# GigaDevice Semiconductor Inc.

# GD32E23x
# Arm® Cortex®-M23 32-bit MCU

**For GD32E230xx, GD32E231xx, GD32E235xx, GD32E225xx**

# User Manual

Revision 2.6

(Jan. 2026)

# Table of Contents

**Table of Contents** ..... **2**

**List of Figures** ..... **13**

**List of Table** ..... **19**

**1. System and memory architecture** ..... **21**

**1.1. Arm® Cortex®-M23 processor** ..... **21**

**1.2. System architecture** ..... **22**

**1.3. Memory map** ..... **23**

1.3.1. On-chip SRAM memory ..... 26

1.3.2. On-chip Flash memory ..... 26

**1.4. Boot configuration** ..... **26**

**1.5. System configuration registers (SYSCFG)** ..... **28**

1.5.1. System configuration register 0 (SYSCFG_CFG0) ..... 28

1.5.2. EXTI sources selection register 0 (SYSCFG_EXTISS0) ..... 29

1.5.3. EXTI sources selection register 1 (SYSCFG_EXTISS1) ..... 30

1.5.4. EXTI sources selection register 2 (SYSCFG_EXTISS2) ..... 32

1.5.5. EXTI sources selection register 3 (SYSCFG_EXTISS3) ..... 33

1.5.6. System configuration register 2 (SYSCFG_CFG2) ..... 34

1.5.7. IRQ Latency register (SYSCFG_CPU_IRQ_LAT) ..... 35

**1.6. Device electronic signature** ..... **36**

1.6.1. Memory density information ..... 36

1.6.2. Unique device ID (96 bits) ..... 36

**2. Flash memory controller (FMC)** ..... **38**

**2.1. Overview** ..... **38**

**2.2. Characteristics** ..... **38**

**2.3. Function overview** ..... **38**

2.3.1. Flash memory architecture ..... 38

2.3.2. Read operations ..... 39

2.3.3. Unlock the FMC_CTL register ..... 40

2.3.4. Page erase ..... 40

2.3.5. Mass erase ..... 42

2.3.6. Main flash programming ..... 43

2.3.7. OTP programming ..... 45

2.3.8. Option byte erase ..... 46

2.3.9. Option byte programming ..... 46

2.3.10. Option byte description ..... 47

2.3.11. Page erase/Program protection ..... 48
2.3.12. Security protection ..... 48
**2.4. Register definition ..... 50**
2.4.1. Wait state register (FMC_WS) ..... 50
2.4.2. Unlock key register (FMC_KEY) ..... 50
2.4.3. Option byte unlock key register (FMC_OBKEY) ..... 51
2.4.4. Status register (FMC_STAT) ..... 51
2.4.5. Control register (FMC_CTL) ..... 52
2.4.6. Address register (FMC_ADDR) ..... 54
2.4.7. Option byte status register (FMC_OBSTAT) ..... 54
2.4.8. Write protection register (FMC_WP) ..... 55
2.4.9. Product ID register (FMC_PID) ..... 55
**3. Power management unit (PMU) ..... 56**
**3.1. Overview ..... 56**
**3.2. Characteristics ..... 56**
**3.3. Function overview ..... 56**
3.3.1. Backup domain ..... 57
3.3.2. $V_{DD}$ / $V_{DDA}$ power domain ..... 58
3.3.3. $V_{CORE}$ power domain ..... 60
3.3.4. Power saving modes ..... 60
**3.4. PMU registers ..... 63**
3.4.1. Control register (PMU_CTL) ..... 63
3.4.2. Control and status register (PMU_CS) ..... 64
**4. Reset and clock unit (RCU) ..... 67**
**4.1. Reset control unit (RCTL) ..... 67**
4.1.1. Overview ..... 67
4.1.2. Function overview ..... 67
**4.2. Clock control unit (CCTL) ..... 68**
4.2.1. Overview ..... 68
4.2.2. Characteristics ..... 70
4.2.3. Function overview ..... 70
**4.3. Register definition ..... 74**
4.3.1. Control register 0 (RCU_CTL0) ..... 74
4.3.2. Configuration register 0 (RCU_CFG0) ..... 75
4.3.3. Interrupt register (RCU_INT) ..... 79
4.3.4. APB2 reset register (RCU_APB2RST) ..... 82
4.3.5. APB1 reset register (RCU_APB1RST) ..... 83
4.3.6. AHB enable register (RCU_AHBEN) ..... 85
4.3.7. APB2 enable register (RCU_APB2EN) ..... 86
4.3.8. APB1 enable register (RCU_APB1EN) ..... 88

4.3.9. Backup domain control register (RCU_BDCTL) ..... 89
4.3.10. Reset source /clock register (RCU_RSTSCK) ..... 91
4.3.11. AHB reset register (RCU_AHBRST) ..... 92
4.3.12. Configuration register 1 (RCU_CFG1) ..... 93
4.3.13. Configuration register 2 (RCU_CFG2) ..... 94
4.3.14. Control register 1 (RCU_CTL1) ..... 95
4.3.15. Voltage key register (RCU_VKEY) ..... 96
4.3.16. Deep-sleep mode voltage register (RCU_DSV) ..... 96

**5. Interrupt / event controller (EXTI) ..... 98**

**5.1. Overview ..... 98**

**5.2. Characteristics ..... 98**

**5.3. Interrupts function overview ..... 98**

**5.4. External interrupt and event block diagram ..... 101**

**5.5. External interrupt and event function overview ..... 101**

**5.6. Register definition ..... 103**

5.6.1. Interrupt enable register (EXTI_INTEN) ..... 103
5.6.2. Event enable register (EXTI_EVEN) ..... 103
5.6.3. Rising edge trigger enable register (EXTI_RTEN) ..... 104
5.6.4. Falling edge trigger enable register (EXTI_FTEN) ..... 104
5.6.5. Software interrupt event register (EXTI_SWIEV) ..... 105
5.6.6. Pending register (EXTI_PD) ..... 106

**6. General-purpose and alternate-function I/Os (GPIO and AFIO) ..... 107**

**6.1. Overview ..... 107**

**6.2. Characteristics ..... 107**

**6.3. Function overview ..... 107**

6.3.1. GPIO pin configuration ..... 108
6.3.2. Alternate functions (AF) ..... 109
6.3.3. Additional functions ..... 109
6.3.4. Input configuration ..... 109
6.3.5. Output configuration ..... 110
6.3.6. Analog configuration ..... 110
6.3.7. Alternate function (AF) configuration ..... 111
6.3.8. GPIO locking function ..... 112
6.3.9. GPIO single cycle toggle function ..... 112

**6.4. Register definition ..... 113**

6.4.1. Port control register (GPIOx_CTL, x=A..C,F) ..... 113
6.4.2. Port output mode register (GPIOx_OMODE, x=A..C,F) ..... 114
6.4.3. Port output speed register (GPIOx_OSPD, x=A..C,F) ..... 116
6.4.4. Port pull-up/down register (GPIOx_PUD, x=A..C,F) ..... 118

6.4.5. Port input status register (GPIOx_ISTAT, x=A..C,F) ..... 119
6.4.6. Port output control register (GPIOx_OCTL, x=A..C,F) ..... 120
6.4.7. Port bit operate register (GPIOx_BOP, x=A..C,F) ..... 120
6.4.8. Port configuration lock register (GPIOx_LOCK, x=A,B) ..... 121
6.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x=A,B,F) ..... 122
6.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x=A,B,F) ..... 123
6.4.11. Bit clear register (GPIOx_BC, x=A..C,F) ..... 124
6.4.12. Port bit toggle register (GPIOx_TG, x=A..C,F) ..... 124

**7. Cyclic redundancy checks management unit (CRC) ..... 126**

**7.1. Overview ..... 126**

**7.2. Characteristics ..... 126**

**7.3. Function overview ..... 127**

**7.4. Register definition ..... 129**

7.4.1. Data register (CRC_DATA) ..... 129
7.4.2. Free data register (CRC_FDATA) ..... 129
7.4.3. Control register (CRC_CTL) ..... 130
7.4.4. Initialization data register (CRC_IDATA) ..... 131
7.4.5. Polynomial register (CRC_POLY) ..... 131

**8. Direct memory access controller (DMA) ..... 132**

**8.1. Overview ..... 132**

**8.2. Characteristics ..... 132**

**8.3. Block diagram ..... 133**

**8.4. Function overview ..... 133**

8.4.1. DMA operation ..... 133
8.4.2. Peripheral handshake ..... 135
8.4.3. Arbitration ..... 136
8.4.4. Address generation ..... 136
8.4.5. Circular mode ..... 136
8.4.6. Memory to memory mode ..... 136
8.4.7. Channel configuration ..... 137
8.4.8. Interrupt ..... 137
8.4.9. DMA request mapping ..... 138

**8.5. Register definition ..... 141**

8.5.1. Interrupt flag register (DMA_INTF) ..... 141
8.5.2. Interrupt flag clear register (DMA_INTC) ..... 141
8.5.3. Channel x control register (DMA_CHxCTL) ..... 142
8.5.4. Channel x counter register (DMA_CHxCNT) ..... 144
8.5.5. Channel x peripheral base address register (DMA_CHxPADDR) ..... 145
8.5.6. Channel x memory base address register (DMA_CHxMADDR) ..... 145

9. Debug (DBG) ..... 147
9.1. Overview ..... 147
9.2. SW function overview ..... 147
9.2.1. Pin assignment ..... 147
9.3. Debug hold function overview ..... 147
9.3.1. Debug support for power saving mode ..... 147
9.3.2. Debug support for TIMER, I2C, RTC, WWDGT and FWDGT ..... 148
9.4. Register definition ..... 149
9.4.1. ID code register (DBG_ID) ..... 149
9.4.2. Control register 0 (DBG_CTL0) ..... 149
9.4.3. Control register 1 (DBG_CTL1) ..... 151
10. Analog to digital converter (ADC) ..... 153
10.1. Overview ..... 153
10.2. Characteristics ..... 153
10.3. Pins and internal signals ..... 154
10.4. Function overview ..... 155
10.4.1. Foreground calibration function ..... 155
10.4.2. Dual clock domain architecture ..... 156
10.4.3. ADC enable ..... 156
10.4.4. Routine sequence ..... 156
10.4.5. Operation mode ..... 156
10.4.6. Conversion result threshold monitor function ..... 159
10.4.7. Data storage mode ..... 160
10.4.8. Sample time configuration ..... 161
10.4.9. External trigger configuration ..... 161
10.4.10. DMA request ..... 161
10.4.11. ADC internal channels ..... 162
10.4.12. Programmable resolution (DRES) - fast conversion mode ..... 162
10.4.13. On-chip hardware oversampling ..... 163
10.4.14. ADC interrupts ..... 165
10.5. Register definition ..... 166
10.5.1. Status register (ADC_STAT) ..... 166
10.5.2. Control register 0 (ADC_CTL0) ..... 166
10.5.3. Control register 1 (ADC_CTL1) ..... 168
10.5.4. Sample time register 0 (ADC_SAMPT0) ..... 169
10.5.5. Sample time register 1 (ADC_SAMPT1) ..... 170
10.5.6. Watchdog low threshold register (ADC_WDLT) ..... 171
10.5.7. Routine sequence register 0 (ADC_RSQ0) ..... 171
10.5.8. Routine sequence register 1 (ADC_RSQ1) ..... 172
10.5.9. Routine sequence register 2 (ADC_RSQ2) ..... 173

10.5.10. Routine data register (ADC_RDATA) ..... 173
10.5.11. Oversampling control register (ADC_OVSAMPCTL) ..... 174
**11. Comparator (CMP) ..... 176**
**11.1. Overview ..... 176**
**11.2. Characteristics ..... 176**
**11.3. Function overview ..... 176**
11.3.1. CMP clock ..... 177
11.3.2. CMP I / O configuration ..... 177
11.3.3. CMP operating mode ..... 178
11.3.4. CMP hysteresis ..... 178
11.3.5. CMP register write protection ..... 178
11.3.6. CMP interrupt ..... 179
**11.4. Register definition ..... 180**
11.4.1. CMP Control / status register (CMPx_CS) ..... 180
**12. Watchdog timer (WDGT) ..... 182**
**12.1. Free watchdog timer (FWDGT) ..... 182**
**12.1.1. Overview ..... 182**
**12.1.2. Characteristics ..... 182**
**12.1.3. Function overview ..... 182**
**12.1.4. Register definition ..... 185**
**12.2. Window watchdog timer (WWDGT) ..... 189**
**12.2.1. Overview ..... 189**
**12.2.2. Characteristics ..... 189**
**12.2.3. Function overview ..... 189**
**12.2.4. Register definition ..... 192**
**13. Real-time clock(RTC) ..... 194**
**13.1. Overview ..... 194**
**13.2. Characteristics ..... 194**
**13.3. Function overview ..... 195**
13.3.1. Block diagram ..... 195
13.3.2. Clock source and prescalers ..... 196
13.3.3. Shadow registers introduction ..... 196
13.3.4. Configurable and field maskable alarm ..... 196
13.3.5. RTC initialization and configuration ..... 197
13.3.6. Calendar reading ..... 198
13.3.7. Resetting the RTC ..... 199
13.3.8. RTC shift function ..... 199
13.3.9. RTC reference clock detection ..... 200
13.3.10. RTC smooth digital calibration ..... 201

13.3.11. Time-stamp function ..... 203
13.3.12. Tamper detection ..... 203
13.3.13. Calibration clock output ..... 204
13.3.14. Alarm output ..... 204
13.3.15. RTC power saving mode management ..... 205
13.3.16. RTC interrupts ..... 205
**13.4. Register definition ..... 206**
13.4.1. Time register (RTC_TIME) ..... 206
13.4.2. Date register (RTC_DATE) ..... 206
13.4.3. Control register (RTC_CTL) ..... 207
13.4.4. Status register (RTC_STAT) ..... 209
13.4.5. Prescaler register (RTC_PSC) ..... 211
13.4.6. Alarm 0 time and date register (RTC_ALRM0TD) ..... 211
13.4.7. Write protection key register (RTC_WPK) ..... 213
13.4.8. Sub second register (RTC_SS) ..... 213
13.4.9. Shift function control register (RTC_SHIFTCTL) ..... 213
13.4.10. Time of time stamp register (RTC_TTS) ..... 214
13.4.11. Date of time stamp register (RTC_DTS) ..... 215
13.4.12. Sub second of time stamp register (RTC_SSTS) ..... 216
13.4.13. High resolution frequency compensation register (RTC_HRFC) ..... 216
13.4.14. Tamper register (RTC_TAMP) ..... 217
13.4.15. Alarm 0 sub second register (RTC_ALRM0SS) ..... 219
13.4.16. Backup registers (RTC_BKPx) (x=0..4) ..... 220
**14. Timer (TIMERx) ..... 222**
**14.1. Advanced timer (TIMERx, x=0) ..... 223**
14.1.1. Overview ..... 223
14.1.2. Characteristics ..... 223
14.1.3. Block diagram ..... 224
14.1.4. Function overview ..... 225
14.1.5. TIMERx registers(x=0) ..... 252
**14.2. General level0 timer (TIMERx, x=2) ..... 280**
14.2.1. Overview ..... 280
14.2.2. Characteristics ..... 280
14.2.3. Block diagram ..... 280
14.2.4. Function overview ..... 281
14.2.5. TIMERx registers(x=2) ..... 295
**14.3. General level2 timer (TIMERx, x=13) ..... 319**
14.3.1. Overview ..... 319
14.3.2. Characteristics ..... 319
14.3.3. Block diagram ..... 319
14.3.4. Function overview ..... 321
14.3.5. TIMERx registers(x=13) ..... 329

**14.4. General level3 timer (TIMERx, x=14) ..... 339**
14.4.1. Overview ..... 339
14.4.2. Characteristics ..... 339
14.4.3. Block diagram ..... 339
14.4.4. Function overview ..... 340
14.4.5. TIMERx registers(x=14) ..... 357
**14.5. General level4 timer (TIMERx, x=15, 16) ..... 377**
14.5.1. Overview ..... 377
14.5.2. Characteristics ..... 377
14.5.3. Block diagram ..... 377
14.5.4. Function overview ..... 378
14.5.5. TIMERx registers(x=15, 16) ..... 392
**14.6. Basic timer (TIMERx, x=5) ..... 408**
14.6.1. Overview ..... 408
14.6.2. Characteristics ..... 408
14.6.3. Block diagram ..... 408
14.6.4. Function overview ..... 408
14.6.5. TIMERx registers(x=5) ..... 413
**15. Infrared ray port (IFRP) ..... 418**
**15.1. Overview ..... 418**
**15.2. Characteristics ..... 418**
**15.3. Function overview ..... 418**
**16. Universal synchronous/asynchronous receiver /transmitter (USART) ..... 420**
**16.1. Overview ..... 420**
**16.2. Characteristics ..... 420**
**16.3. Function overview ..... 422**
16.3.1. USART frame format ..... 422
16.3.2. Baud rate generation ..... 423
16.3.3. USART transmitter ..... 424
16.3.4. USART receiver ..... 425
16.3.5. Use DMA for data buffer access ..... 426
16.3.6. Hardware flow control ..... 428
16.3.7. Multi-processor communication ..... 429
16.3.8. LIN mode ..... 430
16.3.9. Synchronous mode ..... 431
16.3.10. IrDA SIR ENDEC mode ..... 432
16.3.11. Half-duplex communication mode ..... 433
16.3.12. Smartcard (ISO7816-3) mode ..... 433
16.3.13. ModBus communication ..... 435
16.3.14. Receive FIFO ..... 436

