read as 0.
4	PLLRDYF	r	0x00	PLLRDYF: PLL ready interrupt flag Set to '1' by hardware when the PLL is ready and PLLRDYDIE is set to '1'. Cleared by software setting the PLLRDYC bit. 0: No clock ready interrupt caused by PLL lock 1: Clock ready interrupt caused by PLL lock
3	HSERDYF	r	0x00	HSERDYF: HSE ready interrupt flag Set to '1' by hardware when External High Speed clock becomes stable and HSERDYDIE is set to '1'. Cleared by software setting the HSERDYC bit. 0：No clock ready interrupt caused by the external oscillator 1: Clock ready interrupt caused by the external oscillator

Bit	Field	Type	Reset	Description
2	HSIRDYF	r	0x00	HSIRDYF: HSI ready interrupt flag Set to '1' by hardware when the Internal High Speed clock becomes stable and HSIRDYDIE is set. Cleared by software setting the HSIRDYC bit. 0: No clock ready interrupt caused by the internal RC oscillator 1: Clock ready interrupt caused by the internal RC oscillator
1	LSERDYF	r	0x00	LSERDYF: LSE ready interrupt flag Set by hardware when the External Low Speed clock becomes stable and LSERDYDIE is set. Cleared by software setting the LSERDYC bit. 0: No clock ready interrupt caused by the external 32 kHz oscillator 1: Clock ready interrupt caused by the external 32 kHz oscillator
0	LSIRDYF	r	0x00	LSIRDYF: LSI ready interrupt flag Set to '1' by hardware when the internal low speed clock becomes stable and LSIRDYDIE is set. Cleared by software setting the LSIRDYC bit. 0: No clock ready interrupt caused by the internal RC 40 kHz oscillator 1: Clock ready interrupt caused by the internal RC 40 kHz oscillator

6.3.4 APB2 peripheral reset register(RCC_APB2RSTR)

Address offset：0x0C

Reset value：0x0000 0000

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UART1	Res.	SPI1	TIM1	ADC2	ADC1	Reserved		IOPE	IOPD	IOPC	IOPB	IOPA	Res.	AFIO
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
31 : 15	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
14	UART1	rw	0x00	UART1: UART1 reset Set to '1' or cleared by software. 0: No effect 1：Reset UART1
13	Reserved			Always read as 0.
12	SPI1	rw	0x00	SPI1: SPI1 reset Set to '1' or cleared by software. 0: No effect 1：Reset SPI1
11	TIM1	rw	0x00	TIM1: TIM1 timer reset Set to '1' or cleared by software. 0: No effect 1：Reset TIM1 timer
10	ADC2	rw	0x00	ADC2: ADC2 interface reset Set to '1' or cleared by software. 0: No effect 1：Reset ADC2 interface
9	ADC1	rw	0x00	ADC1: ADC1 interface reset Set to '1' or cleared by software. 0: No effect 1：Reset ADC1 interface
8: 7	Reserved			Always read as 0.
6	IOPE	rw	0x00	IOPE: I/O port E reset Set to '1' or cleared by software. 0: No effect 1：Reset I/O port E
5	IOPD	rw	0x00	IOPD: I/O port D reset Set to '1' or cleared by software. 0: No effect 1：Reset I/O port D
4	IOPC	rw	0x00	IOPC: I/O port C reset Set to '1' or cleared by software. 0: No effect 1：Reset I/O port C
3	IOPB	rw	0x00	IOPB: I/O port B reset Set to '1' or cleared by software. 0: No effect 1：Reset I/O port B
2	IOPA	rw	0x00	IOPA: I/O port A reset Set to '1' or cleared by software. 0: No effect 1：Reset I/O port A

Bit	Field	Type	Reset	Description
1	Reserved			Always read as 0.
0	AFIO	rw	0x00	AFIO: Alternate function I/O reset Set to '1' or cleared by software. 0: No effect 1：Reset Alternate Function

6.3.5 APB1 peripheral reset register(RCC_APB1RSTR)

Address offset：0x10

Reset value：0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			PWR	BKP	Res.	CAN	Res.	USB	I2C2	I2C1	Reserved		UART3	UART2	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SIP2	Reserved		WWDG	Reserved								TIM4	TIM3	TIM2

Bit	Field	Type	Reset	Description
31 : 29	Reserved			Always read as 0.
28	PWR	rw	0x00	PWR: Power interface reset Set to '1' or cleared by software. 0: No effect 1: Reset power interface
27	BKP	rw	0x00	BKP: Backup interface reset Set to '1' or cleared by software. 0: No effect 1：Reset backup interface
26	Reserved			Always read as 0.
25	CAN	rw	0x00	CAN: CAN reset Set to '1' or cleared by software. 0: No effect 1: Reset CAN
24	CRS	rw	0x00	CRS: CRS interface reset Set to '1' or cleared by software. 0: No effect 1: Reset CRS
23	USB	rw	0x00	USB: USB reset Set to '1' or cleared by software. 0: No effect 1: Reset USB

Bit	Field	Type	Reset	Description
22	I2C2	rw	0x00	I2C2: I2C2 reset Set to '1' or cleared by software. 0: No effect 1：Reset I2C2
21	I2C1	rw	0x00	I2C1: I2C1 reset Set to '1' or cleared by software. 0: No effect 1: Reset I2C1
20 : 19	Reserved			Always read as 0.
18	UART3	rw	0x00	UART3: UART3 reset Set to '1' or cleared by software. 0: No effect 1：Reset UART3
17	UART2	rw	0x00	UART2: UART2 reset Set to '1' or cleared by software. 0: No effect 1: Reset UART2
16 : 15	Reserved			Always read as 0.
14	SPI2	rw	0x00	SPI2: SPI2 reset Set to '1' or cleared by software. 0: No effect 1: Reset SPI2
13 : 12	Reserved			Always read as 0.
11	WWDG	rw	0x00	WWDG: Window watchdog reset Set to '1' or cleared by software. 0: No effect 1: Reset window watchdog
10 : 3	Reserved			Always read as 0.
2	TIM4	rw	0x00	TIM4: Timer4 reset Set to '1' or cleared by software. 0: No effect 1：Reset TIM4 timer
1	TIM3	rw	0x00	TIM3: Timer3 reset Set to '1' or cleared by software. 0: No effect 1: Reset TIM3 timer
0	TIM2	rw	0x00	TIM2: Timer2 reset Set to '1' or cleared by software. 0: No effect 1: Reset TIM2 timer

6.3.6 AHB peripheral clock enable register(RCC_AHBENR)

Address offset：0x14

Reset value：0x0000 0014

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AES	CRC	Res.	FLASH	Res.	SRAM	Res.	DMA
								rw	rw		rw		rw		rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Always read as 0.
7	AES	rw	0x00	AES: AES clock enable Set to ’1’ or cleared by software. 0: AES clock disabled 1: AES clock enabled
6	CRC	rw	0x00	CRC: CRC clock enable Set to ’1’ or cleared by software. 0: CRC clock disabled 1: CRC clock enabled
5	Reserved			Always read as 0.
4	FLASH	rw	0x01	FLASH: FLASH clock enable 0：FLASH clock disabled 1：FLASH clock enabled
3	Reserved			Always read as 0.
2	SRAM	rw	0x01	SRAM: SRAM interface clock enable Set to ’1’ or clear by software, to enable/disable the SRAM clock in Sleep mode. 0: SRAM clock disabled in Sleep mode. 1: SRAM clock enabled in Sleep mode.
1	Reserved			Always read as 0.
0	DMA	rw	0x00	DMA: DMA clock enable Set to ’1’ or cleared by software. 0：DMA clock disabled 1：DMA clock enabled

6.3.7 APB2 peripheral clock enable register(RCC_APB2ENR)

Address offset：0x18

Reset value：0x0000 0000

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

Note: When the peripheral clock is not active, the peripheral registervalues may not be read by software.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UART1	Res.	SPI1	TIM1	ADC2	ADC1	Reserved			IOPD	IOPC	IOPB	IOPA	Res.	AFIO
	rw		rw	rw	rw	rw				rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
31 : 15	Reserved			Always read as 0.
14	UART1	rw	0x00	UART1: UART1 clock enable Set to ’1’ or cleared by software. 0：UART1 clock disabled 1：UART1 clock enabled
13	Reserved			Always read as 0.
12	SPI1	rw	0x00	SPI1: SPI1 clock enable Set to ’1’ or cleared by software. 0：SPI1 clock disabled 1：SPI1 clock enabled
11	TIM1	rw	0x00	TIM1: TIM1 Timer clock enable Set to ’1’ or cleared by software. 0: TIM1 clock disabled 1: TIM1 clock enabled
10	ADC2	rw	0x00	ADC2: ADC2 interface clock enable Set to ’1’ or cleared by software. 0: ADC2 interface clock disabled 1: ADC2 interface clock enabled
9	ADC1	rw	0x00	ADC1: ADC1 interface clock enable Set to ’1’ or cleared by software. 0: ADC1 interface clock disabled 1: ADC1 interface clock enabled
8：6	Reserved			Always read as 0.
5	IOPD	rw	0x00	IOPD: I/O port D clock enable Set to ’1’ or cleared by software. 0：I/O port D clock disabled 1：I/O port D clock enabled
4	IOPC	rw	0x00	IOPC: I/O port C clock enable Set to ’1’ or cleared by software. 0：I/O port C clock disabled 1：I/O port C clock enabled

Bit	Field	Type	Reset	Description
3	IOPB	rw	0x00	IOPB: I/O port B clock enable Set to '1' or cleared by software. 0：I/O port B clock disabled 1：I/O port B clock enabled
2	IOPA	rw	0x00	IOPA: I/O port A clock enable Set to '1' or cleared by software. 0：I/O port A clock disabled 1：I/O port A clock enabled
1	Reserved			Always read as 0.
0	AFIO	rw	0x00	AFIO: Alternate function I/O clock enable Set to '1' or cleared by software. 0：Alternate Function I/O clock disabled 1：Alternate Function I/O clock enabled

6.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset：0x1C

Reset value：0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			PWR	BKP	Res.	CAN	Res.	USB	I2C2	I2C1	Reserved		UART3	UART2	Res.
			rw	rw		rw		rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SIP2	Reserved		WWDG	Reserved								TIM4	TIM3	TIM2
	rw			rw									rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 29	Reserved			Always read as 0.
28	PWR	rw	0x00	PWR: Power interface clock enable Set to '1' or cleared by software. 0: Power interface clock disabled 1: Power interface clock enabled
27	BKP	rw	0x00	BKP: Backup interface clock enable Set to '1' or cleared by software. 0: Backup interface clock disabled 1: Backup interface clock enabled
26	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
25	CAN	rw	0x00	CAN: CAN clock enable Set to '1' or cleared by software. 0：CAN clock disabled 1：CAN clock enabled
24	Reserved			Always read as 0.
23	USB	rw	0x00	USB: USB clock enable Set to '1' or cleared by software. 0：USB clock disabled 1：USB clock enabled
22	I2C2	rw	0x00	I2C2: I2C2 clock enable Set to '1' or cleared by software. 0：I2C 2 clock disabled 1：I2C 2 clock enabled
20：19	Reserved			Always read as 0.
18	UART3	rw	0x00	UART3: UART3 clock enable Set to '1' or cleared by software. 0：UART3 clock disabled 1：UART3 clock enabled
17	UART2	rw	0x00	UART2: UART2 clock enable Set to '1' or cleared by software. 0：UART2 clock disabled 1：UART2 clock enabled
16：15	Reserved			Always read as 0.
14	SPI2	rw	0x00	SPI2: SPI2 clock enable Set to '1' or cleared by software. 0：SPI2 clock disabled 1：SPI2 clock enabled
13：12	Reserved			Always read as 0.
11	WWDG	rw	0x00	WWDG: Window watchdog clock enable Set to '1' or cleared by software. 0: Window watchdog clock disabled 1: Window watchdog clock enabled
10：3	Reserved			Always read as 0.
2	TIM4	rw	0x00	TIM4: TIM4 clock enable Set to '1' or cleared by software. 0: TIM4 clock disabled 1: TIM4 clock enabled
1	TIM3	rw	0x00	TIM3: TIM3 clock enable Set to '1' or cleared by software. 0: TIM3 clock disabled 1: TIM3 clock enabled

Bit	Field	Type	Reset	Description
0	TIM2	rw	0x00	TIM2: TIM2 clock enable Set to '1' or cleared by software. 0: TIM2 clock disabled 1: TIM2 clock enabled

6.3.9 Backup domain control register(RCC_BDCR)

Address offset：0x20

Reset value：0x0000 0000, reset by Backup domain Reset

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSEL		Reserved					LSEBYP	LSERDY	LSEON
rw						rw	rw						rw	r	rw

Bit	Field	Type	Reset	Description
31 : 17	Reserved			Always read as 0.
16	BDRST	rw	0x00	BDRST: Backup domain software reset Set to '1' or cleared by software. 0: Reset not activated 1: Resets the entire Backup domain
15	RTCEN	rw	0x00	RTCEN: RTC clock enable Set to '1' or cleared by software. 0：RTC clock disabled 1：RTC clock enabled
14：10	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
9：8	RTCSEL	rw	0x00	RTCSEL: RTC clock source selection Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them. 00: No clock 01: LSE oscillator clock used as RTC clock 10: LSI oscillator clock used as RTC clock 11: HSE oscillator clock divided by 128 used as RTC clock
7：3	Reserved			Always read as 0.
2	LSEBYP	rw	0x00	LSEBYP: External low-speed oscillator bypass Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the external 32 kHz oscillator is disabled. 0: LSE oscillator not bypassed 1: LSE oscillator bypassed
1	LSERDY	r	0x00	LSERDY: External low-speed oscillator ready Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles. 0: External 32 kHz oscillator not ready 1: External 32 kHz oscillator ready
0	LSEON	rw	0x00	LSEON: External low-speed oscillator enable Set and cleared by software. 0: External 32 kHz oscillator OFF 1: External 32 kHz oscillator ON

6.3.10 Control status register(RCC_CSR)

Address offset：0x24

Reset value：0xXC00 0000

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

Reset by system Reset, except reset flags by power Reset only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
	r	r	r	r	r		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

Bit	Field	Type	Reset	Description
31	Reserved			Always read as 0.
30	WWDGRSTF	r	0x0x	WDGRSTF: Window watchdog reset flag Set to ’1’ by hardware when a window watchdog reset occurs. Cleared by writing to the RMVF bit. 0: No window watchdog reset occurred 1: Window watchdog reset occurred
29	IWDGRSTF	r	0x0x	IWDGRSTF: Independent watchdog reset flag Set to ’1’ by hardware when an independent watchdog reset from VDD domain occurs. Cleared by writing to the RMVF bit. 0: No watchdog reset occurred 1: Watchdog reset occurred
28	SFTRSTF	r	0x0x	SFTRSTF: Software reset flag Set to ’1’ by hardware when a software reset occurs. Cleared by writing to the RMVF bit. Cleared by writing to the RMVF bit. 0: No software reset occurred 1: Software reset occurred
27	PORRSTF	r	0x01	PORRSTF: POR/PDR reset flag Set to ’1’ by hardware when a POR/PDR reset occurs. Cleared by writing to the RMVF bit. 0: No POR/PDR reset occurred 1: POR/PDR reset occurred
26	PINRSTF	r	0x01	PINRSTF: PIN reset flag Set to ’1’ by hardware when a reset from the NRST pin occurs. Cleared by writing to the RMVF bit. 0: No reset from NRST pin occurred 1: Reset from NRST pin occurred
25	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
24	RMVF	rw	0x00	RMVF: Remove reset flag Set to '1' by software to clear the reset flags. 0: No effect 1: Clear the reset flags
23 : 2	Reserved			Always read as 0.
1	LSIRDY	r	0x00	LSIRDY: Internal low-speed oscillator ready Set to '1' and cleared by software to indicate when the internal RC 40 kHz oscillator is ready. After the LSION bit is cleared, LSIRDY goes low after 3 internal RC 40 kHz oscillator clock cycles. 0: Internal RC 40 kHz oscillator not ready 1: Internal RC 40 kHz oscillator ready
0	LSION	rw	0x00	LSION: Internal low-speed oscillator enable Set to '1' or cleared by software. 0: Internal RC 40 kHz oscillator disabled 1: Internal RC 40 kHz oscillator enabled

6.3.11 System configuration register(RCC_SYSCFG)

Address offset：0x40

Reset value：0x0000 0000

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Reset by system Reset, except reset flags by power Reset only.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SECTOR_1K_CFG	Res.
														rw	

Bit	Field	Type	Reset	Description
31：2	Reserved			Always read as 0.
1	SECTOR_1K-_CFG	rw	0x00	SECTOR_1K_CFG: Flash page erasing. 1：1K bytes 0：512 bytes
0	Reserved			Always read as 0.

7 General-Purpose and Alternate Functions I/O (GPIO and AFIO)

General-Purpose and Alternate Functions I/O (GPIO and AFIO)

7.1 GPIO functional description

Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR),a 32-bit locking register (GPIOx_LCKR).)

Each port bits of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input pull-down
- Analog Input
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bits is freely programmable, however, the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed).

The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. In this way, there is no risk that an IRQ occurs between the read and the modify access.

The following figure shows the basic structure of an I/O port bits.

Figure 14. Basic Structure of I/O Port bits

Table 18. Port bits Configuration Table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-Pull	0	0	01		0 or 1
	Open-Drain		1			0 or 1
Alternate function output	Push-Pull	1	0			Not used
	Open-Drain		1			Not used
Input	Analog input	0	0	00		Not used
	Input floating		1			Not used
	Input pull-down	1	0			0
	Input pull-up					1

Table 19. Output MODE bits

MODE[1: 0]	Meaning
00	Reserved
01	Output

7.1.1 General-Purpose I/O(GPIO)

During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1: 0] = 01, MODEx[1: 0] = 00).

The JTAG pins are in input PU/PD after reset:

- PA15: JTDI is placed in pull-up mode
- PA14: JTCK is placed in pulldown mode
- PA13: JTMS is placed in pull-up mode
- PB4: JNTRST is placed in pull-up mode

When configured as output, the value written to the Output Data Register (GPIOx_ODR) is output to the I/O pin. It is possible to use the output driver in Push-Pull mode or Open-Drain mode (only the N-MOS is activated when outputting 0).

The Input Data register (GPIOx_IDR) captures the data present on the I/O pin at every APB2 clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down that can be activated or not when configured as input.

7.1.2 Atomic bits set or reset

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bits level:

it is possible to modify only one or several bits in a single APB2 write access.

This is achieved by programming to '1' the bits Set/Reset Register (GPIOx_BSRR, or for reset only GPIOx_BRR) to select the bits to modify. The unselected bits will not be modified.

7.1.3 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. For more information on external interrupts, refer to 8.2 External interrupt/event controller (EXTI).

7.1.4 Alternate functions

It is necessary to program the Port bits Configuration Register before using a default alternate function.

- For alternate function inputs, the port must be configured in Input mode (floating, pullup or pull-down) and the input pin must be driven externally.

Note: It is also possible to emulate the AFI input pin by software by programming the GPIO controller. In this case, the port should be configured in Alternate Function Output mode. And obviously, the corresponding port should not be driven externally as it will be driven by the software using the GPIO controller.

- For alternate function outputs, the port must be configured in Alternate Function Output mode (Push-Pull or Open-Drain).
- For bidirectional Alternate Functions, the port bits must be configured in Alternate Function Output mode (Push-Pull or Open-Drain). In this case the input driver is configured in input floating mode.

If a port bits is configured as Alternate Function Output, this disconnects the output register and connects the pin to the output signal of an on-chip peripheral. If software configures

a GPIO pin as Alternate Function Output, but peripheral is not activated, its output is not specified.

7.1.5 Software remapping of I/O alternate functions

To optimize the number of peripheral I/O functions for different device packages, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the corresponding registers (refer to AFIO registers).

In that case, the alternate functions are no longer mapped to their original assignations.

7.1.6 GPIO locking mechanism

The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bits, it is no longer possible to modify the value of the port bits until the next reset.

7.1.7 Input configuration

When the I/O Port is programmed as Input:

- The Output Buffer is disabled
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating);
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register obtains the I/O State

The following figure shows the Input Configuration of the I/O Port bits:

Figure 15. Input Floating/Pull Up/Pull Down Configurations

7.1.8 Output configuration

When the I/O Port is programmed as Output:

- The Output Buffer is enabled
  - Open drain mode: ’0’ on the output register activates N-MOS, and ’1’ on the output register places the port in a high-impedance state (P-MOS is never activated)
  - Push-pull mode: ’0’ on the output register activates N-MOS, and ’1’ on the output register activates P-MOS
- The Schmitt Trigger Input is activated.
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

The following figure shows the Output configuration of the I/O Port bits:

Figure 16. Output Configuration

7.1.9 Alternate functions configuration

When the I/O Port is programmed as Alternate Function:

- The Output Buffer is turned on in Open Drain or Push-Pull configuration
- The Output Buffer is driven by the signal coming from the peripheral (alternate function output)
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

The following figure shows the Alternate Function Configuration of the I/O Port bits. Also, refer to AFIO registers for further information.

A set of Alternate Function I/O registers allow the user to remap some alternate functions to different pins.

Figure 17. Alternate functions configuration

7.1.10 Analog configuration

When the I/O Port is programmed as Analog configuration:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin. The output of the Schmitt Trigger is forced to a constant value ’0’.
- The weak pull-up and pull-down resistors are disabled.
- Read access to the Input Data Register gets the value ’0’.

The following figure shows the high impedance-analog configuration of the I/O Port bits.

Figure 18. High Impedance-analog Configuration

7.1.11 GPIO configurations for device peripherals

The following tables give the GPIO configurations of the device peripherals:

Table 20. Advanced Timers TIM1

TIM1 pin	configuration	GPIO configuration
TIM1_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM1_CHxN	Complementary output channel x	Alternate function push-pull
TIM1_BKIN	Break input	Input floating
TIM1_ETR	External trigger timer input	Input floating

Table 21. General-Purpose timers TIM2/3/4

TIM2/3/4 pin	configuration	GPIO configuration
TIM2/3/4_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM2/3/4_ETR	External trigger timer input	Input floating

Table 22. UART

UART pin	configuration	GPIO configuration
UARTx_TX	Serial-port sending	Alternate function push-pull

UART pin	configuration	GPIO configuration
UARTx_RX	Serial-port receiving	Input floating/Input pull-up
UARTx_RTS	Hardware flow control	Alternate function push-pull
UARTx_CTS	Hardware flow control	Input floating/Input pull-up

Table 23. SPI

SPI pin	configuration	GPIO configuration
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex/Master	Alternate function push-pull
	Full duplex/Slave	Input floating/Input pull-up
SPIx_MISO	Full duplex/Master	Input floating/Input pull-up
	Full duplex/Slave	Alternate function push-pull
SPIx_NSS	Hardware master/Slave	Input floating/Input pull-up/Input pull-down
	Hardware master /NSS output enabled	Alternate function push-pull
	Software	Not used. Can be used as GPIO

Table 24. I2C

I2C pin	configuration	GPIO configuration
I2Cx_SCL	I2C clock	Alternate function open drain
I2Cx_SDA	I2C data	Alternate function open drain

Table 25. CAN

CAN pin	GPIO configuration
CAN_TX	Alternate function push-pull
CAN_RX	Input floating/Input pull-up

Table 26. ADC

ADC pin	GPIO configuration
ADC	Analog input

Table 27. Other I/Os

pin	configuration	GPIO configuration
TAMPER-RTC	RTC output	Forced by hardware when configuring the BKP_CP and BKP_RTCCR registers
	Temper Input	
MCO	Clock output	Alternate function push-pull

pin	configuration	GPIO configuration
EXTI input lines	External input interrupts	Input floating/Input pull-up/ input pull-down

7.2 GPIO register description

Table 28. Overview of GPIO Registers

Offset	Acronym	Register Name	Reset	Section
0x00	GPIOx_CRL	Port configuration low register	0xXXXXXXXX	section 7.2.1
0x04	GPIOx_CRH	Port configuration high register	0xXXXXXXXX	section 7.2.2
0x08	GPIOx_IDR	Port input data register	0x0000XXXX	section 7.2.3
0x0C	GPIOx_ODR	Port output data register	0x00000XXX	section 7.2.4
0x10	GPIOx_BSRR	Port set/reset register	0x00000000	section 7.2.5
0x14	GPIOx_BRR	Port bits reset register	0x00000000	section 7.2.6
0x18	GPIOx_LCKR	Port configuration lock register	0x00000000	section 7.2.7

7.2.1 Port configuration low register(GPIOx_CRL)(x = A..D)

Offset address: 0x00

Reset value:GPIOA_CRL: 0x4444 4848

GPIOB_CRL: 0x4466 6444

GPIOC_CRL: 0x0000 0000

GPIOD_CRL: 0x0000 4444

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CNF7	MODE7	CNF6	MODE6	CNF5	MODE5	CNF4	MODE4
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CNF3	MODE3	CNF2	MODE2	CNF1	MODE1	CNF0	MODE0
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 30 27: 26 23: 22 19: 18 15: 14 11: 10 7: 6 3: 2	CNFy	rw		Port x configuration bits(0…7) These bits are written by software to configure the corresponding I/O port. Refer to Table 18 : Port pin configuration In input mode (MODE = 00): 00: Analog mode 01: Floating input 10: Input with pull-up / pull-down 11: Reserved In output mode (MODE > 00): 00: General-purpose output push-pull 01: General-purpose output Open-drain 10: Alternate functionsoutput push-pull 11: Alternate functionsoutput Open-drain
29: 28 25: 24 21: 20 17: 16 13: 12 9: 8 5: 4 1: 0	MODEy	rw		Port x mode bits(y = 0…7) These bits are written by software to configure the corresponding I/O port. Refer to Table 18 : Port pin configuration 00: Input mode (reset state) 01: Output mode 10: Reserved 11: Reserved

7.2.2 Port configuration high register(GPIOx_CRH)(x = A..D)

Offset address: 0x04

Reset value:GPIOA_CRH: 0x6444 4484

GPIOB_CRH: 0x4444 8844

GPIOC_CRH: 0x4484 4444

GPIOD_CRH: 0x4444 4444

GPIOE_CRH: 0x4444 4444

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CNF15	MODE15	CNF14	MODE14	CNF13	MODE13	CNF12	MODE12
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CNF11	MODE11	CNF10	MODE10	CNF9	MODE9	CNF8	MODE8
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31: 30 27: 26 23: 22 19: 18 15: 14 11: 10 7: 6 3: 2	CNFy	rw		Port x configuration bits(8…15) These bits are written by software to configure the corresponding I/O port. Refer to Table 18 : Port pin configuration In input mode (MODE = 00): 00: Analog mode 01: Floating input 10: Input with pull-up / pull-down 11: Reserved In output mode (MODE[1: 0] > 00): 00: General-purpose output push-pull 01: General-purpose output Open-drain 10: Alternate functionsoutput push-pull 11: Alternate functionsoutput Open-drain
29: 28 25: 24 21: 20 17: 16 13: 12 9: 8 5: 4 1: 0	MODEy	rw		Port x mode bits(y = 8…15) These bits are written by software to configure the corresponding I/O port. Refer to Table 18 : Port pin configuration 00: Input mode (reset state) 01: Output mode 10: Reserved 11: Reserved

7.2.3 Port input data register(GPIOx_IDR)(x = A..D)

Offset address: 0x08

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
IDR															
r															

Bit	Field	Type	Reset	Description
31: 16	Reserved			Always read as 0.
15: 0	IDRy	r	0xXXXX	Port input data(y = 0..15) These bits are read only and can be accessed in Word (16 bits) mode only. They contain the input value of the corresponding I/O port.

7.2.4 Port output data register(GPIOx_ODR)(x = A..D)

Offset address: 0x0C

Reset value: GPIOA_ODR: 0x0204

GPIOB_ODR: 0x0400

GPIOC_ODR: 0x0000

GPIOD_ODR: 0x0000

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
ODR															
rw															

Bit	Field	Type	Reset	Description
31: 16	Reserved			Always read as 0.
15: 0	ODRy	rw		Port output data(y = 0..15) These bits can be read and written by software and can be accessed in Word (16 bits) mode only. Note: For GPIOx_BSRR (x = A-E), the ODR bits can be individually set and cleared.

7.2.5 Port set/reset register(GPIOx_BSRR)(x = A..D)

Offset address: 0x10

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	Field	Type	Reset	Description
31: 16	BRy	w	0x0000	Port x Reset bit y (y =0-15) These bits are write-only and can be accessed in Word (16 bits) mode only. 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit

Bit	Field	Type	Reset	Description
15: 0	BSy	w	0x0000	Port x Set bit y (y =0-15) These bits are write-only and can be accessed in Word (16 bits) mode only. 0: No action on the corresponding ODRy bit 1: Set the corresponding ODRy bit to '1'

7.2.6 Port bits reset register(GPIOx_BRR)(x = A..D)

Offset address: 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
BR															
w															

Bit	Field	Type	Reset	Description
31: 16	Reserved			Always read as 0
15: 0	BRy	w	0x0000	Port x Reset bit y (y =0-15) These bits are write-only and can be accessed in Word (16 bits) mode only. 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit

7.2.7 Port configuration lock register(GPIOx_LCKR)(x = A..D)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit it is no longer possible to modify the value of the port bit until the next reset. Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH).

Address offset: 0x18

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK															
rw															

Bit	Field	Type	Reset	Description
31: 17	Reserved			Always read as 0
16	LCKK	rw	0x00	Lock key This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence. 0: Port configuration ock key not active 1: Port configuration ock key active, GPIOx_LCKR register is locked until the next reset. LOCK key writing sequence: Write 1->Write 0->Write 1->Read 0->Read 1 The last read is optional but confirms that the lock is active. Note: During the LOCK Key Writing sequence, the value of LCK[15:0] must not change. Any error in the lock sequence will abort the lock.
15: 0	LCKy	rw	0x00	Port x Lock bits y(y = 0…15) These bits are read and written but can only be written when the LCKK bit is 0. 0: Port configuration not locked 1: Port configuration locked

7.3 Multiplexed I/O and Debug Configuration (AFIO)

To optimize the number of peripherals, some of the multiplexing functions can be remapped to other pins. Set the multiplex remapping and debug I/O configuration registers (AFIO_MAPR) to implement pin remapping. At this point, the multiplexing functions are no longer mapped to their original allocation.

7.3.1 Use OSC32_IN/OSC32_OUT as GPIO port PC14/PC15

When the LSE oscillator is turned off, the LSE oscillator pin OSC32_IN/OSC32_OUT can be used as the PC14/PC15 of the GPIO, respectively. The LSE function clock takes precedence over the general I/O function.

Note: When the 1.5V voltage zone is turned off (into standby mode) or the backup area is powered by VBAT (no more VDD supply), the GPIO port function of PC14/PC15 cannot be used.

7.3.2 Use the OSC_IN/OSC_OUT pin as GPIO PD0/PD1

The external oscillator pin OSC_IN/OSC_OUT can be used as the PD0/PD1 of the GPIO. The internal high-speed clock must be turned off first, and then by setting the repeat remapping and debugging I/O configuration register (AFIO_MAPR). Note: The external interrupt/event function is not remapped. On the 36, 48, and 64 pin packages, PD0 and PD1 cannot be used to generate external interrupt events.

7.3.3 CAN Multiplexing Function Remapping

CAN signals can be mapped to port A and port B as shown in the following table.

Table 29. CAN reuse function remapping

Reuse	CAN_REMAP = 00	CAN_REMAP = 10
CAN_RX	PA11	PB8
CAN_TX	PA12	PB9

7.3.4 JTAG/SWD Multiplexing Feature Remapping

The debug interface signals are mapped to the GPIO port as shown in the following table.

Table 30. Debug interface signal

reuse function	GPIO port
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4

In order to use more GPIOs during debugging, the above remapping configuration can be changed by setting the SWJ_CFG bit of the Multiplexing Remap and Debug I/O Configuration Register (AFIO_MAPR). See the table below.

Table 31. Debug port image

SWJ_CFG[2:0]	possible debug port	SWJ I/O pin assignment				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO/ TRACESWO	PB4/ JNTRST
000	Full SWJ (JTAG-DP + SWDP) (Reset Status)	I/O Not Available	I/O Not Available	I/O Not Available	I/O Not Available	I/O Not Available
001	Full SWJ (JTAG-DP + SWDP) but no JNTRST	I/O not available	I/O not available	I/O not available	I/O not available	I/O available
010	Enable SW-DP	I/O Unavailable	I/O Un-available	I/O Available	I/O Available	I/O Available
100	Close JTAG-DP Off SW-DP	I/O Available	I/O Available	I/O Available	I/O Available	I/O Available
Other	Disable					

7.3.5 Timer Multiplexing Function Remapping

Table 32. Timer 3 multiplexing function re-image

reuse function	TIM3_REMAP = 00 (no re-imaging)	TIM3_REMAP = 10 (partial re-imaging)	TIM3_REMAP = 11 (completely re-image)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

Table 33. Timer 2 multiplexing function re-image

Reuse	TIM2_REMAP = 00 (no re-imaging)	TIM2 _REMAP =01 (partial re-imaging)	TIM2_REMAP = 10 (partial re-imaging)	TIM2_REMAP = 11 (full re-imaging)
TIM2_CH1_ETR	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

Table 34. Timer 1 multiplexing function re-image

multiplexed fea-ture image	TIM1_REMAP = 00 (no re-imaging)	TIM1_REMAP = 01 (partial re-imaging)
TIM1_ETR	PA12	
TIM1_CH1	PA8	
TIM1_CH2	PA9	
TIM1_CH3	PA10	
TIM1_CH4	PA11	
TIM1_BKIN	PB12	PA6
TIM1_CH1N	PB13	PA7
TIM1_CH2N	PB14	PB0
TIM1_CH3N	PB15	PB1

7.3.6 UART Reuse Function Remapping

See Multiplexing Remap and Debug I/O Configuration Register (AFIO_MAPR)

Table 35. UART3 re-image

Alternate Fea-tures	UART3_REMAP = 00 (no re-imaging)	UART3 _REMAP = 01 (partial reimage)
UART3_TX	PB10	PC10
UART3_RX	PB11	PC11
UART3_CTS	PB13	
UART3_RTS	PB14	

Table 36. UART1 re-image

Reuse	UART1_REMAP= 0	UART1_REMAP= 1
UART1_TX	PA9	PB6
UART1_RX	PA10	PB7

7.3.7 I2C1 Multiplexing Function Remapping

See Multiplexing Remap and Debug I/O Configuration Register (AFIO_MAPR)

Table 37. I2C1 re-image

Reuse	I2C1_REMAP= 0	I2C1_REMAP= 1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

7.3.8 SPI Multiplexing Function Remapping

See Multiplexing Remap and Debug I/O Configuration Register (AFIO_MAPR)

Table 38. SPI1 re-image

Reuse	SPI1_REMAP= 0	SPI1_REMAP= 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

7.4 AFIO register description

Table 39. AFIO Register Overview

Offset	Acronym	Register Name	Reset	Section
0x04	AFIO_MAPR	Multiplexing Remap and Debug I/O Configuration Registers	0x00000000	section 7.4.1
0x08	AFIO_EXTICR1	External Interrupt Configuration Register 1	0x00000000	section 7.4.2
0x0C	AFIO_EXTICR2	External Interrupt Configuration Register 2	0x00000000	section 7.4.3
0x10	AFIO_EXTICR3	External Interrupt Configuration Register 3	0x00000000	section 7.4.4
0x14	AFIO_EXTICR4	External Interrupt Configuration Register 4	0x00000000	section 7.4.5

7.4.1 Multiplex Remapping and Debug I/O Configuration Register (AFIO_MAPR)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_ REMAP	CAN_ REMAP		Res.	TIM3_ REMAP		TIM2_ REMAP		TIM1_ REMAP		UART3_ REMAP		Res.	UART1_ REMAP	I2C1_ REMAP	SPI1_ REMAP
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 27	Reserved			Always read as 0
26 : 24	SWJ_ CFG	w	0x00	Serial Wire JTAG configuration These bits are readable and writable by software and are used to configure the SWJ and I/O ports that track the alternate function. SWJ (serial line JTAG) supports JTAG or SWD access to the debug port of the CPU. The default state after a system reset is to enable SWJ but no tracking. In this state, the JTAG or SW (serial line) mode can be selected by a specific signal on the JTMS/JTCK pin. 000: Full SWJ (JTAG-DP + SW-DP) : Reset Status 001: Full SWJ (JTAG-DP + SW-DP) but no JNTRST 010: Turn off JTAG-DP and start SW-DP 100: Turn off JTAG-DP and turn off SW-DP Other combinations : Disable
23 : 16	Reserved			always reads as 0
15	PD01_ REMAP	rw	0x00	Port D0/Port D1 mapped to OSCIN/OSC_OUT (Port D0/Port D1 mapping on OSC_IN/OSC_OUT) This bit can be set to '1' or set to '0' by software. It controls the GPIO functional image of PD0 and PD1. When the main oscillator HSE is not applied (the system operates on an internal 8MHz RC oscillator), PD0 and PD1 can be mapped to the OSC_IN and OSC_OUT pins. This feature is only available in 36, 48, and 64-pin packages. 0: Do not remap PD0 and PD1 1: PD0 image to OSC_IN, PD1 image to OSC_OUT

Bit	Field	Type	Reset	Description
14 : 13	CAN_ REMAP	rw	0x00	CAN alternate function remapping These bits can be set to '1' or set to '0' by software to control the re-imaging of the multiplexing functions CAN_RX and CAN_TX. 00: CAN_RX image to PA11, CAN_TX image to PA12 01: Unused combination 10: CAN_RX image to PB8, CAN_TX image to PB9 11: Unused combination
12	Reserved			always reads as 0
11 : 10	TIM3_ REMAP	rw	0x00	Re-imaging of Timer 3 (TIM3 remapping) This bit can be set to '1' or set to '0' by software, and channels 1 4 of TIM3 are controlled to be mapped to the GPIO port. 00: No re-image (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1) 01: Unused combination 10: Partial image (CH1 /PB4, CH2/PB5, CH3/PB0, CH4/PB1) 11: Full image (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9) Note: Re-imaging does not affect TIM3_ETR on PD2.
9 : 8	TIM2_ REMAP	rw	0x00	Re-imaging of Timer 2 (TIM2 remapping) This bit can be set to '1' or set to '0' by software to control the channel 1 4H and external trigger (ETR) of TIM2 to be mapped to the GPIO port. 00: No re-imaging (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3) 01: Partial image (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3) 10: Partial image (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11) 11: Full image (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

Bit	Field	Type	Reset	Description
7 : 6	TIM1_ REMAP	rw	0x00	Re-image of Timer 1 (TTIM1 remapping) This bit can be set to '1' or set to '0' by software. Channels 1 4, 1N to 3N, external trigger (ETR) and disconnection input (BKIN) that control TIM1 are mapped to the GPIO port. 00: No re-image (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BAKIN/PB12, CH1N/PB13, CH2.PB14, CH3N/PB15) 01: Partial image (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BAKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)
5 : 4	UART3_ REMAP	rw	0x00	UART3 re-imaging (UART3 remapping) These bits can be set to '1' or set to '0' by software to control the RX of UART3, the image of the TX multiplexing function on the GPIO port. 00: No re-imaging (TX/PB10, RX/PB11) 01: Partial re-image (TX/PC10, RX/PC11)
3	Reserved		0x00	Always read as 0
2	UART1_ REMAP	rw	0x00	UART1 re-imaging (UART1 remapping) These bits can be set to '1' or set to '0' by software to control the RX of UART1, which is the image of the multiplexed function on the GPIO port. 0: no re-imaging (TX/PA9, RX/PA10) 1: Re-image (TX/PB6, RX/PB7)
1	I2C1_ REMAP	rw	0x00	I2C1 re-image These bits can be set to '1' or set to '0' by software, control the SCL of I2C1, and the SDA alternates the image on the GPIO port. 0: No re-imaging (SCL/PB6, SDA/PB7) 1: Re-image (SCL/PB8, SDA/PB9)
0	SPI1_ REMAP	rw	0x00	SPI1 remap (SPI1 remapping) These bits can be set to '1' or set to '0' by software to control the SCK, MISO, and MOSI multiplexing functions of SPI1 on the GPIO port. 0: No re-imaging (SCK/PA5, MISO/PA6, MOSI/PA7) 1: Re-image (SCK/PB3, MISO/PB4, MOSI/PB5)

7.4.2 External Interrupt Configuration Register 1 (AFIO_EXTICR1)

Offset address: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3				EXTI2				EXTI1				EXTI0			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31:16	Reserved			always reads as 0
15:0	EXTI3, EXTI2, EXTI1, EXTI0	rw	0x00	EXTIx configuration (x = 0...3) (EXTI x configuration) These bits are available for software to read and write. Input source for selecting EXTIx external interrupts. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin (EXTI0-EXTI2)

7.4.3 External Interrupt Configuration Register 2 (AFIO_EXTICR2)

Offset address: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7				EXTI6				EXTI5				EXTI4			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31:16	Reserved			always reads as 0
15:0	EXTI7, EXTI6, EXTI5, EXTI4	rw	0x00	EXTIx configuration (x = 7...4) (EXTI x configuration) These bits are available for software to read and write. Input source for selecting EXTIx external interrupts. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin

7.4.4 External Interrupt Configuration Register 3 (AFIO_EXTICR3)

Offset address: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11				EXTI10				EXTI9				EXTI8			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31:16	Reserved			always reads as 0
15:0	EXTI11, EXTI10, EXTI9, EXTI8	rw	0x00	EXTIx configuration (x = 11...8) (EXTI x configuration) These bits are available for software to read and write. Input source for selecting EXTIx external interrupts. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin

7.4.5 External Interrupt Configuration Register 1 (AFIO_EXTICR4)

Offset address: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15				EXTI14				EXTI13				EXTI12			
rw				rw				rw				rw			

Bit	Field	Type	Reset	Description
31:16	Reserved			always reads as 0
15:0	EXTI15, EXTI14, EXTI13, EXTI12	rw	0x00	EXTIx configuration (x = 15...12) (EXTI x configuration) These bits are available for software to read and write. Input source for selecting EXTIx external interrupts. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin

8 Interrupts and events (EXTI)

Interrupts and events (EXTI)

8.1 Nested Vectored Interrupt Controller

Features

- Interrupts can be masked (except NMI)
- 8 programmable priority levels (3 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to CPU programming manual.

8.1.1 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (HCLK/8, HCLK = 72 MHz).

8.1.2 Interrupt and exception vectors

The following tables are the vector tables for the product series.

Table 40. Vectors for the Product Series

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	Fixed	Reset	Reset	0x0000_0004
	-2	Fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	Fixed	HardFault	All class of fault	0x0000_000C
	0	Settable	MemManage	Memory management	0x0000_0010
	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014

Position	Priority	Type of priority	Acronym	Description	Address
	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C ~ 0x0000_002B
	3	Settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	Settable	DebugMonitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	Settable	PendSV	Pendable request for system service	0x0000_0038
	6	Settable	SysTick	System tick timer	0x0000_003C
0	7	Settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	Settable	PVD	PVD through EXTI 16 Line detection interrupt	0x0000_0044
2	9	Settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	Settable	RTC	RTC global interrupt	0x0000_004C
4	11	Settable	FLASH	FLASH global interrupt	0x0000_0050
5	12	Settable	RCC	RCC interrupt	0x0000_0054
6	13	Settable	EXTI0	EXTI line 0 interrupt	0x0000_0058
7	14	Settable	EXTI1	EXTI line 1 interrupt	0x0000_005C
8	15	Settable	EXTI2	EXTI line 2 interrupt	0x0000_0060
9	16	Settable	EXTI3	EXTI line 3 interrupt	0x0000_0064
10	17	Settable	EXTI4	EXTI line 4 interrupt	0x0000_0068
11	18	Settable	DMA1 Channel 1	DMA1 Channel 1 global interrupt	0x0000_006C
12	19	Settable	DMA1 Channel 2	DMA1 Channel 2 global interrupt	0x0000_0070
13	20	Settable	DMA1 Channel 3	DMA1 Channel 3 global interrupt	0x0000_0074
14	21	Settable	DMA1 Channel 4	DMA1 Channel 4 global interrupt	0x0000_0078
15	22	Settable	DMA1 Channel 5	DMA1 Channel 5 global interrupt	0x0000_007C
16	23	Settable	DMA1 Channel 6	DMA1 Channel 6 global interrupt	0x0000_0080
17	24	Settable	DMA1 Channel 7	DMA1 Channel 7 global interrupt	0x0000_0084
18	25	Settable	ADC1_2	ADC1 和 ADC2 的 global interrupt	0x0000_0088
19	26	Settable	USB	USB interrupt	0x0000_008C
20	-	-	-	Reserved	0x0000_0090
21	28	Settable	CAN_RX1	CAN RX1 interrupt	0x0000_0094
22	-	-	-	Reserved	0x0000_0098
23	30	Settable	EXTI9_5	EXTI line [9:5] interrupt	0x0000_009C
24	31	Settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0

Position	Priority	Type of priority	Acronym	Description	Address
25	32	Settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	Settable	TIM1_TRG_COM	TIM1 Trigger and Communication interrupt	0x0000_00A8
27	34	Settable	TIM1_CC	TIM1 Capture and Compare interrupt	0x0000_00AC
28	35	Settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	Settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	Settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	Settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	-	-	-	Reserved	0x0000_00C0
33	40	Settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	-	-	-	Reserved	0x0000_00C8
35	42	Settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	Settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	Settable	UART1	UART1 global interrupt	0x0000_00D4
38	45	Settable	UART2	UART2 global interrupt	0x0000_00D8
39	46	Settable	UART3	UART3 global interrupt	0x0000_00DC
40	47	Settable	EXTI15_10	EXTI line [15:10] interrupt	0x0000_00E0
41	48	Settable	RTC Alarm	RTC Alarm Interrupt combined with EXTI 17	0x0000_00E4
42	49	Settable	USB Wakeup	USB Wakeup Interrupt combined with EXTI 18	0x0000_00E8
43	-	-	-	Reserved	0x0000_00EC
44	-	-	-	Reserved	0x0000_00F0
45	-	-	-	Reserved	0x0000_00F4

8.2 External interrupt/event controller (EXTI)

The external interrupt/event controller, consisting of edge detectors, is used for managing external and internal asynchronous events/interrupts, and for corresponding event requests sent to the CPU/ interrupt controller, and a wake-up request transmitted to the power manager.

The event/interrupt requests can be generated by the edge detector. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also be masked independently. A pending register maintains the status line of the interrupt requests.

8.2.1 Main features

The main features of EXTI controller are as follows:

- Independent trigger and mask on each interrupt/event line
- Dedicated status bit for each interrupt line
- Generation of software event/interrupt requests
- Detection of external signal with pulse width lower than APB2 clock period. Refer to the electrical characteristics section of the datasheet for details on this parameter.

8.2.2 Block diagram

Figure 19. External Interrupt/Event Controller Block Diagram

8.2.3 Wakeup event management

The device(WFE) is able to handle external or internal events (WFE). The wakeup event can be generated either by:

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

To use an external line as a wakeup event, refer to Section "Functional Description".

8.2.4 Functional description

To generate an interrupt, an interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

To generate an event, an event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

Hardware interrupt selection

To configure the several lines as interrupt sources, use the following procedure:

- Configure the mask bits of the Interrupt lines (EXTI_IMR)
- Configure the Trigger Selection bits of the Interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the External Interrupt Controller (EXTI) so that an interrupt coming from one of the lines can be correctly acknowledged.

Hardware event selection

To configure the several lines as event sources, use the following procedure:

- Configure the mask bits of the Event lines (EXTI_EMR)
- Configure the Trigger Selection bits of the Event lines (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

The several lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the Interrupt/Event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit of the software interrupt register (EXTI_SWIER)

8.2.5 External interrupt/event line mapping

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

Figure 20. External Interrupt/Event GPIO Mapping

Note: GPIO corresponding to the above figure may differ due to actual chip package, and the actual package shall prevail.

The other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to RTC Alarm event
- EXTI line 18 is connected to USB Wakeup event

8.3 EXTI register description

Table 41. EXTI Register Overview

Offset	Acronym	Register Name	Reset	Section
0x00	EXTI_IMR	Interrupt Mask Register	0x00000000	section 8.3.1
0x04	EXTI_EMR	Event Mask Register	0x00000000	section 8.3.2
0x08	EXTI_RTSR	Rising Trigger Selection Register	0x00000000	section 8.3.3
0x0C	EXTI_FTSR	Falling Trigger Selection Register	0x00000000	section 8.3.4
0x10	EXTI_SWIER	Software Interrupt Event Register	0x00000000	section 8.3.5
0x14	EXTI_PR	Pending Register	0x00000000	section 8.3.6

8.3.1 Interrupt Mask Register(EXTI_IMR)

Offset address: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													IMR18	IMR17	IMR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.
18 : 0	IMRx	rw	0x00	Interrupt Mask on line x 1 = Interrupt request from Line x is not masked 0 = Interrupt request from Line x is masked

8.3.2 Event Mask Register(EXTI_EMR)

Offset address: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													EMR18	EMR17	EMR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR15	EMR14	EMR13	EMR12	EMR11	EMR10	EMR9	EMR8	EMR7	EMR6	EMR5	EMR4	EMR3	EMR2	EMR1	EMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.
18 : 0	EMRx	rw	0x00	Event Mask on line x 1 = Event request from Line x is not masked 0 = Event request from Line x is masked

8.3.3 Rising Trigger Selection Register(EXTI_RTSR)

Offset address: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.
18 : 0	TRx	rw	0x00	Rising trigger event configuration bit of line x 1 = Rising trigger enabled (for Event and Interrupt) for input line 0 = Rising trigger disabled (for Event and Interrupt) for input line

8.3.4 Falling Trigger Selection Register(EXTI_FTSR)

Offset address: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
18 : 0	TRx	rw	0x00	Falling trigger event configuration bit of line x 1 = Falling trigger enabled (for Event and Interrupt) for input line 0 = Falling trigger disabled (for Event and Interrupt) for input line

8.3.5 Software Interrupt Event Register(EXTI_SWIER)

Offset address: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													SWIER18	SWIER17	SWIER16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER15	SWIER14	SWIER13	SWIER12	SWIER11	SWIER10	SWIER9	SWIER8	SWIER7	SWIER6	SWIER5	SWIER4	SWIER3	SWIER2	SWIER1	SWIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.
18 : 0	SWIERx	rw	0x00	Software interrupt on line x If the interrupt is enabled on this line in EXTI_INTMASK and EXTI_EVNTMASK, write '1' to this bit when it is set to '0', so as to set the corresponding pending bit in EXTI_PR, and to generate an interrupt request. Note: This bit is cleared by clearing the corresponding bit of EXTI_PEND (by writing a '1' into the bit)

8.3.6 Pending register(EXTI_PR)

Offset address: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													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	Field	Type	Reset	Description
31 : 19	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
18 : 0	PRx	rc_w1	0x00	Pending bit 1 = selected trigger request occurred 0 = No trigger request occurred This bit is set to '1' when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a '1' into the bit or by changing the polarity of edge detection.

9 Direct memory access controller (DMA)

Direct memory access controller (DMA)

9.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA controller has 7 channels, each dedicated to managing memory access requests from several peripherals.

9.2 DMA main features

- 7 independently configurable channels.
- Each channel is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from 7 channels are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 0 has priority over request 1, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management.
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically or together in a single interrupt request for each channel
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral transfers
- Access to Flash, SRAM, SRAM peripherals, APB1, APB2 and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65536.

The block diagram is shown in the following figure:

Figure 21. DMA Block Diagram

9.3 Functional description

The DMA controller performs direct memory transfer by sharing the system bus with the CPU. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (RAM or peripheral). The bus arbiter implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

9.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, and the DMA Controller release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

1. The load operation is performed from the memory unit specified by the DMA_CPARx register or the DMA_CMARx register.

2. Store data in the DMA_CPARx register or the memory location specified by the DMA_CMARx register.
3. The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

9.3.2 DMA arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences. The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

9.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transfer depending on the PINC and MINC bits in the DMA_CCRx register. If the incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. If the channel is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero).

Channel configuration procedure

The following sequence should be followed to configure a DMA channel x (where x is the channel number):

1. Set the peripheral register address in the DMA_CPARx register. The data will be moved from/ to this address to/ from the memory after the transfer request of peripheral data.
2. Set the memory address in the DMA_CMARx register. The data will be written to or

read from this memory after the transfer request of peripheral data.

3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each data transmission, this value will be decremented.
4. Configure the channel priority using the PL [1:0] bits in the DMA_CCRx register.
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register.
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register. As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set to '1' and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode. If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

9.3.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in the following table.

Table 43. Programmable Data Width and Endian Behavior (When Bits PINC = MINC = 1)

Source port width	Destination port width	Number of data items to be transferred (NDT)	Source content (address / data)	Transfer operations	Destination content (address / data)
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: read B0[7: 0] @ 0x0, write B0[7: 0] @ 0x0 2: read B1[7: 0] @ 0x1, write B1[7: 0] @ 0x1 3: read B2[7: 0] @ 0x2, write B2[7: 0] @ 0x2 4: read B3[7: 0] @ 0x3, write B3[7: 0] @ 0x3	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: read B0[7: 0] @ 0x0, write 00B0[15: 0] @ 0x0 2: read B1[7: 0] @ 0x1, write 00B1[15: 0] @ 0x2 3: read B2[7: 0] @ 0x2, write 00B2[15: 0] @ 0x4 4: read B3[7: 0] @ 0x3, write 00B3[15: 0] @ 0x6	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: read B0[7: 0] @ 0x0, write 000000B0[31: 0] @ 0x0 2: read B1[7: 0] @ 0x1, write 000000B1[31: 0] @ 0x4 3: read B2[7: 0] @ 0x2, write 000000B2[31: 0] @ 0x8 4: read B3[7: 0] @ 0x3, write 000000B3[31: 0] @ 0xC	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: read B1B0[15: 0] @ 0x0, write B0[7: 0] @ 0x0 2: read B3B2[15: 0] @ 0x2, write B2[7: 0] @ 0x1 3: read B5B4[15: 0] @ 0x4, write B4[7: 0] @ 0x2 4: read B7B6[15: 0] @ 0x6, write B6[7: 0] @ 0x3	0x0/B0 0x1/B2 0x2/B4 0x3/B6

Source port width	Destination port width	Number of data items to be transferred (NDT)	Source content (address / data)	Transfer operations	Destination content (address / data)
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: read B1B0[15: 0] @ 0x0, write B1B0[15: 0] @ 0x0 2: read B3B2[15: 0] @ 0x2, write B3B2[15: 0] @ 0x2 3: read B5B4[15: 0] @ 0x4, write B5B4[15: 0] @ 0x4 4: read B7B6[15: 0] @ 0x6, write B7B6[15: 0] @ 0x6	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: read B1B0[15: 0] @ 0x0, write 0000B1B0[31: 0] @ 0x0 2: read B3B2[15: 0] @ 0x2, write 0000B3B2[31: 0] @ 0x4 3: read B5B4[15: 0] @ 0x4, write 0000B5B4[31: 0] @ 0x8 4: read B7B6[15: 0] @ 0x6, write 0000B7B6[31: 0] @ 0xC	0x0/0000B1B0 0x4/0000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: read B3B2B1B0[31: 0] @ 0x0, write B0[7: 0] @ 0x0 2: read B7B6B5B4[31: 0] @ 0x4, write B4[7: 0] @ 0x1 3: read BBBAB9B8[31: 0] @ 0x8, write B8[7: 0] @ 0x2 4: read BFBEBDBC[31: 0] @ 0xC, write BC[7: 0] @ 0x3	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: read B3B2B1B0[31: 0] @ 0x0, write B1B0[15: 0] @ 0x0 2: read B7B6B5B4[31: 0] @ 0x4, write B5B4[15: 0] @ 0x2 3: read BBBAB9B8[31: 0] @ 0x8, write B8B8[15: 0] @ 0x4 4: read BFBEBDBC[31: 0] @ 0xC, write BDBC[15: 0] @ 0x6	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC

Source port width	Destination port width	Number of data items to be transferred (NDT)	Source content (address / data)	Transfer operations	Destination content (address / data)
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: read B3B2B1B0[31: 0] @ 0x0, write B3B2B1B0[31: 0] @ 0x0 2: read B7B6B5B4[31: 0] @ 0x4, write B7B6B5B4[31: 0] @ 0x4 3: read BBBAB9B8[31: 0] @ 0x8, write BBBAB8B8[31: 0] @ 0x8 4: read BFBEBDBC[31: 0] @ 0xC, write BFBEBDBC[31: 0] @ 0xC	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte ‘0xABABABAB’.

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0.
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0.

For instance, to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), the memory source size (MSIZE) must be configured to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

9.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx). The channel’s transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer

error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

9.3.6 Interrupts

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 44. DMA Interrupt Requests

Interrupt event	Event flag	Enable Control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

9.3.7 DMA request mapping

DMA controller

The 7 requests from the peripherals TIMx、ADCx、USB、SPIx、I2Cx and UARTx are simply logically ORed before entering the DMA1, this means that only one request must be enabled at a time. Refer to the following figure.

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Figure 22. Peripheral DMA Request Mapping

Table 45. Summary of DMA Requests for Each Channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC	ADC1	ADC2					
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
UART		UART3_TX	UART3_RX	UART1_TX	UART1_RX	UART2_RX	UART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
USB				USB_EP1	USB_EP2		

9.4 DMA register description

Table 46. Summary of DMA Registers

Offset	Acronym	Register Name	Reset	Section
0x00	DMA_ISR	DMA interrupt status register	0x00000000	section 9.4.1
0x04	DMA_IFCR	DMA interrupt flag clear register	0x00000000	section 9.4.2
0x08 + 20 × (n - 1)	DMA_CCRx	DMA channel x configuration register	0x00000000	section 9.4.3
0x0C + 20 × (n - 1)	DMA_CNDTRx	DMA channel x number of data register	0x00000000	section 9.4.4
0x10 + 20 × (n - 1)	DMA_CPARx	DMA channel x peripheral address register	0x00000000	section 9.4.5
0x14 + 20 × (n - 1)	DMA_CMARx	DMA channel x memory address register	0x00000000	section 9.4.6

9.4.1 DMA interrupt status register(DMA_ISR)

Offset address: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 28	Reserved			Reserved, always read as 0.
27, 23, 19, 15, 11, 7, 3	TEIFx	r	0x00	Channel x transfer error flag(x = 1 ... 7)) This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. 0: No transfer error (TE) on channel x 1: A transfer error (TE) occurred on channel x
26, 22, 18, 14, 10, 6, 2	HTIFx	r	0x00	Channel x half transfer flag(x = 1 ... 7) This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. 0: No half transfer (HT) event on channel x 1: A half transfer (HT) event occurred on channel x
25, 21, 17, 13, 9, 5, 1	TCIFx	r	0x00	Channel x transfer complete flag(x = 1 ... 7) This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. 0: No transfer complete (TC) event on channel x 1: A transfer complete (TC) event occurred on channel x
24, 20, 16, 12, 8, 4, 0	GIFx	r	0x00	Channel x global interrupt flag(x = 1 ... 7) This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. 0: No TE, HT or TC event on channel x 1: A TE, HT or TC event occurred on channel x

9.4.2 DMA interrupt flag clear register(DMA_IFCR)

Offset address: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTE IF7	CHT IF7	CTC IF7	CG IF7	CTE IF6	CHT IF6	CTC IF6	CG IF6	CTE IF5	CHT IF5	CTC IF5	CG IF5
				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
CTE IF4	CHT IF4	CTC IF4	CG IF4	CTE IF3	CHT IF3	CTC IF3	CG IF3	CTE IF2	CHT IF2	CTC IF2	CG IF2	CTE IF1	CHT IF1	CTC IF1	CG IF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 28	Reserved			Reserved, always read as 0.
27, 23, 19, 15, 11, 7, 3	CTEIFx	w	0x00	Channel x transfer error clear(x = 1 ... 7) This bit is set and cleared by software. 0: No effect 1: Clear the corresponding TEIF flag in the DMA_ISR register
26, 22, 18, 14, 10, 6, 2	CHTIFx	w	0x00	Channel x half transfer clear(x = 1 ... 7) This bit is set and cleared by software. 0: No effect 1: Clear the corresponding HTIF flag in the DMA_ISR register
25, 21, 17, 13, 9, 5, 1	CTCIFx	w	0x00	Channel x transfer complete clear(x = 1 ... 7) This bit is set and cleared by software. 0: No effect 1: Clear the corresponding TCIF flag in the DMA_ISR register
24, 20, 16, 12, 8, 4, 0	CGIFx	w	0x00	Channel x global interrupt clear(x = 1 ... 7) This bit is set and cleared by software. 0: No effect 1: Clear the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

9.4.3 DMA channel x configuration register(DMA_CCRx) (x = 1···7)

Offset address: 0x08 + 20 x (channel number - 1)

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
Res.	MEM2MEM	PL		MSIZE		PSIZE		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 15	Reserved			Reserved, always read as 0.
14	MEM2MEM	rw	0x00	Memory to memory mode This bit is set and cleared by software. 0: Memory to memory mode disabled 1: Memory to memory mode enabled

Bit	Field	Type	Reset	Description
13 : 12	PL	rw	0x00	Channel priority level This bit is set and cleared by software. 00: Low 01: Medium 10: High 11: Very high
11 : 10	MSIZE	rw	0x00	Memory size This bit is set and cleared by software. 00: 8-bits 01: 16-bits 10: 32-bits 11: Reserved
9 : 8	PSIZE	rw	0x00	Peripheral size This bit is set and cleared by software. 00: 8-bits 01: 16-bits 10: 32-bits 11: Reserved
7	MINC	rw	0x00	Memory increment mode This bit is set and cleared by software. 0: Memory increment mode disabled 1: Memory increment mode enabled
6	PINC	rw	0x00	Peripheral increment mode This bit is set and cleared by software. 0: Peripheral increment mode disabled 1: Peripheral increment mode enabled
5	CIRC	rw	0x00	Circular mode This bit is set and cleared by software. 0: Circular mode disabled 1: Circular mode enabled
4	DIR	rw	0x00	Data transfer direction This bit is set and cleared by software. 0: Read from peripheral 1: Read from memory
3	TEIE	rw	0x00	Transfer error interrupt enable This bit is set and cleared by software. 0: TE interrupt disabled 1: TE interrupt enabled
2	HTIE	rw	0x00	Half transfer interrupt enable This bit is set and cleared by software. 0: HT interrupt disabled 1: HT interrupt enabled

Bit	Field	Type	Reset	Description
1	TCIE	rw	0x00	Transfer complete interrupt enable This bit is set and cleared by software. 0: TC interrupt disabled 1: TC interrupt enabled
0	EN	rw	0x00	Channel enable This bit is set and cleared by software. 0: Channel disabled 1: Channel enabled

9.4.4 DMA channel x number of data register(DMA_CNDTRx) (x = 1···7)

Offset address: 0x0C + 20 x (channel number - 1)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT															
rw															

Bit	Field	Type	Reset	Description
31 : 16	Reserved			Reserved, always read as 0.
15 : 0	NDT	rw	0x0000	Number of data to transfer Number of data to be transferred (0 to 65535). This register can only be written when the channel is disabled (EN bit of DMA_CCRx is 0). Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer. Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode. If this register is zero, no transaction can be served no matter whether the channel is enabled or not.

9.4.5 DMA channel x peripheral address register(DMA_CPARx) (x = 1···7)

Offset address: 0x10 + 20 x (channel number - 1)

Reset value: 0x0000 0000

This register must not be written when the channel (EN bit of DMA_CCRx is 0) is enabled.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA															
rw															

Bit	Field	Type	Reset	Description
31 : 0	PA	rw	0x0000 0000	Peripheral address Base address of the peripheral data register is used as a source or target of data transfer. When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half word address. When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

9.4.6 DMA channel x memory address register(DMA_CMARx) (x = 1···7)

Offset address: 0x14 + 20 x (channel number - 1)

Reset value: 0x0000 0000

This register must not be written when the channel (EN bit of DMA_CCRx is 0) is enabled.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA															
rw															

Bit	Field	Type	Reset	Description
31 : 0	MA	rw	0x0000 0000	Memory address The memory address serves as the source or destination of data transmission. When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address. When MSIZE is 10 (32-bit), MA [1:0] are ignored. Access is automatically aligned to a word address.

10 Analog-to-digital converter (ADC)

Analog-to-digital converter (ADC)

10.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter.

A/D conversion of the various channels can be performed in single, continuous, scan mode, and you can choose automatic channel scanning.

Its start-up mode includes the software setting, external pin triggering and activated by timers.

The window comparator (analog watchdog) allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.

The ADC input clock is generated from the PCLK2 clock divided by a prescaler and it must not exceed 15 MHz.

10.2 ADC main features

- 12-bit-resolution SAR ADC, up to 16(8 × 2) external input channels and 2 internal input channels(Used in ADC1 to test temperature, ADC2 is used to test internal reference voltage).
- The internal signal source of ADC1 is connected to the internal temperature sensor, and the internal signal source of ADC2 is connected to the internal 1.2V reference voltage.
- Up to 1 Msps conversion rate
- Supporting multiple operation modes:
  - Single conversion mode: one A/D conversion in specified channel
  - Single-cycle scanning mode: one A/D conversion cycle (from low-number channel to high-number channel) completed in all designated channels
  - Continuous scan mode: A/D converter continuously performs single-cycle scanning until the converter is disabled by software
- Channel sampling time and resolution can be configured by software
- Support DMA transfer
- Conditions of A/D conversion:
  - By software
  - External triggering
  - Timer matching
- In terms of analog watchdog, the conversion result can be compared with the specified value; the user can set whether to generate an interrupt request or not when the conversion value matches the set value.

10.3 ADC functional description

The ADC block diagram is as shown below.

Figure 23. ADC block diagram

10.3.1 ADC on-off control

The ADC can be powered-on by setting the ADEN bit in the ADCFG register. When the ADEN bit is set for the first time, it wakes up the ADC from Power Down mode.

Conversion starts when ADST bit of ADCR register is set after ADC power-up time.

The conversion can be stopped by clearing the ADST bit, and the ADC put in power down mode by resetting the ADEN bit.

10.3.2 Channel selection

There are several external input channels, internal temperature sensor channel and internal 1.2 V reference voltage channel. Among them, each external input channel has independent enabling bit, which can be configured by setting bits concerned of the ADCHS register.

10.4 ADC operating mode

10.4.1 Single conversion mode

In the single conversion mode, the A/D conversion is only performed once on the corresponding channel, and the specific process is as follows:

- The A/D conversion is activated by software, external trigger input, and the ADST bit of the timer overflow setting ADCR register.
- After the A/D conversion, the data value concerned will be saved in the ADDATA and ADDRn data registers of A/D converter.
- ADIF bit of status register ADSTA is set to '1' after the A/D conversion. If ADIE bit of control register ADCR is set to '1' at this time, an AD conversion end interrupt request will be generated.
- During the A/D conversion, the ADST bit remains 1. After that, the ADST bit is cleared automatically and the idle mode is enabled.

Note: If the software, in the single conversion mode, enables more than one channel, the channel with the smallest number will be converted and other channels will be ignored.

Figure 24. Timing Diagram of Single Conversion Mode

10.4.2 Single-cycle scan mode

- Starting from the software or external trigger setting bit ADST, the A/D conversion is enabled from the channel with the minimum number to that with the maximum number.

- After A/D conversion in each channel, the A/D conversion values will be loaded into the data registers in the corresponding channels in an orderly manner, and the ADIF conversion end flag will be set. If the conversion end interrupt flag is set, an interrupt request will be generated after the conversion in all channels.
- After the conversion, ADST bit is cleared automatically, so that A/D converter enters idle state.

Figure 25. Timing Diagram of Enabled Channel During Conversion in Single-cycle Scan Mode

10.4.3 Continuous scan mode

In the continuous scan mode, A/D conversion is performed sequentially in the channel where the CHENn bit in the ADCHS register is enabled. The operation steps are as follows:

- Starting from the software or external trigger setting bit ADST, the A/D conversion is enabled from the channel with the minimum number to that with the maximum number.
- After A/D conversion in all channels, the A/D conversion values will be loaded into the data registers concerned in an orderly manner, and the ADIF conversion end flag will be set. If the conversion end interrupt flag is set, an interrupt request will be generated after the conversion in all channels.
- As long as the ADST bit remains 1, the A/D conversion continues. When ADST bit is cleared and A/D conversion is completed, A/D converter enters the idle mode. When ADST is cleared, the current A/D conversion will be completed.

Figure 26. Timing Diagram of Enabled Channel During Conversion in Continuous Scan Mode

10.4.4 DMA request

In the single-cycle scan and continuous scan modes, the value of channel conversion is saved in the data registers (ADDRn) in respective channel, and the result of the latest conversion is also stored in the ADDATA register. During DMA transmission, you can choose to transfer data in a specific channel or transfer the results of all scanning channels.

10.5 Data alignment

ALIGN bit in the ADCR register selects the alignment of data stored after conversion. Data can be left or right aligned as shown in the following figure .

Data is right-justified

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Data is left-justified

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

730272

Figure 27. Data Alignment Modes

10.5.1 Programmable resolution

The effective ADC conversion bits can be changed by modifying RSLT CTL [2: 0] bits in ADC_CFG register, to improve the data conversion rate. The effective data bits are

aligned at the high bits of 12-bit data.

10.5.2 Programmable sample time

ADCLK, the ADC clock, is generated by dividing PCLK2, and its division factor can be determined by setting the AD-CPRE bit in ADCFG bit, namely, the PCLK2 / (N + 1) / 2 is used as the ADC clock.

The ADC resolution is n (n=8, 9, 10, 11 and 12), the sampling period in each channel is m, and the number of sampling periods can be modified through the SAMCTL bits in the ADC_CFG registers.

The sampling frequency and sampling time are calculated as follows:

$F_{sample} = F_{ADCLK} / (m + n + 1.5)$.

Example:

With an ADCCLK = 15MHz, the resolution of 12 bits and a sampling time of 1.5 cycles

$F_{sample} = F_{ADCLK} / 15$.

$T_{CONV} = 1.5 + 13.5 = 15$ cycles = 1μs

10.6 Conversion on external trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the TRGEN bit of ADCR register is set, then external events are able to trigger a conversion. By setting the TRGSEL bits, the external trigger sources can be selected.

For the selection of specific external trigger sources, please refer to the description of relevant bits in AD control register.

The sampling is initiated after the generation of trigger signal and N PCLK2 cycles. In the trigger scan mode, only the sampling of the first channel is delayed, and the rest channels is sampled immediately after the end of the previous operation.

10.7 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (T_A) of the device.

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor output voltage into a digital value. When not in use, this sensor can be disabled separately by setting relevant bits of the register.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation.

The internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

The temperature is calculated as follows:

$T(°C) = (V_{SENSE} - V_{25}) / Avg_Slope + 25$

V_{25}: V_{SENSE} value for 25°C

V_{SENSE}: the current output voltage of temperature sensor

V_{SENSE} = Value * V_{dd} / 4096 (Value is the conversion result of ADC)

Avg_Slope: Average Slope for curve between Temperature vs. V_{SENSE}(given in mV/°C or μV/°C)

Refer to the temperature sensor section for the actual values of V_{25} and Avg_Slope.

10.8 Internal reference voltage

The input channel of ADC is loaded with an internal reference voltage (1.2V), converting the reference voltage output of 1.2V into a digital value.

The internal reference voltage has a separate enable bit, which can be enabled or disabled by setting the corresponding bit in the register.

10.9 Monitoring of AD conversion results in window comparator mode

The upper limit and lower limit compare registers are enabled in the comparison mode, and the CMPCH bit can be set by software, to select the monitoring channel.

If CPMHDATA is ≥ CPMLDATA, and the comparison result is greater than or equal to the specified value of CMPHDATA in the ADCMPR register or less than the specified value of CMPLDATA, the ADWIF bit of the status register ADSTA is set to 1.

If CPMHDATA is < CPMLDATA and the comparison result is equal to the specified value of CMPHDATA or between the two specified value, the ADWIF bit of the status register ADSTA is set to 1. An interrupt request will be generated if ADWIE bit of the control register ADCR is set. An interrupt request will be generated if ADWIE bit of the control register ADCR is set.

10.10 ADC register description

Table 47. Summary of ADC Registers

Offset	Acronym	Register Name	Reset	Section
0x00	ADC_ADDATA	A/D data register	0x00000000	section 10.10.1
0x04	ADC_ADCFG	A/D configuration register	0x00000000	section 10.10.2
0x08	ADC_ADCR	A/D control register	0x00000000	section 10.10.3
0x0C	ADC_ADCHS	A/D channel select register	0x00000000	section 10.10.4
0x10	ADC_ADCMPR	A/D window compare register	0x00000000	section 10.10.5
0x14	ADC_ADSTA	A/D status register	0x00000000	section 10.10.6
0x18 ~ 0x38	ADC_ADDR0 ~ 8	A/D data register	0x00000000	section 10.10.7

10.10.1 A/D data register(ADC_ADDATA)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										VAILD	OVER RUN	CHANNELSEL			
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 22	Reserved			Reserved, always read as 0.
21	VALID	r	0x00	Valid flag (read-only) 1 = DATA[11: 0] bits are valid 0 = DATA[11: 0] bits are invalid After the conversion in the corresponding analog channel, this bit is set and this bit is cleared by hardware after the ADDATA register is read.
20	OVERRUN	r	0x00	Overrun flag (read-only) 1 = DATA[11: 0] data overwritten 0 = The last conversion result of DATA[11: 0] data Before the new conversion result is loaded into the register, if the data of DATA[11: 0] is not read, OVERRUN will be set to 1. This bit is cleared by hardware after the ADDATA register is read.
19 : 16	CHANNELSEL	r	0x00	Channel selection (the 4 bits show the channel corresponding to the current data) 0000 = convert data for Channel 0 0001 = convert data for Channel 1 0010 = convert data for Channel 2 0011 = convert data for Channel 3 0100 = convert data for Channel 4 0101 = convert data for Channel 5 0110 = convert data for Channel 6 0111 = convert data for Channel 7 1000 = ADC1 is the conversion data of the temperature sensor, ADC2 is the conversion data of the internal reference voltage) Others: invalid
15 : 0	DATA	r	0x00	Transfer data (12-bit A/D conversion result) Left alignment or right alignment, depending on specific settings.

10.10.2 A/D configuration register(ADC_ADCFG)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			SAMCTL			RSLTCTL			ADCPRE			Res.	TVSEN	ADWEN	ADEN
			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 13	Reserved			Reserved, always read as 0.
12 : 10	SAMCTL	rw	0x00	Channel x Sample time selection These bits are used to independently select the sampling time for each channel. The channel select bit must remain unchanged during the sampling period. 000: 1.5 cycles 100: 41.5 cycles 001: 7.5 cycles 101: 55.5 cycles 010: 13.5 cycles 110: 71.5 cycles 011: 28.5 cycles 111: 239.5 cycles
9 : 7	RSLTCTL	rw	0x00	Resolution (select ADCx conversion data resolution) 000: valid in 12 bits 001: valid in 11 bits 010: valid in 10 bits 011: valid in 9 bits 100: valid in 8 bits
6 : 4	ADCPRE	rw	0x00	ADC prescaler Set to '1' or cleared by software to determine the ADC clock frequency. n: PCLK2 is divided by 2*(n + 1) and used as ADC clock
3	Reserved			Reserved, always read as 0.
2	TVSEN	rw	0x00	Temperature sensor and internal reference voltage enable control bits 1 = Enabled 0 = Disabled Note: The temperature sensor is enabled in ADC1 and the internal reference voltage is available in ADC2.
1	ADWEN	rw	0x00	ADC window comparison enable 1 = A/D window comparator enabled 0 = A/D window comparator disabled

Bit	Field	Type	Reset	Description
0	ADEN	rw	0x00	ADC enable 1 = Enabled 0 = Disabled

10.10.3 A/D control register(ADC_ADCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPCH				ALIGN	ADMD		ADST	SYNCEN	TRGSEL			DMAEN	TRGEN	ADWIE	ADIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 16	Reserved			Reserved, always read as 0.
15 : 12	CMPCH	rw	0x00	Window comparison channel selection 0000 = Conversion result of select comparison channel 0 0001 = Conversion result of select comparison channel 1 0010 = Conversion result of select comparison channel 2 0011 = Conversion result of select comparison channel 3 0100 = Conversion result of select comparison channel 4 0101 = Conversion result of select comparison channel 5 0110 = Conversion result of select comparison channel 6 0111 = Conversion result of select comparison channel 7 1000 = Select temperature sensor or internal reference voltage Other: invalid
11	ALIGN	rw	0x00	Data alignment 0: Right alignment 1: Left alignment
10 : 9	ADMD	rw	0x00	ADC mode 00: Single conversion 01: Single-cycle scan 10: Continuous scan When changing the conversion mode, disable the ADST bit by the software.

Bit	Field	Type	Reset	Description
8	ADST	rw	0x00	ADC start 1 = Conversion starts 0 = Conversion ends or it enables idle mode ADST bit can be set in the following two ways: In single mode or single-cycle mode, ADST bit will be automatically cleared by hardware after the conversion. In the continuous scan mode, the A/D conversion continues until the software writes' 0' to this bit or the system resets.
7	SYNCEN	rw	0x00	Two ADC synchronous conversion enable 0: Two ADCs convert separately 1: As a slave ADC, synchronous conversion with another ADC Note: The SYNCEN of both ADCs cannot be set to 1.
6 : 4	TRGSEL	rw	0x00	External trigger selection The trigger configuration of the ADC1 is as follows: 000: TIM1_CC1 001: TIM1_CC2 010: TIM1_CC3 011: TIM2_CC2 100: TIM3_TRGO 101: TIM4_CC4 110: TIM3_CC1 111: EXTI line 11 The trigger configuration of the ADC2 is as follows: 000: TIM1_TRGO 001: TIM1_CC4 010: TIM2_TRGO 011: TIM2_CC1 100: TIM3_CC4 101: TIM4_TRGO 110: TIM3_CC1 111: EXTI line 15
3	DMAEN	rw	0x00	Direct memory access enable 1 = DMA request enabled 0 = DMA disabled
2	TRGEN	rw	0x00	External trigger enable 1 = Start A/D conversion with external trigger signal 0 = Start A/D conversion without external trigger signal
1	ADWIE	rw	0x00	ADC window comparator interrupt enable 1 = A/D window comparator interrupt enabled 0 = A/D window comparator interrupt disabled

Bit	Field	Type	Reset	Description
0	ADIE	rw	0x00	ADC interrupt enable 1 = A/D interrupt enabled 0 = A/D interrupt disabled If ADIF is set, an interrupt request is generated after the A/D conversion.

10.10.4 A/D channel select register(ADC_ADCHS)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SENSOREN	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 9	Reserved			Reserved, always read as 0.
8	SENSOREN	rw	0x00	Sensor enable 1 = Enabled 0 = Disabled
7	CHEN7	rw	0x00	Analog input channel 7 enable 1 = Enabled 0 = Disabled
6	CHEN6	rw	0x00	Analog input channel 6 enable 1 = Enabled 0 = Disabled
5	CHEN5	rw	0x00	Analog input channel 5 enable 1 = Enabled 0 = Disabled
4	CHEN4	rw	0x00	Analog input channel 4 enable 1 = Enabled 0 = Disabled
3	CHEN3	rw	0x00	Analog input channel 3 enable 1 = Enabled 0 = Disabled
2	CHEN2	rw	0x00	Analog input channel 2 enable 1 = Enabled 0 = Disabled

Bit	Field	Type	Reset	Description
1	CHEN1	rw	0x00	Analog input channel 1 enable 1 = Enabled 0 = Disabled
0	CHEN0	rw	0x00	Analog input channel 0 enable 1 = Enabled 0 = Disabled

Note: If channels enabled are all 0, Channel 0 is enabled.

10.10.5 A/D window compare register(ADC_ADCMPR)

Address offset: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CMPLDATA											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 28	Reserved			Reserved, always read as 0.
27 : 16	CMPHDATA	rw	0x00	Compare data high limit The 12-bit value will be compared with the conversion result of the specified channel.
15 : 12	Reserved			Reserved, always read as 0.
11 : 0	CMPLDATA	rw	0x00	Compare data low limit The 12-bit value will be compared with the conversion result of the specified channel.

10.10.6 A/D status register(ADC_ADSTA)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			OVERRUN									Reserved			VALID
			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
VALID								CHANNEL				Res.	BUSY	ADWIF	ADIF
r	r	r	r	r	r	r	r	r	r	r	r		r	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
31 : 29	Reserved			Reserved, always read as 0.
28 : 20	OVERRUN	r	0x00	Overrun flag (Channel 0 ~ 8) Read only.
19 : 17	Reserved			Reserved, always read as 0.
16 : 8	VALID	r	0x00	Valid flag (Channel 0 ~ 8) Read only.
7 : 4	CHANNEL	r	0x00	Current conversion channel In case of BUSY = 1, the 4 bits indicate the channel being converted. In case of BUSY = 0, they indicate the channel to be converted in the next time.
3	Reserved			Reserved, always read as 0.
2	BUSY	r	0x00	Busy/idle 1 = A/D converter is busy 0 = A/D converter is idle
1	ADWIF	rc_w1	0x00	ADC window comparator interrupt flag If the result of selected A/D conversion channel is greater than or equal to ADCMPHR or less than ADCMPLR, this bit is set to '1'. This flag bit is cleared by writing '1'.
0	ADIF	rc_w1	0x00	ADC interrupt flag This bit is set by hardware at the end of channel group conversion and cleared by software. 1 = A/D conversion completed 0 = A/D conversion not completed This flag bit is cleared by writing' 1'.

10.10.7 A/D data register(ADC_ADDR0 ~ 8)

Address offset: 0x18 – 0x38

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 22	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
21	VALID	r	0x00	Valid flag(read-only) 1 = DATA[11: 0] bits are valid 0 = DATA[11: 0]bits are invalid After the conversion in the corresponding analog channel, this bit is set and this bit is cleared by hardware after the ADDATA register is read.
20	OVERRUN	r	0x00	Overrun flag(read-only) 1 = DATA [11: 0]bits are overwritten 0 = The last conversion result of DATA[11: 0] data Before the new conversion result is loaded into the register, if the data of DATA[11: 0] is not read, OVERRUN will be set to 1. This bit is cleared by hardware after the ADDATA register is read.
19 : 16	Reserved			Reserved, always read as 0.
15 : 0	DATA	r	0x00	Transfer data(12-bit A/D conversion result on channel) Left alignment or right alignment, depending on specific settings.

11 Advanced-Control Timer (TIM1)

Advanced-Control Timer (TIM1)

11.1 TIM1 introduction

Advanced-control timer(TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section Timer Synchronization.

11.2 Main features

TIM1 functions include:

- 16-bit up, down, up/down auto-reload register
- 16-bit programmable prescaler allowing dividing (modifing in real time) 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
- outputs with programmable dead-time
- circuit to control the timer with external signals and to interconnect several timers together.
- counter to update the timer registers only after a given number of cycles of the counter
- input to put the timer's output signals in reset state or in a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture

- Output compare
- Break input

- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 28. Block Diagram of Advanced-Control Timer

11.3 Functional description

11.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by

software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR 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 factor is taken into account at the next update event.

The following figures give some examples of the counter behavior when the prescaler factor is changed on the fly:

Figure 29. Counter Timing Diagram with Prescaler Division Change from 1 to 2

Figure 30. Counter Timing Diagram with Prescaler Division Change from 1 to 4

11.3.2 Counter modes

Upcounting mode

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

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

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

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag bit(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 31. Counter Timing Diagram, Internal Clock Divided by 1

Figure 32. Counter Timing Diagram, Internal Clock Divided by 2

Figure 33. Counter Timing Diagram, Internal Clock Divided by 4

Figure 34. Counter Timing Diagram, Internal Clock Divided by N

Figure 35. Counter Timing Diagram, Update Event When ARPE = 0 (TiMx_ARR Not Preloaded)

Figure 36. Counter Timing Diagram, Update Event When ARPE = 1 (TiMx_ARR Preloaded)

Downcounting mode

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

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

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

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag bit(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.

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

Figure 37. Counter Timing Diagram, Internal Clock Divided by 1

Figure 38. Counter Timing Diagram, Internal Clock Divided by 2

Figure 39. Counter Timing Diagram, Internal Clock Divided by 4

Figure 40. Counter Timing Diagram, Internal Clock Divided by N

Figure 41. Counter Timing Diagram, Update Event when Repetition Counter is Not Used

Center-aligned mode (Upcounting/Downcounting))

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) －1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

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) . In this case, the counter restarts counting from 0, so does 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 update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag bit(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 auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 42. Counter Timing Diagram, Internal Clock Divided by 1, TIMx_ARR = 6

Figure 43. Counter Timing Diagram, Internal Clock Divided by 2

Figure 44. Counter Timing Diagram, Internal Clock Divided by 4, TIMx_ARR = 0×36

Figure 45. Counter Timing Diagram, Internal Clock Divided by N

Figure 46. Counter Timing Diagram, Update Event with ARPE = 1(Counter Underflow)

Figure 47. Counter Timing Diagram, Update Event with ARPE = 1(Counter Overflow))

11.3.3 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 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 the TIMx_RCR register value (refer to Figure 49). 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 48. Update Rate Examples Depending on Modes and TIMx_RCR Register Settings

11.3.4 Clock selection

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

- Internal clock (CK_INT).
- External clock mode1: external input pin (TIx).
- External clock mode2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

Internal clock (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.

The following figure shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 49. 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 50. TI2 External Clock Connection Example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ’01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Select the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.

6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it. When a rising edge occurs on TI2, the counter counts once and the TIF flag is set. The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 51. Control Circuit in External Clock Mode 1

External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The following figure gives an overview of the external trigger input block.

Figure 52. External Trigger Input Block

For example, to configure the upcounter to count once each 2 rising edges on ETR, use the following procedure:

- As no filter is needed in this example, write ETF [3:0]=0000 in the TIMx_SMCR register.
- Set the prescaler by writing ETPS [1:0]=01 in the TIMx_SMCR register.
- Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
- Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 53. Control Circuit in External Clock Mode 2

11.3.5 Capture/compare channels

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

Figure 54 to Figure 57 give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 54. Capture/Compare Channel (Example: Channel 1 Input Stage)

Figure 55. Capture/Compare Channel 1 Main Circuit

Figure 56. Output stage of Capture/Compare Channel (Channels 1 to 3)

Figure 57. 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.

11.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When

a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written to '0'.

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

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming ICxF bits in the TIMx_CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at 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).
- Configure 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 to '1'.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

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

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

11.3.7 PWM input mode

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

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.

- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode. For example, user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):
- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P 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 58. PWM Input Mode Timing

The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

11.3.8 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, being independent of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, the user just needs to

write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

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

11.3.9 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 bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output.

The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode)

Procedure:

- Select the counter clock (internal, external, prescaler).
- Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
- Set the CCxIE bit if an interrupt request is to be generated.
- Select the output mode. For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches with CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
- 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', otherwise

TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in the following figure.

Figure 59. Output Compare Mode, Toggle on OC1

11.3.10 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 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 section 11.3.2.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx, otherwise 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 61 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 60. Edge-aligned PWM Waveforms (ARR = 8)

Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to section 11.3.2.

In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx, otherwise 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 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 section 11.3.2 Center-aligned Mode.

Figure 61 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- 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 61. Center-aligned PWM Waveforms (ARR = 8)

Hints in 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.

11.3.11 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 adjusted 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 51 Output Control Bits for Complementary OCx and OCxN Channels with Break Feature 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 OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.
- 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. 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 62. Complementary Output with Dead-time Insertion

Figure 63. Dead-time Waveforms with Delay Greater Than the Negative Pulse

Figure 64. Dead-time Waveforms with Delay Greater than the Positive Pulse

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to section 11.4.18 for delay calculation.