16.3.15. Wakeup from Deep-sleep mode ..... 436
16.3.16. USART interrupts ..... 437
**16.4. Register definition ..... 439**
16.4.1. Control register 0 (USART_CTL0) ..... 439
16.4.2. Control register 1 (USART_CTL1) ..... 441
16.4.3. Control register 2 (USART_CTL2) ..... 444
16.4.4. Baud rate generator register (USART_BAUD) ..... 447
16.4.5. Prescaler and guard time configuration register (USART_GP) ..... 447
16.4.6. Receiver timeout register (USART_RT) ..... 448
16.4.7. Command register (USART_CMD) ..... 449
16.4.8. Status register (USART_STAT) ..... 450
16.4.9. Interrupt status clear register (USART_INTC) ..... 453
16.4.10. Receive data register (USART_RDATA) ..... 455
16.4.11. Transmit data register (USART_TDATA) ..... 455
16.4.12. USART coherence control register (USART_CHC) ..... 456
16.4.13. USART receive FIFOcontrol and status register (USART_RFCS) ..... 456
**17. Inter-integrated circuit interface (I2C) ..... 458**
**17.1. Overview ..... 458**
**17.2. Characteristics ..... 458**
**17.3. Function overview ..... 458**
17.3.1. SDA and SCL lines ..... 459
17.3.2. Data validation ..... 460
17.3.3. START and STOP signal ..... 460
17.3.4. Clock synchronization ..... 460
17.3.5. Arbitration ..... 461
17.3.6. I2C communication flow ..... 461
17.3.7. Programming model ..... 462
17.3.8. SCL line stretching ..... 470
17.3.9. Use DMA for data transfer ..... 471
17.3.10. Packet error checking ..... 471
17.3.11. SMBus support ..... 471
17.3.12. SAM_V support ..... 473
17.3.13. Status, errors and interrupts ..... 473
**17.4. Register definition ..... 475**
17.4.1. Control register 0 (I2C_CTL0) ..... 475
17.4.2. Control register 1 (I2C_CTL1) ..... 477
17.4.3. Slave address register 0 (I2C_SADDR0) ..... 478
17.4.4. Slave address register 1 (I2C_SADDR1) ..... 479
17.4.5. Transfer buffer register (I2C_DATA) ..... 479
17.4.6. Transfer status register 0 (I2C_STAT0) ..... 479
17.4.7. Transfer status register 1 (I2C_STAT1) ..... 482

17.4.8. Clock configure register (I2C_CKCFG) ..... 483
17.4.9. Rise time register (I2C_RT) ..... 484
17.4.10. SAM control and status register (I2C_SAMCS) ..... 484
17.4.11. Fast mode plus configure register (I2C_FMPCFG) ..... 486
**18. Serial peripheral interface/Inter-IC sound (SPI/I2S) ..... 487**
**18.1. Overview ..... 487**
**18.2. Characteristics ..... 487**
**18.2.1. SPI characteristics ..... 487**
**18.2.2. I2S characteristics ..... 487**
**18.3. SPI function overview ..... 488**
**18.3.1. SPI block diagram ..... 488**
**18.3.2. SPI signal description ..... 488**
**18.3.3. SPI clock timing and data format ..... 489**
18.3.4. Separate transmission and reception FIFO ..... 491
**18.3.5. NSS function ..... 492**
**18.3.6. SPI operation modes ..... 493**
**18.3.7. DMA function ..... 503**
**18.3.8. CRC function ..... 504**
**18.3.9. SPI interrupts ..... 505**
**18.4. I2S function overview ..... 507**
**18.4.1. I2S block diagram ..... 507**
**18.4.2. I2S signal description ..... 507**
**18.4.3. I2S audio standards ..... 507**
**18.4.4. I2S clock ..... 516**
**18.4.5. Operation ..... 517**
**18.4.6. DMA function ..... 521**
**18.4.7. I2S interrupts ..... 521**
**18.5. Register definition ..... 523**
**18.5.1. Control register 0 (SPI_CTL0) ..... 523**
**18.5.2. Control register 1 (SPI_CTL1) ..... 525**
**18.5.3. Status register (SPI_STAT) ..... 527**
**18.5.4. Data register (SPI_DATA) ..... 529**
**18.5.5. CRC polynomial register (SPI_CRCPOLY) ..... 529**
**18.5.6. RX CRC register (SPI_RCRC) ..... 530**
**18.5.7. TX CRC register (SPI_TCRC) ..... 530**
**18.5.8. I2S control register (SPI_I2SCTL) ..... 531**
**18.5.9. I2S clock prescaler register (SPI_I2SPSC) ..... 533**
**18.5.10. Quad-SPI mode control register (SPI_QCTL) of SPI1 ..... 533**
**19. Operational amplifiers (OPA) ..... 535**
**19.1. Overview ..... 535**

19.2. Characteristics ..... 535
19.3. Function overview ..... 535
20. Appendix ..... 536
20.1. List of abbreviations used in register ..... 536
20.2. List of terms ..... 536
20.3. Available peripherals ..... 537
21. Revision history ..... 538

# List of Figures

Figure 1-1. The structure of the Cortex®-M23 processor ..... 22
Figure 1-2. Series system architecture of GD32E23x series ..... 23
Figure 2-1. Process of page erase operation ..... 41
Figure 2-2. Process of the mass erase operation ..... 43
Figure 2-3. Process of the word programming operation ..... 45
Figure 3-1. Power supply overview ..... 57
Figure 3-2. Waveform of the POR / PDR ..... 59
Figure 3-3. Waveform of the LVD threshold ..... 59
Figure 4-1. The system reset circuit ..... 68
Figure 4-2. Clock tree ..... 69
Figure 4-3. HXTAL clock source ..... 70
Figure 5-1. Block diagram of EXTI ..... 101
Figure 6-1. Basic structure of of a general-pupose I/O ..... 108
Figure 6-2. Basic structure of Input configuration ..... 110
Figure 6-3. Basic structure of Output configuration ..... 110
Figure 6-4. Basic structure of Analog configuration ..... 111
Figure 6-5. Basic structure of Alternate function configuration ..... 111
Figure 7-1. Block diagram of CRC calculation unit ..... 126
Figure 8-1. Block diagram of DMA ..... 133
Figure 8-2. Handshake mechanism ..... 135
Figure 8-3. DMA interrupt logic ..... 138
Figure 8-4. DMA request mapping ..... 139
Figure 10-1. ADC module block diagram ..... 155
Figure 10-2. Single operation mode ..... 156
Figure 10-3. Continuous operation mode ..... 157
Figure 10-4. Scan operation mode, continuous disable ..... 158
Figure 10-5. Scan operation mode, continuous enable ..... 158
Figure 10-6. Discontinuous operation mode ..... 159
Figure 10-7. Data storage mode of 12-bit resolution ..... 160
Figure 10-8. Data storage mode of 10-bit resolution ..... 160
Figure 10-9. Data storage mode of 8-bit resolution ..... 160
Figure 10-10. Data storage mode of 6-bit resolution ..... 161
Figure 10-11. 20-bit to 16-bit result truncation ..... 163
Figure 10-12. A numerical example with 5-bit shifting and rounding ..... 164
Figure 11-1. CMP block diagram ..... 176
Figure 11-2. CMP hysteresis ..... 178
Figure 12-1. Free watchdog block diagram ..... 183
Figure 12-2. Window watchdog timer block diagram ..... 190
Figure 12-3. Window watchdog timing diagram ..... 191
Figure 13-1. Block diagram of RTC ..... 195

Figure 14-1. Advanced timer block diagram ..... 224
Figure 14-2. Timing chart of internal clock divided by 1 ..... 225
Figure 14-3. Timing chart of PSC value change from 0 to 2 ..... 226
Figure 14-4. Timing chart of up counting mode, PSC=0/2 ..... 227
Figure 14-5. Timing chart of up counting mode, change TIMERx_CAR on the go ..... 228
Figure 14-6. Timing chart of down counting mode, PSC=0/2 ..... 229
Figure 14-7. Timing chart of down counting mode, change TIMERx_CAR on the go ..... 229
Figure 14-8. Timing chart of center-aligned counting mode ..... 231
Figure 14-9. Repetition counter timing chart of center-aligned counting mode ..... 232
Figure 14-10. Repetition counter timing chart of up counting mode ..... 232
Figure 14-11. Repetition counter timing chart of down counting mode ..... 233
Figure 14-12. Channel input capture principle ..... 234
Figure 14-13. Channel output compare principle (with complementary output, x=0,1,2) ..... 235
Figure 14-14. Channel output compare principle (CH3_O) ..... 235
Figure 14-15. Output-compare in three modes ..... 237
Figure 14-16. Timing chart of EAPWM ..... 238
Figure 14-17. Timing chart of CAPWM ..... 238
Figure 14-18. Complementary output with dead time insertion ..... 241
Figure 14-19. Output behavior of the channel in response to a break (the break high active) ..... 242
Figure 14-20. Counter behavior with CI0FE0 polarity non-inverted in mode 2 ..... 243
Figure 14-21. Counter behavior with CI0FE0 polarity inverted in mode 2 ..... 243
Figure 14-22. Hall sensor is used to BLDC motor ..... 244
Figure 14-23. Hall sensor timing between two timers ..... 245
Figure 14-24. Restart mode ..... 246
Figure 14-25. Pause mode ..... 246
Figure 14-26. Event mode ..... 247
Figure 14-27. Single pulse mode TIMERx_CHxCV=4, TIMERx_CAR=99 ..... 248
Figure 14-28. TIMER0 Master/Slave mode timer example ..... 249
Figure 14-29. Triggering TIMER0 with enable signal of TIMER2 ..... 250
Figure 14-30. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input ..... 251
Figure 14-31. General Level 0 timer block diagram ..... 281
Figure 14-32. Timing chart of internal clock divided by 1 ..... 282
Figure 14-33. Timing chart of PSC value change from 0 to 2 ..... 283
Figure 14-34. Timing chart of up counting mode, PSC=0/2 ..... 284
Figure 14-35. Timing chart of up counting mode, change TIMERx_CAR on the go. ..... 285
Figure 14-36. Timing chart of down counting mode, PSC=0/2 ..... 286
Figure 14-37. Timing chart of down counting mode, change TIMERx_CAR on the go. ..... 286
Figure 14-38. Timing chart of center-aligned counting mode ..... 288
Figure 14-39. Channel input capture principle ..... 289
Figure 14-40. Channel output compare principle (x=0,1,2,3) ..... 290
Figure 14-41. Output-compare under three modes ..... 291
Figure 14-42. Timing chart of EAPWM ..... 292
Figure 14-43. Timing chart of CAPWM ..... 293

Figure 14-44. General level2 timer block diagram ..... 320
Figure 14-45. Timing chart of internal clock divided by 1 ..... 321
Figure 14-46. Timing chart of PSC value change from 0 to 2 ..... 322
Figure 14-47. Timing chart of up counting mode, PSC=0/2 ..... 323
Figure 14-48. Timing chart of up counting mode, change TIMERx_CAR on the go ..... 323
Figure 14-49. Channel input capture principle ..... 324
Figure 14-50. Channel output compare principle ..... 325
Figure 14-51. Output-compare in three modes ..... 326
Figure 14-52. PWM mode timechart ..... 327
Figure 14-53. General level3 timer block diagram ..... 340
Figure 14-54. Timing chart of internal clock divided by 1 ..... 341
Figure 14-55. Timing chart of PSC value change from 0 to 2 ..... 342
Figure 14-56. Timing chart of up counting mode, PSC=0/2 ..... 343
Figure 14-57. Timing chart of up counting mode, change TIMERx_CAR on the go ..... 343
Figure 14-58. Repetition counter timing chart of up counting mode ..... 344
Figure 14-59. Channel input capture principle ..... 345
Figure 14-60. Channel output compare principle (with complementary output, x=0) ..... 346
Figure 14-61. Channel output compare principle (CH1_O) ..... 346
Figure 14-62. Output-compare in three modes ..... 348
Figure 14-63. PWM mode timechart ..... 349
Figure 14-64. Complementary output with dead-time insertion. ..... 351
Figure 14-65. Output behavior in response to a break(The break high active) ..... 352
Figure 14-66. Restart mode ..... 353
Figure 14-67. Pause mode ..... 354
Figure 14-68. Event mode ..... 354
Figure 14-69. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99 ..... 355
Figure 14-70. General level4 timer block diagram ..... 378
Figure 14-71. Timing chart of internal clock divided by 1 ..... 379
Figure 14-72. Timing chart of PSC value change from 0 to 2 ..... 380
Figure 14-73. Timing chart of up counting mode, PSC=0/2 ..... 381
Figure 14-74. Timing chart of up counting mode, change TIMERx_CAR on the go ..... 381
Figure 14-75. Repetition counter timing chart of up counting mode ..... 382
Figure 14-76. Channel input capture principle ..... 383
Figure 14-77. Channel output compare principle (with complementary output, x=0) ..... 384
Figure 14-78. Output-compare under three modes ..... 385
Figure 14-79. PWM mode timechart ..... 386
Figure 14-80. Complementary output with dead-time insertion. ..... 389
Figure 14-81. Output behavior in response to a break(The break high active) ..... 390
Figure 14-82. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60 ..... 391
Figure 14-83. Basic timer block diagram ..... 408
Figure 14-84. Timing chart of internal clock divided by 1 ..... 409
Figure 14-85. Timing chart of PSC value change from 0 to 2 ..... 410
Figure 14-86. Timing chart of up counting mode, PSC=0/2 ..... 411
Figure 14-87. Timing chart of up counting mode, change TIMERx_CAR on the go ..... 411

Figure 15-1. IFRP output timechart 1 ..... 419
Figure 15-2. IFRP output timechart 2 ..... 419
Figure 15-3. IFRP output timechart 3 ..... 419
Figure 16-1. USART module block diagram ..... 422
Figure 16-2. USART character frame (8 bits data and 1 stop bit) ..... 423
Figure 16-3.USART transmit procedure ..... 425
Figure 16-4.Oversampling method of a receive frame bit (OSB=0) ..... 426
Figure 16-5. Configuration step when using DMA for USART transmission ..... 427
Figure 16-6. Configuration step when using DMA for USART reception ..... 428
Figure 16-7. Hardware flow control between two USARTs ..... 428
Figure16-8. Hardware flow control ..... 429
Figure 16-9. Break frame occurs during idle state ..... 430
Figure 16-10. Break frame occurs during a frame ..... 431
Figure 16-11. Example of USART in synchronous mode ..... 431
Figure 16-12. 8-bit format USART synchronous waveform (CLEN=1) ..... 432
Figure 16-13. IrDA SIR ENDEC module ..... 432
Figure 16-14. IrDA data modulation ..... 433
Figure 16-15. ISO7816-3 frame format ..... 434
Figure 16-16. USART receive FIFO structure ..... 436
Figure 16-17. USART interrupt mapping diagram ..... 438
Figure 17-1. I2C module block diagram ..... 459
Figure 17-2. Data validation ..... 460
Figure 17-3. START and STOP signal ..... 460
Figure 17-4. Clock synchronization ..... 461
Figure 17-5. SDA line arbitration ..... 461
Figure 17-6. I2C communication flow with 7-bit address ..... 462
Figure 17-7. I2C communication flow with 10-bit address (Master Transmit) ..... 462
Figure 17-8. I2C communication flow with 10-bit address (Master Receive) ..... 462
Figure 17-9. Programming model for slave transmitting (10-bit address mode) ..... 464
Figure 17-10. Programming model for slave receiving (10-bit address mode) ..... 465
Figure 17-11. Programming model for master transmitting (10-bit address mode) ..... 466
9. Figure 17-12. Programming model for master receiving using Solution A (10-bit address mode) ..... 468
Figure 17-13. Programming model for master receiving mode using solution B (10-bit address mode) ..... 470
Figure 18-1. Block diagram of SPI ..... 488
Figure 18-2. SPI0 timing diagram in normal mode ..... 489
Figure 18-3. SPI1 timing diagram in normal mode ..... 490
Figure 18-4. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0) ..... 490
Figure 18-5. SPI1 data frame right-aligned diagram ..... 491
Figure 18-6. Transmission and reception FIFO ..... 491
Figure 18-7. A typical full-duplex connection ..... 495
Figure 18-8. A typical simplex connection (Master: receive, Slave: transmit) ..... 495
Figure 18-9. A typical simplex connection (Master: transmit only, Slave: receive) ..... 495