Re-directing OCxREF to OCx or OCxN

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

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

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

11.3.12 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 51 Output Control Bits for Complementary OCx and OCxN Channels with Break Feature for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller.

When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time.

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

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

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0 then the timer releases the enable output otherwise the enable output remains high.
- In case of complementary output:
  - The outputs are first put in reset state i.e. 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, 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 otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Otherwise, 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.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 11.4.8. The LOCK bits can be written only once after an MCU reset.

The following figure shows an example of behavior of the outputs in response to a break:

Figure 65. Output Behavior in Response to A Break

11.3.13 Clearing the OCxREF signal on an external event

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

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

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

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

The following Figure 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 66. Clearing TIMx OCxREF

11.3.14 Six-step PWM generation

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

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The following figure describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 67. Six-step PWM, COM Example (OSSR = 1)

11.3.15 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 68. 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 the value written in the compare registers (taking into account the clock frequency and the counter prescaler)

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this, enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case the compare value must be written in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

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

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

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

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

11.3.16 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 48. 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 corresponds to the rotation direction of the connected sensor. The following table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 48. Counting Direction Versus Encoder Signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI1FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The following figure gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is restrained 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', (TIMx_CCER register, IC1FP1 non-inverted, IC1FP1=TI1).
- C2P= '0' (TIMx_CCER register, IC2FP2 non-inverted, IC2FP2=TI2).
- SMS= '011' (TIMx_SMCR register, all inputs are active on both rising and falling edges).
- CEN= '1' (TIMx_CR1 register, Counter enabled).

Figure 69. Example of Counter Operation in Encoder Mode

Figure 70. Example of Encoder Interface Mode with Inverted IC1FP1

The timer, when configured in Encoder Interface mode provides information on the sensor's current position.The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times.

This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request generated by a real-time clock.

11.3.17 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in section 11.3.18.

11.3.18 Interfacing with Hall sensors

When using the advanced control timer (TIM1) to generate the PWM signal to drive the motor, another universal TIMx (TIM2, TIM3 or TIM4) timer can be used as the "interface timer" to connect the Hall sensor, see Autoreffig:190862, three timer input pins (CC1, CC2, CC3) are connected to the TI1 input channel through an XOR gate (selected by setting the TI1S bit in the TIMx_CR2 register), 'Interface Timer' Capture this signal.

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 54). 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 TIMx 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 ’01’. 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).

The following figure describes this example.

Figure 71. Example of Hall Sensor Interface

11.3.19 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, or a DMA request can be sent if enabled (depending on the TIE (interrupt enable) and TDE (DMA

enable) bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 72. 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 the trigger input level is)

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 73. Control Circuit in Gated Mode

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are only configured to select CC2P=1 in TIMx_CCER register, so as 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 74. 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:

- Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
- Configure the channel 1 as follows, to detect rising edges on T1:
  - 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).
- 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 75. Control Circuit in External Clock Mode 2 + Trigger Mode

11.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to Section TIM2/3/4 for details.

11.3.21 Debug mode

When the microcontroller enters debug mode (CPU core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to the following debug sections.

11.4 Register description

Table 49. Summary of TIM1 Register

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	Control register 1	0x00000000	section 11.4.1
0x04	TIMx_CR2	Control register 2	0x00000000	section 11.4.2
0x08	TIMx_SMCR	Slave mode control register	0x00000000	section 11.4.3
0x0C	TIMX_DIER	DMA /interrupt enable register	0x00000000	section 11.4.4
0x10	TIMx_SR	Status register	0x00000000	section 11.4.5
0x14	TIMx_EGR	Event generation register	0x00000000	section 11.4.6
0x18	TIMx_CCMR1	Capture/compare mode register 1	0x00000000	section 11.4.7
0x1C	TIMx_CCMR2	Capture/compare mode register 2	0x00000000	section 11.4.8
0x20	TIMx_CCER	Capture/compare enable register	0x00000000	section 11.4.9
0x24	TIMx_CNT	Counter	0x00000000	section 11.4.10
0x28	TIMx_PSC	Prescaler	0x00000000	section 11.4.11
0x2C	TIMx_ARR	Auto-reload register	0x00000000	section 11.4.12
0x30	TIMx_RCR	Repetition counter register	0x00000000	section 11.4.13
0x34	TIMx_CCR1	Capture/compare register 1	0x00000000	section 11.4.14
0x38	TIMx_CCR2	Capture/compare register 2	0x00000000	section 11.4.15
0x3C	TIMx_CCR3	Capture/compare register 3	0x00000000	section 11.4.16
0x40	TIMx_CCR4	Capture/compare register 4	0x00000000	section 11.4.17
0x44	TIMx_BDTR	Break and dead-time register	0x00000000	section 11.4.18
0x48	TIMx_DCR	DMA control register	0x00000000	section 11.4.19
0x4C	TIMx_DMAR	DMA address in continuous mode	0x00000000	section 11.4.20

11.4.1 Control register 1(TIMx_CR1)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 10	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
9: 8	CKD	rw	0x00	Clock division The 2 bits 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: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved, do not program this value
7	ARPE	rw	0x00	Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 5	CMS	rw	0x00	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 when the counter is enabled (CEN=1).
4	DIR	rw	0x00	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	0x00	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit	Field	Type	Reset	Description
2	URS	rw	0x00	Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt or DMA request if enabled.These events can be: - Counter overflow/underflow - Setting the UG bit - Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
1	UDIS	rw	0x00	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	0x00	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.

11.4.2 Control register 2(TIMx_CR2)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit	Field	Type	Reset	Description
15	Reserved			Reserved, always read as 0.
14	OIS4	rw	0x00	Output Idle state 4 (OC4 output).Refer to OIS1 bit.
13	OIS3N	rw	0x00	Output Idle state 3(OC3N output). Refer to OIS1N bit.
12	OIS3	rw	0x00	Output Idle state 4 (OC4 output).Refer to OIS1 bit.
11	OIS2N	rw	0x00	Output Idle state 3(OC3N output). Refer to OIS1N bit.
10	OIS2	rw	0x00	Output Idle state 4 (OC4 output).Refer to OIS1 bit.
9	OIS1N	rw	0x00	Output Idle state 1 (OC1N output) 0: OC1N=0 after a dead-time when MOE=0 1: OC1N=1 after a dead-time when MOE=0 Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).
8	OIS1	rw	0x00	Output Idle state 1 (OC1 output) 0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0 1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0 Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).
7	TI1S	rw	0x00	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)

Bit	Field	Type	Reset	Description
6: 4	MMS	rw	0x00	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	CCDS	rw	0x00	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs

Bit	Field	Type	Reset	Description
2	CCUS	rw	0x00	Capture/compare control update selection 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COM bit only 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COM bit or when an rising edge occurs on TRGI Note: This bit acts only on channels that have a complementary output.
1	Reserved			Reserved, always read as 0.
0	CCPC	rw	0x00	Capture/compare preloaded control 0: CCxE, CCxNE and OCxM bits are not preloaded 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set). Note: This bit acts only on channels that have a complementary output.

11.4.3 Slave mode control register(TIMx_SMCR)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Res.	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	External trigger polarity This bit selects whether ETR or inverted ETR is used for trigger operations. 0: ETR is non-inverted, active at high level or rising edge. 1: ETR is inverted, active at low level or falling edge.

Bit	Field	Type	Reset	Description
14	ECE	rw	0x00	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). Note 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). Note 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	rw	0x00	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

Bit	Field	Type	Reset	Description
11: 8	ETF	rw	0x00	External trigger filter This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS}. 0001: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 2 0010: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 4 0011: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 8 0100: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 6 0101: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 8 0110: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 6 0111: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 8 1000: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 6 1001: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 8 1010: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 5 1011: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 6 1100: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 8 1101: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 5 1110: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 6 1111: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 8
7	MSM	rw	0x00	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.

Bit	Field	Type	Reset	Description
6: 4	TS	rw	0x00	Trigger selection This bit-field selects the trigger input to be used to synchronize the counter. 000: Internal Trigger 0 (ITR0) 001: Internal Trigger 1(ITR1) 010: Internal Trigger 2(ITR2) 011: Internal Trigger 3(ITR3) 100: TI1 Edge Detector (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1) 110: Filtered Timer Input 2(TI2FP2) 111: External Trigger input (ETRF) See the following table for more details on ITRx. 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	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
2: 0	SMS	rw	0x00	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 (refer to 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 input 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 input (TRGI) clock the counter. Note: The gated mode must not be used if TI1F_EN 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.

Table 50. TIMx internal trigger connection

Slave timer	ITR0	ITR1	ITR2	ITR3
TIM1	None	TIM2	TIM3	TIM4
TIM2	TIM1	None	TIM3	None
TIM3	TIM1	TIM2	None	None
TIM4	TIM1	TIM2	TIM3	None

11.4.4 DMA/interrupt enable register (TIMX_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COM DE	CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE	BIE	TIE	COM IE	CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15	Reserved			Reserved, always read as 0.
14	TDE	rw	0x00	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	COMDE	rw	0x00	COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled
12	CC4DE	rw	0x00	Capture/Compare 4 DMA request enable 0: CC4 DMA request disabled 1: CC4 DMA request enabled
11	CC3DE	rw	0x00	Capture/Compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
10	CC2DE	rw	0x00	Capture/Compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	BIE	rw	0x00	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	TIE	rw	0x00	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	COMIE	rw	0x00	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	CC4IE	rw	0x00	Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled 1: CC4 interrupt enabled
3	CC3IE	rw	0x00	Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled 1: CC3 interrupt enabled

Bit	Field	Type	Reset	Description
2	CC2IE	rw	0x00	Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

11.4.5 Status register(TIMx_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4 OF	CC3 OF	CC2 OF	CC1 OF	Res.	BIF	TIF	COM IF	CC4 IF	CC3 IF	CC2 IF	CC1 IF	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	Field	Type	Reset	Description
15: 13	Reserved			Reserved, always read as 0.
12	CC4OF	rc_w0	0x00	Capture/Compare 4 overcapture flag Refer to CC1OF description.
11	CC3OF	rc_w0	0x00	Capture/Compare 3 overcapture flag Refer to CC1OF description.
10	CC2OF	rc_w0	0x00	Capture/Compare 2 overcapture flag Refer to CC1OF description.
9	CC1OF	rc_w0	0x00	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	Reserved			Reserved, always read as 0.
7	BIF	rc_w0	0x00	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

Bit	Field	Type	Reset	Description
6	TIF	rc_w0	0x00	Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
5	COMIF	rc_w0	0x00	COM interrupt flag This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software. 0: No COM event occurred. 1: COM interrupt pending.
4	CC4IF	rc_w0	0x00	Capture/Compare 4 interrupt flag Refer to CC1IF description.
3	CC3IF	rc_w0	0x00	Capture/Compare 3 interrupt flag Refer to CC1IF description.
2	CC2IF	rc_w0	0x00	Capture/Compare 2 interrupt flag Refer to CC1IF description.
1	CC1IF	rc_w0	0x00	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. If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred 1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit	Field	Type	Reset	Description
0	UIF	rc_w0	0x00	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 the description of synchronization control register), if URS=0 and UDIS=0 in the TIMx_CR1 register.

11.4.6 Event generation register (TIMx_EGR)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15: 8	Reserved			Reserved, always read as 0.
7	BG	w	0x00	Break generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.
6	TG	w	0x00	Trigger generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit	Field	Type	Reset	Description
5	COMG	w	0x00	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	0x00	Capture/Compare 4 generation Refer to CC1G description.
3	CC3G	w	0x00	Capture/Compare 3 generation Refer to CC1G description.
2	CC2G	w	0x00	Capture/Compare 2 generation Refer to CC1G description.
1	CC1G	w	0x00	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 CC1 If channel CC1 is configured as output: CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled. If channel CC1 is configured as input: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
0	UG	w	0x00	Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler factor is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), otherwise, it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

11.4.7 Capture/compare mode register 1 (TIMx_CCMR1)

Offset address: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M			OC2 PE	OC2 FE	CC2S		OC1 CE	OC1M			OC1 PE	OC1 FE	CC1S	
IC2F				IC2PSC				IC1F				IC1PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	Output compare 2 clear enable
14: 12	OC2M	rw	0x00	Output compare 2 mode
11	OC2PE	rw	0x00	Output compare 2 preload enable
10	OC2FE	rw	0x00	Output compare 4 fast enable
9: 8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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	0x00	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

Bit	Field	Type	Reset	Description
6: 4	OC1M	rw	0x00	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 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 otherwise inactive. In downcounting, channel 1 is inactive (OC1REF=’0’) as long as TIMx_CNT>TIMx_CCR1 otherwise active (OC1REF=’1’). 111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1otherwise active. In downcounting, channel 1 is active as long as TIMx_CNT > TIMx_CCR1 otherwise 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). Note 2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from ”frozen” mode to ”PWM” mode.

Bit	Field	Type	Reset	Description
3	OC1PE	rw	0x00	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). Note 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	0x00	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	rw	0x00	Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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:

Bit	Field	Type	Reset	Description
15: 12	IC2F	rw	0x00	Input capture 2 filter
11: 10	IC2PSC	rw	0x00	Input capture 2 prescaler
9: 8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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: 4	IC1F	rw	0x00	Input capture 1 filter This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 1000: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 6 0001: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 2 1001: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 8 0010: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 4 1010: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 5 0011: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 8 1011: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 6 0100: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 6 1100: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 8 0101: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 8 1101: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 5 0110: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 6 1110: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 6 0111: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 8 1111: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 8

Bit	Field	Type	Reset	Description
3: 2	IC1PSC	rw	0x00	Input capture 1 prescaler This bit-field defines the factor 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	rw	0x00	Capture/compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the TS bit (TIMx_SMCR register) Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

11.4.8 Capture/compare mode register 2(TIMx_CCMR2)

Offset address: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M			OC4 PE	OC4 FE	CC4S		OC3 CE	OC3M			OC3 PE	OC3 FE	CC3S	
IC4F				IC4PSC				IC3F				IC3PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	Output compare 4 clear enable
14: 12	OC4M	rw	0x00	Output compare 4 mode
11	OC4PE	rw	0x00	Output compare 4 preload enable
10	OC4FE	rw	0x00	Output compare 4 fast enable

Bit	Field	Type	Reset	Description
9: 8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the TS bit (TIMx_SMCR register) Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER)
7	OC3CE	rw	0x00	Output compare 3 clear enable
6: 4	OC3M	rw	0x00	Output compare 3 mode
3	OC3PE	rw	0x00	Output compare 3 preload enable
2	OC3FE	rw	0x00	Output compare 3 fast enable
1: 0	CC3S	rw	0x00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the 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	Field	Type	Reset	Description
15: 12	IC4F	rw	0x00	Input capture 4 filter
11: 10	IC4PSC	rw	0x00	Input capture 4 prescaler

Bit	Field	Type	Reset	Description
9: 8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the 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	0x00	Input capture 3 filter
3: 2	IC3PSC	rw	0x00	Input capture 3 prescaler
1: 0	CC3S	rw	0x00	Capture/compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI4 10: CC3 channel is configured as input, IC3 is mapped on TI3 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 the TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

11.4.9 Capture/compare enable register(TIMx_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3 NP	CC3 NE	CC3P	CC3E	CC2 NP	CC2 NE	CC2P	CC2E	CC1 NP	CC1 NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 14	Reserved			Reserved, always read as 0.
13	CC4P	rw	0x00	Capture/Compare 4 output polarity Refer to CC1P description.
12	CC4E	rw	0x00	Capture/Compare 4 output enable Refer to CC1E description.

Bit	Field	Type	Reset	Description
11	CC3NP	rw	0x00	Capture/Compare 3 complementary output polarity Refer to CC1NP description.
10	CC3NE	rw	0x00	Capture/Compare 3 complementary output enable Refer to CC1NE description.
9	CC3P	rw	0x00	Capture/Compare 3 output polarity Refer to CC1P description.
8	CC3E	rw	0x00	Capture/Compare 3 output enable Refer to CC1E description.
7	CC2NP	rw	0x00	Capture/Compare 2 complementary output polarity Refer to CC1NP description.
6	CC2NE	rw	0x00	Capture/Compare 2 complementary output enable Refer to CC1NE description.
5	CC2P	rw	0x00	Capture/Compare 2 output polarity Refer to CC1P description.
4	CC2E	rw	0x00	Capture/Compare 2 output enable Refer to CC1E description.
3	CC1NP	rw	0x00	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low Note: This bit is not modified 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	0x00	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.
1	CC1P	rw	0x00	Capture/Compare 1 output polarity CC1 channel is configured as output: 0: OC1 active high 1: OC1 active low CC1 channel is configured as input: This bit selects whether IC1 or inverted IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted

Bit	Field	Type	Reset	Description
0	CC1E	rw	0x00	Capture/Compare 1 output enable CC1 channel is configured as output: 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. CC1 channel is configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not. 0: Capture disabled. 1: Capture enabled.

Table 51. Output Control Bits for Complementary OCx and OCxN Channels with Break Feature

Control bits					Output states[1]	
MOE bit	**OSSI bit**	**OSSR bit**	**CCxE bit**	**CCxNE bit**	**OCx output state**	**OCxN output state**
1	X	0	0	0	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	0	1	Output Disabled (not driven by the timer), OCx = 0, OCx_EN = 0	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Output Disabled (not driven by the timer), OCxN = 0, OCxN_EN = 0
		0	1	1	OCREF + Polarity + dead-time, OCx_EN = 1	OCxREF (inverted) + Polarity + dead-time, OCxN_EN = 1
		1	0	0	Output Disabled (not driven by the timer), OCx = CCxP, OCx_EN = 0	Output Disabled (not driven by the timer), OCxN = CCxNP, OCxN_EN = 0
		1	0	1	Off-State (output enabled with inactive state) OCx = CCxP, OCx_EN = 1	OCxREF + Polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	0	OCxREF + Polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Off-State (output enabled with in-active state), OCxN = CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + Polarity + dead-time, OCx_EN = 1	OCxREF (inverted) + 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:after a dead-time OCx = OISx , OCxN = OISxN, Assuming that OISx and OISxN do not correspond to OCX and OCxN in active state.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	Off-State (output enabled with inactive state) Asynchronously: OCx = CCxP, OCx_EN = 1, OCxN = CCxNP, OCxN_EN = 1; Then if the clock is present:after a dead-time OCx = OISx , OCxN = OISxN, Assuming that OISx and OISxN do not correspond to OCX and OCxN in active state.	
	1		0	1		
	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 1: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

2: In case of CCxE=0 and CCxNE=0, OCx and OCxN are in high impedance state after output is disabled.

11.4.10 Counter(TIMx_CNT)

Offset address: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CNT	rw	0x0000	Counter value

11.4.11 Prescaler(TIMx_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	PSC	rw	0x0000	Prescaler value The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} /(PSC + 1). PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in ”reset mode”)

11.4.12 Auto-reload register(TIMx_ARR)

Offset address: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	ARR	rw	0x0000	Prescaler value ARR is the value to be loaded in the actual auto-reload register. Refer to section 11.3.1 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

11.4.13 Repetition counter register(TIMx_RCR)

Offset address: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 8	Reserved			Reserved, always read as 0.
7: 0	REP	rw	0x00	Repetition counter value These bits allow the user to set-up the update rate of the compare registers (i.e. periodically transfers from preload to active registers) when preload registers are enabled, as well as the update interrupt generation rate, if this interrupt is enabled. 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 - number of half PWM period in center-aligned mode

11.4.14 Capture/compare register 1(TIMx_CCR1)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR1	rw	0x0000	Capture/Compare 1 value If CC1 channel is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Otherwise 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 CC1 channel is configured as input: CCR1 contains the counter value transferred by the last input capture 1 event (IC1).

11.4.15 Capture/compare register2(TIMx_CCR2)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR2	rw	0x0000	Capture/Compare 2 value If CC2 channel is configured as output: CCR2 contains the value to be loaded in the actual capture/compare 2 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Otherwise the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output. If CC2 channel is configured as input: CCR2 contains the counter value transferred by the last input capture 2 event (IC2).

11.4.16 Capture/compare register 3(TIMx_CCR3)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR3	rw	0x0000	Capture/Compare 3 value If CC3 channel is configured as output: CCR3 contains the value to be loaded in the actual capture/compare 3 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Otherwise the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output. If CC3 channel is configured as input: CCR3 contains the counter value transferred by the last input capture 3 event (IC3).

11.4.17 Capture/compare register 4(TIMx_CCR4)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR4	rw	0x0000	Capture/Compare 4 value If CC4 channel is configured as output: CCR4 contains the value to be loaded in the actual capture/compare 4 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Otherwise 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 signaled on OC4 output. If CC4 channel is configured as input: CCR4 contains the counter value transferred by the last input capture 4 event (IC4).

11.4.18 Break and dead-time register(TIMx_BDTR)

Offset address: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK		DTG							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSI, OSSR and DTG can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bit	Field	Type	Reset	Description
15	MOE	rw	0x00	Main output enable This bit is cleared asynchronously by hardware as soon as the break input is active. It is cleared by software or set automatically, depending on the AOE bit. It is acting only on the channels which are configured in output. 0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register). See OC/OCN enable description for more details (section 11.4.9: capture/compare enable register).

Bit	Field	Type	Reset	Description
14	AOE	rw	0x00	Automatic output enable 0: MOE can be set only by software 1: MOE can be set by software or automatically at the next update event (if the break input is not be active) Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
13	BKP	rw	0x00	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).
12	BKE	rw	0x00	Break enable 0: Break inputs (BRK and BRK_ACTH) disabled 1: Break inputs (BRK and BRK_ACTH) enabled Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
11	OSSR	rw	0x00	Off-state selection for Run mode This bit is used when MOE=1 on channels configured as complementary outputs. OSSR is not implemented if no complementary output is implemented in the timer. See OC/OCN enable description for more details (section 11.4.9: capture/compare enable register (TIMx_CCER)). 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. Then, set OC/OCN enable output signal=1 Note: This bit can not be modified as long as LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit	Field	Type	Reset	Description
10	OSSI	rw	0x00	Off-state selection for Idle mode This bit is used when MOE=0 on channels configured as outputs. See OC/OCN enable description for more details (section 11.4.9: capture/compare enable register (TIMx_CCER)). 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1 Note: This bit can not be modified as long as LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
9: 8	LOCK	rw	0x00	Lock configuration These bits offer a write protection against software errors. 00: LOCK OFF - No bit is write protected. 01: LOCK Level 1 = DTG, BKE, BKP, AOE bits in TIMx_BDTR register, and OISx/OISxN bits in TIMx_CR2 register can no longer be written. 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written. 11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written. Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bit	Field	Type	Reset	Description
7: 0	DTG	rw	0x00	Dead-time generator setup This bit-field defines the duration of the dead-time inserted between the complementary outputs. It is assumed that DT correspond to this duration. DTG[7: 5] = 0xx: DT = (DTG[7: 0] + 1) × t_{dtg}, t_{dtg} = t_{DTS}; DTG[7: 5] = 10x: DT = (DTG[5: 0] + 1 + 64) × t_{dtg}, t_{dtg} = 2 × t_{DTS}; DTG[7: 5] = 110: DT = (DTG[4: 0] + 1 + 32) × t_{dtg}, t_{dtg} = 8 × t_{DTS}; DTG[7: 5] = 111: DT = (DTG[4: 0] + 1 + 32) × t_{dtg}, t_{dtg} = 16 × t_{DTS}; Example: if t_{DTS} = 125ns(8MHz), dead-time possible values are: 125ns to 15875ns by 125 nS steps; 16μs to 31750ns by 250 nS steps; 32μs to 63μs by 1 μs steps; 64μs to 126μs by 2 μs steps; Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

11.4.19 DMA control register(TIMx_DCR)

Offset address: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			DBL					Res.			DBA				
			w	w	w	w	w				w	w	w	w	w

Bit	Field	Type	Reset	Description
15: 13	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
12: 8	DBL	w	0x00	DMA burst length This bit field defines the burst transfer in the continuous mode (the timer detects a burst transfer when a write access to the TIMx_DMAR register address is performed), namely, the number of transfers, in half-word (double bytes) or bytes. 00000: 1 transfer 00001: 2 transfers 00010: 3 transfers .... 10001: 18 transfers Example: Let us consider the following transfer: DBL = 7 and DBA = TIM2_CR1. - If DBL =7 and DBA = TIM2_CR1 represent the address of data to be transferred, the transfer address is given by: (Address of TIMx_CR1) + DBA + (DMA index), where, DMA index = DBL TIMx_CR1 address + DBA + 7 is the address of data to be written or read, so that the transfer is completed to/from 7 registers starting from the TIMx_CR1 address + DBA. According to the setting of DMA data length, the following case may occur: -If the data is set to half word (16 bits), the data will be transferred to all 7 registers. -If the data is set to bytes, the data will still be transferred to all 7 registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore, the user must specify the data width of DMA transfer for the timer.
7: 5	Reserved			Reserved, always read as 0.
4: 0	DBA	w	0x00	DMA base address These bits define the base-address for DMA transfers in the continuous mode (when write access is done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register. 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR .....

11.4.20 DMA address for full transfer(TIMx_DMAR)

Offset address: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15: 0	DMAB	w	0x0000	DMA register for burst accesses A write operation to the TIMx_DMAR register will access the register located at the following address: TIMx_CR1 address + DBA + DMA index, Where: 'TIMx_CR1 address' is the address of the control register 1; 'DBA' is the DMA base address configured in TIMx_DCR register; 'DMA index' is the offset automatically controlled by the DMA transfer, depending on DBL configured in TIMx_DCR.

12 16-bit General-Purpose Timer (TIMx 16 Bit)

16-bit General-Purpose Timer (TIMx16 Bit)

12.1 TIMx introduction

General-purpose timers 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 and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

TIMx are completely independent, and do not share any resources. They can be synchronized together as described in Section Timer Synchronization.

12.2 TIMx Main features

TIM2TIM3TIM4 functions include:

- 16-bit up, down, up/down auto-reload register
- 16-bit programmable prescaler allowing dividing (modifing in real time) 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
- circuit to control the timer with external signals and to interconnect several timers together.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes

- Trigger input for external clock or cycle-by-cycle current management

Notes: Reg Preload registers transferred to active registers on U event accroding to control bit.

U Event

Interrupt & DMA output

Figure 76. Block Diagram of General-Purpose Timer

12.3 TIMx Functional description

12.3.1 Time-base unit

The main block of the programmable general-purpose 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)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 factor is taken into account at the next update event.

The following figures give some examples of the counter behavior when the prescaler factor is changed on the fly:

Figure 77. Counter Timing Diagram with Prescaler Division Change from 1 to 2

Figure 78. Counter Timing Diagram with Prescaler Division Change from 1 to 4

12.3.2 Counter modes

Upcounting mode

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

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag bit(UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload shadow register is updated with the preload value (TIMx_ARR).

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

Figure 79. Counter Timing Diagram, Internal Clock Divided by 1

Figure 80. Counter Timing Diagram, Internal Clock Divided by 2

Figure 81. Counter Timing Diagram, Internal Clock Divided by 4

Figure 82. Counter Timing Diagram, Internal Clock Divided by N

Figure 83. Counter Timing Diagram, Update Event When ARPE = 0 (TiMx_ARR Not Preloaded)

Figure 84. Counter Timing Diagram, Update Event When ARPE = 1 (TiMx_ARR Preloaded)

Downcounting mode

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

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register).

Note: The auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock

frequencies when TIMx_ARR = 0x36.

Figure 85. Counter Timing Diagram, Internal Clock Divided by 1

Figure 86. Counter Timing Diagram, Internal Clock Divided by 2

Figure 87. Counter Timing Diagram, Internal Clock Divided by 4

Figure 88. Counter Timing Diagram, Internal Clock Divided by N

Figure 89. Counter Timing Diagram, Update Event when Repetition Counter is Not Used

Center-aligned mode (Upcounting/Downcounting))

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) －1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

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) . In this case, the counter restarts counting from 0, so does 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 update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag bit(UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register).

Note: If the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 90. Counter Timing Diagram, Internal Clock Divided by 1, TIMx_ARR = 6

Figure 91. Counter Timing Diagram, Internal Clock Divided by 2

Figure 92. Counter Timing Diagram, Internal Clock Divided by 4, TIMx_ARR = 0×36

Figure 93. Counter Timing Diagram, Internal Clock Divided by N

Figure 94. Counter Timing Diagram, Update Event with ARPE = 1(Counter Underflow)

Figure 95. Counter Timing Diagram, Update Event with ARPE = 1(Counter Overflow))

12.3.3 Clock selection

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

- Internal clock (CK_INT).
- External clock mode1: external input pin (TIx).
- External clock mode2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

Internal clock (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.

The following figure shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 96. 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 97. TI2 External Clock Connection Example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ’01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000). Note: The capture prescaler is not used for triggering, so there: Tno need to configure it.
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Select the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.

5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 98. Control Circuit in External Clock Mode 1

External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The following figure gives an overview of the external trigger input block.

Figure 99. External Trigger Input Block

For example, to configure the upcounter to count once each 2 rising edges on ETR, use the following procedure:

- As no filter is needed in this example, write ETF [3:0]=0000 in the TIMx_SMCR register.
- Set the prescaler by writing ETPS [1:0]=01 in the TIMx_SMCR register.
- Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register.
- Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 100. Control Circuit in External Clock Mode 2

12.3.4 Capture/compare channels

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

The following figures show a capture/compare channel. The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 101. Capture/Compare Channel (Example: Channel 1 Input Stage)

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

Figure 102. Capture/Compare Channel 1 Main Circuit

Figure 103. Output Stage of Capture/Compare Channel (Channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

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

12.3.5 Input capture mode

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

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

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming 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 most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at 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).
- Configure 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 to '1'.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- 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.
- CC1OF is also set to 1.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

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

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

12.3.6 PWM input mode

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

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

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

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P 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 104. Output Stage of Capture/Compare Channel (Channel 1)

The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode.

12.3.7 Forced output mode

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

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

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

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

12.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output

compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output.

The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode)

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, the user must write OCxM = 011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in the following figure.

Figure 105. Output Compare Mode (Toggle OC1)

12.3.9 PWM mode

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

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The 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 the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIM1_CCRx are always compared to determine whether TIM1_CCRx≤TIM1_CNT or TIM1_CNT≤TIM1_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR (OCxREF can be cleared by an external event through the ETR signal until the next PWM period), the OCxREF signal is asserted only:

- When the result of the comparison changes
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the period), the OC enabled by the CCxE bit in the TIMx_CCER register. Refer to the

TIMx_CCERx register

This forces the PWM by software while the timer is running. The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

Upcounting configuration

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

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx, otherwise 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 61 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 106. 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, otherwise 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 sig-

nals). The compare flag is set when the counter counts up, when it counts down or 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 section 11.3.2 Center-aligned Mode.

The following figure shows some center-aligned PWM waveforms in an example, where:

- TIMx_ARR = 8
- 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 107. Center-aligned PWM Waveforms (ARR = 8)

Hints in 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.

12.3.10 One-pulse mode

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

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

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

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

Figure 108. 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 the value written in the compare registers (taking into account the clock frequency and the counter prescaler)

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants to build a waveform with a transition from ’0’ to ’1’ when a compare match occurs and a transition from ’1’ to ’0’ when the counter reaches the auto-reload value. To do this, enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case the compare value must be written in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ’0’ in this example.

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

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

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

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

12.3.11 Clearing the OCxREF signal on an external event

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

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

For example, the OCxREF 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:

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

The following Figure 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 109. Clearing TIMx OCxREF

12.3.12 Encoder interface mode

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

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

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Assuming that the counter is enabled (CEN bit in TIMx_CR1 register written to ’1) in the following table, it 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). 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, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The following table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 52. Counting Direction Versus Encoder Signals

Active edge	**Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)**	**TI1FP1 signal**		**TI1FP2 signal**	
		Rising	**Falling**	**Rising**	**Falling**
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The following figure 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', (TIMx_CCER register, IC1FP1 non-inverted, IC1FP1=TI1).
- C2P= '0' (TIMx_CCER register, IC2FP2 non-inverted, IC2FP2=TI2).
- SMS= '011' (TIMx_SMCR register, all inputs are active on both rising and falling edges).
- CEN= '1' (TIMx_CR1 register, Counter enabled).

Figure 110. Example of Counter Operation in Encoder Mode

The following figure gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 111. Example of Encoder Interface Mode with Inverted Polarity IC1FP1

The timer, when configured in Encoder Interface mode provides information on the sensor's current position.The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be

generated by another timer). When available, it is also possible to read its value through a DMA request generated by a real-time clock.

12.3.13 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in section 11.3.18.

12.3.14 Timers 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 the user does not need to configure it. The CC1S bits select the input capture source only, i.e. 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, or a DMA request can be sent if enabled depending on the TIE (interrupt enable) and TDE (DMA enable) bits in TIMx_DIER register.

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 112. 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 level on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=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.
- Start the counter by writing CEN=1 in the TIMx_CR1 register. In gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 113. Control Circuit in Gated Mode

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are only configured to select CC2P=1 in TIMx_CCER register, so as 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 stop of the counter is due to the resynchronization circuit on TI2 input.

Figure 114. 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:

- Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

- Configure the channel 1 as follows, to detect rising edges on T1:
  - 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).
- 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 115. Control Circuit in External Clock Mode 2 + Trigger Mode

12.3.15 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

The following figure presents an overview of the trigger selection and the master mode selection blocks.

Using one timer as prescaler for another timer

Figure 116. Master/Slave Timer Example

For example, the user can configure Timer 1 to act as a prescaler for Timer 2 (see the above figure). To do this:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM1_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of Timer 1 to Timer 2, Timer 2 must be configured in slave mode using ITR1 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=000).
- Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which corresponds to the Timer 1 counter overflow).
- Finally, both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on Timer 1 as trigger output (MMS=1xx), its rising edge is used to clock the counter of Timer 2.

Using one timer to enable another timer

In this example, we control the enable of Timer 2 with the output compare 1 of Timer 1. Refer to the following figure for connections. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK_CNT} = f_{CK_INT/3}$).

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=001 in the TIM2_SMCR

register).

- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Enable Timer 2 by writing ’1 in the CEN bit (TIM2_CR1 register).
- Enable Timer 1 by writing ’1 in the CEN bit (TIM1_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.

Figure 117. Gating Timer 2 with OC1REF of Timer 1

In the example in the above figure, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing ’0 to the CEN bit in the TIM1_CR1 register:

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Reset Timer 1 by writing ’1 in UG bit (TIM1_EGR register).
- Reset Timer 2 by writing ’1 in UG bit (TIM2_EGR register).
- Initialize Timer 2 to 0xE7 by writing ’0xE7’ in the Timer 2 counter (TIM2_CNT).
- Enable Timer 2 by writing ’1 in the CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing ’1 in the CEN bit (TIM1_CR1 register).
- Stop Timer 1 by writing ’0 in the CEN bit (TIM1_CR1 register).

Figure 118. Gating Timer 2 with Enable of Timer 1

Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to the following figure for connections. Timer 2 starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by Timer 1.

When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0' to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK_CNT} = f_{CK_INT/3}$).

- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
- Configure the Timer 1 period (TIM1_ARR registers).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in TIM2_SMCR register).
- Start Timer 1 by writing '1' in the CEN bit (TIM1_CR1 register)

Figure 119. Triggering Timer 2 with Update of Timer 1

As in the previous example, the user can initialize both counters before starting counting. The following figure shows the behavior with the same configuration as ’0’ but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 120. Triggering Timer 2 with Enable of Timer 1

Using one additional timer as prescaler for another timer

In this example, we use Timer 1 as the prescaler for Timer 2. The configuration is as follows:

- Configure Timer 1 in master mode, togenerate the update event (UEV) as the trigger output (MMS=010 in the TIM1_CR2 register). Then, output a periodic signal in case of each counter overflow.
- Configure the Timer 1 period (TIM1_ARR registers).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in the external clock mode (write SMS=111 in the TIM2_SMCR reg-

ister).

- Start Timer 2 by writing '1' in the CEN bit (TIM1_CR2 register)
- Start Timer 1 by writing '1' in the CEN bit (TIM1_CR1 register).

Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. To ensure the counters are aligned, Timer 1 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer 2):

- Configure Timer 1 master mode to send its Enable as trigger output (MMS=001 in the TIM1_CR2 register).
- Configure Timer 1 slave mode to get the input trigger from TI1 (TS=100 in the TIM1_SMCR register).
- Configure Timer 1 in trigger mode (SMS=110 in the TIM1_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (Timer 1), both counters starts counting synchronously on the internal clock and both TIF flags are set. Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on timer 1.

Figure 121. Triggering Timer 1 and 2 with Timer 1 TI1 input

12.3.16 Debug mode

When the microcontroller enters debug mode (CPU core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in

DBG module. For more details, refer to ”Debug” sections.

12.4 TIMx register description

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

Table 53. Summary of TIMx Register

Offset	Acronym	Register Name	Reset	Section
0x00	TIMx_CR1	Control register 1	0x00000000	section 12.4.1
0x04	TIMx_CR2	Control register 2	0x00000000	section 12.4.2
0x08	TIMx_SMCR	Slave mode control register	0x00000000	section 12.4.3
0x0C	TIMx_DIER	DMA /interrupt enable register	0x00000000	section 12.4.4
0x10	TIMx_SR	Status register	0x00000000	section 12.4.5
0x14	TIMx_EGR	Event generation register	0x00000000	section 12.4.6
0x18	TIMx_CCMR1	Capture/compare mode register 1	0x00000000	section 12.4.7
0x1C	TIMx_CCMR2	Capture/compare mode register 2	0x00000000	section 12.4.8
0x20	TIMx_CCER	Capture/compare enable register	0x00000000	section 12.4.9
0x24	TIMx_CNT	Counter	0x00000000	section 12.4.10
0x28	TIMx_PSC	Prescaler	0x00000000	section 12.4.11
0x2C	TIMx_ARR	Auto-reload register	0x00000000	section 12.4.12
0x34	TIMx_CCR1	Capture/compare register 1	0x00000000	section 12.4.13
0x38	TIMx_CCR2	Capture/compare register 2	0x00000000	section 12.4.14
0x3C	TIMx_CCR3	Capture/compare register 3	0x00000000	section 12.4.15
0x40	TIMx_CCR4	Capture/compare register 4	0x00000000	section 12.4.16
0x48	TIMx_DCR	DMA control register	0x00000000	section 12.4.17
0x4C	TIMx_DMAR	DMA address in continuous mode	0x00000000	section 12.4.18

12.4.1 Control register 1(TIMx_CR1)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 10	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
9: 8	CKD	rw	0x00	Clock division The 2 bits indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock used by the dead-time generators and the digital filters (ETR, TIx). 00: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved, do not program this value
7	ARPE	rw	0x00	Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 5	CMS	rw	0x00	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	0x00	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	0x00	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit	Field	Type	Reset	Description
2	URS	rw	0x00	Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt or DMA request if enabled.These events can be: - Counter overflow/underflow - Setting the UG bit - Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
1	UDIS	rw	0x00	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	0x00	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.

12.4.2 Control register 2(TIMx_CR2)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS			CCDS	Reserved		
								rw	rw	rw	rw	rw			

Bit	Field	Type	Reset	Description
15: 8	Reserved			Reserved, always read as 0.
7	TI1S	rw	0x00	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	rw	0x00	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	CCDS	rw	0x00	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs

Bit	Field	Type	Reset	Description
2: 0	Reserved			Reserved, always read as 0.

12.4.3 Slave mode control register(TIMx_SMCR)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Res.	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Field	Type	Reset	Description
15	ETP	rw	0x00	External trigger polarity This bit selects whether ETR or inverted 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	0x00	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). Note 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). Note 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	rw	0x00	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

Bit	Field	Type	Reset	Description
11: 8	ETF	rw	0x00	External trigger filter This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS}. 0001: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 2 0010: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 4 0011: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 8 0100: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 6 0101: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 8 0110: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 6 0111: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 8 1000: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 6 1001: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 8 1010: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 5 1011: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 6 1100: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 8 1101: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 5 1110: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 6 1111: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 8
7	MSM	rw	0x00	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.

Bit	Field	Type	Reset	Description
6: 4	TS	rw	0x00	Trigger selection This bit-field selects the trigger input to be used to synchronize the counter. 000: Internal Trigger 0 (ITR0) 001: Internal Trigger 1(ITR1) 010: Internal Trigger 2(ITR2) 011: Internal Trigger 3(ITR3) 100: TI1 Edge Detector (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1) 110: Filtered Timer Input 2(TI2FP2) 111: External Trigger input (ETRF) See the following table for more details on ITRx. 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	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
2: 0	SMS	rw	0x00	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 (refer to 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 input 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 input (TRGI) clock the counter. Note: The gated mode must not be used if TI1F_EN 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.

Table 54. TIMx Internal Trigger Connection

Slave timer	ITR0(TS = 000)	ITR1(TS = 001)	ITR2(TS = 010)	ITR3(TS = 011)
TIM2	TIM1	x	TIM3	TIM4
TIM3	TIM1	TIM2	x	TIM4
TIM4	TIM1	TIM2	TIM3	x

12.4.4 DMA/interrupt enable register(TIMx_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15	Reserved			Reserved, always read as 0.
14	TDE	rw	0x00	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	Reserved			Reserved, always read as 0.
12	CC4DE	rw	0x00	Capture/Compare 4 DMA request enable 0: CC4 DMA request disabled 1: CC4 DMA request enabled
11	CC3DE	rw	0x00	Capture/Compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
10	CC2DE	rw	0x00	Capture/Compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
9	CC1DE	rw	0x00	Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
8	UDE	rw	0x00	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	Reserved			Reserved, always read as 0.
6	TIE	rw	0x00	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	Reserved			Reserved, always read as 0.
4	CC4IE	rw	0x00	Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled 1: CC4 interrupt enabled
3	CC3IE	rw	0x00	Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled 1: CC3 interrupt enabled
2	CC2IE	rw	0x00	Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
1	CC1IE	rw	0x00	Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled

Bit	Field	Type	Reset	Description
0	UIE	rw	0x00	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

12.4.5 Status register(TIMx_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			CC4OF	CC3OF	CC2OF	CC1OF	Res.		TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit	Field	Type	Reset	Description
15: 13	Reserved			Reserved, always read as 0.
12	CC4OF	rc_w0	0x00	Capture/Compare 4 overcapture flag Refer to CC1OF description.
11	CC3OF	rc_w0	0x00	Capture/Compare 3 overcapture flag Refer to CC1OF description.
10	CC2OF	rc_w0	0x00	Capture/Compare 2 overcapture flag Refer to CC1OF description.
9	CC1OF	rc_w0	0x00	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: 7	Reserved			Reserved, always read as 0.
6	TIF	rc_w0	0x00	Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
5	Reserved			Reserved, always read as 0.
4	CC4IF	rc_w0	0x00	Capture/Compare 4 interrupt flag Refer to CC1IF description.
3	CC3IF	rc_w0	0x00	Capture/Compare 3 interrupt flag Refer to CC1IF description.
2	CC2IF	rc_w0	0x00	Capture/Compare 2 interrupt flag Refer to CC1IF description.

Bit	Field	Type	Reset	Description
1	CC1IF	rc_w0	0x00	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. 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	0x00	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 the description of synchronization control register), if URS=0 and UDIS=0 in the TIMx_CR1 register.

12.4.6 Event generation register (TIMx_EGR)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bit	Field	Type	Reset	Description
15: 7	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
6	TG	w	0x00	Trigger generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.
5	Reserved			Reserved, always read as 0.
4	CC4G	w	0x00	Capture/Compare 4 generation Refer to CC1G description.
3	CC3G	w	0x00	Capture/Compare 3 generation Refer to CC1G description.
2	CC2G	w	0x00	Capture/Compare 2 generation Refer to CC1G description.
1	CC1G	w	0x00	Capture/Compare 1 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 1: If channel CC1 is configured as output: CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled. If channel CC1 is configured as input: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
0	UG	w	0x00	Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler factor is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), otherwise, it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

12.4.7 Capture/compare mode register 1(TIMx_CCMR1)

Offset address: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The

direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M			OC2PE	OC2FE	CC2S		OC1CE	OC1M			OC1PE	OC1FE	CC1S	
IC2F				IC2PSC				IC1F				IC1PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit	Field	Type	Reset	Description
15	OC2CE	rw	0x00	Output compare 2 clear enable
14: 12	OC2M	rw	0x00	Output compare 2 mode
11	OC2PE	rw	0x00	Output compare 2 preload enable
10	OC2FE	rw	0x00	Output compare 4 fast enable
9: 8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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	0x00	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

Bit	Field	Type	Reset	Description
6: 4	OC1M	rw	0x00	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 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 otherwise inactive. In downcounting, channel 1 is inactive (OC1REF=’0’) as long as TIMx_CNT>TIMx_CCR1 otherwise active (OC1REF=’1’). 111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1otherwise active. In downcounting, channel 1 is active as long as TIMx_CNT > TIMx_CCR1 otherwise 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). Note 2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from ”frozen” mode to ”PWM” mode.

Bit	Field	Type	Reset	Description
3	OC1PE	rw	0x00	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). Note 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	0x00	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	rw	0x00	Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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:

Bit	Field	Type	Reset	Description
15: 12	IC2F	rw	0x00	Input capture 2 filter
11: 10	IC2PSC	rw	0x00	Input capture 2 prescaler
9: 8	CC2S	rw	0x00	Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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: 4	IC1F	rw	0x00	Input capture 1 filter This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 1000: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 6 0001: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 2 1001: sampling frequency $f_{SAMPLING}=f_{DTS}/8$, N = 8 0010: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 4 1010: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 5 0011: sampling frequency $f_{SAMPLING}=f_{CK_INT}$, N = 8 1011: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 6 0100: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 6 1100: sampling frequency $f_{SAMPLING}=f_{DTS}/16$, N = 8 0101: sampling frequency $f_{SAMPLING}=f_{DTS}/2$, N = 8 1101: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 5 0110: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 6 1110: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 6 0111: sampling frequency $f_{SAMPLING}=f_{DTS}/4$, N = 8 1111: sampling frequency $f_{SAMPLING}=f_{DTS}/32$, N = 8

Bit	Field	Type	Reset	Description
3: 2	IC1PSC	rw	0x00	Input capture 1 prescaler This bit-field defines the factor 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	rw	0x00	Capture/compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the TS bit (TIMx_SMCR register) Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

12.4.8 Capture/compare mode register 2(TIMx_CCMR2)

Offset address: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M			OC4PE	OC4FE	CC4S		OC3CE	OC3M			OC3PE	OC3FE	CC3S	
IC4F				IC4PSC				IC3F				IC3PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit	Field	Type	Reset	Description
15	OC4CE	rw	0x00	Output compare 4 clear enable
14: 12	OC4M	rw	0x00	Output compare 4 mode
11	OC4PE	rw	0x00	Output compare 4 preload enable
10	OC4FE	rw	0x00	Output compare 4 fast enable

Bit	Field	Type	Reset	Description
9: 8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the TS bit (TIMx_SMCR register) Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER)
7	OC3CE	rw	0x00	Output compare 3 clear enable
6: 4	OC3M	rw	0x00	Output compare 3 mode
3	OC3PE	rw	0x00	Output compare 3 preload enable
2	OC3FE	rw	0x00	Output compare 3 fast enable
1: 0	CC3S	rw	0x00	Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the 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	Field	Type	Reset	Description
15: 12	IC4F	rw	0x00	Input capture 4 filter
11: 10	IC4PSC	rw	0x00	Input capture 4 prescaler

Bit	Field	Type	Reset	Description
9: 8	CC4S	rw	0x00	Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 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 the 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	0x00	Input capture 3 filter
3: 2	IC3PSC	rw	0x00	Input capture 3 prescaler
1: 0	CC3S	rw	0x00	Capture/compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the input pin. 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI4 10: CC3 channel is configured as input, IC3 is mapped on TI3 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 the TS bit (TIMx_SMCR register) Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

12.4.9 Capture/compare enable register(TIMx_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CC4P	CC4E	Res.		CC3P	CC3E	Res.		CC2P	CC2E	Res.		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Bit	Field	Type	Reset	Description
15: 14	Reserved			Reserved, always read as 0.
13	CC4P	rw	0x00	Capture/Compare 4 output polarity Refer to CC1P description.
12	CC4E	rw	0x00	Capture/Compare 4 output enable Refer to CC1E description.

Bit	Field	Type	Reset	Description
11: 10	Reserved			Reserved, always read as 0.
9	CC3P	rw	0x00	Capture/Compare 3 output polarity Refer to CC1P description.
8	CC3E	rw	0x00	Capture/Compare 3 output enable Refer to CC1E description.
7: 6	Reserved			Reserved, always read as 0.
5	CC2P	rw	0x00	Capture/Compare 2 output polarity Refer to CC1P description.
4	CC2E	rw	0x00	Capture/Compare 2 output enable Refer to CC1E description.
3: 2	Reserved			Reserved, always read as 0.
1	CC1P	rw	0x00	Capture/Compare 1 output polarity CC1 channel is configured as output: 0: OC1 active high 1: OC1 active low CC1 channel is configured as input: This bit selects whether IC1 or inverted IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted Note: This bit can not be modified as long as LOCK level 3 or 2 has been programmed (LOCK bits in TIMx_BDTR register).
0	CC1E	rw	0x00	Capture/Compare 1 output enable CC1 channel is 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 is configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not. 0: Capture disabled. 1: Capture enabled.

Table 55. Output Control Bit for Standard OCx Channels

CCxE bit	OCx output state
0	Output Disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

12.4.10 Counter(TIMx_CNT)

Offset address: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CNT	rw	0x0000	Counter value

12.4.11 Prescaler(TIMx_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	PSC	rw	0x0000	Prescaler value The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} /(PSC + 1). PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in ”reset mode”)

12.4.12 Auto-reload register(TIMx_ARR)

Offset address: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	ARR	rw	0x0000	Prescaler value ARR is the value to be loaded in the actual auto-reload register. Refer to section 12.3.1 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

12.4.13 Capture/compare register 1(TIMx_CCR1)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR1	rw	0x0000	Capture/Compare 1 value If CC1 channel is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Otherwise 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 CC1 channel is configured as input: CCR1 contains the counter value transferred by the last input capture 1 event (IC1).

12.4.14 Capture/compare register2(TIMx_CCR2)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR2	rw	0x0000	Capture/Compare 2 value If CC2 channel is configured as output: CCR2 contains the value to be loaded in the actual capture/compare 2 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Otherwise the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output. If CC2 channel is configured as input: CCR2 contains the counter value transferred by the last input capture 2 event (IC2).

12.4.15 Capture/compare register 3(TIMx_CCR3)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR3	rw	0x0000	Capture/Compare 3 value If CC3 channel is configured as output: CCR3 contains the value to be loaded in the actual capture/compare 3 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Otherwise the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output. If CC3 channel is configured as input: CCR3 contains the counter value transferred by the last input capture 3 event (IC3).

12.4.16 Capture/compare register 4(TIMx_CCR4)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	CCR4	rw	0x0000	Capture/Compare 4 value If CC4 channel is configured as output: CCR4 contains the value to be loaded in the actual capture/compare 4 register (preload value). The written value is transferred to the current register immediately if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Otherwise 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 signaled on OC4 output. If CC4 channel is configured as input: CCR4 contains the counter value transferred by the last input capture 4 event (IC4).

12.4.17 DMA control register(TIMx_DCR)

Offset address: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			DBL					Res.			DBA				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 13	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
12: 8	DBL	w	0x00	DMA burst length This bit field defines the burst transfer in the continuous mode (the timer detects a burst transfer when a write access to the TIMx_DMAR register address is performed), namely, the number of transfers, in half-word (double bytes) or bytes. 00000: 1 transfer 00001: 2 transfers 00010: 3 transfers .... 10001: 18 transfers Example: Let us consider the following transfer: DBL = 7 and DBA = TIM2_CR1. - If DBL =7 and DBA = TIM2_CR1 represent the address of data to be transferred, the transfer address is given by: (Address of TIMx_CR1) + DBA + (DMA index), where, DMA index = DBL TIMx_CR1 address + DBA + 7 is the address of data to be written or read, so that the transfer is completed to/from 7 registers starting from the TIMx_CR1 address + DBA. According to the setting of DMA data length, the following case may occur: -If the data is set to half word (16 bits), the data will be transferred to all 7 registers. -If the data is set to bytes, the data will still be transferred to all 7 registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore, the user must specify the data width of DMA transfer for the timer.
7: 5	Reserved			Reserved, always read as 0.
4: 0	DBA	w	0x00	DMA base address These bits define the base-address for DMA transfers in the continuous mode (when write access is done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register. 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR .....

12.4.18 DMA address for full transfer(TIMx_DMAR)

Offset address: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15: 0	DMAB	w	0x0000	DMA register for burst accesses A write operation to the TIMx_DMAR register will access the register located at the following address: TIMx_CR1 address + DBA + DMA index, Where: 'TIMx_CR1 address' is the address of the control register 1; 'DBA' is the DMA base address configured in TIMx_DCR register; 'DMA index' is the offset automatically controlled by the DMA transfer, depending on DBL configured in TIMx_DCR.

13 Real-time clock (RTC)

Real-time clock (RTC)

13.1 RTC introduction

The real-time clock is an independent timer. The RTC provides a set of continuously running counters which can be used, with suitable software, to provide a clock-calendar function. The counter values can be written to set the current time/date of the system.

The RTC core and clock configuration (RCC_BDCR register) are in the Backup domain, which means that RTC setting and time are kept after reset or wakeup from Standby mode.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- Enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
- Set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

13.2 Main features

- Programmable prescaler: division factor up to 2^{20}
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for the APB1 interface and RTC clock (must be at least four times slower than the PCLK1 clock)
- The RTC clock source could be any of the following ones:
  - HSE clock divided by 128
  - LSE oscillator clock
  - LSI oscillator clock
- Two separate reset types:
  - The APB1 interface is reset by system reset
  - The RTC Core (Prescaler, Alarm, Counter and Divider) is reset only by a Backup domain reset
- Three dedicated maskable interrupt lines:
  - Alarm interrupt, for generating a software programmable alarm interrupt.
  - Seconds interrupt, for generating a periodic interrupt signal with a programmable period length (up to 1 second).
  - Overflow interrupt, to detect when the internal programmable counter rolls over to zero.

13.3 Functional description

13.3.1 Overview

The RTC consists of two main units (see the following figure). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode. The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC Core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC prescaler block, which generates the RTC time base TR_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC Prescaler). Every TR_CLK period, the RTC generates an interrupt (Second Interrupt) if it is enabled in the RTC_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR_CLK rate and compared with a programmable date (stored in the RTC_ALR register) in order to generate an alarm interrupt, if enabled in the RTC_CR control register.

The following is the simplified RTC block diagram

Figure 122. RTC block diagram

13.3.2 Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV.

The RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV registers are reset only by a Backup Domain reset.

13.3.3 Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers.

This can occur if:

- A system reset or power reset has occurred
- The system has just woken up from Standby mode
- The system has just woken up from Stop mode

In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).

Consequently, when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC_CRL register to be set by hardware.

Note: The RTC APB1 interface is not affected by WFI and WFE low-power modes.

13.3.4 Configuring RTC registers

To write in the RTC_PRL, RTC_CNT, and RTC_ALR registers, the peripheral must enter Configuration Mode. This is done by setting the CNF bit in the RTC_CRL register.

In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is ’1’.

Configuration procedure:

- Poll RTOFF, wait until its value goes to ’1’
- Set the CNF bit to enter configuration mode
- Write to one or more RTC registers
- Clear the CNF bit to exit configuration mode
- Poll RTOFF, wait until its value goes to ’1’ to check the end of the write operation
- The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete.

13.3.5 RTC flag assertion

The RTC Second flag (SECF) is asserted on each RTC Core clock cycle before the update of the RTC Counter.

The RTC Overflow flag (OWF) is asserted on the last RTC Core clock cycle before the counter reaches 0x0000.

The RTC_Alarm and RTC Alarm flag (ALRF) are asserted on the last RTC Core clock cycle before the counter reaches the RTC Alarm value stored in the Alarm register increased by one (RTC_ALR + 1). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

- Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated.
- Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

Figure 123. RTC second and alarm waveform example with PR=0003, ALARM=00004

Figure 124. RTC Overflow waveform example with PR=0003

13.4 RTC registers

Table 56. Overview of RTC Registers

Offset	Acronym	Register Name	Reset	Section
0x00	RTC_CRH	RTC control register high	0x00000000	section 13.4.1
0x04	RTC_CRL	RTC control register low	0x00000020	section 13.4.2
0x08	RTC_PRLH	RTC prescaler load register high	0x00000000	section 13.4.3
0x0C	RTC_PRLL	RTC prescaler load register low	0x00000000	section 13.4.3
0x10	RTC_DIVH	RTC prescaler divider register high	0x00000000	section 13.4.4

Offset	Acronym	Register Name	Reset	Section
0x14	RTC_DIVL	RTC prescaler divider register low	0x00000000	section 13.4.4
0x18	RTC_CNTH	RTC counter register high	0x00000000	section 13.4.5
0x1C	RTC_CNTL	RTC counter register low	0x00000000	section 13.4.5
0x20	RTC_ALRH	RTC alarm register high	0x0000FFFF	section 13.4.6
0x24	RTC_ALRL	RTC alarm register low	0x0000FFFF	section 13.4.6

13.4.1 RTC control register high(RTC_CRH)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													OWIE	ALRIE	SECIE
													rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 3	Reserved			Always read as 0.
2	OWIE	rw	0x00	Overflow interrupt enable 0: Overflow interrupt is masked (disabled) 1: Overflow interrupt is enabled
1	ALRIE	rw	0x00	Alarm interrupt enable 0: Alarm interrupt is masked (disabled) 1: Alarm interrupt is enabled
0	SECIE	rw	0x00	Second interrupt enable 0: Second interrupt is masked (disabled) 1: Second interrupt is enabled

These bits are used to mask interrupt requests.

Note: At reset, all interrupts are disabled, so it is possible to write to the RTC registers to ensure that no interrupt requests are pending after initialization. It is not possible to write to the RTC_CRH register when the peripheral is completing a previous write operation (FTOFF bit = 0).

The RTC functions are controlled by this control register. Some bits must be written using a specific configuration procedure.

13.4.2 RTC control register low(RTC_CRL)

Address offset: 0x04

Reset value: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bit	Field	Type	Reset	Description
15 : 6	Reserved			Always read as 0.
5	RTOFF	r	0x01	RTC operation OFF With this bit the RTC reports the status of the last write operation performed on its registers, indicating if it has been completed or not. If its value is '0' then it is not possible to write to any of the RTC registers. This bit is read only. 0: Last write operation on RTC registers is still ongoing 1: Last write operation on RTC registers terminated
4	CNF	rw	0x00	Configuration flag This bit must be set by software to enter in configuration mode so as to allow new values to be written in the RTC_CNTL/H, RTC_ALRL/H or RTC_PRLL/H registers. The write operation is only executed when the CNF bit is reset by software after it has been set. 0: Exit configuration mode (start update of RTC registers) 1: Enter configuration mode
3	RSF	rc_w0	0x00	Registers synchronized flag This bit is set by hardware at each time the RTC_CNT and RTC_DIV registers are updated and cleared by software. Before any read operation after an APB1 reset or an APB1 clock stop, this bit must be cleared by software, and the user application must wait until it is set to be sure that the RTC_CNT, RTC_ALR or RTC_PRL registers are synchronized. 0: Registers not yet synchronized. 1: Registers synchronized
2	OWF	rc_w0	0x00	Overflow flag This bit is set by hardware when the 32-bit programmable counter overflows. An interrupt is generated if OWIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect. 0: Overflow not detected 1: 32-bit programmable counter overflow occurred
1	ALRF	rc_w0	0x00	Alarm flag This bit is set by hardware when the 32-bit programmable counter reaches the threshold set in the RTC_ALR register. An interrupt is generated if ALRIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect. 0: Alarm not detected 1: Alarm detected

Bit	Field	Type	Reset	Description
0	SECF	rc_w0	0x00	Second flag This bit is set by hardware when the 32-bit programmable prescaler overflows, thus incrementing the RTC counter. Hence this flag provides a periodic signal with a period corresponding to the resolution programmed for the RTC counter (usually one second). An interrupt is generated if SECIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect. 0: Second flag condition not met 1: Second flag condition met

The functions of the RTC are controlled by this control register. It is not possible to write to the RTC_CR register while the peripheral is completing a previous write operation (RTOFF = 0).

Note: 1. Any flag remains pending until the appropriate RTC_CR request bit is reset by software, indicating that the interrupt request has been granted. 2. At reset, the interrupts are disabled, no interrupt requests are pending and it is possible to write to the RTC registers. 3. The OWF, ALRF, SECF and RSF bits are not updated when the APB1 clock is not running. 4. The OWF, ALRF, SECF and RSF bits can only be set by hardware and only cleared by software. 5. If ALRF = 1 and ALRIE = 1, the RTC global interrupt is enabled. If EXTI Line 17 is also enabled through the EXTI Controller, both the RTC global interrupt and the RTC Alarm interrupt are enabled. 6. If ALRF = 1, the RTC Alarm interrupt is enabled if EXTI Line 17 is enabled through the EXTI Controller in interrupt mode. When the EXTI Line 17 is enabled in event mode, a pulse is generated on this line (no RTC Alarm interrupt generation).

13.4.3 RTC prescaler load register(RTC_PRLH/RTC_PRLL)

RTC prescaler load register high(RTC_PRLH)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PRL			
												w	w	w	w

Bit	Field	Type	Reset	Description
15 : 4	Reserved			Always read as 0.
3 : 0	PRL	w	0x00	RTC prescaler reload value high These bits are used to define the counter clock frequency according to the following formula: $f_{TR_CLK} = \frac{f_{RTCCLK}}{PRL+1}$ Caution: The zero value is not recommended. RTC interrupts and flags cannot be asserted correctly.

RTC prescaler load register low(RTC_PRLL)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15 : 0	PRL	w	0x0000	RTC prescaler reload value low These bits are used to define the counter clock frequency according to the following formula: $f_{TR_CLK} = \frac{f_{RTCCLK}}{PRL+1}$

Note: If the input clock frequency (f_{rtcclk}) is 32.768 kHz, write 7FFFh in this register to get a signal period of 1 second.

13.4.4 RTC prescaler divider register(RTC_DIVH/RTC_DIVL)

During each period of TR_CLK, the counter inside the RTC prescaler is reloaded with the value stored in the RTC_PRL register. To get an accurate time measurement it is possible to read the current value of the prescaler counter, stored in the RTC_DIV register, without stopping it. This register is read-only and it is reloaded by hardware after any change in the RTC_PRL or RTC_CNT registers.

RTC prescaler divider register high(RTC_DIVH)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												DIV			
												r	r	r	r

Bit	Field	Type	Reset	Description
15 : 4	Reserved			Always read as 0.
3 : 0	DIV	r	0x00	RTC clock divider high

RTC prescaler divider register low(RTC_DIVL)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 0	DIV	r	0x0000	RTC clock divider low

13.4.5 RTC counter register(RTC_CNTH/RTC_CNTL)

The RTC core has one 32-bit programmable counter, accessed through two 16-bit registers; the count rate is based on the TR_CLK time reference, generated by the prescaler. RTC_CNT registers keep the counting value of this counter. They are write-protected by bit RTOFF in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'. A write operation on the upper (RTC_CNTH) or lower (RTC_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC Prescaler. When reading, the current value in the counter (system date) is returned.

RTC counter register high(RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0000	RTC counter high Reading the RTC_CNTH register, the current value of the high part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode.

RTC counter register low(RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0000	RTC counter low Reading the RTC_CNTL register, the current value of the lower part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode.

13.4.6 RTC alarm register high(RTC_ALRH/RTC_ALRL)

When the programmable counter reaches the 32-bit value stored in the RTC_ALR register, an alarm is triggered and the RTC_alarm interrupt request is generated. This register is write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'.

RTC alarm register high(RTC_ALRH)

Address offset: 0x20

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15 : 0	ALR	w	0xFFFF	RTC alarm high The high part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode.

RTC alarm register low(RTC_ALRL)

Address offset: 0x24

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
15 : 0	ALR	w	0xFFFF	RTC counter low The low part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode.

14 Independent watchdog (IWDG)

Independent watchdog (IWDG)

14.1 (IWDG introduction)

The devices have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window.

14.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

14.3 Functional description

The following figure shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Figure 125. Independent Watchdog Block Diagram

Note: The watchdog function is implemented in the V_{DD} voltage domain, still functional in Stop and Standby modes.

Table 57. Min/max IWDG Timeout Period (in ms) at 40 kHz (LSI)

Prescaler divider	PR [2:0] bits	Min timeout RL[11:0] = 0x000	Max timeout
/4	0	0.1	409.6
/8	1	0.2	819.2
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 or 7)	6.4	26214.4

Note: These timings are given for a 40 kHz clock but the microcontroller internal RC frequency can vary between 30 kHz –60 kHz.
In addition, even if the frequency of the oscillator is accurate, the exact timing still depends on the phase difference between the APB interface clock and the oscillator clock.
As a result, there will always be a complete oscillator period that is uncertain.

14.3.1 Hardware watchdog

If the user activates the 'Hardware Watchdog' function in the select bit (refer to the section "Embedded Flash"), the watchdog will automatically run after the system power-on reset; if the software does not write the corresponding value to the key register before the counter ends counting, the system reset will occur.

14.3.2 Register access protection

The IWDG_PR and IWDG_RLR registers are write-protected. To modify the values of these two registers, you must first write 0x5555 to the IWDG_KR register. Writing to this register with a different value will disrupt the sequence and the registers will be reprotected. The reload operation (i.e. writing 0xAAAA) will also activate the write protection.

The status register indicates whether the prescaler value and the downcounter are being updated.

14.3.3 Debug mode

When the microcontroller enters debug mode (CPU core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to sections of debug modules.

14.4 IWDG register description

Table 58. Overview of IWDG Registers

Offset	Acronym	Register Name	Reset	Section
0x00	IWDG_KR	Key register	0x00000000	section 14.4.1
0x04	IWDG_PR	Prescaler register	0x00000000	section 14.4.2
0x08	IWDG_RLR	Reload register	0x00000FFF	section 14.4.3
0x0C	IWDG_SR	Status register	0x00000000	section 14.4.4

14.4.1 Key register(IWDG_KR)

Offset address: 0x00

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 16	Reserved			Always read as 0.
15 : 0	KEY	w	0x0000	Key value (write only, read 0x0000) These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0. Writing the key value 0x5555 to enable access to the IWDG_PR and IWDG_RLR registers. Writing the key value 0xCCCC starts the watchdog.

14.4.2 Prescaler register(IWDG_PR)

Offset address: 0x04

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PR		
													rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 3	Reserved			Always read as 0.
2 : 0	PR	rw	0x00	Prescaler divider These bits are write access protected. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider. 000: divider / 4 100: divider / 64 001: divider / 8 101: divider / 128 010: divider / 16 110: divider / 256 011: divider / 32 111: divider / 256 Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason, the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

14.4.3 Reload register(IWDG_RLR)

Offset address: 0x08

Reset value: 0x0000 0FFF(reset by Standby mode)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RL											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 12	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
11 : 0	RL	rw	0xFFF	Watchdog counter reload value These bits are write access protected. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason, the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

14.4.4 Status register(IWDG_SR)

Offset address: 0x0C

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RVU	PVU
														r	r

Bit	Field	Type	Reset	Description
31 : 2	Reserved			Always read as 0.
1	RVU	r	0x00	Watchdog counter reload value update This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles). Reload value can be updated only when RVU bit is reset.
0	PVU	r	0x00	Watchdog prescaler value update This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles). Prescaler value can be updated only when PVU bit is reset.

Note: If several reload values or prescaler values are used by application, it is mandatory to wait

until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete).

15 Window watchdog (WWDG)

Window watchdog (WWDG)

15.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

15.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset:
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40. It is used to reload the counter, thus preventing WWDG reset.

15.3 Functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 126. Watchdog Block Diagram

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

- Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

- Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register.

The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. The above figure describes the window watchdog process.

Another way to reload the counter is to use the early wakeup interrupt (EWI). This interrupt is enabled by setting the WEI bit in the WWDG_CFR register. It is generated when the downcounter reaches 0x40, and the corresponding interrupt service routine (ISR) can be used to load counters, so as to prevent WWDG reset. The interrupt can be cleared by writing a ’0’ to the WWDG_SR register.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

15.4 How to program the watchdog timeout

The figure below shows the linear relationship (in mS) between the 6-bit count value loaded into the Watchdog Counter (CNT) and the watchdog delay time. This figure can be used as a reference for rapid calculation without taking into account time deviations. If higher precision is required, you can use the calculation formula provided in the figure below.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

The formula to calculate the WWDG timeout value is given by:

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{WDGTB} \times (T[5:0]+1) \quad \text{(mS)}$$

Where:

T_{WWDG}: WWDG timeout

T_{PCLK1}: APB1 clock period measured in ms

Minimum and maximum timeout values at PCLK1=36 MHz

WDGTB	Min. timeout value	Max. timeout value
0	**113µS**	**7.28mS**
1	**227µS**	**14.56mS**
2	**455µS**	**29.12mS**
3	**910µS**	**58.25mS**

399420

Figure 127. Window Watchdog Timing Diagram

15.5 Debug mode

When the microcontroller enters debug mode (CPU core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to sections of debug modules.

15.6 WWDG register description

Table 59. Overview of WWDG Registers

Offset	Acronym	Register Name	Reset	Section
0x00	WWDG_CR	Control register	0x0000007F	section 15.6.1
0x04	WWDG_CFGR	Configuration register	0x0000007F	section 15.6.2
0x08	WWDG_SR	Status register	0x00000000	section 15.6.3

15.6.1 Control register(WWDG_CR)

Offset address: 0x00

Reset value: 0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T						
								rw	rw						

Bit	Field	Type	Reset	Description
31:8	Reserved			Reserved, always read as 0.
7	WDGA	rw	0x00	Activation bit This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset. 0: Watchdog disabled 1: Watchdog enabled
6:0	T	rw	0x7F	7 - bit counter These bits contain the value of the watchdog counter. It is decremented every(4096x2^{WDGTB})PCLK1 cycles.A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

15.6.2 Configuration register(WWDG_CFGR)

Offset address: 0x04

Reset value: 0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB		W						
						rw	rw		rw						

Bit	Field	Type	Reset	Description
31:10	Reserved			Reserved, always read as 0.
9	EWI	rw	0x00	Early wakeup interrupt When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.
8:7	WDGTB	rw	0x00	Timer base The time base of the prescaler can be modified as follows: 00: CK Counter Clock (PCLK1 div 4096) div1 01: CK Counter Clock (PCLK1 div 4096) div2 10: CK Counter Clock (PCLK1 div 4096) div4 11: CK Counter Clock (PCLK1 div 4096) div8
6:0	WINDOW	rw	0x7F	7-bit window value These bits contain the window value to be compared to the downcounter.

15.6.3 Status register(WWDG_SR)

Offset address: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

Bit	Field	Type	Reset	Description
31:1	Reserved			Reserved, always read as 0.
0	EWIF	rc_w0	0x00	Early wakeup interrupt flag This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. A write of ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

16 Universal serial bus full-speed device interface (USB)

Universal serial bus full-speed device interface (USB)

16.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

16.2 USB main features

- USB specification version 2.0 full-speed compliant
- Support full-speed 12M mode
- One control transfer endpoint and four independent universal endpoints for interrupt transfer and bulk transfer.
- Up to 64 bytes of packets for control, bulk and interrupt transfer.
- Cyclic redundancy check (CRC) generation/checking, non-return-to-zero (NRZI) inverted coding/decoding and bit stuffing.
- USB suspend/resume operations
- DMA transfer

The following figure is a block diagram of the USB peripheral:

Figure 128. USB Block Diagram

16.3 Functional description

The USB peripheral provides an USB compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. The USB peripheral interfaces with the host PC, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a 64-bit buffer description block, and the buffer is located in the USB module and cannot be directly accessed by CPU. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data buffered by the USB peripheral is loaded in an internal register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt isgenerated, reading status registers and/or using different interrupt handling routines. The microcontroller can determine:

- Which type of transaction requested by the host
- Which endpoint has to be served
- What type of transmission is in process
- Endpoint response
- Whether the transmission is completed

The USB peripheral can be placed in low-power mode (SUSPEND mode) by writing the USB_POWER register whenever the peripheral is not required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wakeup line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

16.3.1 Description of USB blocks

The USB module contains an 8-bit wide 320-byte FIFO, with a 64-byte FIFO at endpoint. On the APB1 bus, each register uses the lower 8 bits of the 32-bit wide bus. In addition, data is stored through the lower 8 bits.

The USB module contains several registers:

- USB control register: used to configure USB parameters and query status
- Endpoint status register
- Endpoint configuration register
- Setup data register: used to store the data of the SETUP packet
- Endpoint data control register: used to control the data flow

The APB1 bus or DMA writes data or reads data to the data port, and the number of FIFO data increases or decreases accordingly.

The FIFO pointer changes only after the endpoint has successfully received and sent data. Each endpoint is configured with a FIFO data register that serves as the entry address for writing or reading FIFO. Each endpoint is also provided with a dedicated register, to query the current number of valid data in the FIFO.

Only Endpoint 1 and Endpoint 2 support DMA transfer.

16.4 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to simplify application software development.

16.4.1 Overview of USB transmission

The following describes the relationship between packets, things, and transfer.

Packet is the basic unit of information transmission in a USB system. All data is transmitted on the bus after being packaged.

The basic USB packets are shown below, such as token packets, data packets, and handshake packets.

Figure 129. Basic Format of Packet

The process of receiving or transmitting data information on a USB is called a transaction. Types of transaction processing include input (IN) transaction processing, output (OUT) transaction processing, setup (SETUP) transaction processing, etc.

Figure 130. USB Transaction

The following figure depicts the complete transfer process of an endpoint. For bulk/control transfers, a transfer must begin after the last transfer.

Figure 131. USB Transfer

When the USB controller starts working, the USB is in the IDLE state, and then waits for the bus command. After a bus command is received, such as reset, setup, etc., an interrupt is generated and the CPU queries the command type. For example, if it is a reset command, the CPU clears and resets all programmable states. If it is a transmission request command, the CPU will write/read the data in the USB controller FIFO. For bulk transfer or control transfer of input, when data is prepared in CPU or DMA, the CPU will send an ACK handshake signal or STALL handshake signal, to end the transfer. For the bulk transfer of output, the USB automatically sends an ACK signal when the data is transmitted to the corresponding FIFO.

When the data size exceeds the maximum packet size during transmission, the data is divided into more than one packets for transmission, otherwise, only one packet will be transmitted.

16.4.2 USB enumeration

The host sends various requests to the device through the Endpoint 0 in the manner of controlling the transmission. After receiving the request from the host, the device restores the corresponding information and performs enumeration.

After the system power-on and reset, configure the USB controller first:

1. Set the endpoint enable bit
2. Enable the reset and endpoint interrupt
3. Set the connect bit (CONNECT bit of the USB_TOP register)

Figure 132. Enumeration Process

When Endpoint 0 receives the SETUP packet, the system will read the set 8 bytes of data, to determine the next step. For the SetAddress descriptor, the USB controller will automatically load the new address.

The analysis in the first cycle during enumeration covers:

The device is detected and the host sends a bus reset, which is different from USB power-on reset and system reset and is a reset that the SIE notifies the user according to the bus status. The device generates a reset interrupt and the device firmware program determines the handling method.

The host enables the first control transfer:

1. Host SETUP packet (to Address 0 and Endpoint 0), host packet (request device descriptor), device handshake packet ACK.
   The device generates an data output interrupt of Endpoint 0. The firmware program is prepared according to the host requirements in the packet, namely, the device descriptor is prepared in the input buffer of Endpoint 0.
2. Data process: the host sends an IN token packet first, and the device sends a data packet (this data is ready, the SIE receives the IN token and sends it directly to the bus; the user shall not intervene at this time), and the host sends an ACK packet.
   At this point, the SIE generates an data input interrupt of Endpoint 0, indicating that the host has got the data prepared by the device, and the user can also perform relevant operations in the interrupt handling routine (SIE refers to the serial interface engine, which is the 'core' of USB controllers. The SIE handles the underlying protocols such as bit stuffing, CRC generation and checking, and enables issuing error reports. Its main task is to convert low-level signals into bytes for the controller's application).
   At the moment, the host accepts only one data, at least 8 bytes. If the user data is not sent, and is stored in the control endpoint input buffer, the host will ignore it even if it has been prepared.
3. Status process: The host sends an OUT packet (notifying the device to output) and 0 byte status packet (this is 0 bytes, indicating that it received the device descriptor), and the device sends a handshake ACK packet.
   At this time, the device will not generate a data output interrupt of Endpoint 0, and there is no data.

During the enumeration, the address is set in the second cycle.

After the first cycle, the host resets the bus again, to enter the address setting control transfer phase.

1. Host SETUP packet (to address 0 and Endpoint 0), host packet (request setting address), device handshake packet ACK. So the SETUP packet will be followed by a data packet (either GET or SET) indicating the purpose of the host SETUP.
   The device generates a data output interrupt of Endpoint 0. The firmware program is prepared according to the host requirements in the packet, and the address control register is written according to the address sent by the host.
2. Data process: No data is transmitted.
3. Status process: The host sends an IN packet (notifying the device to return data), the device sends a 0 byte status packet (indicating that the address has been set successfully), and the host sends a handshake ACK packet (the address setting has taken effect).
   At this time, the device will not generate a data input interrupt of Endpoint 0, and there is no data.

During the enumeration, the host uses the new address to get the full device descriptor in the third cycle.

The host enables the first control transfer with the new address:

1. Host SETUP packet (to new Address endpoint 0), host packet (request device descriptor), device handshake packet ACK.
   The device generates an data output interrupt of Endpoint 0. The firmware program is prepared according to the host requirements in the packet, namely, the device descriptor is prepared in the input buffer of Endpoint 0.
2. Data process: the host sends an IN token packet first, and the device sends a data packet (this data is ready, the SIE receives the IN token and sends it directly to the bus; the user shall not intervene at this time), and the host sends an ACK packet.
   At this time the SIE generates a data input interrupt of Endpoint 0, indicating that the host has got the data prepared by the device. The user can perform the following operations in the interrupt handling routine: if the descriptor is not sent once, the remaining content is transmitted to the input buffer of Endpoint 0.
   The second data transmission: the host sends another IN token packet, the device sends a data packet, and the host sends an ACK packet.
   At the moment, SIE generates a data input interrupt of Endpoint 0 if the data has been sent (not processed here), to enter the status process.
3. Status process: The host sends an OUT packet (notifying the device to output) and 0 byte status packet (indicating that it received the device descriptor), and the device sends a handshake ACK packet.

16.4.3 USB transfer handling

The system needs to set up an inoperative loop, to wait for a request from the USB host, which sets an interrupt bit, and the system jumps out of the loop by constantly querying the interrupt bit or entering the interrupt service routine. When the system detects the interrupt bit, it jumps to the corresponding subroutine for processing.

The following figure is a flow chart of a typical USB transfer:

Figure 133. USB Transfer Flowchart

16.4.4 IN token packet

When receiving an IN request from a host, if a NACK response is received, the USB host resends the IN request until the endpoint confirms that the request is valid. If the received address matches a configured endpoint address, follow these steps:

1. If the data in the FIFO is less than the size set in register EPX_CTRL6:0, or EPX_CTRL7

is not set, the USB module will automatically send a NACK until the data is ready.

2. The CPU receives the IN_NACK status.
3. The CPU writes the data to the FIFO.
4. The CPU sets the size of the data to be sent and the transmit enable in the EPX_CTRL register.
5. The USB module automatically sends the data in the FIFO when receiving the next IN request. After the last data byte is sent, the calculated CRC is automatically transmitted.
6. The CPU receives the IN_ACK status and the END status after the transmission.
7. If the endpoint is not enabled, or if the EP_HALT register configuration is paused, the USB module will acknowledge IN_STALL without sending data.

16.4.5 OUT token packet

When receiving an OUT request from a host, if a NACK response is received, the USB host resends the OUT request until the endpoint confirms that the request is valid. If the received address matches a configured endpoint address, follow these steps:

1. If the space in the FIFO is less than the size of the packet sent by the host, the USB module will automatically send a NACK until the CPU fetches the data from the FIFO.
2. The CPU receives the OUT_NACK status.
3. The CPU reads data from the FIFO.
4. If the endpoint is not enabled, or if the EP_HALT register configuration is paused, the USB module will acknowledge OUT_STALL without sending data.

16.5 USB register description

Table 60. USB Register Overview

Offset	Acronym	Register Name	Reset	Section
0x00	USB_TOP	USB TOP register	0x00000002	section 16.5.1
0x04	USB_INT_STATE	USB interrupt status register	0x00000000	section 16.5.2
0x08	EP_INT_STATE	USB endpoint interrupt status register	0x00000000	section 16.5.3
0x0C	EP0_INT_STATE	USB endpoint 0 interrupt status register	0x00000000	section 16.5.4
0x10	USB_INT_EN	USB interrupt enable register	0x00000000	section 16.5.5
0x14	EP_INT_EN	USB endpoint interrupt enable register	0x00000000	section 16.5.6
0x18	EP0_INT_EN	USB endpoint 0 interrupt enable register	0x00000000	section 16.5.7
0x20~ 0x2C	EPX_INT_STATE	USB endpoint X interrupt status register	0x00000000	section 16.5.8
0x40~ 0x4C	EPX_INT_EN	USB endpoint X interrupt enable register	0x00000000	section 16.5.9
0x60	USB_ADDR	USB address register	0x00000000	section 16.5.10
0x64	EP_EN	USB endpoint enable register	0x00000000	section 16.5.11
0x78	TOG_CTRL1_4	USB data toggle control register	0x00000000	section 16.5.12
0x80 ~ 0x9C	SETUPX	USB transfer packet data register	0x00000000	section 16.5.13
0xA0, 0xA4	PACKET_SIZEX	USB transfer packet size register	0x00000040	section 16.5.14

Offset		Acronym	Register Name	Reset	Section
0x100 0x110	~	EPX_AVAIL	USB endpoint X valid data register	0x00000000	section 16.5.15
0x140 0x150	~	EPX_CTRL	USB endpoint X control register	0x00000000	section 16.5.16
0x160 0x170	~	EPX_FIFO	USB endpoint X FIFO register	0x00000000	section 16.5.17
0x184		EP_DMA	USB endpoint DMA enable register	0x00000000	section 16.5.18
0x188		EP_HALT	USB endpoint halt register	0x00000000	section 16.5.19
0x1C0		USB_POWER	USB power control register	0x00000003	section 16.5.20

16.5.1 USB TOP register(USB_TOP)

Offset address: 0x00

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ACTIVE	DP/DM STATE		SUSPEND	RESET	Res.	CONNECT	SPEED
								rw	r	r	r	rw		rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	ACTIVE	rw	0x00	USB bus is active 0：USB bus is inactive 1：USB bus is active
6 : 5	DP/DM STATE	r	0x00	Current USB DP/DM line state
4	SUSPEND	r	0x00	USB suspend state This bit monitors the controller status regardless of the APB_POWER register. 0: The controller is in an operating state 1: The controller is in a suspending state
3	RESET	rw	0x00	Reset EP and FIFO in USB controller 0: Not reset 1: Reset Note that this bit needs to be cleared by software after being set.
2	Reserved			Always read as 0.
1	CONNECT	rw	0x01	USB connection 0: Disconnected 1: Connected

Bit	Field	Type	Reset	Description
0	SPEED	rw	0x00	Set USB speed 0: Full speed transmission 1: Low speed transmission

16.5.2 USB interrupt status register(USB_INT_STATE)

Offset address: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP INTF	SOFF	RESUMF	SUS PENDF	RSTF
											r	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
15 : 5	Reserved			Always read as 0.
4	EPINTF	r	0x00	EP interrupt received This bit is set to ’1’ when any endpoint generates an interrupt. Refer to EP_INT_STATE description for details. This bit is read-only.
3	SOFF	rc_w1	0x00	SOF detection flag bit (BUS received) This bit is set to ’1’ when the SOF is detected, and can be cleared by writing ’1’.
2	RESUMF	rc_w1	0x00	Wake-up flag bit (BUS resume received) This bit is set to ’1’ when the USB bus is activated, and can be cleared by writing ’1’.
1	SUSPENDF	rc_w1	0x00	USB bus suspend flag bit (BUS suspend received) This bit is set to ’1’ when a suspend state on the bus is detected, and can be cleared by writing ’1’.
0	RSTF	rc_w1	0x00	USB bus reset request flag bit (BUS reset received) This bit is set to ’1’ when the reset signal input of the bus is detected, and can be cleared by writing ’1’.

16.5.3 USB endpoint interrupt status register(EP_INT_STATE)

Offset address: 0x08

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4F	EP3F	EP2F	EP1F	EP0F
											r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 7	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
4	EP4F	r	0x00	EP4 interrupt received
3	EP3F	r	0x00	EP3 interrupt received
2	EP2F	r	0x00	EP2 interrupt received
1	EP1F	r	0x00	EP1 interrupt received
0	EP0F	r	0x00	EP0 interrupt received

16.5.4 USB endpoint 0 interrupt status register(EP0_INT_STATE)

Offset address: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	SET UP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	OUT-STALLF	rc_w1	0x00	OUT packet acknowledge STALL identifier (OUT-STALL received) After the endpoint receives the OUT packet from the host, the controller acknowledges STALL. When the EP_HALT0 register is set to '1' and EP0_CTRL7 is enabled, the USB controller will acknowledge STALL. This bit is cleared by writing '1'.
6	OUT-ACKF	rc_w1	0x00	OUT packet acknowledge ACK identifier (OUT-ACK received) After Endpoint 0 receives the OUT packet from the host, the FIFO has enough space for the host to complete the current transfer. The USB controller automatically acknowledges the ACK. This bit is cleared by writing '1'.
5	OUT-NACKF	rc_w1	0x00	OUT packet acknowledge NACK identifier (OUT-NACK received) After the endpoint receives the OUT packet from the host, there is no enough space to store the data sent from the host. The USB controller automatically acknowledges the NACK. This bit is cleared by writing '1'.

Bit	Field	Type	Reset	Description
4	IN-STALLF	rc_w1	0x00	IN packet acknowledge STALL identifier (IN-STALL received) After the endpoint receives the IN packet from the host, the controller acknowledges STALL. When the EP_HALT0 register is set to '1' and EP0_CTRL7 is enabled, the USB controller will acknowledge STALL. This bit is cleared by writing '1'.
3	IN-ACKF	rc_w1	0x00	IN packet acknowledge ACK identifier (IN-ACK received) After Endpoint 0 receives the IN packet from the host, the FIFO has enough data for the host to complete the current transfer. The USB controller automatically acknowledges the ACK. This bit is cleared by writing '1'. After the endpoint receives the IN packet from the host, the host completes the current transfer and sets this bit.
2	IN-NACKF	rc_w1	0x00	IN packet acknowledge NACK identifier (IN-NACK received) After Endpoint 0 receives the IN packet from the host, there is no enough data to complete the current transfer. The USB controller automatically acknowledges the NACK. This bit is cleared by writing '1'.
1	END	rc_w1	0x00	Transfer finish flag (Status stage finished) This bit is set to '1' when the endpoint transfer is completed, and can be cleared by writing '1'.
0	SETUP	rc_w1	0x00	Received SETUP packet identifier (SETUP packet received) After this bit is set, the contents of the SETUP packet can be read from the register SETUP0 ~ 7. This bit is cleared by writing '1'.