**Figure 18-10. A typical bidirectional connection** ..... 495
**Figure 18-11. Timing diagram of TI master mode with discontinuous transfer** ..... 498
**Figure 18-12. Timing diagram of TI master mode with continuous transfer** ..... 498
**Figure 18-13. Timing diagram of TI slave mode** ..... 499
**Figure 18-14. Timing diagram of NSS pulse with continuous transmit** ..... 500
**Figure 18-15. Timing diagram of write operation in Quad-SPI mode** ..... 501
**Figure 18-16. Timing diagram of read operation in Quad-SPI mode** ..... 502
**Figure 18-17. Block diagram of I2S** ..... 507
**Figure 18-18. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)** ..... 508
**Figure 18-19. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)** ..... 508
**Figure 18-20. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)** ..... 508
**Figure 18-21. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)** ..... 509
**Figure 18-22. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)** ..... 509
**Figure 18-23. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)** ..... 509
**Figure 18-24. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)** ..... 509
**Figure 18-25. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)** ..... 510
**Figure 18-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)** ... 510
**Figure 18-27. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)** ... 510
**Figure 18-28. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)** ... 510
**Figure 18-29. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)** ... 510
**Figure 18-30. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)** ... 511
**Figure 18-31. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)** ... 511
**Figure 18-32. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)** ... 511
**Figure 18-33. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)** ... 511
**Figure 18-34. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)** .... 511
**Figure 18-35. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)** .... 512
**Figure 18-36. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)** .... 512
**Figure 18-37. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)** .... 512
**Figure 18-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)** ..... 513
**Figure 18-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)** ..... 513
**Figure 18-40. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)** ..... 513
**Figure 18-41. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)** ..... 513
**Figure 18-42. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)** ..... 513
**Figure18-43. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)** ..... 514
**Figure 18-44. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)** ..... 514
**Figure 18-45. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)** ..... 514

**Figure 18-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)** ..... 514
**Figure18-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)** ..... 514
**Figure 18-48. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)** ..... 515
**Figure 18-49. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)** ..... 515
**Figure 18-50. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)** ..... 515
**Figure 18-51. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)** ..... 515
**Figure 18-52. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)** ..... 515
**Figure 18-53. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)** ..... 516
**Figure 18-54. Block diagram of I2S clock generator** ..... 516
**Figure 18-55. I2S initialization sequence** ..... 518
**Figure 18-56. I2S master reception disabling sequence** ..... 520

# List of Table

Table 1-1.Memory map of GD32E23x series ..... 24
Table1-2. Boot modes ..... 27
Table 2-1. Base address and size for flash memory ..... 38
Table 2-2. The relation between WSCNT and AHB clock frequency ..... 39
Table 2-3. Option byte ..... 47
Table 2-4. OB_WP bit for pages protected ..... 48
Table 3-1. Power saving mode summary ..... 61
Table 4-1. Clock source select ..... 72
Table 4-2. Core domain voltage selected in Deep-sleep mode - ..... 73
Table 5-1. NVIC exception types in Cortex®-M23 ..... 99
Table 5-2. Interrupt vector table ..... 99
Table 5-3. EXTI source ..... 102
Table 6-1. GPIO configuration table ..... 108
Table 8-1. DMA transfer operation ..... 134
Table 8-2. interrupt events ..... 137
Table 8-3. DMA requests for each channel ..... 139
Table 10-1. ADC internal input signals ..... 154
Table 10-2. ADC input pins definition ..... 154
Table 10-3. External trigger source for ADC ..... 161
Table 10-4. $t_{CONV}$ timings depending on resolution ..... 163
Table 10-5. Maximum output results for N and M combimations (grayed values indicates truncation) ..... 164
Table 11-1. CMP inputs and outputs summary ..... 177
Table 12-1. Min/max FWDGT timeout period at 40 kHz (IRC40K) ..... 184
Table 12-2. Min-max timeout value at 72 MHz ($f_{PCLK1}$) ..... 191
Table 13-1. RTC power saving mode management ..... 205
Table 13-2. RTC interrupts control ..... 205
Table 14-1. Timers (TIMERx) are devided into six sorts ..... 222
Table 14-2. Complementary outputs controlled by parameters ..... 239
Table 14-3. Counting direction in different quadrature decoder mode ..... 242
Table 14-4. Examples of slave mode ..... 245
Table 14-5. Complementary outputs controlled by parameters ..... 350
Table 14-6. Slave mode example table ..... 353
Table 14-7. Complementary outputs controlled by parameters ..... 387
Table 16-1. Description of USART important pins ..... 422
Table 16-2. Configuration of stop bits ..... 423
Table 16-3. USART interrupt requests ..... 437
Table 17-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors) ..... 459
Table 17-2. Event status flags ..... 473

**Table 17-3. Error flags** ..... 474
**Table 18-1. SPI signal description** ..... 488
**Table 18-2. Quad-SPI signal description** ..... 489
**Table 18-3. NSS function in slave mode** ..... 492
**Table 18-4. NSS function in master mode** ..... 493
**Table 18-5. SPI operation modes** ..... 493
**Table 18-6. SPI interrupt requests** ..... 506
**Table 18-7. I2S bitrate calculation formulas** ..... 516
**Table 18-8. Audio sampling frequency calculation formulas** ..... 516
**Table 18-9. Direction of I2S interface signals for each operation mode** ..... 517
**Table 18-10. I2S interrupt** ..... 522
**Table 20-1. List of abbreviations used in register** ..... 536
**Table 20-2. List of terms** ..... 536
**Table 21-1. Revision history** ..... 538

# 1. System and memory architecture

The GD32E23x series are 32-bit general-purpose microcontrollers based on the Arm® Cortex®-M23 processor. The Cortex®-M23 processor includes AHB buses. All memory accesses of the Cortex®-M23 processor are executed on the AHB buses according to the different purposes and the target memory spaces. The memory organization uses a ARMv8M architecture, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

## 1.1. Arm® Cortex®-M23 processor

The Cortex®-M23 processor is an energy-efficient processor with a very low gate count. It is intended to be used for microcontroller and deeply embedded applications that require an area-optimized processor. It offers significant benefits to developers, including:

- A simple architecture that is easy to learn and program.
- Ultra-low power, energy-efficient operation.
- Excellent code density.
- Deterministic, high-performance interrupt handling.
- Upward compatibility with Cortex®-M processor family.

The processor delivers high energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single-cycle multiplier and a 17-cycle divider.

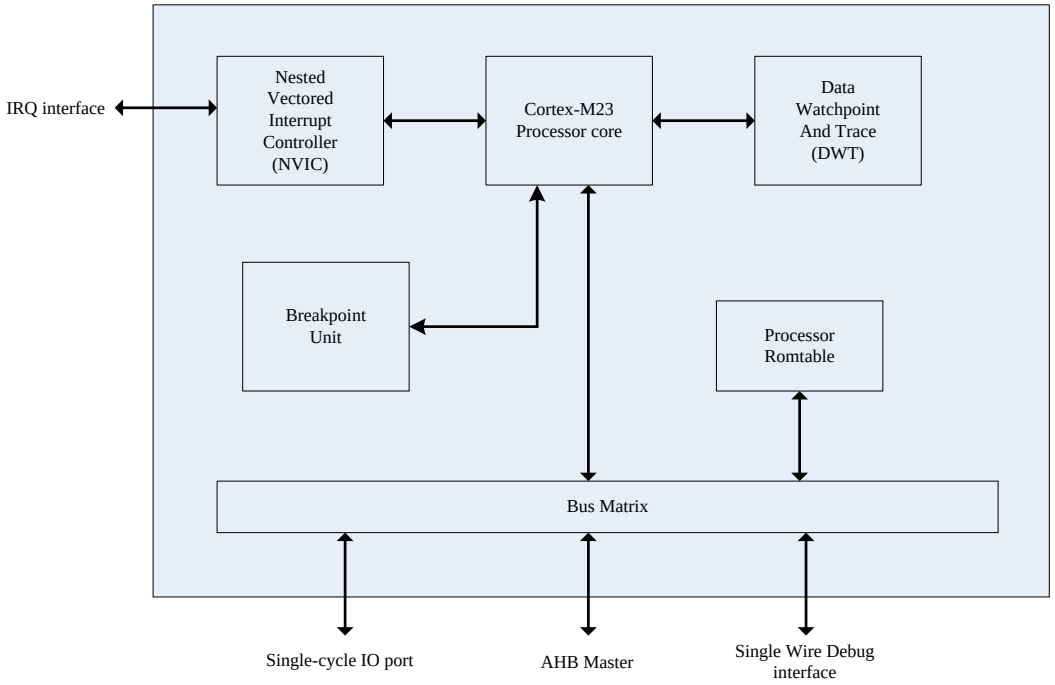
The Cortex®-M23 processor closely integrates a configurable Nested Vectored Interrupt Controller (NVIC), to deliver industry-leading interrupt performance.

Some system peripherals listed below are also provided by Cortex®-M23:

- Low latency, high-speed peripheral I/O port
- A Vector Table Offset Register
- Breakpoint unit
- Data Watchpoint
- Serial Wire Debug Port

The following figure shows the Cortex®-M23 processor block diagram. For more information, refer to the Arm® Cortex®-M23 Technical Reference Manual.

**Figure 1-1. The structure of the Cortex®-M23 processor**

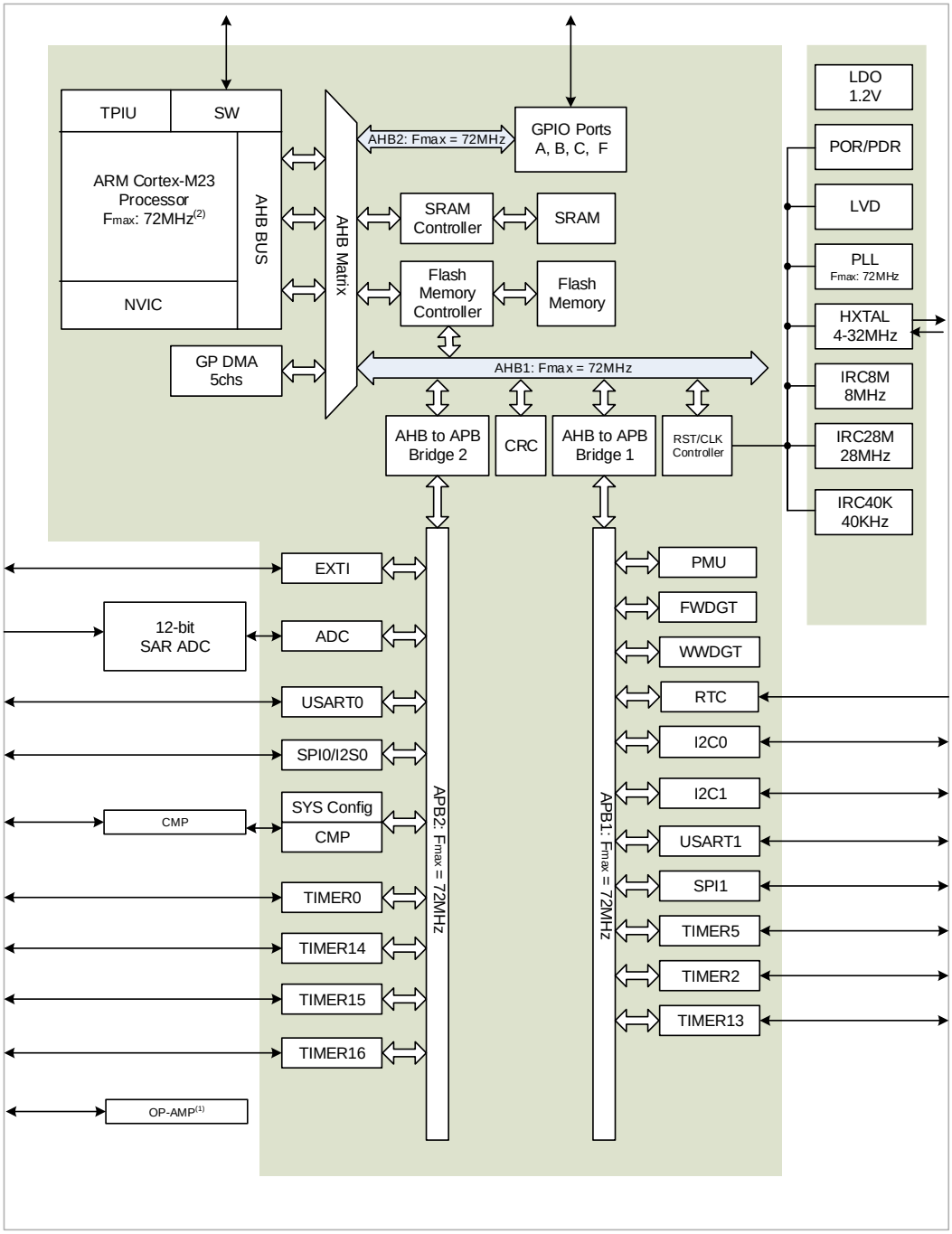



## 1.2. System architecture

The system architecture of GD32E23x series is shown in the following figure. The AHB matrix based on AMBA 5 AHB-LITE is a multi-layer AHB, which enables parallel access paths between multiple masters and slaves in the system. Two masters on the AHB matrix, including AHB bus of the Cortex®-M23 core and DMA. The AHB matrix consists of four slaves, including the flash memory controller, internal SRAM, AHB1 and AHB2.

The AHB2 connects with the GPIO ports. The AHB1 connects with the AHB peripherals including two AHB-to-APB bridges which provide full synchronous connections between the AHB1 and the two APB buses. The two APB buses connect with all the APB peripherals.

**Figure 1-2. Series system architecture of GD32E23x series**




**Note:** (1) only the GD32E231Kx series is available.

(2) The maximum frequency of the GD32E225xx series is 64MHz.

## 1.3. Memory map

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space which is the maximum address range of the Cortex®-M23 since it has a 32-bit bus address width. Additionally, a pre-defined memory map is provided by the Cortex®-M23 processor to reduce the software complexity of repeated implementation of

different device vendors. However, some regions are used by the Arm® Cortex®-M23 system peripherals. The following figure shows the memory map of GD32E23x series, including Code, SRAM, peripheral, and other pre-defined regions. Each peripheral of either type is allocated 1KB of space. This allows simplifying the address decoding for each peripheral.

**Table 1-1.Memory map of GD32E23x series**

| Pre-defined Regions | Bus | ADDRESS | Peripherals |
|---|---|---|---|
| | | 0xE000 0000 - 0xE00F FFFF | Cortex®-M23 internal peripherals |
| External Device | | 0xA000 0000 - 0xDFFF FFFF | Reserved |
| External RAM | | 0x60000000 - 0x9FFFFFFF | Reserved |
| Peripherals | AHB1 | 0x5004 0000 - 0x5FFF FFFF | Reserved |
| | | 0x5000 0000 - 0x5003 FFFF | Reserved |
| | AHB2 | 0x4800 1800 - 0x4FFF FFFF | Reserved |
| | | 0x4800 1400 - 0x4800 17FF | GPIOF |
| | | 0x4800 1000 - 0x4800 13FF | Reserved |
| | | 0x4800 0C00 - 0x4800 0FFF | Reserved |
| | | 0x4800 0800 - 0x4800 0BFF | GPIOC |
| | | 0x4800 0400 - 0x4800 07FF | GPIOB |
| | | 0x4800 0000 - 0x4800 03FF | GPIOA |
| | AHB1 | 0x4002 4400 - 0x47FF FFFF | Reserved |
| | | 0x4002 4000 - 0x4002 43FF | Reserved |
| | | 0x4002 3400 - 0x4002 3FFF | Reserved |
| | | 0x4002 3000 - 0x4002 33FF | CRC |
| | | 0x4002 2400 - 0x4002 2FFF | Reserved |
| | | 0x4002 2000 - 0x4002 23FF | FMC |
| | | 0x4002 1400 - 0x4002 1FFF | Reserved |
| | | 0x4002 1000 - 0x4002 13FF | RCU |
| | | 0x4002 0400 - 0x4002 0FFF | Reserved |
| | | 0x4002 0000 - 0x4002 03FF | DMA |
| | APB2 | 0x4001 8000 - 0x4001 FFFF | Reserved |
| | | 0x4001 5C00 - 0x4001 7FFF | Reserved |
| | | 0x4001 5800 - 0x4001 5BFF | DBG |
| | | 0x4001 4C00 - 0x4001 57FF | Reserved |
| | | 0x4001 4800 - 0x4001 4BFF | TIMER16 |
| | | 0x4001 4400 - 0x4001 47FF | TIMER15 |
| | | 0x4001 4000 - 0x4001 43FF | TIMER14 |
| | | 0x4001 3C00 - 0x4001 3FFF | Reserved |
| | | 0x4001 3800 - 0x4001 3BFF | USART0 |
| | | 0x4001 3400 - 0x4001 37FF | Reserved |
| | | 0x4001 3000 - 0x4001 33FF | SPI0/I2S0 |
| | | 0x4001 2C00 - 0x4001 2FFF | TIMER0 |
| | | 0x4001 2800 - 0x4001 2BFF | Reserved |

| Pre-defined Regions | Bus | ADDRESS | Peripherals |
|---|---|---|---|
| | | 0x4001 2400 - 0x4001 27FF | ADC |
| | | 0x4001 0800 - 0x4001 23FF | Reserved |
| | | 0x4001 0400 - 0x4001 07FF | EXTI |
| | | 0x4001 0000 - 0x4001 03FF | SYSCFG + CMP |
| | APB1 | 0x4000 CC00 - 0x4000 FFFF | Reserved |
| | | 0x4000 C800 - 0x4000 CBFF | Reserved |
| | | 0x4000 C400 - 0x4000 C7FF | Reserved |
| | | 0x4000 C000 - 0x4000 C3FF | Reserved |
| | | 0x4000 8000 - 0x4000 BFFF | Reserved |
| | | 0x4000 7C00 - 0x4000 7FFF | Reserved |
| | | 0x4000 7800 - 0x4000 7BFF | Reserved |
| | | 0x4000 7400 - 0x4000 77FF | Reserved |
| | | 0x4000 7000 - 0x4000 73FF | PMU |
| | | 0x4000 6400 - 0x4000 6FFF | Reserved |
| | | 0x4000 6000 - 0x4000 63FF | Reserved |
| | | 0x4000 5C00 - 0x4000 5FFF | Reserved |
| | | 0x4000 5800 - 0x4000 5BFF | I2C1 |
| | | 0x4000 5400 - 0x4000 57FF | I2C0 |
| | | 0x4000 4800 - 0x4000 53FF | Reserved |
| | | 0x4000 4400 - 0x4000 47FF | USART1 |
| | | 0x4000 4000 - 0x4000 43FF | Reserved |
| | | 0x4000 3C00 - 0x4000 3FFF | Reserved |
| | | 0x4000 3800 - 0x4000 3BFF | SPI1 |
| | | 0x4000 3400 - 0x4000 37FF | Reserved |
| | | 0x4000 3000 - 0x4000 33FF | FWDGT |
| | | 0x4000 2C00 - 0x4000 2FFF | WWDGT |
| | | 0x4000 2800 - 0x4000 2BFF | RTC |
| | | 0x4000 2400 - 0x4000 27FF | Reserved |
| | | 0x4000 2000 - 0x4000 23FF | TIMER13 |
| | | 0x4000 1400 - 0x4000 1FFF | Reserved |
| | | 0x4000 1000 - 0x4000 13FF | TIMER5 |
| | | 0x4000 0800 - 0x4000 0FFF | Reserved |
| | | 0x4000 0400 - 0x4000 07FF | TIMER2 |
| | | 0x4000 0000 - 0x4000 03FF | Reserved |
| SRAM | | 0x2000 4000 - 0x3FFF FFFF | Reserved |
| | | 0x2000 0000 - 0x2000 3FFF | SRAM |
| Code | | 0x1FFF F810 - 0x1FFF FFFF | Reserved |
| | | 0x1FFF F800 - 0x1FFF F80F | Option bytes |
| | | 0x1FFF EC00 - 0x1FFF F7FF | System memory |
| | | 0x0802 0000 - 0x1FFF EBFF | Reserved |

| Pre-defined Regions | Bus | ADDRESS | Peripherals |
|---|---|---|---|
| | | 0x0800 0000 - 0x0801 FFFF | Main Flash memory |
| | | 0x0002 0000 - 0x07FF FFFF | Reserved |
| | | 0x0000 0000 - 0x0001 FFFF | Aliased to Flash or system memory |

### 1.3.1. On-chip SRAM memory

The GD32E23x series contain up to 16KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses. In order to increase memory robustness, parity check is supported. The user can enable the parity check function using the bit SRAM_PARITY_CHECK in the user option byte (refer to Chapter 2.3.9 ***Option byte programming***). When enabled, an NMI is generated if the parity check fails. The SRAM parity check error flag is implemented in the system configuration register 2 (SYSCFG_CFG2). The error flag can be connected to the break input of TIMER 0/ TIMER 14/ TIMER 15/ TIMER 16, if the SRAM_PARITY_ERROR_LOCK control bit in the system configuration register 2 (SYSCFG_CFG2) is set to 1.

The real data width of the SRAM is 36 bits, including 32 bits for data and 4 bits for parity (1 bit per byte). When writing, the parity bits are computed and stored into the SRAM. When reading, the parity bits are also computed using the stored data in SRAM. The computed parity bits are compared with the stored parity bits which are computed during the writing access. If they are different, the parity check fails.

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

### 1.3.2. On-chip Flash memory

The devices provide up to 128 KB of main flash memory. The flash memory consists of up to 128 KB main flash and a 3 KB information block for the boot loader.

All of, byte, half-word (16 bits) and word (32 bits) read accesses are supported. The flash memory can be programmed word (32 bits) or double-word (64 bits) at a time. Each page of the flash memory can be erased individually. The whole flash memory space except information blocks can be erased at a time.

## 1.4. Boot configuration

The GD32E23x series provides three kinds of boot sources which can be selected using the bit BOOT1_n in the user option byte (refer to Chapter 2.3.9 ***Option byte programming***) and the BOOT0 pins. The value on the BOOT0 pin is latched on the 4th rising edge of SYSCLK

after a reset. It is up to the user to set the BOOT1_n and BOOT0 after a power-on reset or a system reset to select the required boot source. The details are shown in the following table.

**Table1-2. Boot modes**

| Selected boot source | Boot mode selection pins | |
|---|---|---|
| | Boot1 | Boot0 |
| Main Flash Memory | x | 0 |
| System Memory | 0 | 1 |
| On-chip SRAM | 1 | 1 |

1. The Boot1 value is the opposite of the BOOT1_n value.

After power-on sequence or a system reset, the Arm® Cortex®-M23 processor fetches the top-of-stack value from address 0x0000 0000 and the base address of boot code from 0x0000 0004 in sequence. Then, it starts executing code from the base address of boot code.

According to the selected boot source, either the main flash memory (original memory space beginning at 0x0800 0000) or the system memory (original memory space beginning at 0x1FFF EC00) is aliased in the boot memory space which begins at the address 0x0000 0000. When the on-chip SRAM whose memory space is beginning at 0x2000 0000 is selected as the boot source, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

The embedded boot loader is located in the System memory, which is used to reprogram the Flash memory. The boot loader can be activated through certain interfaces refer to the boot mode chapter of datasheet.

# 1.5. System configuration registers (SYSCFG)

SYSCFG base address: 0x4001 0000

## 1.5.1. System configuration register 0 (SYSCFG_CFG0)

Address offset: 0x00

Reset value: 0x0000 000X (X indicates BOOT_MODE[1:0] may be any value according to the BOOT0 pin and the nBOOT1 option bit after reset)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PB9_HCCE | Reserved | | |
| | | | | | | | | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | TIMER16_DMA_RMP | TIMER15_DMA_RMP | USART0_RX_DMA_RMP | USART0_TX_DMA_RMP | ADC_DMA_RMP | Reserved | | | PA11_PA12_RMP | Reserved | | BOOT_MODE | |
| | | | rw | rw | rw | rw | rw | | | | rw | | | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value |
| 19 | PB9_HCCE | PB9 pin high current capability enable<br>When it is set, the PB9 pin can be used to control an infrared LED directly.<br>0: High current capability on the PB9 pin is disenabled.<br>1: High current capability on the PB9 pin is enabled, and the speed control of the pin is bypassed. |
| 18:13 | Reserved | Must be kept at reset value |
| 12 | TIMER16_DMA_RMP | Timer 16 DMA request remapping enable<br>0: not remap (TIMER16_CH1 and TIMER16_UP DMA requests are mapped on DMA channel 0)<br>1: remap (TIMER16_CH1 and TIMER16_UP DMA requests are mapped on DMA channel 1) |
| 11 | TIMER15_DMA_RMP | Timer 15 DMA request remapping enable<br>0: not remap (TIMER15_CH1 and TIMER15_UP DMA requests are mapped on DMA channel 2)<br>1: remap (TIMER15_CH1 and TIMER15_UP DMA requests are mapped on DMA channel 3) |
| 10 | USART0_RX_DMA_RMP | USART0_RX DMA request remapping enable<br>0: not remap (USART0_RX DMA requests are mapped on DMA channel 2) |

| | | |
|---|---|---|
| | | 1: remap (USART0_RX DMA requests are mapped on DMA channel 4) |
| 9 | USART0_TX_DMA_RMP | USART0_TX DMA request remapping enable<br>0: not remap (USART0_TX DMA requests are mapped on DMA channel 1)<br>1: remap (USART0_TX DMA requests are mapped on DMA channel 3) |
| 8 | ADC_DMA_RMP | ADC DMA request remapping enable<br>0: not remap (ADC DMA requests are mapped on DMA channel 0)<br>1: remap (ADC DMA requests are mapped on DMA channel 1) |
| 7:5 | Reserved | Must be kept at reset value |
| 4 | PA11_PA12_RMP | PA11 and PA12 remapping bit for small packages (28, 24and 20 pins).<br>This bit is set and cleared by software. It controls the mapping of either PA9/10 or PA11/12 pin pair on small pin-count packages.<br>0: No remap (pin pair PA9/10 mapped on the pins)<br>1: Remap (pin pair PA11/12 mapped instead of PA9/10) |
| 3:2 | Reserved | Must be kept at reset value |
| 1:0 | BOOT_MODE[1:0] | Boot mode (Refer to Chapter 1.4 Boot configuration for details)<br>Bit0 is mapping to the BOOT0 pin; the value of bit1 is the opposite of the nBOOT1 option bit value.<br>x0: Boot from the Main Flash<br>01: Boot from the System Flash memory<br>11: Boot from the embedded SRAM |

### 1.5.2. EXTI sources selection register 0 (SYSCFG_EXTISS0)

Address offset: 0x08

Reset value: 0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI3_SS [3:0] | | | | EXTI2_SS [3:0] | | | | EXTI1_SS [3:0] | | | | EXTI0_SS [3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | must be kept at reset value |
| 15:12 | EXTI3_SS | EXTI 3 sources selection<br>X000: PA3 pin<br>X001: PB3 pin<br>X010: reserved<br>X011: reserved |

| | | |
|---|---|---|
| | | X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 11:8 | EXTI2_SS | EXTI 2 sources selection<br>X000: PA2 pin<br>X001: PB2 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 7:4 | EXTI1_SS | EXTI 1 sources selection<br>X000: PA1 pin<br>X001: PB1 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: PF1 pin<br>X110: reserved<br>X111: reserved |
| 3:0 | EXTI0_SS | EXTI 0 sources selection<br>X000: PA0 pin<br>X001: PB0 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: PF0 pin<br>X110: reserved<br>X111: reserved |

### 1.5.3. EXTI sources selection register 1 (SYSCFG_EXTISS1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI7_SS [3:0] | | | | EXTI6_SS [3:0] | | | | EXTI5_SS [3:0] | | | | EXTI4_SS [3:0] | | | |

rw rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | EXTI7_SS | EXTI 7 sources selection<br>X000: PA7 pin<br>X001: PB7 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: PF7 pin<br>X110: reserved<br>X111: reserved |
| 11:8 | EXTI6_SS | EXTI 6 sources selection<br>X000: PA6 pin<br>X001: PB6 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: PF6 pin<br>X110: reserved<br>X111: reserved |
| 7:4 | EXTI5_SS | EXTI 5 sources selection<br>X000: PA5 pin<br>X001: PB5 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 3:0 | EXTI4_SS | EXTI 4 sources selection<br>X000: PA4 pin<br>X001: PB4 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |

### 1.5.4. EXTI sources selection register 2 (SYSCFG_EXTISS2)

Address offset: 0x10

Reset value: 0x0000 0000

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

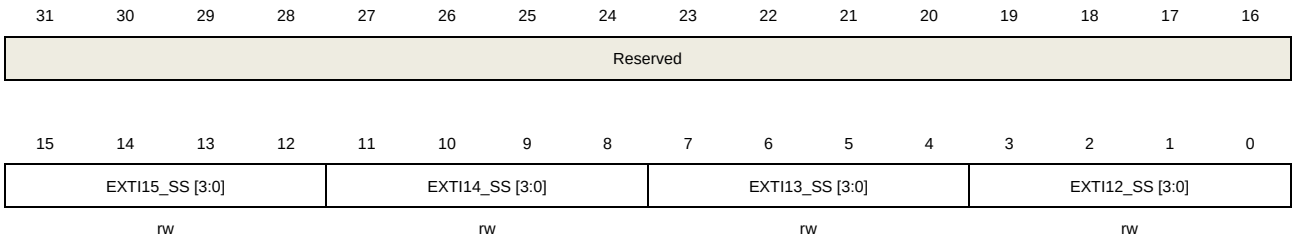
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI11_SS [3:0] | | | | EXTI10_SS [3:0] | | | | EXTI9_SS [3:0] | | | | EXTI8_SS [3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | EXTI11_SS | EXTI 11 sources selection<br>X000: PA11 pin<br>X001: PB11 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 11:8 | EXTI10_SS | EXTI 10 sources selection<br>X000: PA10 pin<br>X001: PB10 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 7:4 | EXTI9_SS | EXTI 9 sources selection<br>X000: PA9 pin<br>X001: PB9 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 3:0 | EXTI8_SS | EXTI 8 sources selection<br>X000: PA8 pin |

X001: PB8 pin
X010: reserved
X011: reserved
X100: reserved
X101: reserved
X110: reserved
X111: reserved

### 1.5.5. EXTI sources selection register 3 (SYSCFG_EXTISS3)

Address offset: 0x14
Reset value: 0x0000 0000




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | EXTI15_SS | EXTI 15 sources selection<br>X000: PA15 pin<br>X001: PB15 pin<br>X010: PC15 pin<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 11:8 | EXTI14_SS | EXTI 14 sources selection<br>X000: PA14 pin<br>X001: PB14 pin<br>X010: PC14 pin<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 7:4 | EXTI13_SS | EXTI 13 sources selection<br>X000: PA13 pin |

| | | |
|---|---|---|
| | | X001: PB13 pin<br>X010: PC13 pin<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |
| 3:0 | EXTI12_SS | EXTI 12 sources selection<br>X000: PA12 pin<br>X001: PB12 pin<br>X010: reserved<br>X011: reserved<br>X100: reserved<br>X101: reserved<br>X110: reserved<br>X111: reserved |

### 1.5.6. System configuration register 2 (SYSCFG_CFG2)

Address offset: 0x18

Reset value: 0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | SRAM_PCEF | Reserved | | | | | LVD_LOCK | SRAM_PARITY_ERROR_LOCK | LOCKUP_LOCK |
| | | | | | | | rc_w1 | | | | | | rw | rw | rw |

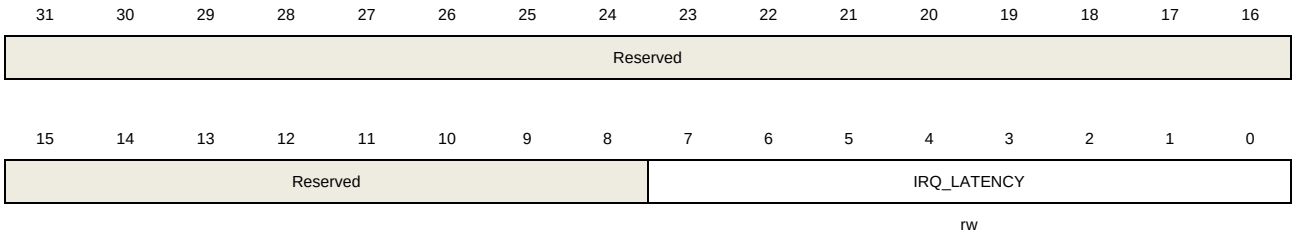
| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value |
| 8 | SRAM_PCEF | SRAM parity check error flag<br>This bit is set by hardware when an SRAM parity check error occurs. It is cleared by software by writing 1.<br>0: No SRAM parity check error detected<br>1: SRAM parity check error detected |
| 7:3 | Reserved | Must be kept at reset value |
| 2 | LVD_LOCK | LVD lock<br>This bit is set by software and cleared by a system reset. |

| | | |
|---|---|---|
| | | 0: The LVD interrupt is disconnected from the break input of TIMER0/14/15/16. LVDE and LVDT[2:0] in the PWR_CTL register can be programmed.<br>1: The LVD interrupt is connected from the break input of TIMER0/14/15/16. LVDE and LVDT[2:0] in the PWR_CTL register are read only. |
| 1 | SRAM_PARITY_ERROR_LOCK | SRAM parity check error lock<br>This bit is set by software and cleared by a system reset.<br>0: The SRAM parity check error is disconnected from the break input of TIMER0/14/15/16.<br>1: The SRAM parity check error is connected from the break input of TIMER0/14/15/16. |
| 0 | LOCKUP_LOCK | Cortex®-M23 LOCKUP output lock<br>This bit is set by software and cleared by a system reset.<br>0: The Cortex®-M23 LOCKUP output is disconnected from the break input of TIMER0/14/15/16.<br>1: The Cortex®-M23 LOCKUP output is connected from the break input of TIMER0/14/15/16. |

## 1.5.7. IRQ Latency register (SYSCFG_CPU_IRQ_LAT)

Address offset: 0x100

Reset value: 0x0000 0000




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | IRQ_LATENCY | IRQ_LATENCY specifies the minimum number of cycles between an interrupt that becomes pended in the NVIC, and the vector fetch for that interrupt being issued on the AHB-Lite interface.<br>If IRQ_LATENCY is set to 0, interrupts are taken as quickly as possible.<br>For non-zero values, the Cortex®-M23 processor ensures that a minimum of IRQ_LATENCY+1 hclk cycles exist between an interrupt becoming pended in the NVIC and the vector fetch for the interrupt being performed. |

# 1.6. Device electronic signature

The device electronic signature contains memory density information and the 96-bit unique device ID. It is stored in the information block of the Flash memory. The 96-bit unique device ID is unique for any device. It can be used as serial numbers, or part of security keys, etc.