16.5.5 USB interrupt enable register (USB_INT_EN)

Offset address: 0x10

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INT MASK	Reserved		EPIE	SOF IE	RESUM IE	SUSPE NDIE	RST IE
								rw			rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
7	INTMASK	rw	0x00	Interrupt mask bit 1: The interrupt flag bit in each interrupt register is controlled by the interrupt enable flag in the corresponding interrupt enable register 0: The interrupt flag bit in each interrupt register is not controlled by the interrupt enable flag in the corresponding interrupt enable register
6 : 5	Reserved			Always read as 0.
4	EPIE	rw	0x00	EP interrupt enable
3	SOFIE	rw	0x00	SOF interrupt enable
2	RESUMIE	rw	0x00	Wake-up interrupt enable bit (BUS resume interrupt enable)
1	SUSPENDIE	rw	0x00	BUS suspend interrupt enable
0	RSTIE	rw	0x00	BUS reset interrupt enable

16.5.6 USB endpoint interrupt enable register (EP_INT_EN)

Offset address: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4IE	EP3IE	EP2IE	EP1IE	EP0IE
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			Always read as 0.
4	EP4IE	rw	0x00	EP4 interrupt enable
3	EP3IE	rw	0x00	EP3 interrupt enable
2	EP2IE	rw	0x00	EP2 interrupt enable
1	EP1IE	rw	0x00	EP1 interrupt enable
0	EP0IE	rw	0x00	EP0 interrupt enable

16.5.7 USB endpoint 0 interrupt enable register(EP0_INT_EN)

Offset address: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	SET UPIE
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
7	OUT-STALLIE	rw	0x00	OUT-STALL interrupt enable
6	OUT-ACKIE	rw	0x00	OUT-ACK interrupt enable
5	OUT-NACKIE	rw	0x00	OUT-NACK interrupt enable)
4	IN-STALLIE	rw	0x00	IN-STALL interrupt enable
3	IN-ACKIE	rw	0x00	IN-ACK interrupt enable
2	IN-NACKIE	rw	0x00	IN-NACK interrupt enable
1	ENDIE	rw	0x00	Status stage finished interrupt enable
0	SETUPIE	rw	0x00	SETUP packet interrupt enable

16.5.8 USB endpoint X interrupt status register (EPX_INT_STATE) (X = 1 ~ 4)

Offset address: 0x20 ~ 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	Res.
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	OUT-STALLF	rc_w1	0x00	OUT packet acknowledge STALL identifier (OUT-STALL received) After the endpoint receives the OUT packet from the host, the controller acknowledges STALL. When the EP_HALTx register is set to '1' and EPx_CTRL7 is enabled, the USB controller will acknowledge STALL. This bit is cleared by writing '1'.
6	OUT-ACKF	rc_w1	0x00	OUT packet acknowledge ACK identifier (OUT-ACK received) After Endpoint x receives the OUT packet from the host, the FIFO has enough space for the host to complete the current transfer. The USB controller automatically acknowledges the ACK. This bit is cleared by writing '1'.

Bit	Field	Type	Reset	Description
5	OUT-NACKF	rc_w1	0x00	OUT packet acknowledge NACK identifier (OUT-NACK received) After the endpoint receives the OUT packet from the host, there is no enough space to store the data sent from the host. The USB controller automatically acknowledges the NACK. This bit is cleared by writing '1'.
4	IN-STALLF	rc_w1	0x00	IN packet acknowledge STALL identifier (IN-STALL received) After the endpoint receives the IN packet from the host, the controller acknowledges STALL. When the EP_HALTx register is set to '1' and EPx_CTRL7 is enabled, the USB controller will acknowledge STALL. This bit is cleared by writing '1'.
3	IN-ACKF	rc_w1	0x00	IN packet acknowledge ACK identifier (IN-ACK received) After Endpoint 0 receives the IN packet from the host, the FIFO has enough data for the host to complete the current transfer. The USB controller automatically acknowledges the ACK. This bit is cleared by writing '1'. After the endpoint receives the IN packet from the host, the host completes the current transfer and sets this bit.
2	IN-NACKF	rc_w1	0x00	IN packet acknowledge NACK identifier (IN-NACK received) After endpoint receives the IN packet from the host, there is no enough data to complete the current transfer. The USB controller automatically acknowledges the NACK. This bit is cleared by writing '1'.
1	END	rc_w1	0x00	Transfer finish flag (Status stage finished) This bit is set to '1' when the endpoint transfer is completed, and can be cleared by writing '1'.
0	Reserved			Always read as 0.

16.5.9 USB endpoint X interrupt status register(EPX_INT_EN) (X = 1 ~ 4)

Offset address: 0x40 ~ 0x4C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	OUT-STALLIE	rw	0x00	OUT-STALL interrupt enable
6	OUT-ACKIE	rw	0x00	OUT-ACK interrupt enable
5	OUT-NACKIE	rw	0x00	OUT-NACK interrupt enable
4	IN-STALLIE	rw	0x00	IN-STALL interrupt enable
3	IN-ACKIE	rw	0x00	IN-ACK interrupt enable
2	IN-NACKIE	rw	0x00	IN-NACK interrupt enable
1	ENDIE	rw	0x00	Status stage finished interrupt enable
0	Reserved			Always read as 0.

16.5.10 USB address register (USB_ADDR)

Offset address: 0x60

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ADDR						
									rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 7	Reserved			Always read as 0.
6 :0	ADDR	rw	0x00	USB address When the setting address descriptor sent by the host is received, the hardware automatically loads the address into this register. The hardware automatically clears the value of this register when a bus reset is received.

16.5.11 USB endpoint enable register(EP_EN)

Offset address: 0x64

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EP4EN	EP3EN	EP2EN	EP1EN	EP0EN
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			Always read as 0.
4	EP4EN	rw	0x00	Enable End Point 4
3	EP3EN	rw	0x00	Enable End Point 3

Bit	Field	Type	Reset	Description
2	EP2EN	rw	0x00	Enable End Point 2
1	EP1EN	rw	0x00	Enable End Point 1
0	EP0EN	rw	0x00	Enable End Point 0

16.5.12 USB data toggle control register(TOG_CTRL1_4)

Offset address: 0x78

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DTOG 4EN	DTOG 4	DTOG 3EN	DTOG 3	DTOG 2EN	DTOG 2	DTOG 1EN	DTOG 1
								w	rw	w	rw	w	rw	w	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	DTOG4EN	w	0x00	Endpoint 4 data toggle enable bit (Set Endpoint 4 enable) 0: The data toggle bit of Endpoint 4 is not changed 1: Bit 6 DTOG4 is set as the data toggle bit of Endpoint 4
6	DTOG4	rw	0x00	Endpoint 4 data toggle bit (Set Endpoint 4 Toggle) 0：DATA0 1：DATA1
5	DTOG3EN	w	0x00	Endpoint 3 data toggle enable bit (Set Endpoint 3 enable) 0: The data toggle bit of Endpoint 3 is not changed 1: Bit 4 DTOG3 is set as the data toggle bit of Endpoint 3
4	DTOG3	rw	0x00	Endpoint 3 data toggle bit (Set Endpoint 3 Toggle) 0：DATA0 1：DATA1
3	DTOG2EN	w	0x00	Endpoint 2 data toggle enable bit (Set Endpoint 2 enable) 0: The data toggle bit of Endpoint 2 is not changed 1: Bit 2 DTOG2 is set as the data toggle bit of Endpoint 2
2	DTOG2	rw	0x00	Endpoint 2 data toggle bit (Set Endpoint 2 Toggle) 0：DATA0 1：DATA1
1	DTOG1EN	w	0x00	Endpoint 1 data toggle enable bit (Set Endpoint 1 enable) 0: The data toggle bit of Endpoint 1 is not changed 1: Bit 0 DTOG1 is set as the data toggle bit of Endpoint 1
0	DTOG1	rw	0x00	Endpoint 1 data toggle bit (Set Endpoint 1 Toggle) 0：DATA0 1：DATA1

16.5.13 USB setup packet data register(SETUPX) (X = 0 ~ 7)

Offset address: 0x80 ~ 0x9C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SETUPDX							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	SETUPDX	r	0x00	USB setup packet data bits (x = 0, 1, 2, .., 7) (Setup Data X) 64-bit SETUP data, which is automatically set by the hardware according to the data sent by the host.

16.5.14 USB transfer packet size register(PACKET_SIZEX)(X = 0 ~ 1)

Offset address: 0xA0, 0xA4

Reset value: 0x0000 0040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SIZE							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	SIZE	rw	0x40	USB DMA Max Packet Size 64 bytes can be set at most.

16.5.15 USB endpoint X valid data register(EPX_AVAIL)

Offset address: 0x100 ~ 0x110

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EPXAVAIL							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	EPXAVIL	r	0x00	EPX FIFO available data number

16.5.16 USB endpoint X control register (EPX_CTRL)

Offset address: 0x140 ~ 0x150

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRAN EN	TRANCOUNT						
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7	TRANEN	rw	0x00	EPX transfer enable If this bit is set to '1', Endpoint X will define the number of data in acknowledge bit 6:0 after the IN transfer, otherwise, Endpoint X will acknowledge NACK. If there is no enough data in the Endpoint x FIFO, the endpoint will also acknowledge the NACK. If the endpoint X HALT enable bit is set to '1', the endpoint will automatically acknowledge STALL. This bit is automatically changed to '0' at the end of the transfer.
6 : 0	TRANCOUNT	rw	0x00	EPX transfer counter Data to transferred by Endpoint X: The data is stored in the FIFO of the endpoint; the maximum transfers cannot exceed the maximum packet size defined by the register PACKAGE_SIZE, and the transmitted data in the last packet may be less than the maximum packet, or even zero, meaning that an empty packet is transmitted.

16.5.17 USB endpoint X FIFO register (EPX_FIFO)

Offset address: 0x160 ~ 0x170

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EPX_FIFO							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	EPX_FIFO	rw	0x00	EPX FIFO port

16.5.18 USB endpoint X DMA enable register (EP_DMA)

Offset address: 0x184

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DMA2EN	DMA1EN
														rw	rw

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1	DMA2EN	rw	0x00	EP2 DMA enable
0	DMA1EN	rw	0x00	EP1 DMA enable

Note: The USB controller only supports DMA operations of Endpoints 1 and 2

16.5.19 USB endpoint halt register(EP_HALT)

Offset address: 0x188

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											HALT4	HALT3	HALT2	HALT1	HALT0
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 5	Reserved			Always read as 0.
4	HALT4	rw	0x00	EP4 halt When this bit is set to ’1’, the device will automatically respond to STALL after IN/OUT transmission. This bit is automatically cleared by hardware when a token packet is received.
3	HALT3	rw	0x00	EP3 halt When this bit is set to ’1’, the device will automatically respond to STALL after IN/OUT transmission. This bit is automatically cleared by hardware when a token packet is received.
2	HALT2	rw	0x00	EP2 halt When this bit is set to ’1’, the device will automatically respond to STALL after IN/OUT transmission. This bit is automatically cleared by hardware when a token packet is received.
1	HALT1	rw	0x00	EP1 halt When this bit is set to ’1’, the device will automatically respond to STALL after IN/OUT transmission. This bit is automatically cleared by hardware when a token packet is received.

Bit	Field	Type	Reset	Description
0	HALT0	rw	0x00	EP0 halt When this bit is set to '1', the device will automatically respond to STALL after IN/OUT transmission. This bit is automatically cleared by hardware when a token packet is received.

16.5.20 USB power control register (USB_POWER)

Offset address: 0x1C0

Reset value: 0x0000 0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WKUP	Res.	SUSP	SUSP EN
												rw		rw	rw

Bit	Field	Type	Reset	Description
15 : 4	Reserved			Always read as 0.
3	WKUP	rw	0x00	Enable controller wake up from suspend state 1: Wake up 0: Not awaken
2	Reserved			Always read as 0.
1	SUSP	rw	0x01	Suspend 1: Normal operating mode 0: Suspend mode
0	SUSPEN	rw	0x01	Bus suspend enable bit 1: The controller directly controls whether the USB is suspended according to the status of Bit 1. 0: The controller determines whether to suspend the signal

17 Controller area network (CAN)

Controller area network (CAN)

17.1 CAN introduction

The CAN is designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmitting messages (priority characteristics configured by software).

For safety-critical applications, the CAN controller provides all hardware functions for supporting the Time Triggered Communication Mode.

17.2 CAN main features

- Supports CAN protocol version 2.0 A, and 2.0 B
- Extended receive buffer (64-byte, first-in and first-out (FIFO))
- Support both 11-digit and 29-digit identifiers
- Bit rate up to 1 Mbits/s
- PeliCAN mode extension
  - Error counter for read/write access
  - Programmable error warning limit
  - Last error code register
  - Interrupt for each CAN bus error
  - Arbitration lost interrupt controlled by specific control bits
  - Single transmission (no retransmission)
  - Listen-only mode (no confirmation and no activity error flag)
  - Software bit rate detection
  - Acceptance filter extension (4-byte code and 4-byte mask)
  - Own information reception (self-receive request)

17.3 CAN general description

In todayAs CAN applications, the number of nodes in a network is increasing and often several CANs are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

Figure 134. CAN Network Topology

17.3.1 CAN 2.0B active core

The CAN module handles the transmission and the reception of CAN messages fully autonomously, fully supporting standard identifiers (11-bit) and extended identifiers.

17.3.2 CAN block diagram

The block diagram of CAN is as follows:

Figure 135. CAN Block Diagram

17.3.3 Interface management logic (IML)

The interface management logic interprets commands from the CPU and controls the addressing of the CAN registers, to provide interrupt information and status information to the master controller.

17.3.4 Transmit buffer (TXB)

The transmit buffer is the interface between the CPU and the BSP (bitstream processor), enabling storing the complete information sent to the CAN network. The buffer is 13 bytes long, written by the CPU and read by the BSP.

17.3.5 Receive buffer (RXB, RXFIFO)

The receive buffer is an interface between the acceptance filter and the CPU, enabling storing the information received from CAN bus and receive buffer (RXB, 13 bytes) as a window for the receive FIFO (RXFIFO, 64 bytes long). Additionally, the buffer can be accessed by the CPU.

With the support of this FIFO, the CPU enables receiving other information while processing information.

17.3.6 Acceptance filter (ACF)

The acceptance filter compares the data therein with the content of the received identifier, to determine whether to receive the information. In a pure reception test, all information is stored in the RXFIFO.

17.3.7 Bitstream processor (BSP)

The bitstream processor is a program device that controls the data flow between the transmit buffer, RXFIFO, and the CAN bus. It also performs error detection, arbitration, padding, and error handling on the CAN bus.

17.3.8 Bit timing logic (BTL)

The bit timing logic monitors the CAN bus of the serial port and handles the bit timing associated with the bus. It synchronizes the CAN bus bit stream (hard sync) during the bus transfer with the information preceded by 'weakness-dominant', and synchronizes the next transfer (soft sync) again when receiving the message. The BTL also provides a programmable time period, to compensate for propagation delay time, phase transitions, and defining sample points and number of samples per bit time.

17.3.9 Error management logic (EML)

EML is used for error control of the transport layer module, receiving an error report from the BSP, and informing the BSP and IML of the error statistics.

17.4 CAN operating mode

CAN has two main operating modes:

- BasicCAN mode
- PeliCAN mode

The default mode, during system reset, is BasicCAN mode.

The PeliCAN mode is a new operating mode, processing all frame types regulated in CAN 2.0B specifications. Moreover, it also supports some enhancements that make it available in a wider field.

The CAN mode is defined in the CAN_CDR.7 register. If CDR.7 is 0, the CAN controller operates in BasicCAN mode. Otherwise, it operates in PeliCAN mode.

17.4.1 Difference between Basic CAN and Peli CAN modes

In Peli CAN mode, the CAN controller has a recombination register with many functions. The Peli CAN mode supports all functions specified by the CAN 2.0B protocol (29-byte identifier). The following are the main new features of the PeliCAN mode:

- Reception and transmission of standard and extended frames
- Receive FIFO (64 bytes)
- Single- / dual- acceptance filters (including mask and code registers) in both standard and extended formats
- Error counter for read/write access
- Programmable error limit alarm
- Last error register
- Error interrupt for each CAN bus error
- Loss of arbitration and detailed bit position

- One-time transmission (no retransmission in case of error or arbitration lost)
- Listen-only mode (CAN bus monitoring, no response and no error flag)

17.5 CAN Functional description

17.5.1 Basic CAN mode

Reset mode

The reset mode is the initialization mode. The reset request bit (CAN_CR.0) is set to '1' (current) when the hardware is started or the bus status is set to '1' (bus-off). If these bits are accessed by software, their values will change and will affect the next rising edge of the internal clock. The read reset request bit of the internal frequency-division clock synchronization reflects the synchronization status when the reset request bit changes. The reset mode is mainly used for CAN communication parameter configuration, and the core has different access rights to the CAN register in different operating modes.

After the reset request bit is set to '0', the CAN controller will wait:

a) A bus idle signal (11 weak bits) if the previous reset request is a hardware reset or an initial CPU reset.

b) 128 buses are idle if the previous reset request is caused by the CAN controller initializing the bus before re-entering the bus-on mode; it must be noted that the values of some registers will be changed if the reset request bit is set.

Operating mode

After the reset mode, the software shall set the hardware into normal mode, to receive and send messages normally. In the reset mode, when a falling edge of '1 - 0' is transmitted to the reset bit, the CAN controller returns to the operating mode, to transmit and receive the message.

Table 61. Permission Assignment for Basic CAN Mode Register

Offset	Field	Operating mode		Reset mode	
		Read	Write	Read	Write
00	Control	Control	Control	Control	Control
04		(FFH)	Command	(FFH)	Command
08		Status	–	Status	–
0C		(FFH)	–	Interrupt	–
10		(FFH)	–	Acceptance code	Acceptance code
14		(FFH)	–	Acceptance mask	Acceptance mask
18		(FFH)	–	Bus timing 0	Bus timing 0
1C		(FFH)	–	Bus timing 1	Bus timing 1
20		(FFH)	–	–	–
24		Test	Test	Test	Test

Offset	Field	Operating mode		Reset mode	
		Read	Write	Read	Write
28	Transmit buffer	Identifier (10 ~ 3)	Identifier (10 ~ 3)	(0xFF)	–
2C		Identifier (2~0) RTR and DLC	Identifier (2~0) RTR and DLC	(0xFF)	–
30		DATA1	DATA1	(0xFF)	–
34		DATA2	DATA2	(0xFF)	–
38		DATA3	DATA3	(0xFF)	–
3C		DATA4	DATA4	(0xFF)	–
40		DATA5	DATA5	(0xFF)	–
44		DATA6	DATA6	(0xFF)	–
48		DATA7	DATA7	(0xFF)	–
4C		DATA8	DATA8	(0xFF)	–
50	Receive buffer	Identifier(10 ~ 3)	Identifier(10 ~ 3)	Identifier(10 ~ 3)	Identifier(10 ~ 3)
54		Identifier(2 ~ 0) RTR and DLC	Identifier(2 ~ 0) RTR and DLC	Identifier(2 ~ 0) RTR and DLC	Identifier(2 ~ 0) RTR and DLC
58		DATA1	DATA1	DATA1	DATA1
5C	Receive buffer	DATA2	DATA2	DATA2	DATA2
60		DATA3	DATA3	DATA3	DATA3
64		DATA4	DATA4	DATA4	DATA4
68		DATA5	DATA5	DATA5	DATA5
6C		DATA6	DATA6	DATA6	DATA6
70		DATA7	DATA7	DATA7	DATA7
74		DATA8	DATA8	DATA8	DATA8
78		(FFH)	–	(FFH)	–
7C		Clock divider	Clock divider	Clock divider	Clock divider

Note: ’(FFH)’ represents read data is all 1, ’—’ represents no write operation permission, and the rest represents ’permissible’.The ’clock divider’ with offset address 0x7C is used to select BasicCAN and PeliCAN.

17.5.2 Peli CAN mode

Reset mode

The reset mode is the initialization mode. The reset mode bit is set to ’1’ (current) when the hardware reset or bus status bit is ’1’ (bus-off). If this bit is accessed by software, the value will change and the rising edge of the next internal clock (frequency is 1/2 of the external oscillator) is active. The change of the reset request bit is synchronized with the internal frequency-division clock. The read reset request bit can reflect this synchronization state. After the reset mode bit is ’0’, the CAN controller will wait:

1. A bus idle signal (11 hidden (weak) bits) if the previous reset request is a hardware reset or an initial CPU reset.
2. 128 buses are idle if the previous reset request is caused by the CAN controller initializing before re-entering the bus-on mode.

Operating mode

After the reset mode, the software shall set the hardware into normal mode, to receive and send messages normally. In the reset mode, when a falling edge of '1 - 0' is detected on the RM bit of CAN_MOD register, the CAN controller returns to the operating mode, to transmit and receive the message.

Self-test mode

This mode is mainly used for testing. Set the self-test mode bit (CAN_MOD.2) to 1, to enter the self-test mode. In this mode, all nodes can be detected, and no self-receive command is used for the active node; even if there is no response, the CAN controller will send message successfully.

Listen-only mode

This is mainly used for testing. Set the Listen-only mode bit (CAN_MOD.1) to 1, to enable the Listen-only mode. In this operating mode, the CAN controller is in a false negative state, making information transfer impossible. The Listen-only mode is available for software-driven bit rate detection, and other functions can be used as they would in normal operating mode.

In this mode, the CAN controller is unable to write a dominant bit on the CAN bus. The activation error flag or the overload flag cannot be written, and the response signal will not be sent after successful reception.

Note: You must enter the reset mode before enabling the Listen-only mode.

Table 62. Permission Assignment for Peli CAN Mode Register

Offset	Operating mode		Reset mode	
	Read	Write	Read	Write
00	Mode	Mode	Mode	Mode
04	(00H)	Command	(00H)	Command
08	Status	–	Status	–
0C	Interrupt	–	Interrupt	–
10	Interrupt enable	–	Interrupt enable	Interrupt enable
14	(00H)	–	(00H)	–
18	Bus timing 0	–	Bus timing 0	Bus timing 0
1C	Bus timing 1	–	Bus timing 1	Bus timing 1
20	Reserved	–	–	–
24	Test	Test	Test	Test
28	Reserved	–	Reserved	–

Offset	Operating mode				Reset mode	
	Read		Write		Read	Write
2C	Capture of lost arbitration		–		Capture of lost arbitration	–
30	Error code capture		–		Error code capture	–
34	Error warning limit		–		Error warning limit	Error warning limit
38	RX error counter		–		RX error counter	RX error counter
3C	TX error counter		–		TX error counter	TX error counter
40	RX frame information SFF	RX frame information EFF	TX frame information SFF	TX frame information EFF	Acceptance code 0	Acceptance code 0
44	RX identifier 1	RX identifier 1	TX identifier 1	TX identifier 1	Acceptance code 1	Acceptance code 1
48	RX identifier 2	RX identifier 2	TX identifier 2	TX identifier 2	Acceptance code 2	Acceptance code 2
4C	RX data1	RX identifier 3	TX data 1	TX identifier 3	Acceptance code 3	Acceptance code 3
50	RX data 2	RX identifier 4	TX data 2	TX identifier 4	Acceptance mask 0	Acceptance mask 0
54	RX data 3	RX data 1	TX data 3	TX data 1	Acceptance mask 1	Acceptance mask 1
58	RX data 4	RX data 2	TX data 4	TX data 2	Acceptance mask 2	Acceptance mask 2
5C	RX data 5	RX data 3	TX data 5	TX data 3	Acceptance mask 3	Acceptance mask 3
60	RX data 6	RX data 4	TX data 6	TX data 4	Reserved	–
64	RX data 7	RX data 5	TX data 7	TX data 5	Reserved	–
68	RX data 8	RX data 6	TX data 8	TX data 6	Reserved	–
6C	(FIFO RAM)	RX data 7	–	TX data 7	Reserved	–
70	(FIFO RAM)	RX data 8	–	TX data 8	Reserved	–
74	RX information counter		–		RX information counter	–
78	RX buffer start address		–		RX buffer start address	RX buffer start address

Offset	Operating mode		Reset mode	
	Read	Write	Read	Write
7C	Clock divider	Clock divider	Clock divider	Clock divider
80	Internal RAM address 0 (FIFO)	–	Internal RAM address 0	Internal RAM address 0
84	Internal RAM address 1 (FIFO)	–	Internal RAM address 1	Internal RAM address 1
...	...	...	...	...
17C	Internal RAM address 63 (FIFO)	–	Internal RAM address 63	Internal RAM address 63
180	Internal RAM address 64 (TX buffer)	–	Internal RAM address 64	Internal RAM address 64
...	...	...	...	...
1B0	Internal RAM address 76 (TX buffer)	–	Internal RAM address 76	Internal RAM address 76
1B4	Internal RAM address 77 (Idle)	–	Internal RAM address 77	Internal RAM address 77
1B8	Internal RAM address 78 (Idle)	–	Internal RAM address 78	Internal RAM address 78
1BC	Internal RAM address 79 (Idle)	–	Internal RAM address 79	Internal RAM address 79
1C0	(00H)	–	(00H)	–
...	...	...	...	...
1FC	(00H)	–	(00H)	–

17.5.3 Transmission handling

According to the CAN protocol specification, the message is transmitted independently by the CAN controller. The microcontroller is used to set the identifier, the data length and the data to be transmitted. Then, the 'Send Request' bit in the command register is set to '1' for RTS (request to send). The transmit buffer is write-locked when the CAN controller is transmitting a message. Therefore, before preventing a new message from being sent to the transmit buffer, the microcontroller must check the status register's 'Transmit Buffer Status' flag (TBS).

After the command bits CMR.0 and CMR.1 are set, a message transmission will occur immediately and no retransmission will appear in case of transmission error or arbitration lost (single transmission). The retransmission will be performed if only the command bit CMR.0 is set and data is failed to be sent. In the self-test mode, a self-receiving message will be generated immediately once the command bits CMR.4 and CMR.1 are set.

Abort

A transmission request of message can be aborted by setting the corresponding bit in the command register, namely, AT bit in CAN_CMR register is set to ’1’.

If CPU requires the pending of current transmission request, for example, a more urgent message needs to be sent first and the current transfer shall not stopped. To know if the source information is successfully sent, you can view it by sending the completion status bit. However, this shall be done after the status bit of transmit buffer is set to ’1’ or a transmit interrupt is generated.

It should be noted that a transmission interrupt occurs even if the information is aborted since the status bit of transmission buffer becomes ’released’.

If the transmission request in the previous instruction is set to ’1’, it cannot be disabled by setting the transmission request bit to ’0’, but shall be canceled by setting the abort transmission bit to ’0’.

17.5.4 Reception handling

The received message is processed independently by the CAN controller, and is sent to the receive buffer. The message that can be sent to the microcontroller is marked by the receive buffer status flag ’RBS’ in the status register and the receive interrupt flag ’RI’.

Query control reception

The microcontroller reads the status register of the CAN controller and checks the receive buffer status (RBS), to check if a message has been received.

When the RBS bit read is 1, it indicates that one or more messages have been received. The microcontroller retrieves the message from the CAN, and then sets the response flag ’RRB’ of the command register, to send a releasing receive buffer command.

Interrupt control reception

The interrupt enable flag is set in the CAN controller register (for BasicCAN mode) or in the interrupt enable register (for PeliCAN mode). If the CAN controller has received a message and the message has been processed by the acceptance filter and sent to the receive FIFO, a receive interrupt will be generated. Then, the interrupt service routine is enabled, the message is fetched by the microcontroller, and the response flag bit ’RRB’ of the command register is set, to send a releasing receive buffer command.

Overrun

An overrun occurs when the receive FIFO is full but another message is received, and the data overload status bit in the status register is set (if enabled), to inform the microcontroller of the data overrrun. The status can be cleared by setting CMR.3 bit to ’1’.

Valid message

A received message is considered as valid when it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) and it passed through the identifier filtering successfully.

The RRB bit in the CAN_CMR register is used to clear the data overrun indicated by the

data overrrun status bit.

17.5.5 Identifier filtering

In the CAN protocol, the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently, a transmitter broadcasts its message to all receivers. During message reception, a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the BUFFER. If not, the message must be discarded without intervention by the software.

The stand-alone CAN controller is equipped with a multifunctional acceptance filter that enables automatic checking of identifiers and data bytes. With these efficient filtering methods, it is possible to prevent invalid messages or message groups in a node from being stored in the receive buffer, thus reducing the processing load on the microcontroller.

The filter is controlled by the acceptance code register and the mask register according to a given algorithm. The received data is compared bit by bit with the values in the acceptance code register. The receive mask register defines the location associated with the comparison (0 = correlated, 1 = uncorrelated). The message will only be received if the corresponding bit of the received message is the same as that in the acceptance code register.

Acceptance filter in BasicCAN mode

The filter is controlled by two registers, the acceptance code register (ACR) and the acceptance mask register (AMR). The upper 8 bits of the CAN message identifier are compared to the values in these registers, to define the identifiers of several groups to be received by any node.

Example: acceptance code register (ACR) includes:

Figure 136. Example of CAN Identifier Reception

In position of ’1’ in the acceptance mask register, the corresponding bit of the identifier can be any value. This is the same for the three lowest bits. Therefore, in this example, you can receive 64 different identifiers; the other bits of the identifier must be equal to the value of the corresponding bit in the acceptance code register.

Acceptance filter in PeliCAN mode

With the help of the acceptance filter, the CAN controller allows the received information to be stored in the RXFIFO only when the identification bits in the received message and the predefined values of the acceptance filter are equal.

In PeliCAN mode, the acceptance filter is defined by the acceptance code register (ACRn) and the acceptance mask register (AMRn), and the bit pattern of the information to be received is defined in the acceptance code register. The corresponding acceptance mask register allows certain bits to be defined as 'no effect' (i.e. any value).

There are two different filtering modes that can be selected in bit 3 of the mode register:

- Single filter mode$^{(1)}$
- Dual-filter mode$^{(0)}$

Single filter configuration

This filter configuration defines a long filter (4 bytes). The correspondence between the bits of the filter byte and the information byte depends on the current received frame format.

Standard frame: If the information in the standard frame format is received, only the first two data bytes are used during the acceptance filtering, to store the complete identifier including the RTR bit. In case of no data byte due to the RTR bit set, or no or only one data byte due to corresponding data length code configured, the message will be also received. For a successfully received message, the receipt signal must be sent after comparison of individul bits.

Note: The lower four bits of ACR1 and AMR1 are not used. For compatibility with future products, these bits can be defined as 'no effect' by setting AMR1.3, AMR1.2, AMR1.1 and AMR1.0 to 1.

Figure 137. Single Filter Configuration When Receiving Information of Standard Structure

Extended frame: If the received message is in extended frame format, all identifiers including the RTR bit will be used for receive filtering.

In order to successfully receive information, each bit must be compared.

Note: The lowest two bits of AMR3 and ACR3 are not used. These bits shall be defined as 'no effect' by setting AMR3.1 and AMR3.0.

Figure 138. Single Filter Configuration for Receiving Extended Frame Information

Dual-filter configuration

This configuration can define two short filters. A received message is compared to that of two filters, to determine if it is stored in the receive buffer. The received message is valid only when at least one filter sends a received signal. The correspondence between the bits of the filter byte and the information byte depends on the currently received frame format.

Standard frame: If the standard frame information is received, the two filters defined are different. For the first filter, the entire standard identifier of the RTR bit and the first data byte of the message are compared; for the second filter, only the entire standard identifier including the RTR bit is compared.

In order to receive information successfully, at least one filter shall indicate the reception when all individual bits are compared. If the RTR bit is set or data length code is 0, no data byte exists. In any case, as long as the part from the first bit to the RTR bit indicates the reception, the information can be processed by Filter 1.

If no data byte filtering is requested from the filter, the lower four bits of AMR1 and AMR3 must be set to ’1’ (no effect). Both filters operate when the entire standard identifier including the RTR bit is used.

Figure 139. Dual-filter Configuration When Receiving Information of Standard Structure

Extended frame: If the extended frame information is received, the two defined filters are identical. For both filters, only the first two bytes of the extended identifier are compared.

In order to receive the information successfully, at least one filter indicates reception during the comparison of all the individual bits.

Figure 140. Dual-flter Configuration for Receiving Extended Frame Information

Example 1: Assume that the following 1 standard frame message will be filtered in Peli-CAN mode, which can be achieved with a long filter (single filter mode).

The acceptance code register (ACRn) and the acceptance mask register (AMRn) include:

n	0	1(upper four bits)	2	3
ACRn	01XX X010	XXXX	XXXX XXXX	XXXX XXXX
AMRn	0011 1000	1111	1111 1111	1111 1111

n	0	1(upper four bits)	2	3
Received message(ID28 ~ ID.18, RTR)	01xx x010 xxxx			

'X' = uncorrelated, 'x' = any value, only the upper four bits of ACR1 and AMR1 are used.

Example 2: Assume that the following two messages with standard frame identifiers are received without further decoding. Data and remote frames must be received correctly and the data byte requires no acceptance filtering.

Message 1: (ID.28)1011 1100 101(ID.18)

Message 2: (ID.28)1111 0100 101(ID.18)

With the single filter mode, you can receive four messages instead of the two required:

n	0	1(upper four bits)	2	3
ACRn	1X11 X100	101X	XXXX XXXX	XXXX XXXX
AMRn	0100 1000	0001	1111 1111	1111 1111
Received message(ID28 ~ ID.18, RTR)	1011 0100 101x 1111 0100 101x (Message 2) 1011 1100 101x (Message 1) 1111 1100 101x			

'X' = uncorrelated, 'x' = any value, only the upper four bits of ACR1 and AMR1 are used.

This result requires further decoding, to meet the requirements for receiving two messages.

Get the right results by using double filters.

	Filter 1			Filter 2	
n	0	1	3 lower four bits	2	3 upper four bits
ACRn	1011 1100	101X XXXX	... XXXX	1111 0100	101X ...
AMRn	0000 0000	0001 1111	... 1111	0000 0000	0001 ...
Received information(ID28 ~ ID.18, RTR)	1011 1100 101X(Message 1)			1111 0100 101X(Message 2)	

'X' = uncorrelated, 'x' = any value.

Message 1 is received by Filter 1 and message 2 received by Filter 2. If the message is received by at least one of the two filters, the message will be stored in the receive FIFO. This method satisfies this requirement.

Example 3: In this example, a long acceptance filter is used to filter message groups with extended identifiers.

n	0	1	2	3(upper six bits)
ACRn	1011 0100	1011 000X	1100 XXXX	0011 0XXX
AMRn	0000 0000	0001 0001	0000 1111	0000 0111
Received information(ID28 ~ ID.18, RTR)	1011 0100 101x 000x 1100 xxxx 0011 0x			

'X' = uncorrelated, 'x' = any value, only the upper six bits of ACR1 and AMR1 are used.

Example 4: The standard frame system is used to identify messages only with the 11-bit identifier and the first two data bytes. If the message exceeds 8 data bytes, the first two data bytes are defined as the message header and the segmented storage protocol uses such a protocol, such as DeviceNet. For this type of system, the CAN controller enables filtering two data bytes in single filter mode in addition to the 11-bit identifier and RTR bit, and enables filtering one data byte in dual filter mode (except for the 11-bit flag and RTR bits).

The following example shows the process of effectively filtering the message in such a system in the dual-filter mode:

	Filter 1			Filter 2	
n	0	1	3 lower four bits	2	3 upper four bits
ACRn	1110 1011	0010 1111	... 1001	1111 0100	XXX0 ...
AMRn	0000 0000	0000 0000	... 0000	0000 0000	1110 ...
Received information(ID28 ~ ID.18, RTR)	1110 1011 0010 + 1111 ... 1001 Identifier RTR + first data byte			1111 0100 Identifier	xxx0 RTR

('X' = uncorrelated, 'x' = any value)

- Filtered messages in Filter 1 include:
  - Identifier '11101011001'
  - RTR = '0', namely, the data frame
  - Data byte '11111001' (this means that, such as DeviceNet, all segments of a message are filtered)
- Filter 2 is used to filter a group of 8 messages, in which the message includes:
  - Identifier '11110100 000' to '11110100111'
  - RTR = '0', namely, the data frame

17.5.6 Message storage

The data to be sent on the CAN bus is loaded into the CAN controller's memory area, which is called the 'transmit buffer'; the data received from the CAN bus is also stored in the CAN controller's memory area, the 'receive buffer'. These buffers contains 2- , 3- or 5-byte identifiers and frame information (depending on mode and frame type), with the maximum capacity of 8 data bytes.

BasicCAN mode

The buffer up to 10 bytes contains:

- 2 identifier bytes
- 8 data bytes

Table 63. RX and TX Buffers in BasicCAN Mode

Relative CAN offset		Register		Composition and notes
TX (hexadecimal)	RX (hexadecimal)	TX	RX	
28	50	CAN_TXIDR1	CAN_RXIDR1	8-bit identifier
2C	54	CAN_TXIDR2	CAN_RXIDR2	3-bit identifier and 1-bit remote transmission request bit (4 bits), According to the length code, it represents the number of data bytes
30	58	CAN_TXDR1	CAN_RXDR1	Represented by the data length code (up to 8 data bytes)
34	5C	CAN_TXDR2	CAN_RXDR2	
38	60	CAN_TXDR3	CAN_RXDR3	
3C	64	CAN_TXDR4	CAN_RXDR4	Represented by the data length code (up to 8 data bytes)
40	68	CAN_TXDR5	CAN_RXDR5	
44	6C	CAN_TXDR6	CAN_RXDR6	
48	70	CAN_TXDR7	CAN_RXDR7	
4C	74	CAN_TXDR8	CAN_RXDR8	

PeliCAN mode

The 13-byte buffer contains:

- 1-byte frame information
- 2 or 4 identifier bytes (standard or extended frame)
- Up to 8 data bytes

Transmit buffer

The complete list of transmit buffers is as shown below. Note to distinguish between standard frame format (SFF) and extended frame format (EFF) configurations. The transmit buffer allows to define the received message up to 8 data bytes.

Figure 141. List of Standard frame and Extended Frame Format Configured in Transmit Buffers

The FF bit in the frame information determines whether the CAN controller sends messages in the extended frame format or those in the standard frame format.

Receive buffer

The list of receive buffers is very similar to the transmit buffers. The receive buffer is the accessible part of the RXFIFO, occupying 40 ~ 70 of the CAN address. Each message is divided into a description area and a data area.

The received byte length code in the frame information byte represents the actual transmitted data length code, which may be greater than 8 (depending on the transmitter). In any case, the maximum received data byte is 8, and this should be considered when reading the information in the receive buffer.

As shown in the figure below, the RXFIFO has a space of 64 information bytes in total. The information that can be stored at a time depends on the length of the data. If there is not enough space in the RXFIFO to store new information, a data overrun will occurs in the CAN controller, at the moment, the message is valid and the receive test is required. In case of data overflow, information that has been written to the RXFIFO will be deleted, and

the conditionn can be reported to CPU through the status register and the data overrun interrupt (interrupt enable).

Figure 142. List of Standard frame and Extended Frame Format Configured in Transmit Buffers

17.5.7 Error management

Based on the value of the error counter, each CAN controller operates in one of three error states: Error Active, Error Pasitive, or Bus Offline. If the value of the error counter is between 0 ～127, the CAN controller is incorrectly activated. At this time, an Error Active flag (6 dominant bits) is generated. If the value of an error counter is between 128～255, the CAN controller is incorrectly recognized. At this time, an Error Pasitive flag (6 recessive bits) is generated before an error is detected. If the value of the transmit error counter is higher than 255, the bus is offline. In this state, the reset request is automatically set and the CAN controller has no effect on the bus. The bus offline status can only be exited through the microcontroller with the command 'reset request = 0'. This will activate the bus offline recovery timer and send 128 bus release signals to the error counter. After the counting, both error counters are 0 and the device is again in the Error Active state.

Error counter

As described above, the error state of the CAN is directly related to the value of the transmit error counter and the receive error counter.

In order to carefully study the error definition, the CAN controller supports the enhanced error analysis function, and provides a readable error counter. In addition, in reset mode, write access is allowed for both error counters.

Error interrupt

Three interrupt sources are used to send the error status to the microprocessor. Each interrupt can be separately enabled in the interrupt enable register.

- Bus error interrupt: an interrupt is generated when any error is detected on the CAN bus
- Error warning interrupt: If the error warning limit is exceeded, an error warning interrupt is enabled. Moreover, this interrupt is also generatged when the CAN controller enters the bus offline state and reenters the Error Active state. The error warning limit of the CAN controller is programmable in reset mode, and the default value after reset is 96.
- Error Pasitive interrupt: If the error status changes from Error Active to Error Pasitive or vice versa, an Error Pasitive interrupt will be generated.

Error code capture

The CAN controller can perform all the error definitions regulated in the CAN2.0B specifications. The entire process of error handling for each CAN controller is completely automatic. However, in order to provide the user with the details of an error, the CAN controller supports the error code capture function. Whenever a CAN bus error occurs, it forces a corresponding bus error interrupt. At the same time, the current bit is captured into the error code capture register; the captured data is saved in the register before being read by the main controller. Then, the capture mechanism is activated again; the register enables distinguishing between four types of errors: format errors, padding errors, bit errors, and other errors. As shown in the figure below, the register additionally indicates whether the error occurs during the reception or transmission of the message. The five bits in this register indicate the bit of the frame error in the CAN. Refer to the table and data table below for more information.

Figure 143. Example of Error Code Capture Function

The CAN specifications define that each bit on the CAN bus has only a special type of error. The following two figures show all the errors that may occur during CAN message transmission and reception. The left part includes the position and error type, which are captured by the error code capture register; the right part of each table is an error description that converts the error code into the upper layer, which can be directly interpreted from the contents of the register. By using these tables, you can get more information about the error counter changes and the error status of the transmit and receive pins. When using these tables, such as in the error analysis software, each error state can be analyzed in detail. Information about the type and location of CAN errors can be used for error statistics and system maintenance or for corrections in system optimization devices.

Table 64. Possible Errors During Reception

Error location in CAN bitstream	Error type	RX error count	Error code capture	Description
Identifier SRR、IDE and RTR bits Reserved bit Data length code Data field CRC sequence	Padding	+1	Receive 5 consecutive bits of the same level	–
CRC delimiter	Format Padding	+1 +1	RX = dominant, Receive 5 consecutive bits of the same level	Bit must be recessive
Acknowledge bit	Bit	+1	RX = dominant, or CRC error detected	Critical bus timing or bus length CRC sequence incorrect
Acknowledge delimiter	Format	+1	RX = dominant, or CRC error detected	Critical bus timing or bus length CRC sequence incorrect
End of frame	Format Others	+1 ±0	RX = the first six bits are dominant, RX = the last bit is dominant	– Reaction: An overload flag is issued, and the data may be repeated if it is resent by the transmitter
Interval	Others	±0	RX = dominant	Reaction: Receiver sends an overload flag
Activation error flag	Bit	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits
Allowable dominant bit	Others	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits
Error delimiter	Format Others	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits
Overload flag	Bit	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits

Table 65. Possible Errors During Reception

Error location in CAN bitstream	Error type	TX error count	Error code capture	Description
Start of frame	Bit	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits

Error location in CAN bitstream	Error type	TX error count	Error code capture	Description
Identifier	Bit padding	+ 8 ± 0	TX = dominant, but RX = recessive; TX = recessive, but RX = dominant	Not allowed to write dominant bits –
SRR bit	Bit padding	+ 8 ± 0	TX = dominant, but RX = recessive; TX = recessive, but RX = dominant	Not allowed to write dominant bits –
IDE and RTR bits	Bit padding	+ 8 ± 8	TX = dominant, but RX = recessive; TX = recessive, but RX = dominant	Not allowed to write dominant bits –
Reserved bit Data length code Data field CRC sequence	Bit	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits
CRC delimiter	Format	+8	RX = dominant	Bit must be recessive
Acknowledge slot	Others Others	+ 8 ± 0	RX = recessive (Error Active), RX = recessive (Error Pasitive)	No response No response; the node may be on the bus separately
Acknowledge delimiter	Format	+8	RX = dominant	Critical bus timing or bus length
End of frame	Formats Others	+8 +8	RX = the first six bits are dominant, RX = the last bit is dominant	– The frame has been received by some nodes, and resending may lead to repeated data in the receiver
Interval	Others	±0	RX = dominant	Overload flag from the 'old' CAN controller
Activation error flag Overload flag	Bit	+8	TX = dominant, but RX = recessive	Not allowed to write dominant bits
Allowable dominant bit	Format	+8	RX = more than 7 dominant bits preceded by activation error or overload flag	–

Error location in CAN bitstream	Error type	TX error count	Error code capture	Description
Error delimiter	Format Others	+8 ±0	RX = the first seven bits are dominant, RX = the last bit of the delimiter is a dominant bit	–
Passive Error flag	Others	+ 8	RX = dominant(Error Pasitive)	No response; the node is not on the bus separately

Offline recovery

If the value of the transmit error counter is higher than 255, the bus is offline, and the bus status bit is set to '1'. In this state, the reset request bit is automatically set and the CAN controller has no effect on the bus. In case of an error interrupt enabled, an error interrupt is generated. This state continues until the CPU clears the reset request bit. After that, the CAN controller will wait for the minimum time specified by the protocol (128 idle bus signals).