## 1.6.1. Memory density information

Base address: 0x1FFF F7E0
The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRAM_DENSITY[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLASH_DENSITY[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | SRAM_DENSITY [15:0] | SRAM density<br>The value indicates the on-chip SRAM density of the device in Kbytes.<br>Example: 0x0008 indicates 8 Kbytes. |
| 15:0 | FLASH_DENSITY [15:0] | Flash memory density<br>The value indicates the Flash memory density of the device in Kbytes.<br>Example: 0x0020 indicates 32 Kbytes. |

## 1.6.2. Unique device ID (96 bits)

Base address: 0x1FFF F7AC
The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:0 | UNIQUE_ID[31:0] | Unique device ID |

Base address: 0x1FFF F7B0

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[63:48] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[47:32] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | UNIQUE_ID[63:32] | Unique device ID |

Base address: 0x1FFF F7B4

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[95:80] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[79:64] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | UNIQUE_ID[95:64] | Unique device ID |

# 2. Flash memory controller (FMC)

## 2.1. Overview

The Flash Memory Controller, FMC, provides all the necessary functions for the on-chip flash memory. A little waiting time is needed while CPU executes instructions stored in the 128K bytes of the flash. It also provides page erase, mass erase, and word/double word program for flash memory.

## 2.2. Characteristics

**For GD32E23x series：**

- Up to 128 KB of on-chip flash memory for storing instruction/data
- 0~2 waiting time within 128K bytes when CPU executes instruction
- Flash memory supports 64-btis wide data reading
- Pre-fetch buffer to speed read operations
- 3K bytes information block for boot loader
- 16 bytes option bytes block for user application requirements
- 1K bytes page size
- Word or double word programming, page erase and mass erase capability
- Flash read protection to prevent illegal code/data access
- Page erase/program protection to prevent unexpected operation

## 2.3. Function overview

### 2.3.1. Flash memory architecture

The flash memory consists of up to 128 KB main flash organized into 128 pages with 1 KB capacity per page and a 3 KB Information Block for the Boot Loader. The main flash memory contains a total of up to 128 pages which can be erased individually. The ***Table 2-1. Base address and size for flash memory*** shows the base address and size.

**Table 2-1. Base address and size for flash memory**

| Block | Name | Address | size(bytes) |
|---|---|---|---|
| Main Flash Block | Page 0 | 0x0800 0000 - 0x0800 03FF | 1KB |
| | Page 1 | 0x0800 0400 - 0x0800 07FF | 1KB |
| | Page 2 | 0x0800 0800 - 0x0800 0BFF | 1KB |
| | . | . | . |
| | . | . | . |
| | . | . | . |

| Block | Name | Address | size(bytes) |
|---|---|---|---|
| | Page 127 | 0x0801 FC00 - 0x0801 FFFF | 1KB |
| Information Block | Boot Loader | 0x1FFF EC00 - 0x1FFF F7FF | 3KB |
| Option byte Block | Option byte | 0x1FFF F800 - 0x1FFF F80F | 16B |
| One-time program Block | OTP bytes | 0x1FFF_7000~0x1FFF_73FF | 1KB |

**Note:** The Information Block stores the bootloader - this block cannot be programmed or erased by user.

### 2.3.2. Read operations

The flash can be addressed directly as a common memory space. Any instruction fetch and the data access from the flash are through the AHB BUS from the CPU.

#### Wait state added:

Configure the WSCNT bits in the FMC_WS register correctly depend on the AHB clock frequency. The relation between WSCNT and AHB clock frequency is shown in ***Table 2-2. The relation between WSCNT and AHB clock frequency***.

**Table 2-2. The relation between WSCNT and AHB clock frequency**

| AHB clock frequency | WSCNT configured |
|---|---|
| <= 24MHz | 0 (0 wait state added) |
| <= 48MHz | 1 (1 wait state added) |
| <= 72MHz | 2 (2 wait state added) |

If system reset occurs, the AHB clock frequency is 8MHz and the WSCNT is 0.

**Note:**

1. If it is needed to increase the AHB clock frequency. First, refer to ***Table 2-2. The relation between WSCNT and AHB clock frequency***, configure the WSCNT bits according to the target AHB clock frequency. Then, increase the AHB clock frequency to the target frequency. It is forbidden to increase the AHB clock frequency before configuring the WSCNT.

2. If it is needed to decrease the AHB clock frequency. First, decrease the target AHB clock frequency. Then refer to ***Table 2-2. The relation between WSCNT and AHB clock frequency***, configure the WSCNT bits according the target AHB clock frequency. It is forbidden to configure the WSCNT bits before decrease the AHB clock frequency.

Because the wait state is added, the read efficiency is very low (such as add 2 wait state when 72MHz). In order to speed up the read access, there are some functions performed.

#### Current buffer:

The current buffer is always enabled. Each time read from flash memory, 64-bit data will be get and store in current buffer. The CPU only need 32-bit or 16-bit in each read operation. So

in the case of sequential code, the next data can get from current buffer without repeat fetch from flash memory.

**Pre-fetch buffer:**

The pre-fetch buffer is enabled by setting the PFEN bit in the FMC_WS register. In the case of sequential code, when CPU executes the current buffer data (64-bit), 32-bit needs at least 2 clocks and 16-bit needs at least 4 clocks. In this case, pre-fetch the data of next double-word address from flash memory and store to Pre-fetch buffer. So when the CPU finishes the current buffer and needs execute the next data, the pre-fetch buffer hits.

### 2.3.3. Unlock the FMC_CTL register

After reset, the FMC_CTL register is not accessible in write mode, except for the OBRLD bit, which is used for reloading the option byte, and the LK bit in FMC_CTL register is 1. An unlocking sequence consists of two write operations to the FMC_KEY register can open the access to the FMC_CTL register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register. After the two write operations, the LK bit in FMC_CTL register is set to 0 by hardware. The software can lock the FMC_CTL again by setting the LK bit in FMC_CTL register to 1. If there is any wrong operations on the FMC_KEY register, the LK bit in FMC_CTL register will be set, and the FMC_CTL register will be locked, then it will generate a bus error.

The OBPG bit and OBER bit in FMC_CTL are also protected by FMC_OBKEY register. The unlocking sequence includes two write operations, which are writing 0x45670123 and 0xCDEF89AB to FMC_OBKEY register. And then set the OBWEN bit in FMC_CTL register to 1. The software can set OBWEN bit to 0 to protect the OBPG bit and OBER bit in FMC_CTL register again.

### 2.3.4. Page erase

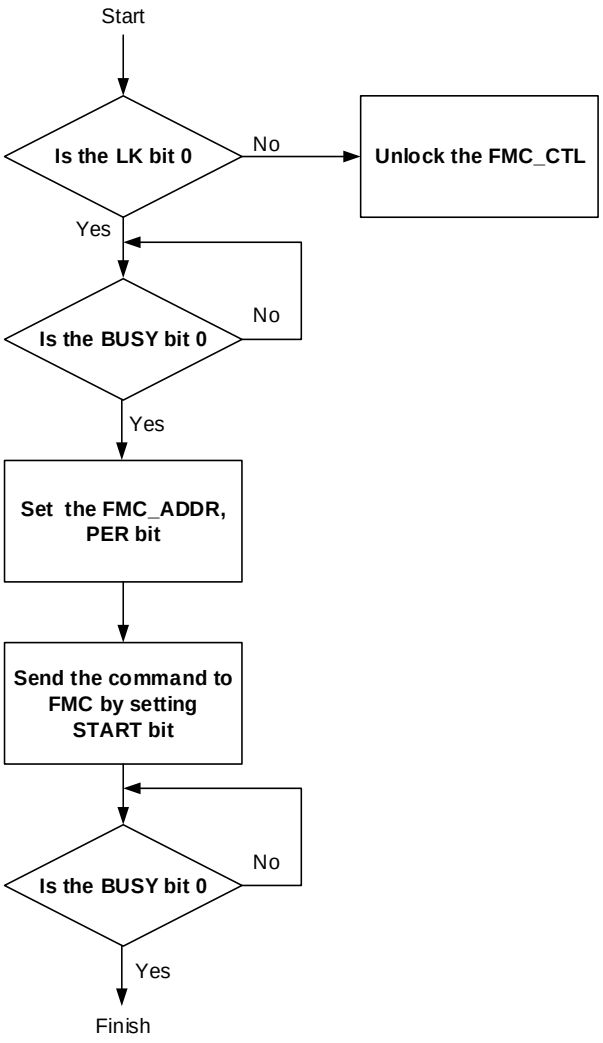
The FMC provides a page erase function which is used for initializing the contents of a main flash memory page to a high state. Each page can be erased independently without affecting the contents of other pages. The following steps show the access sequence of the register for a page erase operation.

- Unlock the FMC_CTL register if necessary.
- Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the page address into the FMC_ADDR register.
- Write the page erase command into PER bit in FMC_CTL register.
- Send the page erase command to the FMC by setting the START bit in FMC_CTL register.

- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC_STAT register.
- Read and verify the page if required using a AHB BUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set, and the ENDF in FMC_STAT register is set. Note that a correct target page address must be confirmed. Or the software may run out of control if the target erase page is being used for fetching codes or accessing data. The FMC will not provide any notification when this occurs. Additionally, the page erase operation will be ignored on protected pages. A Flash Operation Error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the PGERR bit in the FMC_STAT register to detect this condition in the interrupt handler. The end of this operation is indicated by the ENDF bit in the FMC_STAT register. The ***Figure 2-1. Process of page erase operation*** shows the page erase operation flow.

**Figure 2-1. Process of page erase operation**



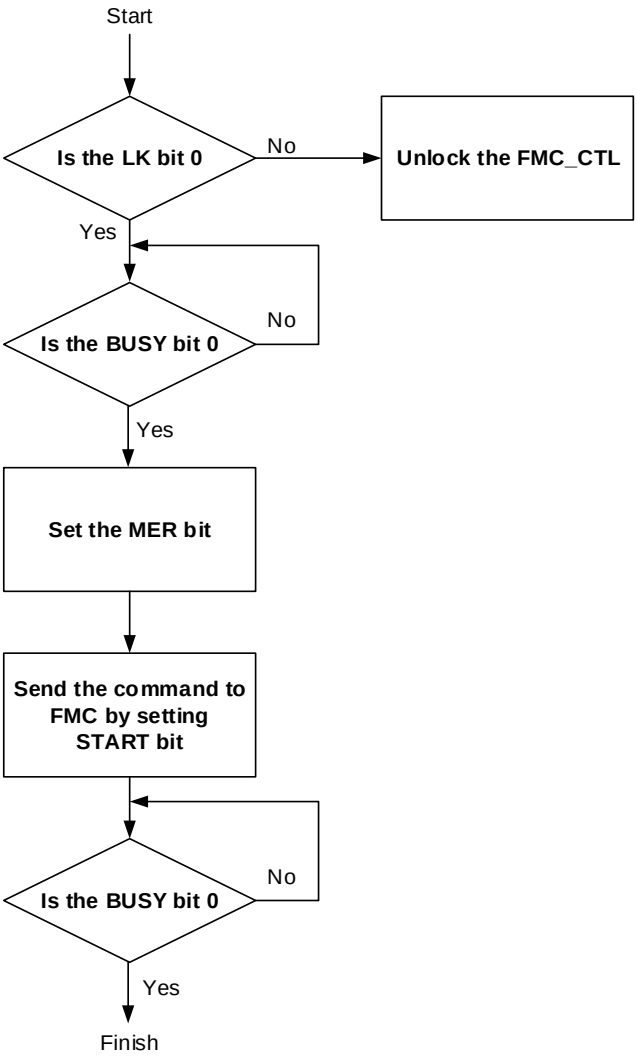
### 2.3.5. Mass erase

The FMC provides a complete erase function which is used for initializing the Main Flash Block contents. The following steps show the mass erase register access sequence.

- Unlock the FMC_CTL register if necessary.
- Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the mass erase command into MER bit in FMC_CTL register.
- Send the mass erase command to the FMC by setting the START bit in FMC_CTL register.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC_STAT register.
- Read and verify the flash memory if required using a AHB BUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set, and the ENDF in FMC_STAT register is set. Since all flash data will be reset to a value of 0xFFFF FFFF, the mass erase operation can be implemented using a program that runs in SRAM or by using the debugging tool to access the FMC registers directly. The end of this operation is indicated by the ENDF bit in the FMC_STAT register. (The starting address of programming operation should be 0x0800 0000) The ***Figure 2-2. Process of the mass erase operation*** indicates the mass erase operation flow.

**Figure 2-2. Process of the mass erase operation**




### 2.3.6. Main flash programming

The FMC provides a 32-bit word/16-bit half word programming function by AHB BUS which is used to modify the main flash memory contents. While actually, the data program to flash memory is 32-bits or 64-bits which is defined by the PGW bit in the FMC_WS register.

The following steps show the register access sequence of the programming operation.

- Unlock the FMC_CTL register if necessary.
- Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Set the PGW bit if needed.
- Write the word program command into the PG bit in FMC_CTL register.
- Write the data to be programed by AHB BUS with desired absolute address (0x08XX

XXXX).

If AHB BUS program is 32-bit word and the PGW bit is set to 0(32-bit program to flash memory), the AHB BUS write once and the data program to flash memory. The data to be programed must word alignment.

If AHB BUS program is 32-bit and the PGW bit is set to 1(64-bit program to flash memory), the AHB BUS write twice to form a 64-bit data and then the 64-bit data program to flash memory. The data to be programed must double-word alignment.

If AHB BUS program is 16-bit and the PGW bit is set to 0(32-bit program to flash memory), the AHB BUS write twice to form a 32-bit data and then the 32-bit data program to flash memory. The data to be programed must word alignment.

If AHB BUS program is 16-bit and the PGW bit is set to 1(64-bit program to flash memory), the AHB BUS write four times to form a 64-bit data and then the 64-bit data program to flash memory. The data to be programed must double-word alignment.

For less program time, suggest the AHB BUS program use 32-bit, set the PGW to 1 if the data to be programed is double-word alignment, or set PGW to 0 if the data to be programed is word alignment

- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC_STAT register.
- Read and verify the flash memory if required using a AHB BUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set, and the ENDF in FMC_STAT register is set. Note that before the word/half word programming operation you should check the address that it has been erased. If the address has not been erased, PGERR bit will set when programming the address even if programming 0x0. Each word can be programmed only one time after erase and before next erase. Additionally, the program operation will be ignored on protected pages. A flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the PGERR bit in the FMC_STAT register to detect this condition in the interrupt handler. The end of this operation is indicated by the ENDF bit in the FMC_STAT register.

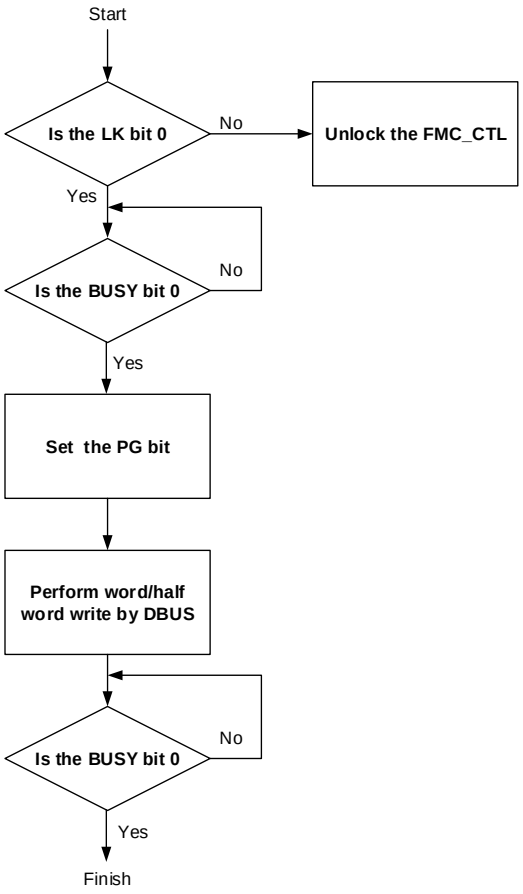
In the following cases, the PGAERR bit in the FMC_STAT register will be set.

- The AHB BUS program use byte write (not 32-bit or 16-bit write)
- The AHB BUS program size is not equal previous size. It not allow mix 32-bit with 16-bit write.
- The AHB BUS write is not alignment. If AHB BUS program is 32-bit and the PGW bit is set to 1(64-bit program to flash memory), the second AHB BUS write must double-word alignment and belong to same double-word address. If AHB BUS program is 16-bit and the PGW bit is set to 0(32-bit program to flash memory), the second AHB BUS write must word alignment and belong to same word address. If AHB BUS program is 16-bit and the PGW bit is set to 1(64-bit program to flash memory), the 2$^{nd}$/3$^{rd}$/4$^{th}$ AHB BUS write must double-word alignment and belong to same double-word address.

**Note:** If the program is not write total 64bits/32bits (by setting the PGW bit in the FMC_WS register), the data is not program to the flash memory without any notice.

In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the PGERR bit, PGAERR bit or WPERR bit in the FMC_STAT register to detect which condition occurred in the interrupt handler. The ***Figure 2-3. Process of the word programming operation*** displays the word programming operation flow.

**Figure 2-3. Process of the word programming operation**




## 2.3.7. OTP programming

The OTP programming method is same as the main flash programming. The OTP block can only be programed once and cannot be erased.

**Note:** It must ensure the OTP programming sequence completely without any unexpected interrupt, such as system reset or power down. If unexpected interrupt occurs, there is very little probability of corrupt the data stored in flash memory.

### 2.3.8. Option byte erase

The FMC provides an erase function which is used for initializing the option byte block in flash. The following steps show the erase sequence.

- Unlock the FMC_CTL register if necessary.
- Unlock the OBWEN bit in FMC_CTL register if necessary.
- Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the option byte erase command into OBER bit in FMC_CTL register.
- Send the option byte erase command to the FMC by setting the START bit in FMC_CTL register.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC_STAT register.
- Read and verify the flash memory if required using a AHB BUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set, and the ENDF in FMC_STAT register is set. The end of this operation is indicated by the ENDF bit in the FMC_STAT register.

### 2.3.9. Option byte programming

The FMC provides a 32-bit word/32-bit double word programming function which is used for modifying the option byte block contents. The following steps show the programming operation sequence.

- Unlock the FMC_CTL register if necessary.
- Unlock the OBWEN bit in FMC_CTL register if necessary.
- Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the program command into the OBPG bit in FMC_CTL register.
- A 32-bit word write at desired address by AHB BUS.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC_STAT register.
- Read and verify the flash memory if required using a AHB BUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set, and the ENDF in FMC_STAT register is set. Note that before the word/half word programming operation you should check the address that it has been erased. If the address has not been erased, PGERR bit will set when programming

the address include programming 0x0. The end of this operation is indicated by the ENDF bit in the FMC_STAT register.

### 2.3.10. Option byte description

The option bytes block of flash memory reloaded to FMC_OBSTAT and FMC_WP registers after each system reset or OBRLD bit set in FMC_CTL register, and the option bytes work. The option complement bytes are the opposite of option bytes. When option bytes reload, if the option complement bytes and option bytes does not match, the OBERR bit in FMC_OBSTAT register is set, and the option byte is set to 0xFF. The ***Table 2-3. Option byte*** is the detail of option bytes.

**Table 2-3. Option byte**

| Address | Name | Description |
|---|---|---|
| 0x1fff f800 | OB_SPC | option byte Security Protection Code (factory value=0x5A)<br>0xA5: No protection<br>any value except 0xA5 or 0xCC: Protection level low<br>0xCC: Protection level high |
| 0x1fff f801 | OB_SPC_N | OB_SPC complement value (factory value=0xFF) |
| 0x1fff f802 | OB_USER | option byte which user defined (factory value=0xFF)<br>[7]: Reserved<br>[6]: SRAM_PARITY_CHECK<br>0: Enable SRAM parity check<br>1: Disable SRAM parity check<br>[5]: VDDA_VISOR<br>0: Disable $V_{DDA}$ monitor<br>1: Enable $V_{DDA}$ monitor<br>[4]: BOOT1_n<br>0: BOOT1 bit is 1<br>1: BOOT1 bit is 0<br>[3]: Reserved<br>[2]: nRST_STDBY<br>0: Generate a reset instead of entering standby mode<br>1: No reset when entering standby mode<br>[1]: nRST_DPSLP<br>0: Generate a reset instead of entering Deep-sleep mode<br>1: No reset when entering Deep-sleep mode<br>[0]: nWDG_SW<br>0: Hardware free watchdog timer<br>1: Software free watchdog timer |
| 0x1fff f803 | OB_USER_N | OB_USER complement value (factory value=0xFF) |
| 0x1fff f804 | OB_DATA[7:0] | user defined data bit 7 to 0 (factory value=0xFF) |
| 0x1fff f805 | OB_DATA_N[7:0] | OB_DATA complement value bit 7 to 0 (factory value=0xFF) |
| 0x1fff f806 | OB_DATA[15:8] | user defined data bit 15 to 8 (factory value=0xFF) |

| Address | Name | Description |
|---|---|---|
| 0x1fff f807 | OB_DATA_N[15:8] | OB_DATA complement value bit 15 to 8 (factory value=0xFF) |
| 0x1fff f808 | OB_WP[7:0] | Page Erase/Program Protection bit 7 to 0 (factory value=0xFF) |
| 0x1fff f809 | OB_WP_N[7:0] | OB_WP complement value bit 7 to 0 (factory value=0xFF) |
| 0x1fff f80a | OB_WP[15:8] | Page Erase/Program Protection bit 15 to 8 (factory value=0xFF) |
| 0x1fff f80b | OB_WP_N[15:8] | OB_WP complement value bit 15 to 8 (factory value=0xFF) |
| 0x1fff f80c | OB_WP[23:16] | Page Erase/Program Protection bit 23 to 16 (factory value=0xFF) |
| 0x1fff f80d | OB_WP_N[23:16] | OB_WP complement value bit 23 to 16 (factory value=0xFF) |
| 0x1fff f80e | OB_WP[31:24] | Page Erase/Program Protection bit 31 to 24 (factory value=0xFF) |
| 0x1fff f80f | OB_WP_N[31:24] | OB_WP complement value bit 31 to 24 (factory value=0xFF) |

### 2.3.11. Page erase/Program protection

The FMC provides page erase/program protection functions to prevent inadvertent operations on the flash memory. The page erase or program will not be accepted by the FMC on protected pages. If the page erase or program command is sent to the FMC on a protected page, the WPERR bit in the FMC_STAT register will then be set by the FMC. If the WPERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the flash operation error interrupt will be triggered by the FMC to get the attention of the CPU. The page protection function can be individually enabled by configuring the OB_WP[31:0] bit field to 0 in the option byte. If a page erase operation is executed on the Option Byte region, all the flash memory page protection functions will be disabled. When setting or resetting OB_WP in the option byte, the software need to set OBRLD in FMC_CTL register or a system reset to reload the OB_WP bits. The ***Table 2-4. OB_WP bit for pages protected*** shows which pages are protected by set OB_WP[31:0].

**Table 2-4. OB_WP bit for pages protected**

| OB_WP bit | pages protected |
|---|---|
| OB_WP[0] | page 0 ~ page 3 |
| OB_WP[1] | page 4 ~ page 7 |
| OB_WP[2] | page 8 ~ page 11 |
| .<br>.<br>. | .<br>.<br>. |
| OB_WP[30] | page 120 ~ page 123 |
| OB_WP[31] | page 124 ~ page 127 |

### 2.3.12. Security protection

The FMC provides a security protection function to prevent illegal code/data access on the flash memory. This function is useful for protecting the software/firmware from illegal users.

There are 3 levels for protecting:

No protection: when setting OB_SPC byte and its complement value to 0xA55A, no protection performed. The main flash and option bytes block are accessible by all operations.

Protection level low: when setting OB_SPC byte and its complement value to any value except 0xA55A or 0xCC33, protection level low performed. The main flash can only be accessed by user code. In debug mode, boot from SRAM or boot from boot loader mode, all operations to main flash is forbidden. If a read operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, a bus error will be generated. If a program/erase operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, the PGERR bit in FMC_STAT register will be set. At protection level low, option bytes block are accessible by all operations. If program back to no protection level by setting OB_SPC byte and its complement value to 0xA55A, a mass erase for main flash will be performed.

Protection level high: when set OB_SPC byte and its complement value to 0xCC33, protection level high performed. When this level is programmed in debug mode, boot from SRAM or boot from boot loader mode is disabled. The main flash block is accessible by all operations from user code. The option byte cannot be erased, and the option byte cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

**Note:** In the case of read protection, the first 4k of flash cannot be programmed or erased.

## 2.4. Register definition

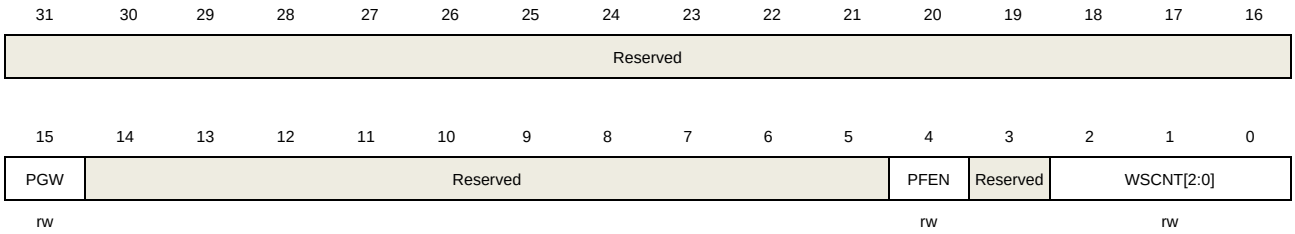
FMC base address: 0x4002 2000

### 2.4.1. Wait state register (FMC_WS)

Address offset: 0x00

Reset value: 0x0000 0030

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | PGW | Program width to flash memory<br>0: 32-bit program width to flash memory<br>1: 64-bit program width to flash memory |
| 14:5 | Reserved | Must be kept at reset value |
| 4 | PFEN | Pre-fetch enable<br>0: Pre-fetch disable<br>1: Pre-fetch enable |
| 3 | Reserved | Must be kept at reset value |
| 2:0 | WSCNT[2:0] | Wait state counter register<br>These bits set and reset by software.<br>000: 0 wait state added<br>001: 1 wait state added<br>010: 2 wait state added<br>011 ~ 111: Reserved |

### 2.4.2. Unlock key register (FMC_KEY)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | KEY[31:0] | FMC_CTL unlock registers<br>These bits are only be written by software<br>Write KEY[31:0] with key to unlock FMC_CTL register. |

### 2.4.3. Option byte unlock key register (FMC_OBKEY)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OBKEY[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OBKEY[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | OBKEY[31:0] | FMC_CTL option byte operation unlock registers<br>These bits are only be written by software<br>Write OBKEY[31:0] with key to unlock option byte command in FMC_CTL register. |

### 2.4.4. Status register (FMC_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | ENDF | WPERR | PGAERR | PGERR | Reserved | BUSY |
| | | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value |
| 5 | ENDF | End of operation flag bit<br>When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1. |
| 4 | WPERR | Erase/Program protection error flag bit<br>When erasing/programming on protected pages, this bit is set by hardware. The software can clear it by writing 1. |
| 3 | PGAERR | Program alignment error flag bit<br>This bit is set by hardware when AHB BUS write data is not alignment. The software can clear it by writing 1. |
| 2 | PGERR | Program error flag bit<br>When programming to the flash while it is not 0xFFFF, this bit is set by hardware. The software can clear it by writing 1. |
| 1 | Reserved | Must be kept at reset value |
| 0 | BUSY | The flash busy bit<br>When the operation is in progress, this bit is set to 1. When the operation is end or an error generated, this bit is clear to 0. |

### 2.4.5. Control register (FMC_CTL)

Address offset: 0x10

Reset value: 0x0000 0080

This register has to be accessed by word(32-bit)

| 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 | | OBRLD | ENDIE | Reserved | ERRIE | OBWEN | Reserved | LK | START | OBER | OBPG | Reserved | MER | PER | PG |
| | | rw | rw | | rw | rw | | rw | rw | rw | rw | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13 | OBRLD | Option byte reload bit<br>This bit is set by software.<br>0: No effect<br>1: Force option byte reload, and generate a system reset |
| 12 | ENDIE | End of operation interrupt enable bit |

| | | |
|---|---|---|
| | | This bit is set or cleared by software.<br>0: No interrupt generated by hardware<br>1: End of operation interrupt enable |
| 11 | Reserved | Must be kept at reset value |
| 10 | ERRIE | Error interrupt enable bit<br>This bit is set or cleared by software.<br>0: No interrupt generated by hardware<br>1: Error interrupt enable |
| 9 | OBWEN | Option byte erase/program enable bit<br>This bit is set by hardware when right sequence written to FMC_OBKEY register.<br>This bit can be cleared by software. |
| 8 | Reserved | Must be kept at reset value |
| 7 | LK | FMC_CTL lock bit<br>This bit is cleared by hardware when right sequent written to FMC_KEY register.<br>This bit can be set by software. |
| 6 | START | Send erase command to FMC bit<br>This bit is set by software to send erase command to FMC. This bit is cleared by hardware when the BUSY bit is cleared. |
| 5 | OBER | Option byte erase command bit<br>This bit is set or cleared by software.<br>0: No effect<br>1: Option byte erase command |
| 4 | OBPG | Option byte program command bit<br>This bit is set or cleared by software.<br>0: No effect<br>1: Option byte program command |
| 3 | Reserved | Must be kept at reset value |
| 2 | MER | Main flash mass erase command bit<br>This bit is set or cleared by software.<br>0: No effect<br>1: Main flash mass erase command |
| 1 | PER | Main flash page erase command bit<br>This bit is set or cleared by software.<br>0: No effect<br>1: Main flash page erase command |
| 0 | PG | Main flash page program command bit<br>This bit is set or cleared by software. |

0: No effect

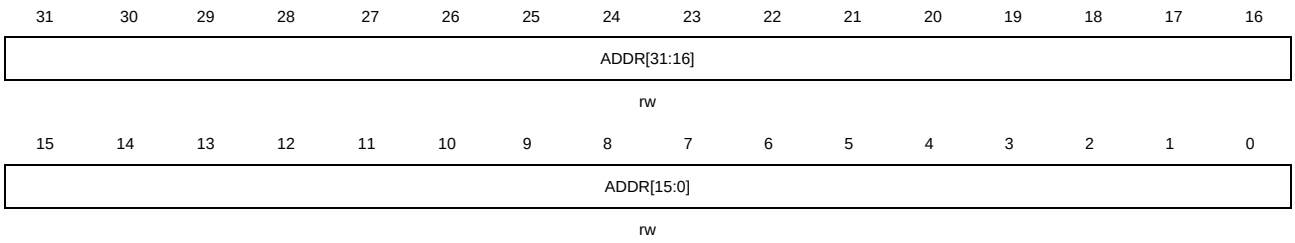
1: Main flash page program command

### 2.4.6. Address register (FMC_ADDR)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



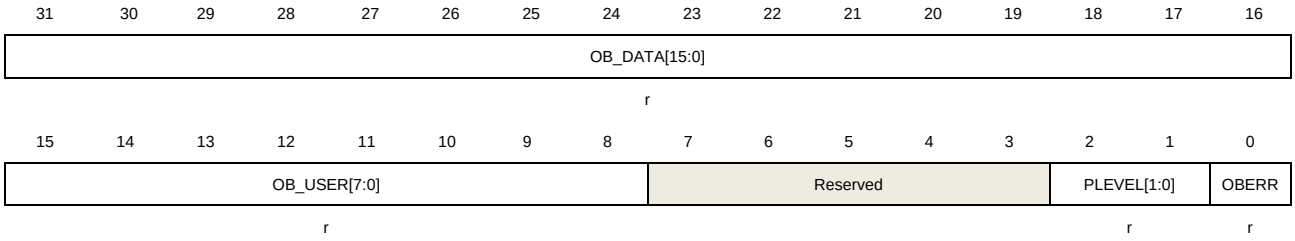

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR[31:0] | Flash command address bits<br>These bits are set by software.<br>ADDR bits are the address of flash erase command |

### 2.4.7. Option byte status register (FMC_OBSTAT)

Address offset: 0x1C

Reset value: 0xXXXX XX0X

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | OB_DATA[15:0] | Store OB_DATA[15:0] of option byte block after system reset |
| 15:8 | OB_USER[7:0] | Store OB_USER byte of option byte block after system reset |
| 7:3 | Reserved | Must be kept at reset value |
| 2:1 | PLEVEL[1:0] | Security Protection level<br>00: No protection level<br>01: Protect level low |

| | | |
|---|---|---|
| | | 11: Protect level high |
| 0 | OBERR | Option byte read error bit.<br>This bit is set by hardware when the option byte and its complement byte do not match, and the option byte set 0xFF. |

## 2.4.8. Write protection register (FMC_WP)

Address offset: 0x20
Reset value: 0x0000 XXXX

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OB_WP[31:16] | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OB_WP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | OB_WP[31:0] | Store OB_WP[31:0] of option byte block after system reset<br>0: Protection active<br>1: Unprotected |

## 2.4.9. Product ID register (FMC_PID)

Address offset: 0x100
Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PID[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PID[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PID[31:0] | Product reserved ID code register<br>These bits are read only by software.<br>These bits are unchanged constantly after power on. These bits are one time programmed when the chip product. |

# 3. Power management unit (PMU)

## 3.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32E23x series. Power management unit (PMU) provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. For GD32E23x series, there are three power domains, including $V_{DD}$ / $V_{DDA}$ domain, $V_{CORE}$ domain, and Backup domain, as is shown in ***Figure 3-1. Power supply overview***. The power of the $V_{DD}$ domain is supplied directly by $V_{DD}$. An embedded LDO in the $V_{DD}$ / $V_{DDA}$ domain is used to supply the $V_{CORE}$ domain power. Backup domain is powered from the main $V_{DD}$ supply.