17.5.8 Bit-time characteristics

The bit timing logic monitors the serial CAN bus by sampling and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- Synchronization segment ($t_{SYNCSEG}$): a bit change is expected to occur within this time segment. It has a fixed length of one time quantum (1 x t_{CAN}).
- Bit segment 1(t_{TSEG1}): It defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2(t_{TESG2}):It defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The period of t_{SCL}, the CAN system clock, is programmable and determines the corresponding bit timing.

The CAN system clock is calculated by the following equation: $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$. Where, t_{CLK} = APB1 frequency period

Synchronization jump width (SJW) defines the upper limit of how many time units can be lengthened or shortened in each bit. In order to compensate for the phase offset between the clock oscillators of the different bus controllers, any bus controller must be resynchronized at the edge of the associated signal currently being transmitted; the synchroniza-

tion jump width defines the maximum number of clock cycles that can be shortened or extended per each bit cycle.

$t_{SJW} = t_{SCL} \times (SJW + 1)$; Time period 1 (TSEG1) and Time period 2 (TSEG2) determine the number of clocks per bit and the position of the sampling point, where:

$t_{SYNCSEG} = 1 \times t_{SCL}$

$t_{TESG1} = t_{SCL} \times (TESG1 + 1)$

$t_{TESG2} = t_{SCL} \times (TESG2 + 1)$

A valid jump is defined as the first transition from a dominant bit to a recessive bit when CAN sends no recessive bit. If a valid jump is detected during Time period 1 (t_{TSEG1}) instead of the synchronization segment ($t_{SYNCSEG}$), the time of t t_{TSEG1} will be extended as long as SJW, and the sampling point will be delayed.

Conversely, if a valid jump is detected in Time period 2 (t_{TSEG2}) instead of $t_{SYNCSEG}$, then the time of t_{TSEG2} will be shortened by up to SJW, so that the sampling point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing Register (CAN_BTR) is only possible while CAN is in Standby mode.

CAN baud rate = APB1 /(2*(BRP + 1)*(TSEG1 + 1 + TSEG2 + 1 + 1)).

17.5.9 Arbitration lost

In case of arbitration lost, a corresponding arbitration lost interrupt (interrupt enable) is generated. At the same time, the current bit of the bit stream processor is captured and sent to the arbitration lost capture register. The contents of the register will not change until the user reads this value through software. Then, the capture mechanism is then activated again.

When the interrupt register is read, the corresponding interrupt flag bit in the interrupt register is cleared. The new arbitration lost interrupt is not valid until the arbitration lost capture register is read once.

The figure below explains the arbitration lost bits:

Figure 144. Example of Arbitration lost Interpretation

17.5.10 CAN interrupts

Four interrupts are dedicated to the BasicCAN mode:

- Receive interrupt
  It occurs when the receive FIFO is not null and the receive interrupt is enabled (CAN_CR register bit 1), and the RI bit of the CAN_IR register is set to '1'.
- Transmit interrupt
  It occurs when the transmit buffer status changes from 0 to 1 (release) and the transmit interrupt is enabled (CAN_CR register bit 2).
- Error interrupt
  This bit is set due to the change in the error status bit or the bus status bit when the error interrupt is enabled (CAN_CR register bit 3).
- Data overrun interrupt
  It toggles to the data overrun status bit '0 - 1' when the data overrun interrupt enable bit (CAN_CR register bit 4) is set to '1'.

Seven interrupts are dedicated to the PeliCAN mode:

- Receive interrupt
  It occurs when the receive FIFO is not null and the RIE bit of the interrupt register is set.
- Transmit interrupt
  It occurs when the transmit buffer state jumps from '0 - 1' (release) and the TIE bit of the interrupt register is set.
- Error warning interrupt
  It occurs when the error status bit and the bus status bit are changed and the EIE bit of the interrupt register is set.
- Data overrun interrupt
  It occurs when the data overrun status bit jumps from '0 - 1' and the DOIE bit of the

interrupt register is set.

- Error Passive interrupt
  It occurs when the CAN controller reaches the Error Passive state (at least one error counter exceeds the value (127) specified by the protocol) or changes from the Error Passive state to the Error Active state and the EPIE bit of the interrupt register is set.
- Arbitration lost interrupt
  It occurs when the CAN controller loses the arbitration and acts as a receiver and ALIE of the interrupt enable register is set.
- Bus error interrupt
  It occurs when the CAN controller detects a bus error and the BEIE in the interrupt enable register is set.

17.6 Description of CAN register

Table 66. Overview of CAN Registers

Offset	Acronym	Register Name	Reset	Section	Note
0x00	CAN_MOD	CAN mode register	0x00000001	section 17.6.1	PeliCAN mode only
0x00	CAN_CR	CAN control register	0x00000001	section 17.6.2	BasicCAN mode only
0x04	CAN_CMR	CAN command register	0x000000XX	section 17.6.3	Reset mode: BasicCAN mode: 0x00FF PeliCAN mode: 0x0000
0x08	CAN_SR	CAN status register	0x00000000	section 17.6.4	–
0x0C	CAN_IR	CAN interrupt register	0x00000000	section 17.6.5	–
0x10	CAN_IER	CAN interrupt enable register	0x00000000	section 17.6.6	Only applicable to PeliCAN mode
0x10	CAN_ACR	CAN acceptance code register	0x000000XX	section 17.6.7	BasicCAN mode
0x14	CAN_AMR	CAN acceptance mask register	0x000000XX	section 17.6.8	BasicCAN mode
0x18	CAN_BTR0	CAN bus timing 0	0x00000000	section 17.6.9	–
0x1C	CAN_BTR1	CAN bus timing 1	0x00000000	section 17.6.10	–
0x28	CAN_TXID0	CAN transmit identifier register 0	0x000000XX	section 17.6.11	Only applicable to BasicCAN mode; reset mode: 0xFF
0x2C	CAN_TXID1	CAN transmit identifier register 1	0x000000XX	section 17.6.12	Only applicable to BasicCAN mode; reset mode:0xFF
0x2C	CAN_ALC	CAN arbitration lost capture register	0x00000000	section 17.6.13	Only applicable to PeliCAN mode
0x30	CAN_ECC	CAN error code capture	0x00000000	section 17.6.14	Only applicable to PeliCAN mode

Offset	Acronym	Register Name	Reset	Section	Note
0x34	CAN_EWLR	CAN error warning limit register	0x00000096	section 17.6.15	Only applicable to PeliCAN mode
0x38	CAN_RXERR	CAN RX error count register	0x00000000	section 17.6.16	Only applicable to PeliCAN mode
0x3C	CAN_TXERR	CAN TX error count register	0x00000000	section 17.6.17	Only applicable to PeliCAN mode
0x40	CAN_SFF	CAN transmit frame information register	0x0000000X	section 17.6.18	Only applicable to PeliCAN mode
0x44	CAN_TXID0	CAN transmit identifier register 0	0x000000XX	section 17.6.19	Only applicable to PeliCAN mode
0x48	CAN_TXID1	CAN transmit identifier register 1	0x000000XX	section 17.6.20	Only applicable to PeliCAN mode
0x4C	CAN_TXDATA0	CAN transmit identifier register 0	0x000000XX	section 17.6.21	Only applicable to PeliCAN mode
0x50	CAN_TXDATA1	CAN transmit identifier register 1	0x000000XX	section 17.6.22	Only applicable to PeliCAN mode
0x7C	CAN_CDR	CAN clock divider register	0x00000000	section 17.6.23	–

17.6.1 CAN mode register(CAN_MOD)

PeliCAN mode only

Address offset: 0x00

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.												AFM	STM	LOM	RM
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 4	Reserved			Reserved, always read as 0.
3	AFM	rw	0x00	Acceptance filter mode 1: Single; select a single acceptance filter (32-bit length) 0: Dual; select two acceptance filters (each with 16 bits active)

Bit	Field	Type	Reset	Description
2	STM	rw	0x00	Self test mode 1: Self-test mode: all nodes can be detected, and no self-receive command is used for the active node; even if there is no response, the CAN controller will send message successfully. 0: Normal mode: acknowledgement signal must be sent in case of successful transmission
1	LOM	rw	0x00	Listen only mode 1: Listen only mode: the CAN controller sends no acknowledgement signal to the bus even if the message is successfully received; the error counter stops at the current value. 0: Normal mode
0	RM	rw	0x01	Reset mode 1: Reset mode: detect that the reset mode bit is set, then abort the information currently being received/transmitted, to enter the reset mode. 0: Normal mode: the CAN controller returns to the operating mode after the reset mode bit receives a '1-0' jump.

17.6.2 CAN control register(CAN_CR)

BasicCAN mode only

Address offset: 0x00

Reset value: 0x0000 0001

The contents of the control register are used to change the behavior of the CAN controller. These bits can be configured or set by the microcontroller, which enables read/write the control registers.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.											OIE	EIE	TIE	RIE	RR
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 5	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
4	OIE	rw	0x00	Overflow interrupt enable 1: Enabled: if the data overflow bit is set, the microcontroller receives the overflow interrupt signal 0: Disabled: the microcontroller receives no overflow interrupt signal from the CAN controller
3	EIE	rw	0x00	Error interrupt enable 1: Enabled: if the error or bus status changes, the microcontroller receives the error interrupt signal 0: Disabled: the microcontroller receives no error interrupt signal from the CAN controller
2	TIE	rw	0x00	Transmit interrupt enable 1: Enabled: when the message is successfully transmitted or the transmit buffer is accessed again (for example, after aborting the send command), the microcontroller receives a transmit interrupt signal from the CAN controller 0: Disabled: the microcontroller receives no transmit interrupt signal from the CAN controller
1	RIE	rw	0x00	Receive interrupt enable 1: Enabled: when the message is received corretly, the CAN controller sends a receive interrupt signal to the microcontroller. 0: Disabled: the microcontroller receives no receive interrupt signal from the CAN controller
0	RR	rw	0x01	Reset request 1: Current: after detecting the reset request, the CAN controller aborts the information currently transmitted/received and enters the reset mode. 0: Null: After the reset request bit receives a falling edge, the CAN controller returns to the operating mode

17.6.3 CAN control register(CAN_CMR)

Address offset: 0x04

Reset value: BasicCAN mode: 0x0000 00FF

PeliCAN mode: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.											SRR	CDO	RRB	AT	TR
											w	w	w	w	w

Bit	Field	Type	Reset	Description
31 : 5	Reserved			Reserved, always read as 0.
4	SRR	w		PeliCAN mode: SRR: Self reset request 1: Current information can be sent and received simultaneously 0: (Null)
3	CDO	w		Clear data overrun 1: Clear: clear data overrun status bit 0: No action The clear data overrrun command bit is used to clear the data overflow indicated by the data overrun status bit. If this bit is set, no data overrun interrupt will be generated. The clear data overrun command can be issued while the receive buffer command is released.
2	RRB	w		Release receive buffer 1: Release: the memory space in the receive buffer will be released 0: No action After reading the receive buffer, the microcontroller can release the memory space of the current information in the RXFIFO by setting the receive buffer bit to '1', possibly making another message in the receive buffer valid immediately, thus enabling another receive interrupt (enable condition). In case of no other available information, the receive interrupt will no longer be generated and the receive buffer status bit will be cleared.
1	AT	w		Abort transmission 1: Current: if it is not in the process, the send request of waiting process will be canceled 0: Null: no action The abort transmission bit is used when the CPU requests the current transfer to be paused, for example, to send an emergency message, the ongoing transfer will not be stopped. In order to check if the original message is successfully sent, it can be detected with the successful transmit status bit, provided that, the transmit buffer status bit is 1 (release) or a transmit interrupt is generated.

Bit	Field	Type	Reset	Description
0	TR	w		Transmission request 1: Current: information is sent 0: Null: no action If the transmission request is set in the previous command, it cannot be canceled by setting it directly to 0 but by setting the abort transmission bit to 0.

17.6.4 CAN status register(CAN_SR)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BS	ES	TS	RS	TCS	TBS	DOS	RBS
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	BS	rw	0x00	BS: Bus status 1: Bus-off: the CAN controller disables the bus activity 0: Bus-on: the CAN controller enables the bus activity When the transmission error counter exceeds the limit (255) (bus status bit set to ’1’ - bus off), the CAN controller will set the reset request bit to ’1’ (current); if allowable, the error interrupt will be generated. This state continues until the CPU clears the reset request bit. After that, the CAN controller will wait for the minimum time specified by the protocol (128 bus idle signals). After the bus status bit is cleared (bus on), the error status bit is set to ’0’ (ok), the error counter is reset and an error interrupt is generated (interrupt enable).
6	ES	rw	0x00	Error status 1: Error; at least one error counter is full or overflows CPU alarm limit 0: ok: both error counters are below the alarm limit According to the CAN 2.0B protocol, an error detected during reception or transmission affects the error count. The error status bit is set when at least one error counter is full or the CPU warning limit (96) is exceeded. If allowed, an error interrupt will be generated.

Bit	Field	Type	Reset	Description
5	TS	rw	0x00	Transmit status 1: Transmit: the CAN controller is transmitting information 0: Idle: no information is being sent If both the receive status bit and the transmit status bit are 0, the CAN bus is idle.
4	RS	rw	0x00	Receive status 1: Receive: the CAN controller is receiving information 0: Idle: no information is being received If both the receive status bit and the transmit status bit are 0, the CAN bus is idle.
3	TCS	rw	0x00	Transmission complete status 1: Completed: the last request is successfully processed 0: Not completed: the current transmission request has not been processed Whenever the transmission request bit is set to '1', the transmission complete bit will be set to '0' (not completed). The '0' of the transmission complete bit will remain until the message is successfully sent.
2	TBS	rw	0x00	Transmit buffer status 1: Released: the CPU is allowed to write information to the transmit buffer 0: Locked: the CPU is not allowed to access the transmit buffer; the information is waiting to be sent or being sent If the CPU attempts to write to the transmit buffer when the transmit buffer status bit is 0 (locked), the written byte is rejected and will be lost without any prompt.
1	DOS	rw	0x00	Data overrun status 1: Overrun: the information is lost since there is no enough space in RXFIFO to store it 0: Null: no data overrun occurrs since the last clear data overrun command is executed When the information to be received is successfully processed by the acceptance filter (for example, at the beginning of arbitration), the CAN controller requires some space in the RXFIFO to store the descriptor of this information. Therefore, enough space shall be configured, to store every data byte received. In case of insufficient space, the information will be lost and only the CPU will receive the prompt of data overflow; if the received message has no errors except the last bit, the message is valid.

Bit	Field	Type	Reset	Description
0	RBS	rw	0x00	Receive buffer status 1: Full: available information in RXFIFO 0: Null: no information available This bit is cleared after the information in the RXFIFO is read and the memory space is released with the release receive buffer command. If there is still information available in the FIFO, this bit will be reset in the next time limit (t_{SCL}).

17.6.5 CAN interrupt register(CAN_IR)

Address offset: 0x0C

Reset value: 0x0000 0000

The interrupt register allows to identifying the interrupt source; the interrupt is eanbled when one or more bits of the register are set; the interrupt register acts as a read-only memory for the microcontroller.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BEI	ALI	EPI	Res.	DOI	EI	TI	RI
								r	r	r		r	r	r	r

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	BEI	r	0x00	Basic mode: Reserved, with the read value of 1 PeliCAN mode: BEI: Bus error interrupt 1: Set: this bit is set when the CAN controller detects a bus error and the BEIE in the interrupt enable register is set. 0: Reset
6	ALI	r	0x00	Basic mode: Reserved, with the read value of 1 PeliCAN mode: ALI: Arbitration lost interrupt 1: Set: this bit is set when the CAN controller loses the arbitration and acts as a receiver and ALIE of the interrupt enable register is set. 0: Reset

Bit	Field	Type	Reset	Description
5	EPI	r	0x00	Basic mode: Reserved, with the read value of 1 PeliCAN mode: EPI: Error passive interrupt 1: Set: This bit is set when the CAN controller reaches the Error Passive state (at least one error counter exceeds the value (127) specified by the protocol) or changes from the Error Passive state to the Error Active state and the EPIE bit of the interrupt register is set. 0: Reset
4	Reserved			Reserved, always read as 0.
3	DOI	r	0x00	Data overrun interrupt 1: Set: this bit is set when the data overrun interrupt enable bit, set to '1' jumps to the data overrun status bit '0-1'. 0: Reset: any read access by the microcontroller will clear this bit The overrun interrupt bit (interrupt enabled) and the overrun status bit are set simultaneously.
2	EI	r	0x00	Error interrupt 1: Set: this bit will be set when the error interrupt is enabled and the error status bit or the bus status bit changes. 0: Reset: any read access by the microcontroller will clear this bit
1	TI	r	0x00	Transmit interrupt 1: Set: this bit is set when the transmit buffer status changes from 0 to 1 (release) and the transmit interrupt is enabled. 0: Reset: any read access by the microcontroller will clear this bit
0	RI	r	0x00	Receive interrupt 1: Set: this bit is set when the receive FIFO is not null and the receive interrupt is enabled. 0: Reset: any read access by the microcontroller will clear this bit. The receive interrupt bit (interrupt enabled) and the receive buffer status bit are set simultaneously. It must be noted that the receive interrupt bit is cleared in the reading process even if there is other information available in the FIFO. When the release receive buffer command is executed, and there is other available information in the receive buffer, the receive interrupt (interrupt enabled) is reset at the next t_{SCL}.

17.6.6 CAN interrupt enable register(CAN_IER)

Only applicable to PeliCAN mode

Address offset: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								BEIE	ALIE	EPIE	Res.	DOIE	EIE	TIE	RIE
								rw	rw	rw		rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	BEIE	rw	0x00	Bus error interrupt enable 1: Enabled; if a bus error is detected, the CAN controller requests the corresponding interrupt 0: Disabled
6	ALIE	rw	0x00	Arbitration lost interrupt enable 1: Enabled: if the CAN controller has lost arbitration, the corresponding interrupt will be enabled 0: Disabled
5	EPIE	rw	0x00	Error passive interrupt enable 1: Enabled: if the error status of the CAN controller changes (from negative to active or vice versa), the corresponding interrupt will be enabled 0: Disabled
4	Reserved			Reserved, always read as 0.
3	DOIE	rw	0x00	Data overrun interrupt enable 1: Enable: if the data overrun status bit is set (see the Status Register), the CAN controller will enable the corresponding interrupt 0: Disabled
2	EIE	rw	0x00	Error interrupt enable 1: Enabled: if the error or bus status changes (see the Status Register), the CAN controller will enable the corresponding interrupt 0: Disabled

Bit	Field	Type	Reset	Description
1	TIE	rw	0x00	Transmit interrupt enable 1: Enabled: when the message is successfully transmitted or the transmit buffer is accessible (for example, after aborting the transmit command), the CAN controller will enable the corresponding interrupt. 0: Disabled
0	RIE	rw	0x00	Receive interrupt enable 1: Enabled: when the receive buffer status is 'full', the CAN controller will enable the corresponding interrupt. 0: Disabled

17.6.7 CAN acceptance code register(CAN_ACR)

BasicCAN mode: CAN_ACR

Address offset: 0x10

Reset value: 0x0000 00XX

With the acceptance filter, the CAN controller allows the RXFIFO to receive only the information that is consistent with the preset values in the identifier and the acceptance filter. The acceptance filter is defined by the acceptance code register and the acceptance mask register.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								AC							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
7 : 0	AC	rw	0xXX	Acceptance code: This register is accessible (read/write) when the reset request bit is tied high (current). If a message passes the acceptance filter test and the receive buffer has enough space, the descriptor and data are written to the RXFIFO in sequence. When the information is received correctly: Receive status bit is set to high (full) Receive interrupt enable bit is set to high (enable) Receive interrupt is set to high (interrupt enabled) The acceptance code bit (AC.7-AC.0) is equal to the upper 8 bits of the message identifier (ID.10-ID.3), with OR relationship (1) to the corresponding bit of the acceptance mask bit (AM.7-AM.0). Namely, it is received if the following equation is satisfied: $[(ID.10-ID.3)\equiv(AC.7-AC.0)] \vee (AM.7-AM.0)\equiv 11111111$

PeliCAN mode: There are four acceptance code registers: CAN_ACR0, CAN_ACR1, CAN_ACR2, CAN_ACR3

CAN_ACR0: Address offset: 0x40

Reset value: 0x00XX

CAN_ACR1: Address offset: 0x44

Reset value: 0x00XX

CAN_ACR2: Address offset: 0x48

Reset value: 0x00XX

CAN_ACR3: Address offset: 0x4C

Reset value: 0x00XX

Note: See the description of the peliCAN mode in the identifier filtering.

17.6.8 CAN acceptance mask register(CAN_AMR)

BasicCAN mode: CAN_AMR

Address offset: 0x14

Reset value: 0x0000 00XX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								AM							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	AM	rw	0xXX	Acceptance mask: This register can be accessed (read/write) if the reset request is high (current). The acceptance mask register defines the corresponding bit of the acceptance code register to be 'correlated' or 'effect-free' (i.e. any value) for the acceptance filter.

PeliCAN mode: There are four acceptance mask registers: CAN_AMR0, CAN_AMR1, CAN_AMR2, CAN_AMR3

CAN_AMR0: Address offset: 0x50

Reset value: 0x00XX

CAN_AMR1: Address offset: 0x54

Reset value: 0x00XX

CAN_AMR2: Address offset: 0x58

Reset value: 0x00XX

CAN_AMR3: Address offset: 0x5C

Reset value: 0x00XX

Note: See the description of the peliCAN mode in the identifier filtering.

17.6.9 CAN bus timing 0(CAN_BTR0)

Address offset: 0x18

Reset value: 0x0000 0000

Bus timing register 0 defines the baud rate preset (BRP) and synchronization jump width (SJW) values. This register can be accessed (read/write) when the reset mode is active.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								SJW		BRP					
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
7 : 6	SJW	rw	0x00	Synchronization jump width In order to compensate for the phase offset between the clock oscillators of the different bus controllers, any bus controller must be resynchronized at the edge of the associated signal currently being transmitted; the synchronization jump width defines the maximum number of clock cycles that can be shortened or extended per each bit cycle. $t_{SJW} = t_{SCL} \times (SJW + 1)$
5 : 0	BRP	rw	0x00	Baud rate prescaler The cycle of t_{SCL}, the CAN system clock, is programmable and determines the corresponding bit timing. The CAN system clock can be calculated by the following formula: $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ Where, t_{SCL} = the clock period of APB1

17.6.10 CAN bus timing 1(CAN_BTR1)

Address offset: 0x1C

Reset value: 0x0000 0000

Bus timing register 1 defines the length of each bit period, the location of the sampling points, and the number of samples at each sampling point. In the reset mode, this register allows read/write access.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								SAM	TESG2			TESG1			
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	SAM	rw	0x00	Sampling 1: Three times: the bus is sampled for three times; it is recommended to use the method on low/medium speed buses (A and B), which is beneficial for filtering the glitch on the bus. 0: Single: the bus is sampled once; the method is recommended on the high-speed bus (SAE C level)

Bit	Field	Type	Reset	Description
6 : 0	TSEG2 TSEG1	rw	0x00	Time segment 1 and Time segment 2 TSEG1 and TSEG2 determine the number of clocks per bit and the location of the sampling point, where: $t_{SYNCSEG}$ = 1 x t_{SCL} t_{TSEG1} = t_{SCL} × (8 × TSEG1.3 + 4 × TSEG1.2 + 2 × TSEG1.1 + TSEG1.0 + 1) t_{TSEG2} = t_{SCL} × (4 × TSEG2.2 + 2 × TSEG2.1 + TSEG2.1 + 1)

17.6.11 CAN transmit identifier register (CAN_TXID0)

In BasicCAN mode:

Address offset: 0x28

Reset value: 0x0000 00XX

Note: it is 0xFF in the reset mode

Transmit identifier register 0 defines the type and length of the transmitted frame, and enables read/write access only in the operating mode.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	IDx	rw	0xXX	CAN identifier byte 10 ~ 3 Note: In Peli mode, Bit [3:0] is DLC [3:0].

17.6.12 CAN transmit identifier register 1(CAN_TXID1)

Only applicable to BasicCAN mode:

Address offset: 0x2C

Reset value: 0x0000 00XX

Note: it is 0xFF in the reset mode.

Transmit identifier register 1 defines the type and length of the transmitted frame, and enables read/write access only in the operating mode.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 5	IDx	rw	0x0X	CAN identifier byte 2 ~ 0
4	RTR	rw	0x0X	Remote transmission request 1: Remote: CAN will send remote frames 0: Data: CAN will send data frame
3 : 0	DLCx	rw	0x0X	Data length code 0 ~ 8

Only applicable to BasicCAN mode:

Offset address: 0x30 ~ 0x4C

Reset value: 0x0000 0000

Transmit data register CAN_TXDR0 ~ 7: This register enables read/write access only in the operating mode, and the data format of the receive buffer is consistent with that of the transmit buffer.

17.6.13 CAN arbitration lost capture register(CAN_ALC)

Only applicable to PeliCAN mode:

Address offset: 0x2C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.											BITNO				
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 5	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
4 : 0	BITNO	rw	0x00	For values and functions, refer to the following table (Bit number) In case of arbitration lost, a corresponding arbitration lost interrupt (interrupt enable) is generated. At the same time, the current bit of the bit stream processor is captured and sent to the arbitration lost capture register. The contents of the register will not change until the user reads this value through software. Then, the capture mechanism is then activated again. When the interrupt register is read, the corresponding interrupt flag bit in the interrupt register is cleared. The new arbitration lost interrupt is not valid until the arbitration lost capture register is read once.

Table 67. Functions of Arbitration lost Capture Register's Bit 4 - Bit 0

Bit					Decimal value	Features
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	Arbitration lost in Bit 1 of the identifier
0	0	0	0	1	1	Arbitration lost in Bit 2 of the identifier
0	0	0	1	0	2	Arbitration lost in Bit 3 of the identifier
0	0	0	1	1	3	Arbitration lost in Bit 4 of the identifier
0	0	1	0	0	4	Arbitration lost in Bit 5 of the identifier
0	0	1	0	1	5	Arbitration lost in Bit 6 of the identifier
0	0	1	1	0	6	Arbitration lost in Bit 7 of the identifier
0	0	1	1	1	7	Arbitration lost in Bit 8 of the identifier
0	1	0	0	0	8	Arbitration lost in Bit 9 of the identifier
0	1	0	0	1	9	Arbitration lost in Bit 10 of the identifier
0	1	0	1	0	10	Arbitration lost in Bit 11 of the identifier

Bit					Decimal value	Features
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	1	0	1	1	11	Arbitration lost in SRTR bit (Note 2)
0	1	1	0	0	12	Arbitration lost in IDE bit
0	1	1	0	1	13	Arbitration lost in Bit 12 of the identifier (Note 3)
0	1	1	1	0	14	Arbitration lost in Bit 13 of the identifier (Note 3)
0	1	1	1	1	15	Arbitration lost in Bit 14 of the identifier (Note 3)
1	0	0	0	0	16	Arbitration lost in Bit 15 of the identifier (Note 3)
1	0	0	0	1	17	Arbitration lost in Bit 16 of the identifier (Note 3)
1	0	0	1	0	18	Arbitration lost in Bit 17 of the identifier (Note 3)
1	0	0	1	1	19	Arbitration lost in Bit 18 of the identifier (Note 3)
1	0	1	0	0	20	Arbitration lost in Bit 19 of the identifier (Note 3)
1	0	1	0	1	21	Arbitration lost in Bit 20 of the identifier (Note 3)
1	0	1	1	0	22	Arbitration lost in Bit 21 of the identifier (Note 3)
1	0	1	1	1	23	Arbitration lost in Bit 22 of the identifier (Note 3)
1	1	0	0	0	24	Arbitration lost in Bit 23 of the identifier (Note 3)
1	1	0	0	1	25	Arbitration lost in Bit 24 of the identifier (Note 3)
1	1	0	1	0	26	Arbitration lost in Bit 25 of the identifier (Note 3)
1	1	0	1	1	27	Arbitration lost in Bit 26 of the identifier (Note 3)
1	1	1	0	0	28	Arbitration lost in Bit 27 of the identifier (Note 3)
1	1	1	0	1	29	Arbitration lost in Bit 28 of the identifier (Note 3)

Bit					Decimal value	Features
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
1	1	1	1	0	30	Arbitration lost in Bit 29 of the identifier (Note 3)
1	1	1	1	1	31	Arbitration lost in RTR bit (Note 3)

Note: The number of the binary coded structure bits of arbitration lost; the RTR bit of the standard frame information; It is used only for extended frame information.

17.6.14 CAN error code capture register(CAN_ECC)

Only applicable to PeliCAN mode:

Offset address: 0x30

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ERRC		DIR	SEG				
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 6	ERRC	r	0x00	Error code 00: Bit error 01: Format error 10: Padding error 11: Other errors
5	DIR	r	0x00	Direction 1: RX: error occurrs during reception 0: TX: error occurrs during transmission

Bit	Field	Type	Reset	Description
4 : 0	SEG	r	0x00	Segment Segment 00010: ID.28-ID.21 00011: Start of frame 00100: SRTR bit 00101: IDE bit 00110: ID.20-ID.18 00111: ID.17-ID.13 01000: CRC sequence 01001: Reserved bit 0 01010: Data field 01011: Data length code 01100: RTR bit 01101: Reserved bit 1 01110: ID.4-ID.0 01111: ID.12-ID.5 10001: Activity error flag 10010: Abort 10011: Dominant (control) bit error 10110: Error Passive flag 10111: Error delimiter 11000: CRC delimiter 11001: Acknowledge channel 11010: End of frame 11011: Acknowledge delimiter 11100: Overrun flag Other: reserved

Note: The bit configurations reflect different error events for the current structure segment.

When an error occurs on the bus, it is forced to generate a corresponding error interrupt (interrupt enabled). At the same time, the current position of the bitstream processor is captured and sent to the error code capture register. Its content is unchanged until the user reads it through the software. After reading, the capture mechanism is activated again. In the process of accessing the interrupt register, the corresponding interrupt flag bit in the interrupt register is cleared; the new bus interrupt is not active until the capture register is read once.

17.6.15 CAN error warning limit register(CAN_EWLR)

Only applicable to PeliCAN mode:

Offset address: 0x34

Reset value: 0x0000 0096

Error warning limits are defined in this register, with the default value (reset value) of 0x96. In the reset mode, this register is readable/writable to the CPU, and it is read-only in the operating mode.

Note: EWLR may only be changed if it has previously entered the reset mode. It is not possible to change the error state (see the Status Register) and the error warning interrupt caused by the contents of the new register until the reset mode is canceled again.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								EWL							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	EWL	rw	0x96	Programmable error warning limit When only one error counter exceeds the error limit programming value, the error status bit is set.

17.6.16 CAN RX error count register(CAN_RXERR)

Only applicable to PeliCAN mode:

Offset address: 0x38

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								RXERR							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
7 : 0	RXERR	rw	0x00	RX error counter register It reflects the current value of the receive error counter; the register is initialized to 0 after a hardware reset. In the operating mode, it is read-only to the CPU. This register can only be accessed by writing in the reset mode. The RX error counter is initialized to 0 if a bus-off condition occurs. Writing this register is not valid during bus-off. Note: It is only possible to force the RX error counter to change by the CPU before entering the reset mode. The error status change (see Status Register), error warning, and error interrupts caused by the contents of the new registers may be invalid until the reset mode is canceled.

17.6.17 CAN TX error count register(CAN_TXERR)

Only applicable to PeliCAN mode:

Offset address: 0x3C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								TXERR							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.

Bit	Field	Type	Reset	Description
7 : 0	TXERR	rw	0x00	TX error counter register It reflects the current value of the transmit error counter. In the operating mode, this register is a read-only memory to the CPU; accessing this register is only possible in the reset mode, and the register is initialized to 0 after a hardware reset. If the bus is off, the TX error counter is initialized to 127, to calculate the minimum time defined by the bus (128 bus idle signals). Reading the TX error counter during this time will reflect the status information of the bus-off recovery. If the bus-off is active, the bus off flag will be cleared in the 0 ～254 unit of write access TXERR, and the controller waits for an 11-bit continuous hidden (weak) bit (bus idle) after the reset mode is cleared. Writing 255 to TXERR will initiate a bus-off event for the CPU driver. It is only possible to change the contents of the TX error counter caused by the CPU before entering the reset mode. Until the reset mode is canceled again, an error or bus status change (see status register), an error warning, and an error interrupt caused by the contents of the new register may be invalid. After exiting from the reset mode, like the cause of bus error, the new contents of TX counter is given and the bus-off is enabled by the same way, which means that the reset mode is enabled again, the TX error counter is initialized to 127, the RX counter is cleared, and all status and interrupt register bits concerned are set. The protocol-defined bus-off recovery sequence (waiting for 128 bus idle signals) can be performed by clearing the reset mode. If the reset mode is enabled before the bus is off (TXERR > 0), the bus-off status remains active and TXERR is locked.

17.6.18 CAN transmit frame information register(CAN_SFF)

Only applicable to PeliCAN mode:

Offset address: 0x40

Reset value: 0x0000 000X

The transmit frame information register sets the type of the frame to be sent and the data length. This register can be read/written only in the operating mode, and acts as the transmit register when being written, and as the receive register when being read, with the same data structure.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								FF	RTR	Res.		DLC.3	DLC.2	DLC.1	DLC.0
								rw	rw			rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	FF	rw	0x00	Frame format 1:EFF: CAN will send/receive extended frame format 0: SFF: CAN will send/receive standard frame format
6	RTR	rw	0x00	Remote transmission request 1: Remote: CAN will send/receive remote frames 0: Data: CAN will send/receive data frame
5 : 4	Reserved			Reserved, always read as 0.
3 : 0	DLC	rw	0x0X	Data length code bit The length of the transmit data area is 0 ~ 8.

17.6.19 CAN transmit identifier register 0 (CAN_TXID0)

Only applicable to PeliCAN mode:

Offset address: 0x44

Reset value: 0x0000 00XX

The transmit identifier register 0 sets the identifier of the frame. This register enables read/write access only in the operating mode, and acts as the transmit register when being written, and as the receive register when being read, with the same data structure.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	IDx	rw	0xXX	CAN identifier bit 28 ~ 21 For standard frames, an 11-bit identifier is formed with ID20 ~ ID18

17.6.20 CAN transmit identifier register 1(CAN_TXID1)

Only applicable to PeliCAN mode:

Offset address: 0x48

Reset value: 0x0000 00XX

The transmit identifier register 1 sets the identifier of the frame. This register enables read/write access only in the operating mode, and acts as the transmit register when being written, and as the receive register when being read, with the same data structure.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	IDx	rw	0xXX	CAN identifier bit 20 ~ 13

17.6.21 CAN transmit data register 0 (CAN_TXDATA0)

Only applicable to PeliCAN mode:

Offset address: 0x4C

Reset value: 0x0000 00XX

The transmit data register 0 is used send and save CAN data. This register enables read/write access only in the operating mode, and acts as the transmit register when being written, and as the receive register when being read, with the same data structure.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								DATA0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	DATA0	rw	0xXX	CAN transmit data 0 Extended frame ID12 ~ 5

17.6.22 CAN transmit data register 1(CAN_TXDATA1)

Only applicable to PeliCAN mode:

Offset address: 0x50

Reset value: 0x0000 00XX

The transmit data register 1 is used send and save CAN data. This register enables read/write access only in the operating mode, and acts as the transmit register when being written, and as the receive register when being read, with the same data structure.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								DATA1							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7 : 0	DATA1	rw	0xXX	CAN transmit data 1 Extended frame ID4 ~ 0, lower 3 bits meaningless

Only applicable to PeliCAN mode:

Offset address: 0x54 ~ 0x70

Reset value: 0x0000 00XX

The above offset addresses all poitns to CAN data registers. In case of the extended frame, CAN transmit data register 0 stores the remaining data in DATA2 and so on. These registers are written as a transmit register and read as a receive register, with the same data structure.

17.6.23 CAN clock divider register (CAN_CDR)

Offset address: 0x7C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								MODE	Res.						
								rw							

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Reserved, always read as 0.
7	MODE	rw	0x00	CAN mode If MODE is 0, the CAN controller operates in BasicCAN mode. Otherwise, it operates in PeliCAN mode, and is writable only in the reset mode.
6 : 0	Reserved			Reserved, always read as 0.

18 Serial peripheral interface (SPI)

Serial peripheral interface (SPI)

18.1 SPI description

The SPI interface is widely used for board-level communication between different devices, such as the extended serial Flash, ADC, etc. Devices from many IC manufacturers support the SPI interface.

The serial peripheral interface (SPI) allows full-duplex, synchronous, serial communication with external devices. Applications can communicate by querying status or SPI interrupts.

18.2 Main features

- Fully compatible with Motorola SPI specification
- Support DMA requests
- Full-duplex synchronous transfers on three lines
- 16-bit programmable baud rate generator
- Master or slave operation
- 8-byte FIFO receiving/transmitting
- In master mode, SPI clock reaches pclk/2 (pclk: APB clock); in slave mode, SPI clock is up to pclk/4
- Programmable clock polarity and phase
- Programmable data sequence, MSB first or LSB first
- Support one-host and multiple-slave operations
- Support simultaneous transmission and reception of 1 ~ 32-bit data
- In addition to 8-bit data transmission and reception, the remaining 1 ~ 32-bit data transmission and reception only supports LSB mode, not supporting MSB mode.
- Support 8 transmit buffers and receive buffers for corresponding configuration data bits (Data size)
- Interrupt drive operation
  - The transmitting end is null and the end overflows
  - Received data is valid, and data overflows at the receiving end
  - Complete reception in SPI master mode, and the transmitting end is null

18.3 Functional description

18.3.1 General

The block diagram of SPI is shown below

Figure 145. SPI Block Diagram

The SPI enables receiving and transmitting 1 ~ 32-bit data simultaneously. The SPI can be configured as the slave mode or the master mode in a host environment. Four possible timing relationships can be selected by configuring the clock polarity CPOL and phase CPHA. It supports programmable data sequence, i.e. MSB first or LSB first.

The same clock is used for the transmission and reception. Data is clocked on the rising or falling edge of the clock, latching the data on the opposite valid edge of SCLK. Since the SPI is used to exchange data, the data must be read after transferring even if the data is invalid. In SPI mode, the clock and polarity of the master shall be the same as that of the slave to be communicated.

Usually, the SPI is connected to external devices through four pins:

MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.

SCK: Serial Clock output for SPI masters and input for SPI slaves.

NSS: Slave select. This is an optional pin to select a master/slave device. This pin acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled and driven low if the SPI is in master mode. At this time, all NSS pins from devices connected to the Master NSS pin see a low level.

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode

and receive data in slave mode.

- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a master/slave device. This pin acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled and driven low if the SPI is in master mode. At this time, all NSS pins from devices connected to the Master NSS pin see a low level.

An example of interconnections between a single master and a single slave is illustrated below.

Figure 146. Single Master/Single Slave Application

The MOSI pins are connected together and the MISO pins are connected together. In this way, data is transferred serially between master and slave (most significant bit (MSB) first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CCTL register. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state, namely, the low level

between two transfers. If CPOL is set, the SCK pin has a high-level idle state, namely, the high level between two transfers.

If the CPHA (clock phase) bit is set, the first data bit is latched on the second edge on the SCK (falling edge if the CPOL bit is set; rising edge if the CPOL bit is reset), and the data bit received is sampled. The SPI changes the serial data during the first SCK clock transition (when the clock changes in the opposite direction of the idle state) and captures the data on the next edge.

If the CPHA (clock phase) bit is cleared, the first data bit is latched on the first edge on the SCK (falling edge if the CPOL bit is reset; rising edge if the CPOL bit is set), and the data bit received is sampled. The SPI captures the the serial data during the first SCK clock transition (the clock changes in the opposite direction of the idle state) and the data is changed on the next edge.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge. Figure 147 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin,the MISO pin, the MOSI pin are directly connected between the master and the slave device.

High-speed transmission

Considering the sensitivity to board-level delay in high-speed transmission mode, adjust the time for adjust the time of the transmitting phase and the received samples by setting TXEDGE and RXEDGE control bits in the SPI_CCTL register.

- In the slave mode, if TXEDGE is 1, the data is immediately sent to the data bus for high-speed mode (SPBRG = 4); when the bit is 0, the data is sent to the data bus after a valid clock edge for low-speed mode (SPBRG > 4).
- In master mode, if RXEDGE is 1, the data is sampled in the middle of the transmitted data bit; when the bit is 0, the data is sampled on the tail clock edge of the transmitted data bit (for high-speed mode)

1. The SPIEN bit must be cleared to disable the SPI before changing the CPOL/CPHA bit.
2. The master and slave must be configured as the same timing mode.
3. The idle state of SCK must be the same as the polarity specified by the SPI_CCTL register. When CPOL is 1, the SCK shall be set high in the idle state. When CPOL is 0, SCK shall be set low in the idle state.

Figure 147. Data Clock Timing Diagram

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFE bit in the SPI_CCTL Register. Each data frame is 7 or 8 bits long depending on the SPILEN bit in the SPI_CCTL register. The selected data frame format is applicable to transmission and/or reception.

In addition, the register SPI_EXTCTL can be configured, with the data frame length of 1 ~ 32 bits. Configuration is required to during the application: the DW8_32 bit of the SPI_CCTL register is set to ’0’, and the LSBFE bit of the SPI_CCTL register is set to ’1’

and the SPILEN bit is set to '1'. In conjunction with DMA data transmission, the data length of the DMA needs to be configured as 8 bits.

18.3.2 SPI slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The setting in the SPI_SPBRG register does not affect the data transfer rate.

Procedure

1. Set the SPILEN bit to define 7- or 8-bit data frame format.
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock. For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFE bit in the SPI_CCTL register) must be the same as the master device.
4. Clear the MDOE bit and set the SPIEN bit, making corresponding pins work in SPI mode. In this configuration, the MOSI pin is used as the data input, and MISO as data output.

Transmit sequence

The data byte is parallel-loaded into the transmitting buffer during the write operation.

The transmit sequence begins when the slave device receives the clock signal and the first data bit appears on its MOSI pin, then the first bit is sent. The remaining bits are loaded into the shift-register. The TX_INTF flag in the SPI_INTSTAT register is set on the transfer of data from the transmitting buffer to the shift register, and an interrupt is generated if the TXIEN bit in the SPI_INTEN register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to receiving buffer and the RX_INTF flag (SPI_INTSTAT register) is set.
- An Interrupt is generated if the RXIEN bit is set in the SPI_INTEN register.

After the last sampling clock edge, the RXNE bit is set, a copy of the data byte received in the shift register is moved to the receiving buffer. When the SPI_RXREG register is read, the SPI peripheral returns this buffered value.

18.3.3 SPI master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Define the serial clock baud rate through SPI_SPBRG register.
2. Select the CPOL and CPHA bits to define the phase relation between the data transfer and the serial clock.
3. Set the SPILEN bit to define 8- or 7-bit data frame format.
4. Configure the LSBFE bit in the SPI_CCTL register to define the frame format.