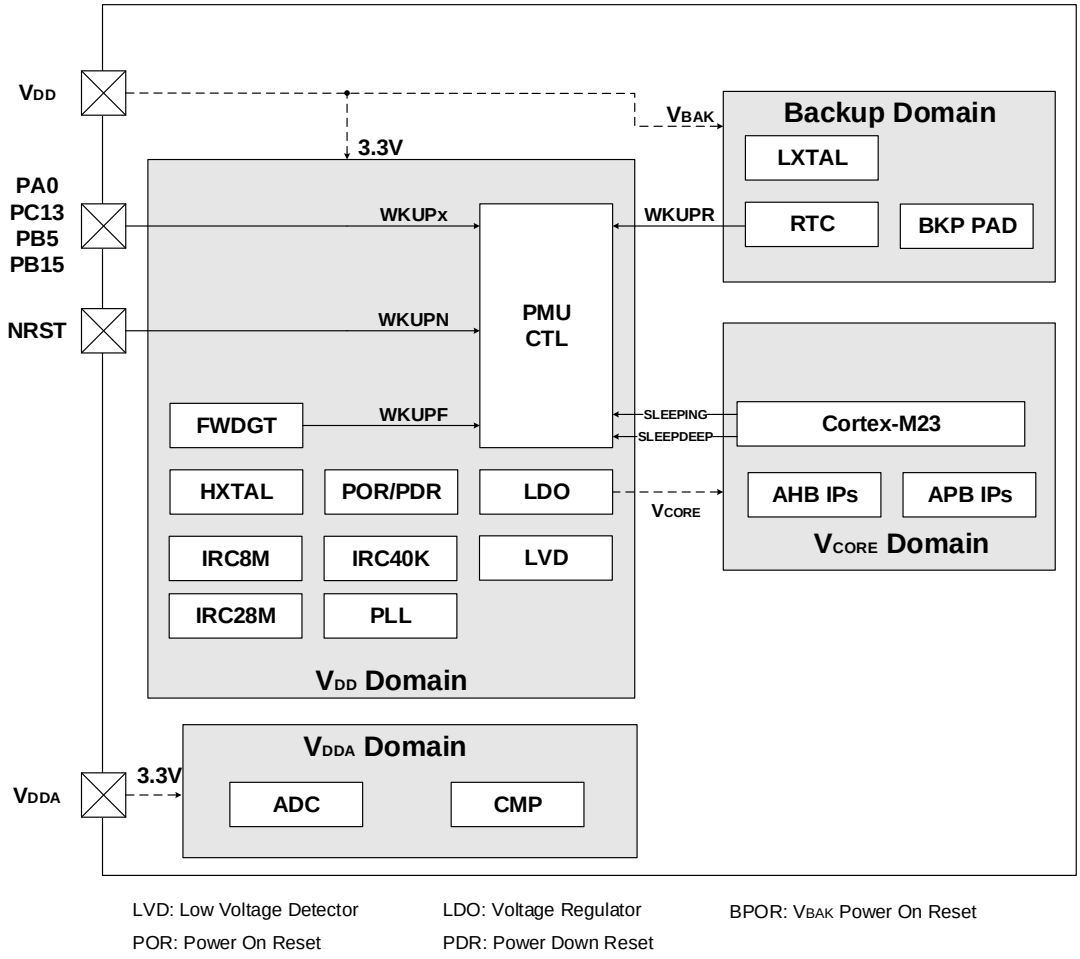
## 3.2. Characteristics

- Three power domains: $V_{BAK}$, $V_{DD}$ / $V_{DDA}$ and $V_{CORE}$ power domains.
- Three power saving modes: Sleep, Deep-sleep and Standby modes.
- Internal Voltage regulator (LDO) supplies around $V_{CORE}$ voltage source for $V_{CORE}$ domain.
- Low Voltage Detector (LVD) issue an interrupt or event when the power is lower than a programmed threshold.
- LDO output voltage select for power saving.

## 3.3. Function overview

***Figure 3-1. Power supply overview*** provides details on the internal configuration of the PMU and the relevant power domains.

**Figure 3-1. Power supply overview**




### 3.3.1. Backup domain

The Backup domain is powered by the $V_{DD}$, and the $V_{BAK}$ pin which drives Backup Domain, power supply for RTC unit, LXTAL oscillator, and three pads, including PC13 to PC15.

The Backup domain reset source only include the Backup Domain software reset. The application software can trigger the Backup domain software reset by setting the BKPRST bit in the RCU_BDCTL register to reset the Backup domain.

The clock source of the Real Time Clock (RTC) circuit can be derived from the Internal 40KHz RC oscillator (IRC40K) or the Low Speed Crystal oscillator (LXTAL), or HXTAL clock divided by 32. Before entering the power saving mode by executing the WFI / WFE instruction, the Cortex®-M23 can setup the RTC register with an expected alarm time and enable the alarm function to achieve the RTC alarm event. After entering the power saving mode for a certain amount of time, the RTC alarm will wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the ***Real-time clock(RTC)***.

When the Backup domain is supplied by $V_{DD}$ ($V_{BAK}$ pin is connected to $V_{DD}$), the following functions are available:

- PC13 can be used as GPIO or RTC function pin described in the ***Real-time clock(RTC)***.
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

**Note:** Since PC13, PC14, PC15 can only be obtained by a small current, the speed of GPIOs PC13 to PC15 should not exceed 2MHz when they are in output mode (maximum load: 30pF).

### 3.3.2. $V_{DD}$ / $V_{DDA}$ power domain

$V_{DD}$ / $V_{DDA}$ domain includes two parts: $V_{DD}$ domain and $V_{DDA}$ domain. $V_{DD}$ domain includes HXTAL (high speed crystal oscillator), LDO (voltage regulator), POR / PDR (power on / down reset), FWDGT (free watchdog timer), IRC8M (Internal 8MHz RC oscillator), IRC28M (Internal 28MHz RC oscillator), IRC40K (Internal 40KHz RC oscillator), PLL (phase locking loop), LVD (low voltage detector), all pads except PC13 / PC14 / PC15, etc. $V_{DDA}$ domain includes ADC (AD converter), CMP(comparator), etc.

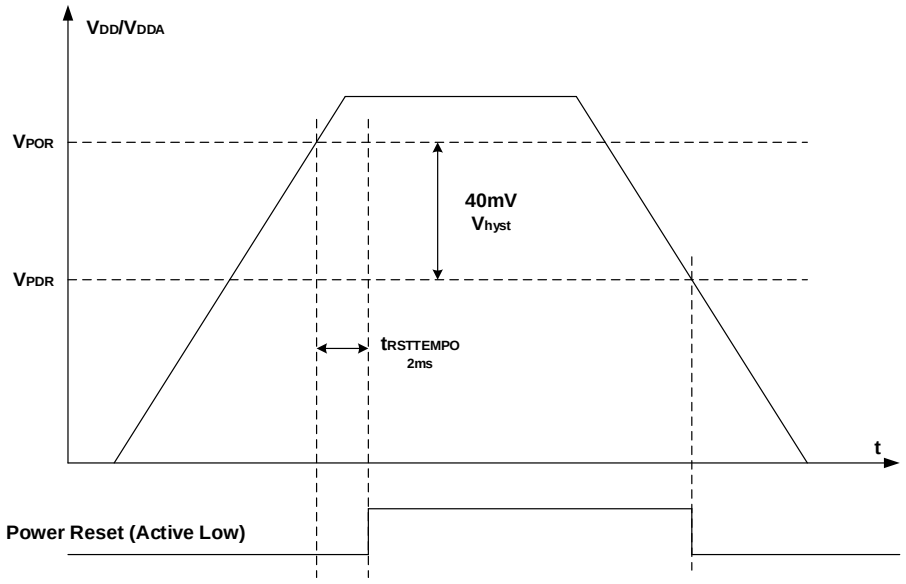
#### $V_{DD}$ domain

The LDO, which is implemented to supply power for the $V_{CORE}$ domain, is always enabled after reset. It can be configured to operate in three different statuses, including in the Sleep mode (full power on), in the Deep-sleep mode (on or low power), and in the Standby mode (power off).

The POR / PDR circuit is implemented to detect $V_{DD}$ / $V_{DDA}$ and generate the power reset signal which resets the whole chip except the Backup domain when the supply voltage is lower than the specified threshold. The POR circuit only detects $V_{DD}$ supply voltage, and the PDR circuit detects both $V_{DD}$ and $V_{DDA}$ supply voltages. ***Figure 3-2. Waveform of the POR / PDR*** shows the relationship between the supply voltage and the power reset signal. $V_{POR}$, which typical value is 1.71V, indicates the threshold of power on reset, while $V_{PDR}$, which typical value is 1.67V, means the threshold of power down reset. The hysteresis voltage ($V_{hyst}$) is around 40mV.

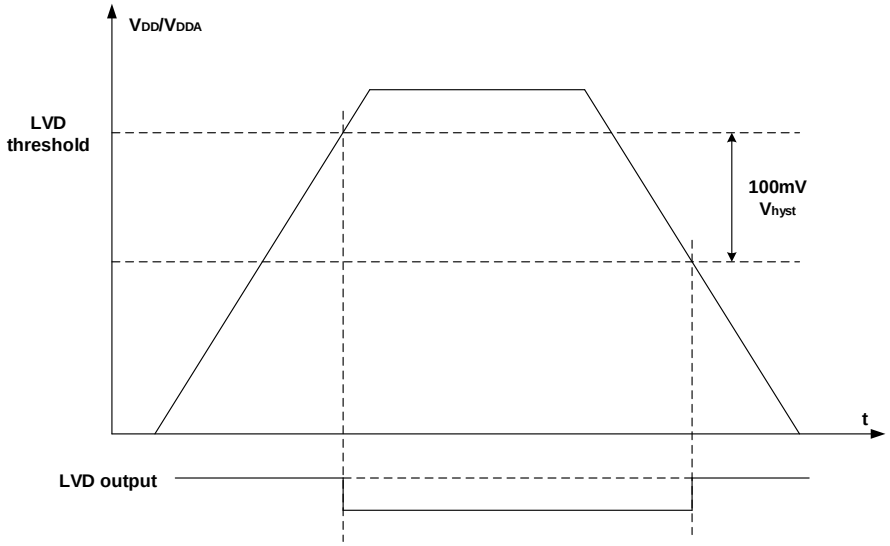
**Note:** $V_{DDA}$ monitor function detected by the PDR circuit can be disabled by reset VDDA_VISOR bit in option byte register OB_USER, to reduce power consumption when users are sure that the $V_{DDA}$ is higher than or equal to $V_{DD}$.

**Figure 3-2. Waveform of the POR / PDR**




The LVD is used to detect whether the $V_{DD}$ / $V_{DDA}$ supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU_CTL). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register (PMU_CS), indicates if $V_{DD}$ / $V_{DDA}$ is higher or lower than the LVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. ***Figure 3-3. Waveform of the LVD threshold*** shows the relationship between the LVD threshold and the LVD output (LVD interrupt signal depends on EXTI line 16 rising or falling edge configuration). The following figure shows the relationship between the supply voltage and the LVD signal. The hysteresis voltage ($V_{hyst}$) is 100mV.

**Figure 3-3. Waveform of the LVD threshold**



### $V_{DDA}$ domain

Generally, digital circuits are powered by $V_{DD}$, while most of analog circuits are powered by $V_{DDA}$. To improve the ADC conversion accuracy, the independent power supply $V_{DDA}$ is implemented to achieve better performance of analog circuits. $V_{DDA}$ can be externally connected to $V_{DD}$ through the external filtering circuit that avoids noise on $V_{DDA}$, and $V_{SSA}$ should be connected to $V_{SS}$ through the specific circuit independently. If $V_{DD}$ and $V_{DDA}$ are not provided by the same power supply (the voltage difference is less than 0.3V), $V_{DDA}$ must be greater than or equal to the $V_{DD}$ during power-on process.

## 3.3.3. $V_{CORE}$ power domain

The main functions that include Cortex®-M23 logic, AHB / APB peripherals, the APB interfaces for the Backup domain and the $V_{DD}$ / $V_{DDA}$ domain, etc, are located in this power domain. Once the $V_{CORE}$ is powered up, the POR will generate a reset sequence on the $V_{CORE}$ power domain. If need to enter the expected power saving mode, the associated control bits must be configured. Then, once a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed, the device will enter an expected power saving mode which will be discussed in the following section. Refer to the datasheet for the $V_{CORE}$ voltage value.

## 3.3.4. Power saving modes

After a system reset or a power reset, the GD32E23x MCU operates at full function and all power domains are active. Users can achieve lower power consumption through slowing down the system clocks (HCLK, PCLK1, PCLK2) or gating the clocks of the unused peripherals or configuring the LDO output voltage by LDOVS bits in PMU_CTL register. The LDOVS bits should be configured only when the PLL is off. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, and Standby mode.

### Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex®-M23. In Sleep mode, only clock of Cortex®-M23 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex®-M23 System Control Register, and execute a WFI or WFE instruction. If the Sleep mode is entered by executing a WFI instruction, any interrupt can wake up the system. If it is entered by executing a WFE instruction, any wakeup event can wake up the system (If SEVONPEND is 1, any interrupt can wake up the system, refer to Cortex®-M23 Technical Reference Manual). The mode offers the lowest wakeup time as no time is wasted in interrupt entry or exit.

According to the SLEEPONEXIT bit in the Cortex®-M23 System Control Register, there are two options to select the Sleep mode entry mechanism.

- 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 from the lowest priority ISR.

### Deep-sleep mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the Cortex®-M23. In Deep-sleep mode, all clocks in the $V_{CORE}$ domain are off, and all of IRC8M, IRC28M, HXTAL and PLL are disabled. The contents of SRAM and registers are preserved. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU_CTL register. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M23 System Control Register, and clear the STBMOD bit in the PMU_CTL register. Then, the device enters the Deep-sleep mode after a WFI or WFE instruction is executed. If the Deep-sleep mode is entered by executing a WFI instruction, any interrupt from EXTI lines can wake up the system. If it is entered by executing a WFE instruction, any wakeup event from EXTI lines can wake up the system (If SEVONPEND is 1, any interrupt from EXTI lines can wake up the system, refer to Cortex®-M23 Technical Reference Manual). When exiting the Deep-sleep mode, the IRC8M is selected as the system clock. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

**Note:** In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI_PD register) and related peripheral flags must be reset, refer to ***Table 5-3. EXTI source***. If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

### Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex®-M23, too. In Standby mode, the whole $V_{CORE}$ domain is power off, the LDO is shut down, and all of IRC8M, IRC28M, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M23 System Control Register, and set the STBMOD bit in the PMU_CTL register, and clear WUF bit in the PMU_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed, and the STBF status flag in the PMU_CS register indicates that the MCU has been in Standby mode. There are four wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC alarm / time stamp / tamper events, the FWDGT reset, and the rising edge on WKUP pins. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers in $V_{CORE}$ power domain are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex®-M23 will execute instruction code from the 0x00000000 address.

**Table 3-1. Power saving mode summary**

| Mode | Sleep | Deep-sleep | Standby |
|---|---|---|---|
| Description | Only CPU clock is off | 1. All clocks in the $V_{CORE}$ domain are off. | 1. The $V_{CORE}$ domain is power off. |

| Mode | Sleep | Deep-sleep | Standby |
|---|---|---|---|
| | | 2. Disable IRC8M, IRC28M, HXTAL and PLL | 2. Disable IRC8M, IRC28M, HXTAL and PLL. |
| LDO Status | On (normal power mode) | On (normal or low power mode) | Off |
| Configuration | SLEEPDEEP = 0 | SLEEPDEEP = 1<br>STBMOD = 0 | SLEEPDEEP = 1<br>STBMOD = 1, WURST=1 |
| Entry | WFI or WFE | WFI or WFE | WFI or WFE |
| Wakeup | Any interrupt for WFI<br>Any event (or interrupt when SEVONPEND is 1) for WFE | Any interrupt from EXTI lines for WFI<br>Any event(or interrupt when SEVONPEND is 1) from EXTI for WFE | 1. NRST pin<br>2. WKUP pins<br>3. FWDGT reset<br>4. RTC |
| Wakeup Latency | None | IRC8M wakeup time, LDO wakeup time added if LDO is in low power mode | Power on sequence |

**Note:** In Standby mode, all I / Os are in high-impedance state except NRST pin, PC13 pin when configured for RTC function, PC14 and PC15 pins when used as LXTAL crystal oscillator pins, and WKUP pins if enabled.