5. If data is only received and not sent, the SPI_RNDNR register shall be set, and the bytes to be received shall be defined.
6. The MDOE and SPIEN bits must be set.

In this configuration, the MOSI pin is a data output, the MISO pin is a data input, and NSS is the select signal output of slave device.

Transmit sequence

The transmit sequence begins when a byte is written in the transmitting buffer. The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI pin; MSB first or LSB first depends on the LSBFE bit in the SPI_CCTL register. The TX_INTF flag is set on the transfer of data from the transmitting buffer to the shift register, and an interrupt is generated if the TXIEN bit in the SPI_INTEN register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to receiving buffer and the RX_INTF flag (SPI_INTSTAT register) is set.
- An Interrupt is generated if the RXIEN bit is set in the SPI_INTEN register.

After the last sampling clock edge, the RXNE bit is set, a copy of the data byte received in the shift register is moved to the receiving buffer. When the SPI_RXREG register is read, the SPI peripheral returns this buffered value.

If only data is received and not transmitted, the RXMATCH_INTF bit is set to '1' after receiving the bytes defined by RXDNR, indicating that all data has been received, and the clock signal is no longer transmitted in Master mode.

18.3.4 Status flags

For convenient software operation, the application can monitor the state of the SPI bus with four current status flags and seven interrupt status flags. The current status flag is read-only and is automatically set and cleared by hardware; the interrupt status flag is set when an event occurs and generates a CPU interrupt when the interrupt is enabled, and it can be cleared by software.

The SPI is configured with 8-byte transmit buffer and receive buffer. The CPU enables reading and writing 1 or 4 bytes at a time according to the DW8_32-bit setting of SPI_GCTL. Based on the configuration of DW8_32, the transmit and receive buffers respectively have one-byte flag or a status flag of valid data.

Table 68. SPI Status

Classification	Status flag	Buffer and signal status
Interrupt status	TX_INTF	According to the DW8_32 setting, there is at least one space for valid data, enabling writing the transmitting data register for one time.

Interrupt status	RX_INTF	According to the DW8_32 setting, there is at least one space for valid data, enabling reading the transmitting data register for one time.
Interrupt status	UNDERRUN_INTF	Transmitting buffer is null and repeated transmission occurs
Interrupt status	RXOERR_INTF	Receiving buffer is non-null and overwritten
Interrupt status	RXMATCH_INTF	Non-null, the last data is transferred to the receiving buffer
Interrupt status	RXFULL_INTF	Receiving buffer is full and unable to receive new data
Interrupt status	TXEPT_INTF	Transmitting buffer is null and unable to send data
Current status	RXAVL_4BYTE	Receiving buffer is loaded with valid data more than 4 bytes
Current status	TXFULL	Transmitting buffer is full
Current status	TXEPT	Transmitting buffer is null
Current status	RXAVL	The receiving buffer is non-null and enables receiving at least one byte

When the TXTLF of the SPI_GCTL register is 00, TX_INTF is set when the transmitting buffer is configured with one or more free data spaces. When TXTLF is 01, TX_INTF is set when the transmitting buffer is configured with more than half of the free space.

When the RXTLF of the SPI_GCTL register is 00, RX_INTF is set when the receiving buffer is loaded with one or more valid data. When RXTLF is 01, RX_INTF is set when the receiving buffer is loaded with more than half of the valid data.

18.3.5 Baud rate setting

The baud rate is the frequency of the generated SCLK, which is typically the division of PCLK. The BRG is a 16-bit baud rate generator, and the SPBREG register is used to control the count period of the 16-bit counter.

The desired baud rate and f_{pclk} (frequency of the APB module) are given, and the value calculated from the formula shown in the table below is set to the SPBRG register. The X in the table below is equal to the value of the SPBRG register (2 ~ 65535).

Table 69. Baud Rate Formula

Mode	Formula
SPI mode	Baud rate = f_{pclk}/X

18.3.6 SPI communication using DMA

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the receive buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when DMAEN bit in the SPI_GCTL register is enabled. DMA requests of transmitting buffer and receive buffer are enabled by DMAEN.

- In transmission, a DMA request is issued if TXTLF in SPI_GCTL register is set to 00 and the transmitting buffer is configured with one or more free data spaces; when TXTLF is set to 01 and the transmitting buffer is configured with more than half free space, a DMA request is generated, enabling only one DMA transfer. In the process, the data size and that of transmitting buffer depend on DW8_32.
- In reception, a DMA transfer request is issued if RXTLF in SPI_GCTL register is set to 00 and the receive buffer is loaded with one or more valid data; when RXTLF is set to 01 and the receive buffer is loaded with more than half valid data, a DMA request is generated, enabling only one DMA transfer. In the process, the data size and that of receive buffer depend on DW8_32.

18.4 Register file and memory mapping description

Table 70. Overview of SPI Register

Offset	Acronym	Register Name	Reset	Section
0x00	SPI_TXREG	Transmit data register	0x00000000	section 18.4.1
0x04	SPI_RXREG	Receive data register	0x00000000	section 18.4.2
0x08	SPI_CSTAT	Current status register	0x00000001	section 18.4.3
0x0C	SPI_INTSTAT	Interrupt status register	0x00000000	section 18.4.4
0x10	SPI_INTEN	Interrupt enable register	0x00000000	section 18.4.5
0x14	SPI_INTCLR	Interrupt clear register	0x00000000	section 18.4.6
0x18	SPI_GCTL	Global control register	0x00000004	section 18.4.7
0x1C	SPI_CCTL	General-purpose control register	0x00000008	section 18.4.8
0x20	SPI_SPBRG	Baud rate generator register	0x00000002	section 18.4.9
0x24	SPI_RXDNR	Receive data count register	0x00000001	section 18.4.10
0x28	SPI_NSSR	Slave chip select register	0x000000FF	section 18.4.11
0x2C	SPI_EXTCTL	Data control register	0x00000008	section 18.4.12

18.4.1 Transmit data register(SPI_TXREG)

Offset address: 0x00

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:0	TXREG	rw	0x0000 0000	Transmit data register Valid data bits are controlled by DW8_32. 0: Lower 8 bits active 1:TXREG 31:0 active

18.4.2 Receive data register(SPI_RXREG)

Offset address: 0x04

Reset value: 0x0000 0000

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

Bit	Field	Type	Reset	Description
31:0	RXREG	r	0x0000 0000	Receive data register Valid data bits are controlled by DW8_32. 0: Lower 8 bits active 1: RXREG 31:0 active This register is readable and non-writable.

18.4.3 Current status register(SPI_CSTAT)

Offset address: 0x08

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RXFADDR				TXFADDR				RXAVL_4BYTE	TXFULL	RXAVL	TXEPT
				r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:12	Reserved			Always read as 0.
11:8	RXFADDR	r	0x00	Receive FIFO address
7:4	TXFADDR	r	0x00	Transmit FIFO address
3	RXAVL_4BYTE	r	0x00	Receive available 4 byte data message 1: Receive buffer is loaded with data more than 4 bytes 0: Receive buffer is loaded with data less than 4 bytes
2	TXFULL	r	0x00	Transmitter FIFO full status bit 1: Transmitting buffer full 0: Transmitting buffer not full
1	RXAVL	r	0x00	Receive available byte data message This bit is set when the receiver buffer receives data of one full byte. 1: Receiver buffer has received a valid byte of data 0: Receiver buffer is null This bit is read-only and is automatically set and cleared by hardware.
0	TXEPT	r	0x01	Transmitter empty bit The transmitter buffer and the transmit shift register are null. 0: The transmitter buffer is non-null This bit is read-only and is automatically set and cleared by hardware.

18.4.4 Interrupt status register(SPI_INTSTAT)

Offset address: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TXEPT_INTF	RXFULL_INTF	RX MATCH_INFT	RXO ERR_INFT	UNDE RRUN_INTF	RX_INTF	TX_INTF
									r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31:7	Reserved			Always read as 0.
6	TXEPT_INTF	r	0x00	Transmitter empty interrupt flag bit This bit is automatically reset by hardware, and can be cleared by writing TXEPT_ICLR bit in INTCLR register. 1: The transmitter buffer and TX shift register are null 0: The transmitter buffer is non-null Note: This bit is the interrupt status signal and TXEPT is the status signal.
5	RXFULL_INTF	r	0x00	RX FIFO full interrupt flag bit This bit is automatically reset by hardware, and can be cleared by writing RXFULL_ICLR bit in INTCLR register 1: RX buffer full 0: RX buffer not full
4	RXMATCH_INTF	r	0x00	Receive specified bytes interrupt flag bit(Receive data match the RXDNR number, the receive process will be completed and generate the interrupt) This bit is automatically reset by hardware, and can be cleared by writing RXMATCH_ICLR bit in INTCLR register. 1: Bytes specified by the RXDNR register are received 0: Bytes specified by the RXDNR register are not received
3	RXOERR_INTF	r	0x00	Receive overrun error interrupt flag bit This bit is automatically reset by hardware, and can be cleared by writing RXOERR_ICLR bit in INTCLR register. 1: Overrun error 0: No overrun error
2	UNDERRUN_INTF	r	0x00	SPI underrun interrupt flag bit This bit is automatically reset by hardware, and can be cleared by writing UNDERRUN_ICLR bit in INTCLR register. 1: Underrun error 0: No underrun error

Bit	Field	Type	Reset	Description
1	RX_INTF	r	0x00	Receive data available interrupt flag bit This bit is automatically reset by hardware, and can be cleared by writing RX_ICLR bit in INTCLR register. This bit is set when the receiver buffer receives data of one full byte. 1: Receiver buffer has received valid data 0: Receiver buffer is null
0	TX_INTF	r	0x00	Transmit FIFO available interrupt flag bit (sending data of one byte) This bit is automatically reset by hardware, and can be cleared by writing TX_ICLR bit in INTCLR register. 1: Transmitter buffer enabled 0: Transmitter buffer disabled

18.4.5 Interrupt enable register(SPI_INTEN)

Offset address: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TXEPT_IEN	RXFULL_IEN	RX MATCH _IEN	RXO ERR_ IEN	UNDE RRUN _IEN	RX_ IEN	TX_ IEN
									rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:7	Reserved			Always read as 0.
6	TXEPT_IEN	rw	0x00	Transmit empty interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled
5	RXFULL_IEN	rw	0x00	Receive FIFO full interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled
4	RXMATCH_ IEN	rw	0x00	Receive data complete interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled
3	RXOERR_ IEN	rw	0x00	Overrun error interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled

Bit	Field	Type	Reset	Description
2	UNDERRUN_IEN	rw	0x00	Transmitter underrun interrupt enable bit(SPI slave mode only) 1: Interrupt enabled 0: Interrupt disabled
1	RX_IEN	rw	0x00	Receive FIFO interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled
0	TX_IEN	rw	0x00	Transmit FIFO empty interrupt enable bit 1: Interrupt enabled 0: Interrupt disabled

18.4.6 Interrupt clear register

Offset address: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TXEPT_ICLR	RXFULL_ICLR	RX MATCH _ICLR	RXO ERR_ ICLR	UNDE RRUN _ICLR	RX_ ICLR	TX_ ICLR
									w	w	w	w	w	w	r

Bit	Field	Type	Reset	Description
31:7	Reserved			Always read as 0.
6	TXEPT_ICLR	w	0x00	Transmitter empty interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared
5	RXFULL_ICLR	w	0x00	Receiver buffer full interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared
4	RXMATCH_ICLR	w	0x00	Receive completed interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared
3	RXOERR_ICLR	w	0x00	Overrun error interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared
2	UNDERRUN_ICLR	w	0x00	Transmitter underrun interrupt clear bit (SPI slave mode only) 1: Interrupt cleared 0: Interrupt not cleared

Bit	Field	Type	Reset	Description
1	RX_ICLR	w	0x00	Receive interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared
0	TX_ICLR	r	0x00	Transmitter FIFO empty interrupt clear bit 1: Interrupt cleared 0: Interrupt not cleared

18.4.7 Global control register(SPI_GCTL)

Offset address: 0x18

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DW8_32	NSS	DMAEN	TXTLF		RXTLF		RXEN	TXEN	MODE	INTEN	SPIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:12	Reserved			Always read as 0.
11	DW8_32	rw	0x00	Valid byte or double-word data select signal 0: Lower 8 bits active 1: 32-bit data active Note: When using CPU or DMA, access data in the specified format.
10	NSS	rw	0x00	NSS select signal that from software or hardware 0: Controlled by the NSSR register value 1: Automatically controlled by hardware during data transmission
9	DMAEN	rw	0x00	DMA access mode enable 0: DMA mode disabled 1: DMA mode enabled

Bit	Field	Type	Reset	Description
8:7	TXTLF	rw	0x00	TX FIFO trigger level bit 00: The DMA request or transmit interrupt request is generated if the transmitting buffer is configured with 1 or more free data spaces. 01: The DMA request or transmit interrupt request is generated if the transmitting buffer is configured with more than half free spaces. 1x: Reserved Note: If DW8_32 is 0, one data space represents 1 byte; when it is 1, a data space represents 4 bytes.
6:5	RXTLF	rw	0x00	RX FIFO trigger level bit 00: The DMA request or receive interrupt request is generated if the receive buffer is loaded with 1 or more valid data. 01: The DMA request or receive interrupt request is generated if the receive buffer is loaded with more than half valid data. 1x: Reserved Note: If DW8_32 is 0, one valid data represents 1 byte; when it is 1, one valid data represents 4 bytes.
4	RXEN	rw	0x00	Receive enable bit 1: Reception enabled 0: Reception disabled and RX buffer cleared Note: txen must be set to 0 when the SPI only runs in master receive mode.
3	TXEN	rw	0x00	Transmit enable bit 1: Transmission enabled 0: Transmission disabled and TX buffer cleared Note: Transmission and receiving are completed simultaneously in master mode.
2	MODE	rw	0x01	Master mode bit 1: Master mode (serial clock generated by internal BRG) 0: Slave mode (serial clock from external master)
1	INTEN	rw	0x00	SPI interrupt enable bit 1: SPI interrupt enabled 0: SPI interrupt disabled
0	SPIEN	rw	0x00	SPI select bit 0: SPI disabled (reset state) 1: SPI enabled

18.4.8 General-purpose control register(SPI_CCTL)

Offset address: 0x1C

Reset value: 0x0000 0008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TX EDGE	RX EDGE	SPI LEN	LSB FE	CPOL	CPHA
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:6	Reserved			Always read as 0.
5	TXEDGE	rw	0x00	Transmit data edge select (slave mode) 1: Data is immediately sent to the data bus when the high-speed mode is available (SPBRG = 4). 0: The data is sent to the data bus after a valid clock edge when the low-speed mode is available (SPBRG > 4).
4	RXEDGE	rw	0x00	Receive data edge select (master mode) 1: Sample data at the tail clock edge of the transmit data bit (for high speed mode) 0: Intermediate sample data in the transmit data bit
3	SPILEN	rw	0x01	SPI character length bit This bit is active after DW8_32 is reset (DW8_32=0). 1: 8-bit data (default) 0: 7-bit data
2	LSBFE	rw	0x00	LSBFE: LSI first enable bit 1: Data transmission or reception lowest bit first 0: Data transmission or reception highest bit first
1	CPOL	rw	0x00	Clock polarity select bit 1: The clock is high in the idle state (between two transmissions) 0: The clock is low in the idle state (between two transmissions)
0	CPHA	rw	0x00	Clock phase select bit 1: The first data bit is sampled from the first clock edge 0: The first data bit is sampled from the second clock edge

18.4.9 Baud rate generator(SPI_SPBRG)

Offset address: 0x20

Reset value: 0x0000 0002

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
SPBRG															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	Reserved			Always read as 0.
15:0	SPBRG	rw	0x0002	SPI baud rate control register for baud rate Baud rate formula: Baud rate = fpclk/SPBRG ($f_{pclk}$is the APB clock frequency) Note: Do not write 0 and 1 to this register.

18.4.10 Receive data count register (SPI_RXDNR)

Offset address: 0x24

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDNR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:16	Reserved			Always read as 0.
15:0	RXDNR	rw	0x0001	The register is used to hold a count of to be received bytes in next receive process The value (1 by default) of this register is valid when the SPI is in master receive mode. This register value is modified by writing MCU. Note: Do not write 0 to this register.

18.4.11 Slave chip select register(SPI_NSSR)

Offset address: 0x28

Reset value: 0x0000 00FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															NSS
															rw

Bit	Field	Type	Reset	Description
15:1	Reserved			Always read as 0.
0	NSS	rw	0xFF	Chip select output signal in Master mode This bit is active-low, and is inactive in the slave mode. 0: Slave device selected 1: Slave device not selected

18.4.12 Data control register(SPI_EXTCTL)

Offset address: 0x2C

Reset value: 0x0000 0008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											EXTLEN				
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31:5	Reserved			Always read as 0.
4：0	EXTLEN	rw	0x08	This bit is used to control SPI data length. 0 0000：32 bit 0 0001：1 bit 0 0010：2 bit 0 0011：3 bit … 1 1100：28 bit 1 1101：29 bit 1 1110：30 bit 1 1111：31 bit Note: It is valid only when the DW8_32 bit of the SPI_GCTL register is set to ’0’, the LSBFE bit of the SPI_CCTL register is set to ’1’, and SPILEN is also set to ’1’.

19 I2C interface (I2C)

I2C interface (I2C)

19.1 I2C introduction

I2C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing.

The I2C bus is a two-wire serial interface, in which two-wire bit serial data (SDA) and serial clock (SCL) lines are used to transmit information between the devices connected to bus. Each device is configured with a unique address identification and can be used as a transmitter or receiver. Moreover, the device can also be considered as a master or slave during data transmission. The master is the device that initializes the data transfer of the bus and generates a clock signal enabling the transmission. At that time, any addressed device is considered as a slave.

I2C can operate in standard mode (data transmission rate: 0 ～100 Kbps) and fast mode (maximum data transmission rate: 400 Kbps).

19.2 I2C main features

- Parallel-bus2C protocol converter
- Half-duplex synchronous operation
- Act as Master or Slave
- Support 7-bit and 10-bit addresses
- Support standard mode (100 Kbps) and fast mode (400 Kbps)
- Generate Start, Stop, Resend Start, Acknowledge Signals for detection
- Only support one master device in master mode
- 2 bytes of transmitting and receive buffers respectively
- Provide burr-free circuit to SCLI and SDAI
- Support DMA operation
- Support interrupt and query operations

19.3 I2C protocol

19.3.1 Start and Stop conditions

When the bus is in the idle state, SCL and SDA are simultaneously tied high with the external pull-up resistor. Before the master enables the data transfer, a Start condition must be generated. When the SCL line is high, the SDA line switches from high to low, to indicate the Start condition. A Stop condition is generated when the master disables

the transfer. The SCL line is high and the SDA line switches from low to high, indicating a Stop condition.

The figure below shows the timing diagram for the Start and Stop conditions. During data transfer, SDA must remain stable when SCL is 1.

Figure 148. Start and Stop Conditions

19.3.2 Slave address protocol

I2C has two address formats: 7-bit address format and 10-bit address format.

7-bit address format

The following figure shows the first 7 bits (bit 7:1; slave address) of a byte transmitted after the Start condition (S), and the lowest bit (bit 0) is the data direction bit. When bit 0 is 0, it indicates that the master writes data to the slave; 1 represents that the master reads data from the slave.

Figure 149. 7-bit Address Format

10-bit address format

In the 10-bit address format, 2 bytes are transmitted to transfer the 10-bit address. The description of the first byte of the transmission bit is as follows: The first 5 bits (bit 7:3) are used to signal the next 10-bit transmission of the slave. The last two bits (bit 2:1) of the first byte are bit 9:8 of the slave address, and the lowest bit (bit 0) is the data direction bit

(R/W). The second byte of the data transferred is the lower eight bits of the 10-bit address.

The details are as follows:

Figure 150. 10-bit Address Format

The following table defines the special purpose and reserved addresses of the first byte of I2C:

Table 71. First Byte of I2C

Slave address	R/W bit	Description
0000 000	0	General call address: the data is sent to the receive buffer via I2C, so as to generate a general call interrupt
0000 000	1	Start byte
0000 001	X	CBUS address: I2C interface ignores this access
0000 010	X	Reserved
0000 011	X	Reserved
0000 1xx	X	Reserved
1111 1xx	X	Reserved
1111 0xx	X	10-bit slave addressing

19.3.3 Transmission and reception protocols

The master, master transmitter or receiver, initializes data transfer and sends or receives data from the bus. The slave, as a slave transmitter or slave receiver, responds to the master's request, sending or receiving data.

Master transmitter and slave receiver

All data is transmitted in byte format, with the unlimited bytes per transfer. When the master sends the address and R/W bit or the master sends a byte of data to the slave, the slave shall generate a response signal (ACK). When the slave receiver fails to generate an ACK response, the master will generate a Stop condition, to abort the transmission. When the slave fails to respond, SDA shall be set high, making the master generate a Stop condition.

When the master transmitter transmits the data, the slave receiver responds to the master transmitter by generating an ACK after each byte received, as shown in the following figure.

Figure 151. Master Transmitting Protocol

Master receiver and slave transmitter

When the master receives the data, as shown in the figure below, the master shall respond to the slave transmitter after receiving data of one byte, except for the last byte. In this way, the master receiver sends signal indicating whether it is the last byte to the slave transmitter. The slave transmitter shall release the SDA when detecting a NACK so that the master generates a Stop condition.

Figure 152. Master Receiving Protocol

When the master needs not to generate Stop conditions, to release the bus, a repeated Start condition can be generated, which is the same as the Start condition except that it is generated after the ACK. In the master mode, the I2C interface communicates with the same slave using different directions of transmission.

Start byte transmission protocol

The start byte transmission protocol is used by systems without dedicated I2C hardware module. When the I2C module is used as the master, the start byte output can be generated for the required slave at the beginning of each transfer.

The protocol consists of seven 0s and one 1, as shown in the following figure. The processor enables inquiring the bus with a low speed sampling flag 0 during the address phase. Once 0 is detected, the processor switches from the low-speed sampling mode to the normal-speed mode of the master.

Figure 153. Start Byte Transmission

The start byte procedure is as follows:

1. The master generates a starting condition
2. The master sends the start byte (0000 0001)
3. The master sends an ACK clock pulse (ACK)
4. No slave responds to the ACK signal
5. The master generates a repeated Start condition (RESTART)

The hardware I2C receiver needs not to respond to the start byte since it is a reserved address and the address is reset after RESTART.

19.3.4 Transmitting buffer management and generation of Start, Stop, and repeated Start conditions

In the master mode, the I2C module generates a Stop condition on the bus whenever the transmitter is null. If the repeated Start condition generation is enabled (RESTART = 1), a repeated Start condition is generated when the transfer direction changes from read to write or from write to read. If the repeated Start condition is disabled, a Start condition will be generated after the Stop condition.

The figure below shows the bits of the DR register.

Figure 154. DR Register

The timing diagram below describes the behavior of the I2C module in the master transmitting mode when the Tx FIFO becomes null.

Figure 155. Master Transmitting - Null Tx FIFO

The timing diagram below describes the behavior of the I2C module in the master receiving mode when the Tx FIFO becomes null.

Figure 156. Master Receiving - Null Tx FIFO

19.3.5 Multiple-master arbitration

The I2C bus is a multi-master bus. Arbitration is a process that multiple masters simultaneously attempt to control the bus, but only one of them is allowed to control the bus and the message is not damaged. Once one of the masters has controlled the bus, the other master can not control the bus until the master sends a Stop condition and sets the bus to the idle state.

Arbitration occurs on the SDA when the SCL line is high. If two or more masters attempt to send information to the bus, and if the other master generates a ”0”, the master that first generated a ”1” will lose the arbitration. Those that lost the arbitration continue to generate clock pulses until the end of the byte transfer. If each master attempts to address the same device, arbitration will continue during the data phase.

After detecting the arbitration lost, the I2C interface stops generating the SCL signal.

The following figure shows the bus timing for arbitration of two masters

Figure 157. Multiple-master Arbitration

19.3.6 Clock synchronization

When two or more masters attempt to transmit information on the bus simultaneously, they must arbitrate and synchronize the SCL clock. All masters generate their own clocks to transmit messages. The data is only active when the clock is high. Clock synchronization is achieved with ’AND’ connection of the SCL signal. When the master turns the SCL clock to 0, the master will calculate the SCL low time and set the SCL clock to 1 at the beginning of next clock cycle. However, the master will enter a wait state until the SCL clock changes to 1 if another master keeps SCL at 0.

All masters will calculate their high time, and those with the shortest high time will change SCL to 0. Then, the master will calculate the low time, and those with the longest low time will force other masters to enter the wait state, generating a synchronized SCL clock, as shown in the following figure.

Figure 158. Clock Synchronization of Multiple Masters

19.4 I2C operating mode

The I2C interface operates in one of four modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

Note: The I2C interface module can only operates in either master mode or slave mode, and not in both modes at the same time. Therefore, Bit 6 (DISSLAVE) and Bit 0 (MASTER) in register CR shall not be set to 0 and 1 respectively (or 1 and 0 respectively).

The functional block diagram of I2C is as follows:

Figure 159. I2C Functional Block Diagram

19.4.1 Slave mode

The following describes the procedure flow chart of the slave mode

Initial configuration

1. Write 0 to Bit 0 of ENR register, to disable I2C.
2. Configure the slave address by initializing the SAR register. The address is that the I2C interface responds to.
3. Configure the specified address format of CR register (set bit 3 to select the 7-bit or 10-bit address format). Write 0 to Bit 6 of the CR register (DISSLAVE) and write 0 to bit 0 (MASTER).
4. Write 1 to bit 0 in the ENR register, enabling the I2C interface module.

Single byte operation of slave transmitter

When the I2C interface is addressed by another I2C master and requests data, the I2C interface operates in the slave transmitter mode as follows:

1. Other I2C master devices initialize the I2C transfer, with the transmit address matching the slave address in the SAR register.
2. The I2C interface responds to the address sent, identifying the transmission direction is in the slave transmitter mode.
3. The I2C interface generates the RD_REQ interrupt (Bit 5 of Register RAWISR) and sets the SCL line low. The bus is always in waiting state until the software responds.

If the RD_REQ interrupt is masked (register IMR5 = 0), it is recommended that the CPU

periodically inquires the RAWISR register.

1. Setting bit 5 of RAWISR is equivalent to generating an RD_REQ interrupt.
2. The software shall meet the requirements for I2C transmission.
3. The time interval is usually around 10 SCL clock cycles. For example, at 400 kbps, the time interval is 25 us.
4. If the Tx FIFO is still loaded with data before receiving a read request, the I2C interface will generate a TX_ABRT interrupt (RAWISR6) and clear the data in the Tx FIFO.
5. The software writes the data to the DR register (its bit 8 is set to 0).
6. Software shall first clear RD_REQ and TX_ABRT interrupts in the RAWISR registers (bits 5 and 6 respectively)
7. The I2C interface releases SCL and sends a data byte.
8. The master device sends a repeated Start condition, to control the bus, or sends a Stop condition to release the bus.

Single byte operation of slave receiver

When the I2C interface is addressed by other master devices and data is sent, the I2C interface operates in the slave receiver mode, as follows:

1. Other I2C master devices initialize the I2C transfer, with the transmit address matching the slave address in the SAR register.
2. The I2C interface responds to the address sent, identifying the transmission direction is in the slave receiver mode.
3. The I2C interface receives the data sent by the master and stores it in the receive buffer.
4. The I2C interface generates an RX_FULL interrupt (RAWISR2). If the RX_FULL interrupt is masked (IMR2 = 0), it is recommended that the software periodically inquire the SR register. When bit 3 of SR register (RFNE) is 1, it is equivalent to the RX_FULL interrupt generated.
5. The software obtains the received data by reading bit 7:0 in the DR register.
6. The master device sends a repeated Start condition, to control the bus, or sends a Stop condition to release the bus.

Block transmission of slave

In the standard I2C protocol, all data is processed in single byte, and the program responds to the master's read request by writing a byte to the slave's Tx FIFO. When a slave (salve transmitter) receives a read request (RD_REQ) from the master (master receiver), at least one data is sent to the Tx FIFO of slave transmitter. This I2C interface module enables processing multiple data in the x FIFO, therefore, for the next read request, an interrupt is not needed to fetch data, thus greatly reducing the waiting time caused by each data interruption.

This mode only acts when the I2C interface is in slave transmitter mode. If the master transmitter responds to the data transferred by the slave transmitter, there is no data in the slave's TX FIFO; the I2C interface will set the SCL line of the I2C bus low until the read request interrupt (RD_REQ) is generated and the TX FIFO data is ready before releasing

the SCL line.

If the RX_REQ interrupt is masked (ISR5 = 0), the software can periodically inquire and read RAWISR register. When reading RAWISR5, returning to 1 is equivalent to generating the RX_REQ interrupt.

The RD_REQ interrupt is generated due to the read request, like an interrupt, it must be cleared when exiting the Interrupt Service Routine (ISR). One or more bytes of data can be written to the TX FIFO in an Interrupt Service Routine (ISR). In the process of transferring these bytes to the master, if the master responds to the last byte, the slave must generate the RD_REQ interrupt request again. This is because that the master requires more data.

If the master receives n bytes from the I2C interface, but the number of data written to the Tx FIFO by the program is greater than n, the slave will clear the Tx FIFO and ignore the extra words after transmitting data of required n bytes.

19.4.2 Main mode

Initial configuration

1. Disable the I2C interface by setting ENR0 = 0
2. Configure bit 2:1 of the CR register, to set the rate mode (standard mode and fast mode) for I2C operation. In addition, ensure bit 6 (DISSLAVE) is 1, and bit 0 (MASTER) is 1.
3. Write the I2C device address to the TAR register. Set this register, which can be configured as a broadcast address or start byte command.
4. Set ENR0, to enable the I2C interface.
5. Write the transferred data and the transfer direction to the DR register. If the DR register is configured before the I2C interface is enabled, data and commands will be lost since the buffer is cleared when the I2C interface is disabled.

By following the above steps, I2C interface will generate a Start condition and send the address byte data to the I2C bus.

Master transmitter and master receiver

The I2C interface supports dynamic switching of reads and writes. When transmitting data, write data to the lower byte (DR) of the I2C RX/TX data buffer and command register, and set the CMD bit to 0, to allow a write operation. For the next read command, it is unnecessary to set the low byte of the DR register, and the CMD bit shall be 1. If the transmitter's FIFO is null, the I2C module sets SCL low until the next command is written to the transmitter's FIFO.

Program flow chart

The following flow chart is a program example of the I2C interface used as a master:

Figure 160. Flow Chart of I2C Interface Master

19.4.3 I2C abort transmission

The ABRT control bit in the ENR register allows the software to disable the I2C bus before transmitting the command in the TX FIFO. In response to the ABORT request, the I2C module issues a Stop condition to the I2C bus while clearing the TX FIFO. The transfer of operation value can be aborted in the master mode.

Procedure

1. Stop writing new commands to the Tx FIFO (DR)
2. Disable the transmission of DMA by setting TDMAE = 0 when operating in DMA mode
3. Set the ABRT bit of ENR register to 1
4. Wait for TX_ABRT interrupt

19.5 Communication using DMA

The I2C interface supports data transmission and reception through DMA, namely, DMA transmission or DMA reception can be enabled separately by setting the corresponding bit in the DMA register. A DMA request will be generated when the data register becomes null during the transmission or becomes full upon reception. The DMA request must be acknowledged before the end of the current byte transfer.

Transmission using DMA

DMA mode can be enabled for transmission by setting the TXEN bit in the DMA register. Data will be loaded from a Memory area predefined to the DR register during data transfer.

Reception using DMA

DMA mode can be enabled for reception by setting the RDMAE bit in the DMA register. Data will be loaded from the DR register to a Memory area predefined during each data reception after DMA channel is configured by I2C.

19.6 I2C interrupt

The following table lists the I2C interrupt bits and how they are set and cleared. Some bits are set by hardware and cleared by software; the other bits are set and cleared by hardware.

Table 72. Set and Clear Interrupt Bits

Interrupt bit	Set by hardware/cleared by software	Set and cleared by hardware
GEN_CALL	√	x
START_DET	√	x
STOP_DET	√	x
ACTIVITY	√	x
RX_DONE	√	x
TX_ABRT	√	x
RD_REQ	√	x
TX_EMPTY	x	√
TX_OVER	√	x
RX_FULL	x	√
RX_OVER	√	x
RX_UNDER	√	x

The following figure depicts that interrupt bits in the interrupt register are set by hardware and cleared by software.

Figure 161. Interrupt Mechanism

19.7 I2C register description

Table 73. I2C Register Overview

Offset	Acronym	Register Name	Reset	Section
0x00	I2C_CR	I2C control register	0x0011	section 19.7.1
0x04	I2C_TAR	I2C destination address register	0x0055	section 19.7.2
0x08	I2C_SAR	I2C slave address register	0x0055	section 19.7.3
0x10	I2C_DR	I2C data command register	0x0001	section 19.7.4
0x14	I2C_SSHR	Standard mode I2C clock high counter register	0x0190	section 19.7.5
0x18	I2C_SSLR	Standard mode I2C clock low counter register	0x01D6	section 19.7.6
0x1C	I2C_FSHR	Fast mode I2C clock high counter register	0x0036	section 19.7.7
0x20	I2C_FSLR	Fast mode I2C clock low counter register	0x0082	section 19.7.8
0x2C	I2C_ISR	I2C interrupt status register	0x0000	section 19.7.9
0x30	I2C_IMR	I2C interrupt mask register	0x08FF	section 19.7.10
0x34	I2C_RAWISR	I2C RAW interrupt register	0x0000	section 19.7.11
0x38	I2C_RXTLR	I2C reception threshold	0x0000	section 19.7.12
0x3C	I2C_TXTLR	I2C transmission threshold	0x0000	section 19.7.13
0x40	I2C_ICR	I2C combined and independent interrupt clear register	0x0000	section 19.7.14
0x44	I2C_RX_UNDER	I2C clear RX_UNDER interrupt register	0x0000	section 19.7.15
0x48	I2C_RX_OVER	I2C clears RX_OVER interrupt register	0x0000	section 19.7.16

Offset	Acronym	Register Name	Reset	Section
0x4C	I2C_TX_OVER	I2C clear TX_OVER interrupt register	0x0000	section 19.7.17
0x50	I2C_RD_REQ	I2C clear RD_REQ interrupt register	0x0000	section 19.7.18
0x54	I2C_TX_ABRT	I2C clear TX_ABRT interrupt register	0x0000	section 19.7.19
0x58	I2C_RX_DONE	I2C clear RX_DONE interrupt register	0x0000	section 19.7.20
0x5C	I2C_ACTIV	I2C clear ACTIVITY interrupt register	0x0000	section 19.7.21
0x60	I2C_STOP	I2C clear STOP_DET interrupt register	0x0000	section 19.7.22
0x64	I2C_START	I2C clear START_DET interrupt register	0x0000	section 19.7.23
0x68	I2C_GC	I2C clear GEN_CALL interrupt register	0x0000	section 19.7.24
0x6C	I2C_ENR	I2C enable register	0x0000	section 19.7.25
0x70	I2C_SR	I2C status register	0x0006	section 19.7.26
0x74	I2C_TXFLR	I2C transmitter buffer level register	0x0000	section 19.7.27
0x78	I2C_RXFLR	I2C receiver buffer level register	0x0000	section 19.7.28
0x7C	I2C_HOLD	I2C SDA hold time register	0x0001	section 19.7.29
0x88	I2C_DMA	I2C DMA control register	0x0000	section 19.7.30
0x94	I2C_SETUP	I2C SDA setup time register	0x0064	section 19.7.31
0x98	I2C_GCR	I2C general call ACK register	0x0001	section 19.7.32

19.7.1 I2C control register(I2C_CR)

Offset address: 0x00

Reset value: 0x0011

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EMPINT	STOPINT	DISSLAVE	REPEN	MASTER10	SLAVE10	SPEED		MASTER
							rw	rw	rw	rw	r	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 9	Reserved			Always read as 0.
8	EMPINT	rw	0x00	This bit controls the generation of TX_EMPTY interrupt. Refer to the IC_RAW_INTR_STAT register for details.
7	STOPINT	rw	0x00	In the slave mode, whether a STOP_DET interrupt is generated. 1: STOP_DET interrupt is generated when the address matches 0: STOP_DET interrupt is generated regardless of the address match. This bit is only applicable to slave mode. Note: During the addressing of broadcasting address, if this bit is set, the slave will not generate a STOP_DET interrupt. The STOP_DET interrupt is generated only when the transmitted address matches the slave address.

Bit	Field	Type	Reset	Description
6	DISSLAVE	rw	0x00	This bit controls whether I2C has its slave disabled 0: Slave enabled 1: Slave disabled
5	REPEN	rw	0x00	Determines whether RESTART conditions may be sent when acting as a master 0: Disabled 1: Enabled When RESTART is disabled, the following functions cannot be performed when the I2C interface acts as a master: Send start byte Change the transmission direction in combined format mode Read data 10-bit address format The RESTART condition is replaced by sending a Stop condition first and then transmitting a Start condition. If the above operation is executed, bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register is set.
4	MASTER10	r	0x01	Address mode when acting as a master 0: 7-bit address format 1: 10-bit address format
3	SLAVE10	rw	0x00	When acting as a slave, this bit controls whether the I2C re- sponds to 7- or 10-bit addresses 0: 7-bit addressing address. The I2C interface ignores 10-bit addressing. For 7-bit addressing, only the lower 7 bits of IC_SAR register is compared. 1: 10-bit addressing address. I2C only responds to 10-bit addressing; the receiving address is compared with 10 bits of IC_SAR.
2 : 1	SPEED	rw	0x00	These bits control at which speed the I2C operates This configuration is only valid when the I2C interface operates in the master mode. 1: Standard mode (0 ~ 100 Kbps) 2: Fast mode (400 Kbps)
0	MASTER	rw	0x01	This bit controls whether the I2C master is enabled 0: Master disabled 1: Master enabled

The DISSLAVE (bit 6) and MASTER (bit 0) configurations are listed in the following table:

Table 74. DISSLAVE (Bit 6) and MASTER (Bit 0) Configurations

DISSLAVE CR[6]	MASTER CR[0]	Status
0	0	Slave device

DISSLAVE CR[6]	MASTER CR[0]	Status
0	1	Configuration error
1	0	Configuration error
1	1	Master device

19.7.2 I2C destination address register(I2C_TAR)

Offset address: 0x04

Reset value: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SPECIAL	GC	ADDR									
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 12	Reserved			Always read as 0.
11	SPECIAL	rw	0x00	This bit indicates whether the software executes a special command (general call or start byte command) 0: Ignore the 10-bit GC, and use the ADDR bit normally 1: Execute special I2C commands such as GC bit
10	GC	rw	0x00	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the I2C 0: General call address. Only perform write operation when sending a general call address. The I2C interface always operates in broadcast address mode until SPECIAL (bit 11) is cleared. 1: Start byte command
9 : 0	ADDR	rw	0x55	This is the target address for any master transaction When a broadcast address is sent, these bits can be ignored. To generate the start byte command, the CPU only needs to write these bits once.

19.7.3 I2C slave address register(I2C_SAR)

Offset address: 0x08

Reset value: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ADDR									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 10	Reserved			Always read as 0.
9 : 0	ADDR	rw	0x55	When the I2C interface operates in the slave mode, for slave memory address in 7-bit address format, ADDR [6:0] is only valid.

19.7.4 I2C data command register(I2C_DR)

Offset address: 0x10

Reset value: 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RESTART	STOP	CMD	DAT							
					w	w	w	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 11	Reserved			Always read as 0.
10	RESTART	w	0x00	Whether to generate a RESTART signal before transmission or reception This bit is only valid when IC_EMPTYFIFO_HOLD_MASTER_EN is set to ’1’ 1: If the RESTART signal is ”1”, the data is received or sent (according to a RESTART signal will be generated before data reception or transmission (depending on CMD), regardless of the previous command changes the direction of data transmission. If IC_RESTART_EN signal is set to ’0’, the STOP signal will follow the START signal 0: If the RESTART signal is set to ’1’, the RESTART signal is generated only when the previous command changes the direction of data transmission. If the RESTART signal is set to ’0’, the STOP signal will follow the START signal

Bit	Field	Type	Reset	Description
9	STOP	w	0x00	STOP: Whether to generate a STOP signal after transmission or reception This bit is only valid when IC_EMPTYFIFO_HOLD_MASTER_EN is set to '1' 1: A STOP signal is generated after the current byte, regardless of the Tx FIFO is null. If he Tx FIFO is not null, the master immediately issues a new transmission and bus arbitration signal. 0: A STOP signal is not generated after the current byte, regardless of the Tx FIFO is null. The master continues the current transmission (data transmission or reception depending on CMD). If the Tx FIFO is null, the master will set SCL low, to pend the bus until Tx FIFO receives new data
8	CMD	w	0x00	Control read or write operations in master mode 1: Read 0: Write When a command is sent to TX FIFO, this bit is used to distinguish between read and write commands. In the slave receiver mode, the write operation of this bit is ignored. In the slave transmitter mode, writing 0 indicates that the data of the DR register is ready to be sent.
7 : 0	DAT	rw	0x01	I2C bus data to be sent or received

19.7.5 Standard mode I2C clock high counter register(I2C_SSHR)

Offset address: 0x14

Reset value: 0x0190

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0190	SCL clock high period in standard mode of I2C interface Note: This register has a configurable value between 6 and 65525. This is because the I2C interface uses a 16-bit counter; the I2C bus is in the idle state when the counter value is SSHR + 10.

19.7.6 Standard mode I2C clock low counter register(I2C_SSLR)

Offset address: 0x18

Reset value: 0x01D6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x01D6	The minimum SCL clock low period is 8 in the I2C interface standard mode.

19.7.7 Fast mode I2C clock high counter register(I2C_FSHR)

Offset address: 0x1C

Reset value: 0x0036

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0036	SCL clock high period in fast mode of I2C interface This register is read-only and returns 0 when I2C operates in standard mode, with the minimum value of 6.

19.7.8 Fast mode I2C clock low counter register(I2C_FSLR)

Offset address: 0x20

Reset value: 0x0082

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 0	CNT	rw	0x0082	SCL clock low period in fast mode of I2C interface This register is read-only and returns 0 when I2C operates in standard mode, with the minimum value of 8.

19.7.9 I2C interrupt status register(I2C_ISR)

Offset address: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		D	RESTART	GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 14	Reserved			Always read as 0.
13 : 0	ISR	r	0x0000	Refer to the RAWISR register for specific description of each bit

19.7.10 I2C interrupt mask register(I2C_IMR)

Offset address: 0x30

Reset value: 0x08FF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 12	Reserved			Always read as 0.
11 : 0	IMR	rw	0x08FF	Each bit is masked and mapped to the ISR.

19.7.11 I2C RAW interrupt register(I2C_RAWISR)

Offset address: 0x34

Reset value: 0x0000

The difference between RAWISR and ISR registers is that the former is not masked.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GC	START	STOP	ACTIV	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
				r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 12	Reserved			Always read as 0.
11	GC	r	0x00	General call This bit is set when a general call address is received. The I2C interface is disabled or cleared when the CPU reads the GC register. I2C stores the received data in the receiver buffer.

Bit	Field	Type	Reset	Description
10	START	r	0x00	Start condition detection Regardless of the I2C interface operates in the master or slave mode, this bit is set once the Start or repeated Start condition is detected on the I2C interface.
9	STOP	r	0x00	Stop condition detection The status of this bit is based on the status of the STOPINT in the CR register when STOPINT = 0 Regardless of the I2C interface operates in the master or slave mode, this bit is set once the Stop condition is detected on the I2C interface.In slave mode, a STOP interrupt is generated regardless of whether the addressing is matched or not. When STOPINT = 1 In master mode (MASTER = 1), this bit shows if a Stop condition occurs on the I2C interface. In slave mode (MASTER = 0), a STOP interrupt is generated only when the slave address is matched successfully.
8	ACTIV	r	0x00	I2C interface is enabled, this bit is used to capture the active state of the I2C module After being set, this bit can only be cleared by the following four methods: Disable I2C interface Read ACTIV register Rread ICR register System reset Once set, this bit can only be cleared by the above method. Even if I2C is idle, this bit also remains high until it is cleared.
7	RX_DONE	r	0x00	Transmit done When I2C is used as a slave transmitter, this bit will be set if the master fails to respond after sending one byte of data. This case happens at the last byte transferred, indicating the end of the transfer.
6	TX_ABRT	r	0x00	Transmit abort When the I2C interface acts as a transmitter, this bit is set if the data in the buffer is failed to be fully sent. Note: The transmit abort bit will clear the receiver and transmitter buffers in the I2C interface. The transmitter buffer is in a refresh state until the TX_ABRT register is read. Once the read operation is performed, the transmitter will receive new data from the APB bus.

Bit	Field	Type	Reset	Description
5	RD_REQ	r	0x00	Read request When I2C acts as a slave, other masters are set when attemptting to read data from the I2C interface. The I2C interface keeps the bus in a wait state (SCL = 0) until the interrupt is processed, which means that the I2C interface is successfully addressed by other masters as a slave and requires data transmission. The processor must respond to the interrupt and then write data to the DR register. This bit is cleared when the processor reads the RD_REQ register.
4	TX_EMPTY	r	0x00	Transmit buffer empty The status of this bit depends on the EMPINT state in the CR register: when EMPINT is 0 and the transmitter buffer is null, this bit is set; when EMPINT is 1, the transmitter buffer is null and the operation of internal shift register is finished, this bit is set This bit is automatically cleared by hardware when the transmitter buffer is not null.
3	TX_OVER	r	0x00	Transmit buffer over If the transmit buffer is full, and the processor writes new data, causing overflow, this bit will be set.
2	RX_FULL	r	0x00	Receive buffer not empty This bit is set when the receive buffer is not null. This bit will be cleared by hardware when the receive buffer is not null.
1	RX_OVER	r	0x00	Receive buffer over This bit will be set when the receive buffer is full and new data is received. The I2C interface will respond at this point, but new data will be lost.
0	RX_UNDER	r	0x00	Receive buffer under This bit will be set when the processor reads the DR register and the RX FIFO is null.

19.7.12 I2C reception threshold(I2C_RXTLR)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TL							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	TL	r	0x00	Receive FIFO threshold level This bit enables controlling the RX_FULL interrupt trigger.

19.7.13 I2C transmission threshold(I2C_TXTLR)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TL							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	TL	r	0x00	Receive FIFO threshold level This bit enables controlling the TX_EMPTY interrupt trigger.

19.7.14 I2C combined and independent interrupt clear register(I2C_ICR)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ICR
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	ICR	r	0x00	When this register is read, all combined interrupts and independent interrupts. Those that can be automatically cleared by hardware will not be cleared b this bit, and only those that can be cleared by software will be cleared.

19.7.15 I2C clear RX_UNDER interrupt register(I2C_RX_UNDER)

Offset address: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RX_UNDER
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	RX_UNDER	r	0x00	Clear RX_UNDER interrupt (RAWISR0) by reading this register.

19.7.16 I2C clears RX_OVER interrupt register(I2C_RX_OVER)

Offset address: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RX_OVER
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	RX_OVER	r	0x00	Clear RX_OVER interrupt (RAWISR1) by reading this register.

19.7.17 I2C clear TX_OVER interrupt register(I2C_TX_OVER)

Offset address: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															TX_OVER
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	TX_OVER	r	0x00	Clear TX_OVER interrupt (RAW_ISR3) by reading this register.

19.7.18 I2C clear RD_REQ interrupt register(I2C_RD_REQ)

Offset address: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RD_REQ
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	RD_REQ	r	0x00	Clear RD_REQ interrupt (RAW_ISR5) by reading this register.

19.7.19 I2C clear TX_ABRT interrupt register(I2C_TX_ABRT)

Offset address: 0x54

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															TX_ABRT
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	TX_ABRT	r	0x00	Clear the TX_ABRT interrupt (RAWISR6) by reading this register Release the TX FIFO from the refresh/reset state, to receive the written data.

19.7.20 I2C clear RX_DONE interrupt register(I2C_RX_DONE)

Offset address: 0x58

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RX_DONE
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	RX_DONE	r	0x00	Clear RX_DONE interrupt (RAWISR7) by reading this register.

19.7.21 I2C clear ACTIVITY interrupt register (I2C_ACTIV)

Offset address: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ACTIV
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	ACTIV	r	0x00	Read this register to clear ACTIV interrupt (RAWISR8) if the I2C bus is inactive If I2C is still active, the ACTIV interrupt will continue to be set. This bit is cleared by hardware when the I2C module is disabled or when the I2C bus is no longer active. The status of ACTIV (bit 8) in the RAWISR can be obtained by reading this register.

19.7.22 I2C clear STOP_DET interrupt register(I2C_STOP)

Offset address: 0x60

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															STOP
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	STOP	r	0x00	Clear STOP interrupt (RAWISR9) by reading this register.

19.7.23 I2C clear START_DET interrupt register(I2C_START)

Offset address: 0x64

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															START
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	START	r	0x00	Clear START interrupt (RAWISR10) by reading this register

19.7.24 I2C clear GEN_CALL interrupt register(I2C_GC)

Offset address: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															GC
															r

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	GC	r	0x00	Clear GC interrupt (RAWISR11) by reading this register

19.7.25 I2C enable register(I2C_ENR)

Offset address: 0x6C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ABORT	ENABLE
														rw	rw

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1	ABORT	rw	0x00	I2C transfer abort 0: The abort did not occur or has ended 1: Abort is in progress The I2C transfer can be aborted by software when the I2C module is set as a master. Once being set, this bit cannot be cleared immediately, the I2C module control logic will generate a STOP condition and clear the transmit buffer after completing the current transfer, and generates a TX_ABRT interrupt after the abort. This ABORT bit is automatically cleared after the abort operation.

Bit	Field	Type	Reset	Description
0	ENABLE	rw	0x00	I2C mode enable 0: Disable the I2C module (transmit and receive buffers remain erased) 1: Enable the I2C module

19.7.26 I2C status register(I2C_SR)

Offset address: 0x70

Reset value: 0x0006

This register is read-only and indicates the current transfer and buffer status. The status bits do not generate an interrupt.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SLV_ACTIV	MST_ACTIV	RFF	RFNE	TFE	TFNE	ACTIV
									r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 : 7	Reserved			Always read as 0.
6	SLV _ACTIV	r	0x00	Slave FSM activity status 0: The slave state machine is in the IDLE state, so the I2C slave part is inactive. 1: The slave state machine is not in the IDLE state, so the I2C slave part is active.
5	MST _ACTIV	r	0x00	Master FSM activity status 0: The master state machine is in the IDLE state, so the I2C master part is inactive. 1: The master state machine is not in the IDLE state, so the I2C master part is active.
4	RFF	r	0x00	Receive FIFO completely full 0: receive buffer not full 1: Receive buffer full
3	RFNE	r	0x00	Receive FIFO not empty 0: Receive buffer empty 1: Receive buffer not empty
2	TFE	r	0x01	Transmit FIFO completely empty 0: Transmit buffer not empty 1: Transmit buffer empty
1	TFNF	r	0x01	Transmit FIFO not full 0: Transmit buffer full 1: Transmit buffer not full
0	ACTIV	r	0x00	I2C activity status The MST_ACTIV bit has the OR relationship with SLV_ACTIV bit.

19.7.27 I2C transmitter buffer level register(I2C_TXFLR)

Offset address: 0x74

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CNT	
														r	r

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1 : 0	CNT	r	0x00	The number of valid data in the transmit buffer (0 ～2)

19.7.28 I2C receiver buffer level register(I2C_RXFLR)

Offset address: 0x78

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CNT	
														r	r

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1 : 0	CNT	r	0x00	The number of valid data in the receive buffer (0 ～2)

19.7.29 I2C SDA hold time register(I2C_HOLD)

Offset address: 0x7C

Reset value: 0x0001

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

Bit	Field	Type	Reset	Description
31 : 24	Reserved			Always read as 0.
23 : 16	RX_HOLD	r	0x00	SDA hold time when the I2C device acts as receiver, with the unit of APB1 system clock cycle

Bit	Field	Type	Reset	Description
15 : 0	TX_HOLD	r	0x01	SDA hold time when the I2C device acts as transmitter, with the unit of APB1 system clock cycle

19.7.30 I2C DMA control register(I2C_DMA)

Offset address: 0x88

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TXEN	RXEN
														rw	rw

Bit	Field	Type	Reset	Description
15 : 2	Reserved			Always read as 0.
1	TXEN	rw	0x00	Transmit DMA enable 0: transmit DMA disabled 1: Receive DMA enabled
0	RXEN	rw	0x00	Receive DMA enable 0: Receive DMA disabled 1: Receive DMA enabled

19.7.31 I2C SDA setup time register(I2C_SETUP)

Offset address: 0x94

Reset value: 0x0064

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CNT							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
15 : 8	Reserved			Always read as 0.
7 : 0	CNT	rw	0x64	SDA setup If the recommended delay time is 1000 nS, and the APB1 clock frequency is 10 MHz, the register is set to 11, with the minimum of 2.

19.7.32 I2C general call ACK register(I2C_GCR)

Offset address: 0x98

Reset value: 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															GC
															rw

Bit	Field	Type	Reset	Description
15 : 1	Reserved			Always read as 0.
0	GC	rw	0x01	ACK general call 1: Response after receiving a general call 0: No response and no interruption after receiving a general call

20 Universal asynchronous receiver transmitter (UART)

Universal asynchronous receiver transmitter (UART)

20.1 UART introduction

The universal asynchronous receiver transmitter (UART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The UART offers a very wide range of baud rates using a fractional baud rate generator. It supports synchronous one-way communication and half-duplex single wire communication as well as modem operations (CTS/RTS).

High speed data communication is possible by using the DMA for multibuffer configuration.

20.2 UART main features

- Support RS-232S protocol in asynchronous mode, and meet Industry Standard 16550
- Support DMA requests
- Full-duplex synchronous operation
- Fractional baud rate generator system
- Programmable baud rate shared by transmitter and receiver
- Separate transmit and receive buffer registers
- Embedded 1 byte transmit and 32 byte receive buffer
- low level first for data transmission and reception
- Starting from start bit, followed by a data bit (the output data length includes 5 bits, 6 bits, 7 bits, 8 bits), and ending with the stop bit. Alternatively, parity check bit is optional, which is set before the stop bit and after the data bit.
- Support hardware odd or even check, generation and detection
- Line break generation and detection
- Support hardware auto flow control
- Support the following interrupt sources:
  - Transmitter BUFFER empty
  - Receiver data valid
  - Receive buffer overflow
  - Frame error
  - Parity error
  - Receive break frame

20.3 UART functional description

At least two pins are required for any UART bidirectional communication: receive data input (RX) and transmit data output (TX).

RX: Receive data serial input. Data is recovered by oversampling techniques, to distinguish between data and noise.

TX: Transmit data output. When the transmitter is disabled, the output pin is returned to its I/O port configuration. The TX pin is high when the transmitter is activated and no data is transmitted.

- The bus shall be idle before transmission or reception
- One start bit
- One data word (5, 6, 7 or 8 bits) with the least significant bit first
- 1 or 2 stop bits, indicating the end of the data frame
- Use the fractional baud rate generator –a representation of 16-bit integers and 4bit decimals

The following pin is required in hardware flow mode:

- nCTS: Clear transmission. If it is high, it blocks the next data transmission at the end of the current data transmission.
- nRTS: Transmit request; the low level indicates that the UART is ready to receive data.

Figure 162. UART Block Diagram

20.3.1 UART character description

Word length may be selected as 5～8 bits by programming the CHAR bit in the UART_CCR register. The TX pin is in low state during the start bit, and is in high state 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 which contains data (the number of ‘1’ s will include the number of stop bits).

A Break character is interpreted on receiving ‘0’ s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic “1” bit) to acknowledge the start bit.

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 163. UART Timing

20.3.2 Transmitter

The transmitter can send data words of 5 ~ 8 bits depending on the CHAR bit status. When the transmit enable bit (TXEN) is set, the data in the transmit shift register is output on the TX pin.

Character transmission

During a UART transmission, data shifts out least significant bit first on the TX pin. In this mode, the UART_TDR register consists of a buffer between the internal bus and the transmit shift register.

Every character is preceded by a start bit. The character is terminated by a configurable number of stop bits.

The TXEN bit shall not be reset during transmission of data. Otherwise, the data on the TX pin will be corrupted as the baud rate counters will get frozen, and the current data being transmitted will

be lost.

Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed by setting SPB bits.

A break frame will be 10 low bits followed by the stop bits, or 11 low bits followed by the stop bits.The RXBRK_INTF bit in the interrupt status register will be set when the break frame is received.

Procedure

1. Enable the UART by writing the UARTEN bit in UART_GCR register to 1.
2. Program the CHAR bit in UART_CCR to define the word length.
3. Program the number of stop bits (SPB) in UART_CCR.
4. Set the TXEN bit in UART_GCR.
5. Select the desired baud rate using the UART_BRR register.
6. Write the data to be sent in the UART_TDR register (this clears the TX_INTF bit). Repeat Step 6 for each data to be transmitted in case of single buffer.

Single byte communication

The TX_INTF bit is always cleared by a write to the data register. The TX_INTF bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is cleared.
- The next data can be written in the UART_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXIEN bit is set. When a transmission is taking place in UART, a write instruction to the UART_TDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place in idle UART, a write instruction to the UART_TDR register places the data directly in the shift register, the data transmission starts, and the TX_INTF bit is immediately set. Meanwhile, TXBUF_EMPTY of UART_CSR is set. If a frame is transmitted (after the stop bit) and no new data is written to UART_TDR (TDR register null), the TXC bit will be set, indicating all transmission has been completed.

Figure 164. Status Bit Change During Transmission

Break character

Setting the BRK bit transmits a break character. If the BRK bit is set to '1', a break character is sent on the TX line after completing the current character transmission. This bit is reset by software when the break character is completed (during the stop bit of the break character). The UART inserts a logic 1 bit at the end of the last break frame, to guarantee the recognition of the start bit of the next frame.

20.3.3 Receiver

Character reception

During a UART reception, data shifts in least significant bit first through the RX pin. In this mode, the UART_RDR register consists of a buffer between the internal bus and the received shift register.

Procedure:

1. Enable the UART by writing the UARTEN bit in UART_GCR register to 1.
2. Program the CHAR bit in UART_CCR to define the word length.
3. Program the number of stop bits (SPB) in UART_CCR.
4. Select the desired baud rate using the UART_BRR register.
5. Set the RXEN bit UART_GCR. This enables the receiver which begins searching for a start bit.

When a character is received:

- The RX_INTF bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXIEN bit is set.
- The error flags can be set if a frame error or an overrun error has been detected during

reception.

- UART_RDR register is read by software, RX_INTF bit shall be cleared before the next character is received.

The RXEN bit shall not be reset while receiving data. If the RXEN bit is cleared during reception, the current byte to be received will be lost.

Break character

When a break frame is received, the UART is set and RXBRK_INTF interrupt is generated.

Overrun error

An overrun error occurs when a character is received before being read by UART_RDR.

When an overrun error occurs:

- The RXOERR_INTF bit is set.
- The RDR content will not be lost. The previous data is available when a read to UART_RDR is performed.
- The shift register will be overwritten. After that point, any data received will be lost.
- An interrupt is generated if the RXOERREN bit is set.

Frame error

The frame error is detected when the Stop bit is failed to be received and identified, then:

- RXFERR_INTF bit is set by hardware.
- Invalid data will not be transmitted to UART_RDR register from the shift register.
- An interrupt will be generated if RXFERREN bit is set.

20.3.4 Fractional baud rate generator

Set the BRR and FRA registers to set the corresponding baud rate. Refer to the following formula:

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times UARTDIV}$$

$$UARTDIV = BRR + \frac{FRA}{16}$$

Get:

$$f_{baudrate} = \frac{f_{PCLK}}{16 \times BRR + FRA}$$

The BRR register has a minimum value of 4.

20.3.5 Sampling

Since no separate clock is provided for asynchronous operation, the receiver requires synchronization to the receiver. In order to obtain the correct character data on the receive pin 'RX', the UART is configured with a detection circuit. The UART samples the RX pin

through a 'bclk16' clock with a 16-times data baud rate. Each data has 16 clock samples, and the sampled values of the 7th, 8th, and 9th falling edges are taken.

Figure 165. RX Pin Sampling Plan

20.3.6 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PEN bit in the UART_CCR register. The invalid data will not be transferred to UART_RDR register from the shift register in case of parity error.

Even parity: the parity bit is calculated to obtain an even number of "1s" inside the frame and the parity bit.

Example: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PSEL bit in UART_CCR = 1).

Odd parity: the parity bit is calculated to obtain an odd number of "1s" inside the frame and the parity bit.

Example: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PSEL bit in UART_CCR = 0).

Transmission mode: If the PEN bit is set in UART_CCR, 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 RXPERR_INTF flag is set in the UART_ISR register and an interrupt is generated if RXPERREN is preset.

20.3.7 Hardware flow control

It is possible to control the serial data flow between two devices by using the nCTS input and the nRTS output. The following figure shows how to connect two devices in this mode.

Figure 166. Hardware Flow Control Between Two UARTs

RTS and CTS flow control can be enabled by setting the AUTOFLOWEN bit in the UART_GCR.

RTS flow control

If the RTS flow control is enabled, then nRTS is active (tied low) as long as the UART receiver is ready to receive new data. When the receive register receives data, nRTS is released, indicating that the transmission is expected to stop at the end of the current frame. The following figure shows an example of communication with RTS flow control enabled.

Figure 167. RTS Flow Control

CTS flow control

If the CTS flow control is enabled, then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is active (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words), else the transmission does not occur. When nCTS is inactive during a transmission, the current transmission is completed before the transmitter stops. The figure below shows an example of communication with

CTS flow control enabled.

Figure 168. CTS Flow Control

20.3.8 Communication using DMA

The UART is capable of communicating using the DMA.

Transmission using DMA

During transmission using DMA, first configure the address of the UART_TDR register as the destination address of the DMA transfer in the DMA control register, configure the memory address as the source address of the DMA transfer, and configure the data volume. Then, enable DMA mode by setting the DMAMODE bit in the UART_GCR register. When the TXEN bit is set to '1', the DMA transfers data from the specified SRAM zone to the UART_TDR register.

Reception using DMA

During reception using DMA, first configure the address of the UART_RDR register as the source address of the DMA transfer in the DMA control register, configure the memory address as the destination address of the DMA transfer, and configure the data volume. Then, enable DMA mode by setting the DMAMODE bit in the UART_GCR register. When the RXEN bit is enabled, the DMA transfers data from the specified SRAM zone to the UART_RDR register.

20.4 UART interrupt requests

Table 76. UART interrupt requests

Interrupt event	Interrupt status	Enable bit
Transmit buffer null	TX_INTF	TXIEN
Valid data received	RX_INTF	RXIEN

Interrupt event	Interrupt status	Enable bit
Receive overrun error	RXOERR_INTF	RXOERREN
Parity error	RXPERR_INTF	RXPERREN
Frame error	RXFERR_INTF	RXFERREN
Break frame	RXBRK_INTF	RXBRKEN

These events generate an interrupt if the corresponding Enable Control Bit is set.

20.5 UART registers

Table 77. UART Register Overview

Offset	Acronym	Register Name	Reset	Section
0x00	UART_TDR	UART transmit data register	0x00000000	section 20.5.1
0x04	UART_RDR	UART receive data register	0x00000000	section 20.5.2
0x08	UART_CSR	UART current status register	0x00000009	section 20.5.3
0x0C	UART_ISR	UART interrupt status register	0x00000000	section 20.5.4
0x10	UART_IER	UART interrupt enable register	0x00000000	section 20.5.5
0x14	UART_ICR	UART interrupt clear register	0x00000000	section 20.5.6
0x18	UART_GCR	UART global control register	0x00000000	section 20.5.7
0x1C	UART_CCR	UART general control register	0x00000030	section 20.5.8
0x20	UART_BRR	UART baud rate register	0x00000001	section 20.5.9
0x24	UART_FRA	UART fractional baud rate register	0x00000000	section 20.5.10

20.5.1 UART transmit data register(UART_TDR)

Offset address: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXREG							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Always read as 0.
7 : 0	TXREG	rw	0x00	Transmit data register

20.5.2 UART receive data register(UART_RDR)

Offset address: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXREG							
								r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
31 : 8	Reserved			Always read as 0.
7 : 0	RXREG	r	0x00	Receive data register This register is read-only.

20.5.3 UART current status register(UART_CSR)

Offset address: 0x08

Reset value: 0x0000 0009

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TXBUF_EMPTY	TXFULL	RXAVL	TXC
												r	r	r	r

Bit	Field	Type	Reset	Description
31 : 4	Reserved			Always read as 0.
3	TXBUF_EMPTY	r	0x01	Transmit buffer empty flag bit 1 : Transmit buffer null 0 : Transmit buffer non-null
2	TXFULL	r	0x00	Transmit buffer full flag bit 1 : Transmit buffer full 0 : Transmit buffer non-null
1	RXAVL	r	0x00	Receive valid data flag bit This bit is set when the receive buffer receives data of one full byte. 1 : Receive buffer has received a complete and valid byte of data 0 : Receive buffer null

Bit	Field	Type	Reset	Description
0	TXC	r	0x01	Transmit complete flag bit 1 : Both the transmit buffer and the transmit shift register are null 0 : The transmit register is non-null

20.5.4 UART interrupt status register

Offset address: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RX BRK_ INTF	RXF ERR_ INTF	RXP ERR_ INTF	RXO ERR_ INTF	Res.	RX_ INTF	TX_ INTF
									r	r	r	r		r	r

Bit	Field	Type	Reset	Description
31 : 7	Reserved			Always read as 0.
6	RXBRK_ INTF	r	0x00	UART receive frame break interrupt flag bit After the abnormal stop bit, the RX pin receives 10 or more low levels for a period of time. 1 : Break frame detected 0 : No break frame detected
4	RXPERR_ INTF	r	0x00	Parity error interrupt flag bit 1 : Parity error detected 0 : No parity error detected
3	RXOERR_ INTF	r	0x00	Receive overflow error interrupt flag bit This bit is set only when autoflowen=0. 1 : Receive overrun error 0 : No overrun error
2	Reserved			Always read as 0.
1	RX_INTF	r	0x00	Receive valid data interrupt flag bit This bit is set when the receive buffer receives data of one full byte. 1 : Receive buffer receives valid byte data 0 : Receive buffer null
0	TX_INTF	r	0x00	Transmit buffer empty interrupt flag bit 1 : Transmit buffer null 0 : Transmit buffer non-null

20.5.5 UART interrupt enable register(UART_IER)

Offset address: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RX BRK EN	RXF ERR EN	RXP ERR EN	RXO ERR EN	Res.	RXIEN	TXIEN
									rw	rw	rw	rw		rw	rw

Bit	Field	Type	Reset	Description
31 : 7	Reserved			Always read as 0.
6	RXBRKEN	rw	0x00	Receive frame break interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled
5	RXFERREN	rw	0x00	Frame error interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled
4	RXPERREN	rw	0x00	Parity error interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled
3	RXOERREN	rw	0x00	Receive overflow error interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled
2	Reserved			Always read as 0.
1	RXIEN	rw	0x00	Receive buffer interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled
0	TXIEN	rw	0x00	Transmit buffer empty interrupt enable bit 1 : Interrupt enabled 0 : Interrupt disabled

20.5.6 UART interrupt clear register(UART_ICR)

Offset address: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RX BRK CLR	RXF ERR CLR	RXP ERR CLR	RXO ERR CLR	Res.	RXICLR	TXICLR
									w	w	w	w		w	w

Bit	Field	Type	Reset	Description
31 : 7	Reserved			Always read as 0.
6	RXBRKCLR	w	0x00	Receive frame break interrupt clear bit 1 : Interrupt cleared 0 : Interrupt not cleared
5	RXFERRCLR	w	0x00	Frame error interrupt enable bit 1 : Interrupt cleared 0 : Interrupt not cleared
4	RXPERRCLR	w	0x00	Parity error interrupt clear bit 1 : Interrupt cleared 0 : Interrupt not cleared
3	RXOERRCLR	w	0x00	Receive overflow error interrupt clear bit 1 : Interrupt cleared 0 : Interrupt not cleared
2	Reserved			Always read as 0.
1	RXICLR	w	0x00	Receive interrupt clear bit 1 : Interrupt cleared 0 : Interrupt not cleared
0	TXICLR	w	0x00	Transmit buffer empty interrupt clear bit 1 : Interrupt cleared 0 : Interrupt not cleared

20.5.7 UART global control register(UART_GCR)

Offset address: 0x18

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TXEN	RXEN	AUTO FLOW EN	DMA MODE	UARTEN
											rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 5	Reserved			Always read as 0.
4	TXEN	rw	0x00	Enable transmit 1 : Transmission enabled 0 : Transmission disabled and TX buffer cleared
3	RXEN	rw	0x00	Enable receive 1 : Reception enabled 0 : Reception disabled and RX buffer cleared
2	AUTO FLOWEN	rw	0x00	Automatic flow control enable bit 1 : Automatic flow control enabled 0 : Automatic flow control disabled
1	DMAMODE	rw	0x00	DMA mode selection bit 1 : Select DMA mode 0 : Select the normal mode
0	UARTEN	rw	0x00	UART mode selection bit 1 : UART module enabled 0 : UART mode disabled

20.5.8 UART general control register(UART_CCR)

Offset address: 0x1C

Reset value: 0x0000 0030

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CHAR		BRK	SPB	PSEL	PEN
										rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 6	Reserved			Always read as 0.
5 : 4	CHAR	rw	0x03	UART width bit 00: 5bits 01: 6bits 10: 7bits 11: 8bits
3	BRK	rw	0x00	UART transmit frame break 1 : Serial forced output logic ’0’ (break frame) 0 : Break disabled

Bit	Field	Type	Reset	Description
2	SPB	rw	0x00	Stop bit selection Set the transmit stop bits. The receiver usually detects a stop bit. 1 : 2 stop bits (5-bit data bit; SPB setting is not used, and stop bit is forced to 1 bit) 0 : 1 stop bit
1	PSEL	rw	0x00	Parity selection bit When the check is enabled, this bit is used to select to use either even or odd parity. 1 : Even parity 0 : Odd parity
0	PEN	rw	0x00	Parity enable bit 1 : Transmit and receive check enabled 0 : Check disabled

20.5.9 UART baud rate register(UART_BRR)

Offset address: 0x20

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 16	Reserved			Always read as 0.
15 : 0	DIV_Mantissa	rw	0x0001	The integer part of UARTDIV These 16 bits define the integer part of the UART divider division factor (UARTDIV). DIV_Mantissa Minimum value is 4

20.5.10 UART fractional baud rate register(UART_FRA)

Offset address: 0x24

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DIV_Fraction			
												rw	rw	rw	rw

Bit	Field	Type	Reset	Description
31 : 4	Reserved			Always read as 0.
3 : 0	DIV_Fraction	rw	0x00	The decimal part of UARTDIV These 4 bits define the decimal part of the UART divider division factor (UARTDIV).

21 Device electronic signature (Device)

Device electronic signature (Device)

The electronic signature is stored in the System memory area in the Flash memory module, and can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match microcontrollers with different configurations.

21.1 Memory size registers

21.1.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications).
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory.
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes (8 bits) /half-words (16 bits) /full words (32 bits) in different ways and then be concatenated using a custom algorithm.

21.2 UID register description

Table 78. Memory Capacity Register Description Overview

Offset	Acronym	Register Name	Reset	Section
0x00	UID1	Unique identification code	0xXXXXXXXX	section 21.2.1
0x02	UID2	Unique identification code	0xXXXXXXXX	section 21.2.2
0x04	UID3	Unique identification code	0xXXXXXXXX	section 21.2.3
0x08	UID4	Unique identification code	0xXXXXXXXX	section 21.2.4

21.2.1 UID1

Base address: 0x1FFF F7E8

Address offset: 0x00

Read-only, the value is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 :0	U_ID	r		U_ID: 15: 0 unique ID bits This field value is also reserved for a future feature.

21.2.2 UID2

Address offset: 0x02

Read-only, the value is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Field	Type	Reset	Description
15 :0	U_ID	r		U_ID: 31: 16 unique ID bits This field value is also reserved for a future feature.

21.2.3 UID3

Address offset: 0x04

Read-only, the value is factory-programmed

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

Bit	Field	Type	Reset	Description
31 :0	U_ID	r		U_ID: 63: 32 unique ID bits This field value is also reserved for a future feature.

21.2.4 UID4

Address offset: 0x08

Read-only, the value is factory-programmed

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

Bit	Field	Type	Reset	Description
31：0	U_ID	r		U_ID: 95: 64 unique ID bits This field value is also reserved for a future feature.

22 Debug Support (DBG)

Debug Support (DBG)

22.1 Overview

The core of the series contains hardware debugging modules for complex debugging operations. The hardware debugging modules 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 completed, the core and the system may be restored and program execution resumed.

The hardware debugging module is used by the debugger for relevant operations when it is connected to and used for debugging the microcontroller of the series.

Support for two debugging interfaces:

- Serial debug interface
- JTAG debug port

Figure 169. Block Diagram of MM32 Series Level and CPU Level Debug Support

The core provides integrated on-device debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface

22.2 SWJ debug port (serial wire and JTAG)

The chip core integrates the serial/JTAG debug port (SWJ-DP). This is the standard Core-Sight debug port, including the JTAG-DP (5 pins) and the SW-DP interface (2 pins).

- The JTAG Debug port (JTAG-DP) provides a 5-pin standard JTAG port for the AHP-AP module.
- The Serial Debug port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP module.

In the SWJ-DP interface, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins.

Figure 170. SWJ debug port

The above figure shows that the asynchronous trace output pin (TRACESWO) and TDO are multiplexed.This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

22.2.1 Mechanism to select the JTAG-DP or the SW-DP

The chip supports SW-DP and JTAG-DP debug interfaces. By default, the JTAG-Debug Port is active

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SW-DP using only the SWCLK and SWDIO pins.

The specified sequence is:

- Send more than 50 TCK cycles with TMS (SWDIO) =1
- Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)

- Send more than 50 TCK cycles with TMS (SWDIO) =1

22.3 Pinout and debug port pins

The device microcontroller is available in various packages with different numbers of available pins. As a result, some functionality related to pin availability may differ between packages.

22.3.1 SWJ Debug Port Pins

Five pins are used as outputs from the core for the SWJ-DP as alternate functions of general-purpose I/Os. These pins are available on all packages.

Table 79. SWJ debug port pin

SWJ-DP port pin Name	JTAG debug interface		SWDebug Interface		Pin assignment
	Type	Description	Type	Debugging Features	
JTMS/SWDIO	Input	JTAG Mode Selection	Input/Output	Serial Data Input/Output	PA13
JTCK/SWCLK	Input	JTAG Clock	Input	Serial Clock	PA14
JTDI	Input	JTAG Data Entry	-	-	PA15
JTDO/TRACESWO	Output	JTAG Data Output	-	Tracking is TRACESWO Signal	PB3
JNTRST	Input	JTAG Module Reset	-	-	PB4

22.3.2 Flexible SWJ-DP pin assignment

After reset (SYSRESETn or PORESETn), all five pins belonging to SWJ-DP are immediately initialized to dedicated pins that can be used by the debugger (note that the trace output pin is not initialized unless the debugger defines this pin)).

However, the family of microcontrollers can use the AFIO_MAPR register to disable the function of some or all of the pins of the SWJ-DP interface. These dedicated pins will be released for use as ordinary I/Os. This register is mapped to and The CPU system bus is connected to the APB bridge. The setting of this register will be done by the user code instead of the debugger.

Three control bits are used to configure the pins of the SWJ-DP interface. These three bits are reset on a system reset.

- AFIO_MAPR (the address in the microcontroller is 0x40010004)
- Read: APB, no wait state
- Write: APB, if the Write buffer of the AHB-APB bridge is full, then a wait state

Bit 26: 24=SWJ_CFG[2:0] is set and reset by software. These three bits are used to set the number of dedicated pins assigned to the SWJ debug interface. The purpose is to release as much as possible when using different debug interfaces. Multiple pins are used as ordinary I/Os.

The initial value after reset is 000 (all pins are set to JTAG-DP interface dedicated pins), and only one of the three bits can be set (no more than one bit can be set at the same time).

Table 80. Flexible SWJ_DP pin assignment

SWJ-CFG[2:0]	Configured as a debug-specific pin	I/O port assignment for SWJ interface				
		PA13/JTMS /SWDIO	PA14/JTCK /SWCLK	PA15/JTDI	PB3/JTDO	PB4/JNTRST
000	all SWJ pins (JTAG-DP + SW-DP) reset status	special	special	special	special	dedicated
001	all SWJ pins (JTAG-DP + SW-DP) except JNTRST pin	Dedicated	Dedicated	Dedicated	Dedicated	Release
002	JTAG-DP interface is disabled, SW-DP interface is allowed	Dedicated	Dedicated	release		
100	JTAG-DP interface and SW-DP interface are disabled	release				
Other	Forbidden					

When the Write Buffer of the APB Bridge is full, an additional APB period is required in the WriteAFIO_MAPR register. This is because the release of the JTAGSW pin requires 2 APB cycles to ensure smoothing of the nTRST and TCK signals of the input core.

- Cycle 1: Enter 1 / 0 of the JTAGSW signal to the core (nTRST, TDI and TMS are 1, TCK is 0).
- Cycle 2: The GPI/O controller obtains control signals for the SWJTAG I/O pins (eg, direction, pull-up/pull-down, Schmitt trigger, etc.).

22.3.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating because they are directly connected to the D flip-flops to control the debug mode. Special attention must be paid to the SWCLK/TCK pins because they are directly connected to the clock terminals of some D flip-flops. Par To avoid any uncontrolled I/O levels, the chip embeds internal pull-up and pull-down on the JTAG input pin.

- JINTRST: internal pull-up
- JTDI: internal pull-up
- JTMS/SWDIO: internal pull-up
- TCK/SWCLK: internal pull-down

Once the JTAG I/O is released by the user code, the GPIO controller takes control again. The state of these I/O ports will return to the state at reset.

- JINTRST: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- TCK/SWCLK: Input pull-down
- JTDO: Floating input

Software can use these I/Os as ordinary I/Os.

The JTAG IEEE standard recommends adding pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down. Having embedded pull-ups and pull-downs removes the need to add external resistors.

22.3.4 Using the serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must set SWJ_CFG=010 just after reset. This releases PA15, PB3 and PB4 which now become available as GPIOs. When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

For user software designs, note that:

To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

22.4 JTAG TAP Connection

The microcontroller integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the TAP (IR is 4-bit wide).

To access the TAP of the core for debug purposes:

- First, the BYPASS instruction must be shifted into the TMC TAP.
- Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
- For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

Important: Once Serial-Wire is selected using the dedicated JTAG sequence, the TMC TAP is automatically disabled (JTMS forced high).

Figure 171. JTAG TAP connection

22.5 ID codes and locking mechanism

There are several ID codes inside the device.

22.5.1 MCU device ID code

The MCU integrates an MCU ID code. This ID identifies the MCU part number and the die revision. It is part of the DBG_MCU component and is mapped on the external APB bus. This code is accessible using JTAG debug port(4 ~ 5 pins) or SW debug port(2 pins) or by the user code.

DBGMCU_IDCODE

Address:0x40007080 Only 32-bits access supported.

Read-only =0xXXXXXXXX, where X is a bit with undefined content.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

31: 0	DEV_ID: Device identifier

22.5.2 Boundary scan TAP JTAG ID code

The TAP of the MCU (boundary scan) integrates a JTAG ID code.

22.5.3 CPU JTAG TAP

The TAP of the CPU JTAG integrates a JTAG ID code. This ID code is the default one of the CPU and has not been modified. This code is only accessible by the JTAG Debug Port.

22.5.4 Cortex JEDEC-106 ID Code

The CPU integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00F F000_0xE00F FFFF.

The following table shows the individual ID codes for the series:

Table 82. chip ID encoding

ID name	chip
DEV_ID	0xCC88 xxx3
CPU TAP JTAG ID	0x4BA0 0477
CPU TAP SW ID	0x2BA0 1477

22.6 JTAG Debug Port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers.

Table 83. JTAG debug port data register

IR(3:0)	Data register	Description
1111	BYPASS[1Bit]	
1110	IDCODE[32Bit]	ID code register 0x4BA0 0477

IR(3:0)	Data register	Description
1010	DPACC[35Bit]	Debug Interface Register Initialize the debug port and allow access to the debug interface register When entering data: Bits34: 3 = DATA[31: 0]: 32-bit data to transfer for a write request Bits2: 1 = A[3: 2]: 2-bit address of a debug port register Bit0 = RnW: Read operation (1) or Write operation (0) When outputting data: Bits34: 3 = DATA[31: 0]: 32-bit data which is read following a read request Bits2: 0 = ACK[2: 0]: 3Bit response (3-bit Acknowledge) 010 = success/failure 001 = wait other = undefined
1011	APACC[35Bit]	Access Interface Register] Initialize the access interface and allow access to the access interface register When entering data: Bits34: 3 = DATA[31: 0]: 32-bit data to shift in for a write request Bits2: 1 = A[3: 2]: 2-bit address (sub-address AP registers) Bit0 = RnW: Read request (1) or write request (0) par When outputting data: Bits34: 3 = DATA[31: 0]: 32-bit data which is read following a read request Bits2: 0 = ACK[2: 0]: 3-bit Acknowledge 010 = success/failure 001 = wait other = undefined Refer to the AHB-AP chapter for the AP registers. The addresses of these registers consist of the following: A[3: 2] Shift value A[3: 2]. The current value of the DP SELECT register.
1000	ABORT [35Bit]	Abort Register Bits 31: 1 undefined Bit 0 = DAPABORT: write 1 to generate a DAP abort

Table 84. A[3: 2]Defined 32-bit debug interface register address

IR(3:0)	data register	Description
Address	A(3:2) Value	Description
0x0	00	undefined
0x4	01	DP CTRL/STAT register Request a system or debug power-on operation Configure the AP access mode of operation Control comparison, check operation Read takes some status bits (overflow, power-on response)

IR(3:0)	data register	Description
0x8	10	DP SELECT Register: Used to select the current access port and a valid 4-word register window Bits31: 24: APSEL select the current AP Bits23: 8: Undefined Bits7: 4: APBANKSEL: select the active 4-words register window on the current AP Bits3: 0: Undefined
0xC	11	DP RDBUFF Register: Used to get the debugger to get the final result of the previous operation (no need to request a new JTAG-DP operation)

22.7 SW debug port

22.7.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to. Bits are transferred LSB-first on the wire. For SWDIO bidirectional management, the pin must be pulled-up on the board (100 KΩ recommended). 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.

22.7.2 SW 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 85. Packet Request (8-bit)

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

Refer to the CPU TRM (Technical Reference Manual) for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drives the line.

Table 86. Packet Request (3-bit)

Bit	Name	Description
0..2	ACK	001: Fault 010: Wait 100: OK

The ACK 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 87. Packet Request (33-bit)

Bit	Name	Description
0..31	WDATA/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.

22.7.3 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.

The SW-DP state machine is inactive until the debugger reads this ID code.

- The SW-DP state machine will be in the RESET state, after a power-on reset, or after the DP switches from JTAG to SWD, or has a high level of more than 50 cycles.
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is mandatory to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the debugger will issue a FAULT acknowledge response on another transactions.

22.7.4 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 and AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the debugger acknowledge response is ”WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which is 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.

22.7.5 SW-DP register

Access to these registers are initiated when APnDP=0.

A(3: 2)	Read/write	CTRLSEL bit of SELECT register	Register	Description
00	Read		IDCODE	It is set to 0x1BA01477 (identifies the SW-DP)
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	Request a system or debug power-up; configure the transfer operation for AP accesses; control the compare and verify operations; read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	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	Select the current access port and the active 4-word 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

22.7.6 SW-AP register

Access to these registers are initiated when APnDP=1.

There are many AP Registers addressed as the combination of:

- A[3:2]
- The current value of the DP SELECT register

22.8 MCU debug module (MCUDBG)

The MCU debug module assists the debugger with the following features:

- Low power mode
- Provide timer at breakpoint, and enable the clock control of watchdog
- Control the assignment of trace pin

22.8.1 Debugg support in low power mode

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 code in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

22.8.2 Support timer, watchdog

When generating a breakpoint, it is necessary to select the operating mode of the counter based on the different uses of the timer and the watchdog:

- The counter continues to count when a breakpoint is generated. This is often used when outputting a PWM controlled motor.
- When a breakpoint is generated, the counter stops counting. This is required for the watchdog counter.

22.8.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- Trace pin assignment

This DBGMCU_CR is mapped on the External APB bus at address 0x40007084. It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

This DBGMCU_CR is mapped on the External APB bus at address 0x40013404. It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software

to write to these registers.

22.9 Description of DBG Register

Table 89. Summary of DBG Register

Offset	Acronym	Register Name	Reset	Section
0x00	DBG	DBG Control Register	0x00000000	section 22.9.1

22.9.1 DBG Control Register(DBG_CR)

Address: 0x40007084 Only 32-bits access supported.

POR Reset: 0x0000 0000 (not reset by system reset)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DBG_TIMx_STOP				DBG_WWDG_STOP	DBG_IWDG_STOP	Reserved					DBG_STANDBY	DBG_STOP	DBG_SLEEP
		w	w	w	w	w	w						w	w	w

Bit	Field	Type	Reset	Description
31:14	Reserved			Always read as 0.
13:10	DBG_TIMx_STOP	w	0x00	TIMx counter stopped when core is halted 0: The clock of the involved Timer Counter is fed even if the core is halted 1: The clock of the involved Timer counter is stopped when the core is halted
9	DBG_WWDG_STOP	w	0x00	Debug window watchdog stopped when core is halted 0: The window watchdog counter clock continues even if the core is halted 1: The window watchdog counter clock is stopped when the core is halted
8	DBG_IWDG_STOP	w	0x00	Debug independent watchdog stopped when core is halted 0: The watchdog counter clock continues even if the core is halted 1: The watchdog counter clock is stopped when the core is halted
7:3	Reserved			Always read as 0.

Bit	Field	Type	Reset	Description
2	DBG_STANDBY	w	0x00	Debug Standby mode 0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered. From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby) 1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset
1	DBG_STOP	w	0x00	Debug Stop mode 0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. 1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)
0	DBG_SLEEP	w	0x00	Debug Sleep mode 0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled. In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller. 1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

23 Revision history

Revision history

Table 90. Revision History

Date	Rversion	Changes
2019/09/03	Rev1.67	Modify the DMA channel description
2019/08/06	Rev1.66	Modify the typo at the memory and bus architecture
2019/07/23	Rev1.65	SPI_EXTCTL is only valid when the DW8_32 bit is ’0’
2019/07/16	Rev1.64	The minimum UART BRR register is 4
2019/05/09	Rev1.63	Modify port mode configuration Brake and deadband registers in the advanced timer BDTR deadband generator setting bit DTG Correction
2019/04/16	Rev1.62	Modify the adc calculation formula
2019/01/10	Rev1.61	Modify the PWR description
2018/12/22	Rev1.61	In the TIMX_16bit Function Description section, the Input Capture section modifies ”CC1S = 01 in the TIMx_CCR1 register” to ”CC1S = 01” in the TIMx_CCMR1 register. TIM1/8 replaces ”capture” in the text with ”capture”. In the TIM1/8 Function Description section, the Input Capture section modifies ”CC1S = 01 in the TIMx_CCR1 register” to ”CC1S=01 in the TIMx_CCMR1 register”. The synchronization part of the TIM1/8 TIMx timer and external trigger changes the incorrect writing ”IMx_CR1” to ”TIMx_CR1”. DIV_Mantiss cannot be 0 in UART_BRR.
2018/12/20	Rev1.60	UART_CSR changed to UART current status register