## 3.4. PMU registers

PMU base address: 0x4000 7000

### 3.4.1. Control register (PMU_CTL)

Address offset: 0x00

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

This register can be accessed by half-word(16-bit) or word(32-bit).

| 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 |
| LDOVS[1:0] | | Reserved | | | | | BKPWEN | LVDT[2:0] | | | LVDEN | STBRST | WURST | STBMOD | LDOLP |
| rs | | | | | | | rw | rw | | | rw | rc_w1 | rc_w1 | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:14 | LDOVS[1:0] | LDO output voltage select<br>These bits are set by software when the main PLL closed.<br>00: Reserved (LDO output voltage high mode)<br>01: LDO output voltage high mode<br>1x: LDO output voltage low mode |
| 13:9 | Reserved | Must be kept at reset value. |
| 8 | BKPWEN | Backup Domain Write Enable<br>0: Disable write access to the registers in Backup domain<br>1: Enable write access to the registers in Backup domain<br>After reset, any write access to the registers in Backup domain is ignored. This bit has to be set to enable write access to these registers. |
| 7:5 | LVDT[2:0] | Low Voltage Detector Threshold<br>000: 2.1V<br>001: 2.3V<br>010: 2.4V<br>011: 2.6V<br>100: 2.7V<br>101: 2.9V<br>110: 3.0V<br>111: 3.1V |
| 4 | LVDEN | Low Voltage Detector Enable<br>0: Disable Low Voltage Detector<br>1: Enable Low Voltage Detector |

| | | |
|---|---|---|
| | | **Note:** When LVD_LOCK bit is set to 1 in the SYSCFG_CFG2 register, LVDEN and LVDT[2:0] are read only. |
| 3 | STBRST | Standby Flag Reset<br>0: No effect<br>1: Reset the standby flag<br>This bit is always read as 0. |
| 2 | WURST | Wakeup Flag Reset<br>0: No effect<br>1: Reset the wakeup flag<br>This bit is always read as 0. |
| 1 | STBMOD | Standby Mode<br>0: Enter the Deep-sleep mode when the Cortex®-M23 enters SLEEPDEEP mode<br>1: Enter the Standby mode when the Cortex®-M23 enters SLEEPDEEP mode |
| 0 | LDOLP | LDO Low Power Mode<br>0: The LDO operates normally during the Deep-sleep mode<br>1: The LDO is in low power mode during the Deep-sleep mode<br>**Note:** Some peripherals may work with the IRC8M clock in the Deep-sleep mode. In this case, the LDO automatically switches from the low power mode to the normal mode and remains in this mode until the peripheral stop working. |

### 3.4.2. Control and status register (PMU_CS)

Address offset: 0x04

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

This register can be accessed by half-word(16-bit) or word(32-bit).

| 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 | WUPEN6 | WUPEN5 | Reserved | | | WUPEN1 | WUPEN0 | Reserved | | | | | LVDF | STBF | WUF |
| | rw | rw | | | | rw | rw | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | WUPEN6 | WKUP Pin6 (PB15) Enable<br>0: Disable WKUP pin6 function<br>1: Enable WKUP pin6 function<br>If WUPEN6 is set before entering the power saving mode, a rising edge on the WKUP pin6 wakes up the system from the power saving mode. As the WKUP pin6 is active high, the WKUP pin6 is internally configured to input pull down mode. And |

| | | |
|---|---|---|
| | | set this bit will trigger a wakeup event when the input is already high. |
| 13 | WUPEN5 | WKUP Pin5 (PB5) Enable<br>0: Disable WKUP pin5 function<br>1: Enable WKUP pin5 function<br>If WUPEN5 is set before entering the power saving mode, a rising edge on the WKUP pin5 wakes up the system from the power saving mode. As the WKUP pin5 is active high, the WKUP pin5 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 12:10 | Reserved | Must be kept at reset value. |
| 9 | WUPEN1 | WKUP Pin 1 (PC13) Enable<br>0: Disable WKUP pin1 function<br>1: Enable WKUP pin1 function<br>If WUPEN1 is set before entering the power saving mode, a rising edge on the WKUP pin1 wakes up the system from the power saving mode. As the WKUP pin1 is active high, the WKUP pin1 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 8 | WUPEN0 | WKUP Pin 0 (PA0) Enable<br>0: Disable WKUP pin0 function<br>1: Enable WKUP pin0 function<br>If WUPEN0 is set before entering the power saving mode, a rising edge on the WKUP pin0 wakes up the system from the power saving mode. As the WKUP pin0 is active high, the WKUP pin0 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | LVDF | Low Voltage Detector Status Flag<br>0: Low Voltage event has not occurred ($V_{DD}$ is higher than the specified LVD threshold)<br>1: Low Voltage event occurred ($V_{DD}$ is equal to or lower than the specified LVD threshold)<br>**Note:** The LVD function is stopped in Standby mode. |
| 1 | STBF | Standby Flag<br>0: The device has not entered the Standby mode<br>1: The device has been in the Standby mode<br>This bit is cleared only by a POR/PDR or by setting the STBRST bit in the PMU_CTL register. |
| 0 | WUF | Wakeup Flag<br>0: No wakeup event has been received<br>1: Wakeup event occurred from the WKUP pin or the RTC wakeup event including RTC Tamper event, RTC alarm event，RTC Time Stamp event |

This bit is cleared only by a POR / PDR or by setting the WURST bit in the PMU_CTL register.

# 4. Reset and clock unit (RCU)

## 4.1. Reset control unit (RCTL)

### 4.1.1. Overview

GD32E23x reset control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power on reset, known as a cold reset, resets the full system except the backup domain during a power up. A system reset resets the processor core and peripheral IP components with the exception of the SW-DP controller and the backup domain. A backup domain reset resets the backup domain. These resets can be triggered by an external signal, internal events and the reset generators. More information about these resets will be described in the following sections.

### 4.1.2. Function overview

#### Power Reset

The power reset is generated by either an external reset as power on and power down reset (POR/PDR reset), or by the internal reset generator when exiting Standby mode. The power reset sets all registers to their reset values except the backup domain. The power reset which active signal is low will be de-asserted when the internal LDO voltage regulator is ready to provide $V_{CORE}$ power for GD32E23x series. The reset service routine vector is fixed at address 0x0000_0004 in the memory map.

#### System Reset

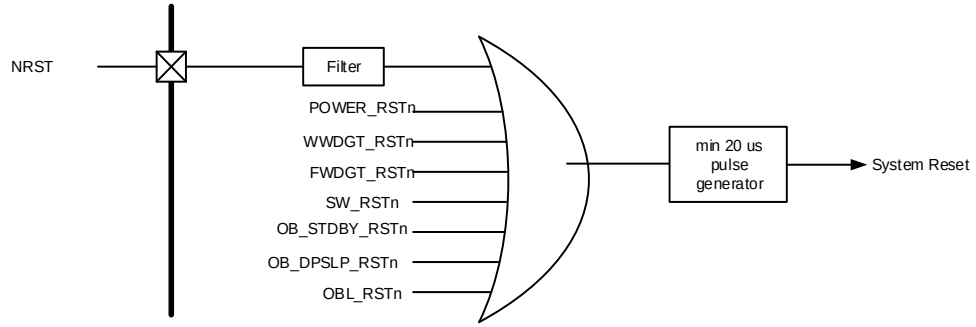
A system reset is generated by the following events:

- A power reset (POWER_RSTn)
- An external pin reset (NRST)
- A window watchdog timer reset (WWDGT_RSTn)
- A free watchdog timer reset (FWDGT_RSTn)
- The SYSRESETREQ bit in Cortex®-M23 application interrupt and reset control register is set (SW_RSTn)
- Option byte loader reset (OBL_RSTn)
- Reset generated when entering Standby mode when resetting nRST_STDBY bit in user option bytes (OB_STDBY_RSTn)
- Reset generated when entering Deep-sleep mode when resetting nRST_DPSLP bit in user option bytes (OB_DPSLP_RSTn)

A system reset resets the processor core and peripheral IP components except for the SW-DP controller and the backup domain.

A system reset pulse generator guarantees low level pulse duration of 20 μs for each reset source (external or internal reset).

**Figure 4-1. The system reset circuit**




**Backup domain reset**

A backup domain reset is generated by setting the BKPRST bit in the backup domain control register or backup domain power on reset ($V_{DD}$ power on).

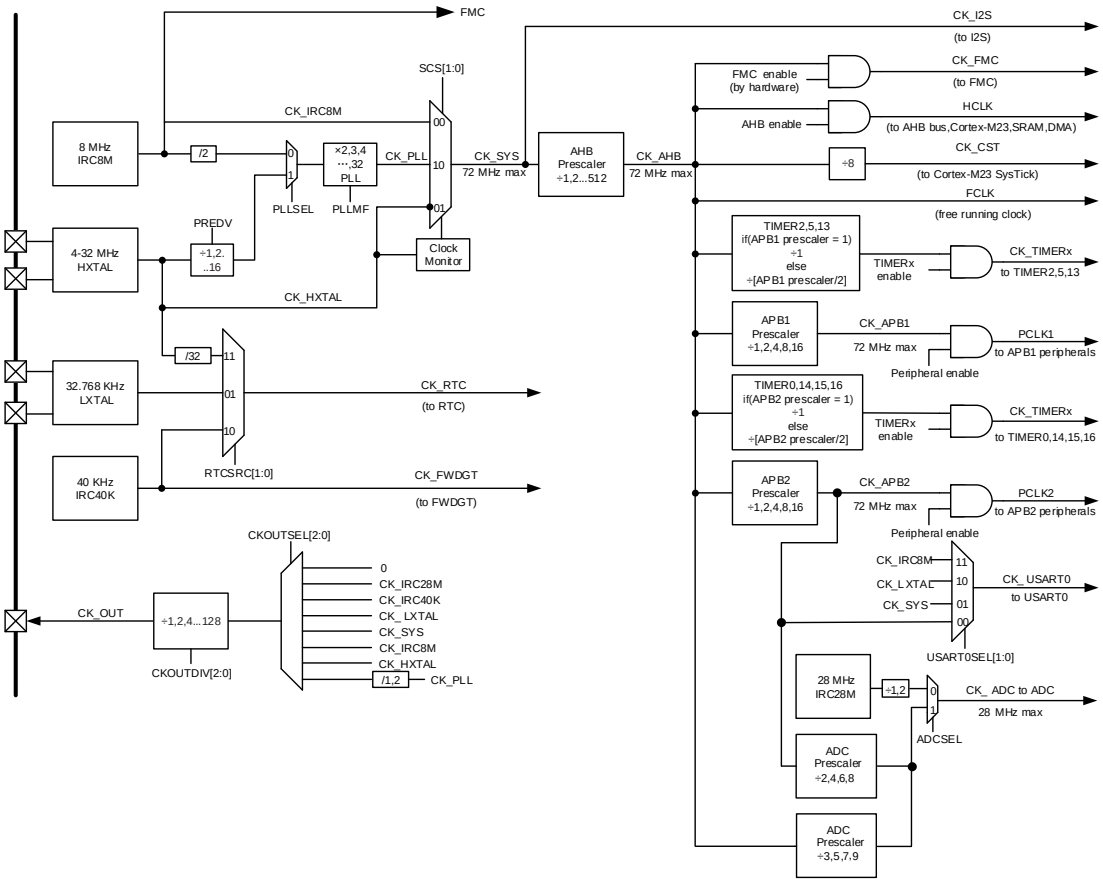
## 4.2. Clock control unit (CCTL)

### 4.2.1. Overview

The clock control unit provides a range of frequencies and clock functions. These include an Internal 8 MHz RC oscillator (IRC8M), an Internal 28 MHz RC oscillator (IRC28M), a High speed crystal oscillator (HXTAL), Internal 40KHz RC oscillator (IRC40K), a Low speed crystal oscillator (LXTAL), a Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex®-M23 are derived from the system clock (CK_SYS) which can source from the IRC8M, HXTAL or PLL. The maximum operating frequency of the system clock (CK_SYS) can be up to 72 MHz. The Free Watchdog Timer has independent clock source (IRC40K), and Real Time Clock (RTC) use the IRC40K, LXTAL or HXTAL/32 as its clock source.

**Figure 4-2. Clock tree**




The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB, APB2 and APB1 domains is 72 MHz/72 MHz/72 MHz. The Cortex System Timer (SysTick) external clock is clocked with the AHB clock (HCLK) divided by 8. The systick can work either with this clock or with the AHB clock (HCLK), configurable in the systick control and status register.

The ADC are clocked by the clock of APB2 divided by 2, 4, 6, 8 or by the clock of AHB divided by 3, 5, 7, 9 or IRC28M or IRC28M/2 clock for GD32E23x series selected by ADCSEL bit in configuration register 2 (RCU_CFG2). The USART0 is clocked by IRC8M clock or LXTAL clock or system clock or APB2 clock, which selected by USART0SEL bits in configuration register 2 (RCU_CFG2).

The RTC is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 32 which select by RTCSRC bits in backup domain control register (RCU_BDCTL).

The FWDGT is clocked by IRC40K clock, which is forced on when FWDGT started.

If the APB prescaler is 1, the timer clock frequencies are set to AHB frequency divide by 1. Otherwise, they are set to the AHB frequency divide by half of APB prescaler.

### 4.2.2. Characteristics

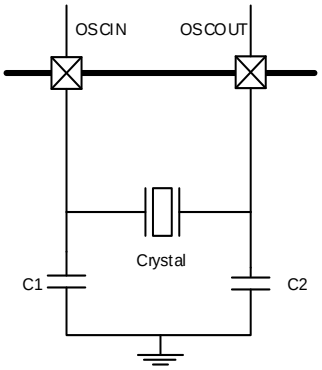
- 4 to 32 MHz High speed crystal oscillator (HXTAL)
- Internal 8 MHz RC oscillator (IRC8M)
- Internal 28 MHz RC oscillator (IRC28M)
- 32.768 KHz Low speed crystal oscillator (LXTAL)
- Internal 40 KHz RC oscillator (IRC40K)
- PLL clock source can be HXTAL or IRC8M
- HXTAL clock monitor

### 4.2.3. Function overview

**High Speed Crystal Oscillator (HXTAL)**

The high speed crystal oscillator (HXTAL), which has a frequency from 4 to 32 MHz, produces a highly accurate clock source for use as the system clock. A crystal with a specific frequency must be connected and located close to the two HXTAL pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

**Figure 4-3. HXTAL clock source**




The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the control register 0, RCU_CTL0. The HXTALSTB flag in control register 0, RCU_CTL0 indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator “Start-up time”. As the HXTAL becomes stable, an interrupt will be generated if the related interrupt enable bit HXTALSTBIE in the Interrupt register RCU_INT is set. At this point the HXTAL clock can be used directly as the system clock source or the PLL input clock.

Select external clock bypass mode by setting the HXTALBPS and HXTALEN bits in the control register 0, RCU_CTL0. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

**Internal 8 MHz RC Oscillator (IRC8M)**

The internal 8 MHz RC oscillator, IRC8M, has a fixed frequency of 8 MHz and is the default clock source selection for the CPU when the device is powered up. The IRC8M oscillator provides a lower cost type clock source as no external components are required. The IRC8M RC oscillator can be switched on or off using the IRC8MEN bit in the Control register 0, RCU_CTL0. The IRC8MSTB flag in the control register 0, RCU_CTL0 is used to indicate if the internal RC oscillator is stable. The start-up time of the IRC8M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC8MSTBIE, in the Interrupt register, RCU_INT, is set when the IRC8M becomes stable. The IRC8M clock can also be used as the PLL input clock.

The frequency accuracy of the IRC8M can be calibrated by the manufacturer, but its operating frequency is still less accurate than HXTAL. The application requirements, environment and cost will determine which oscillator type is selected.

If the HXTAL or PLL is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the IRC8M clock to be the system clock when the system initially wakes-up.

**Phase Locked Loop (PLL)**

The internal Phase Locked Loop, PLL, can provide 16~72 MHz clock output which is 2 ~32 multiples of a fundamental reference frequency of 4 ~ 32 MHz.

The PLL can be switched on or off by using the PLLEN bit in the Control register 0, RCU_CTL0. The PLLSTB flag in the Control register 0, RCU_CTL0 will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the Interrupt register, RCU_INT, is set as the PLL becomes stable.

**Internal 28 MHz RC Oscillator (IRC28M)**

The internal 28 MHz RC Oscillator, IRC28M, has a fixed frequency of 28 MHz and dedicated as ADC clock. The IRC28M RC oscillator can be switched on or off using the IRC28MEN bit in the control register 1 (RCU_CTL1). The IRC28MSTB flag in the control register 1 (RCU_CTL1) is used to indicate if the internal 28M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC28MSTBIE, in the Interrupt register, RCU_INT, is set when the IRC28M becomes stable.

**Low Speed Crystal Oscillator (LXTAL)**

The low speed crystal or ceramic resonator oscillator, which has a frequency of 32,768 Hz, produces a low power but highly accurate clock source for the real time clock circuit. The LXTAL oscillator can be switched on or off using the LXTALEN bit in the backup domain control register(RCU_BDCTL). The LXTALSTB flag in the backup domain control register(RCU_BDCTL) will indicate if the LXTAL clock is stable. An interrupt can be generated if the related interrupt enable bit, LXTALSTBIE, in the interrupt register RCU_INT is set when the LXTAL becomes stable.

Select external clock bypass mode by setting the LXTALBPS and LXTALEN bits in the backup domain control r2egister(RCU_BDCTL). The CK_LXTAL is equal to the external clock which drives the OSC32IN pin.

**Internal 40 KHz RC Oscillator (IRC40K)**

The internal 40 KHz RC Oscillator has a frequency of about 40 kHz and is a low power clock source for the real time clock circuit or the free watchdog timer. The IRC40K offers a low cost clock source as no external components are required. The IRC40K RC oscillator can be switched on or off by using the IRC40KEN bit in the reset source/clock register, RCU_RSTSCK. The IRC40KSTB flag in the reset source/clock register RCU_RSTSCK will indicate if the IRC40K clock is stable. An interrupt can be generated if the related interrupt enable bit IRC40KSTBIE in the interrupt register RCU_INT is set when the IRC40K becomes stable.

**System Clock (CK_SYS) Selection**

After the system reset, the default CK_SYS source will be IRC8M and can be switched to HXTAL or PLL by changing the system clock switch bits, SCS, in the configuration register 0, RCU_CFG0. When the SCS value is changed, the CK_SYS will continue to operate using the original clock source until the target clock source is stable. When a clock source is used directly by the CK_SYS or the PLL, it is not possible to stop it.

**HXTAL Clock Monitor (CKM)**

The HXTAL clock monitor function is enabled by the HXTAL clock monitor enable bit, CKMEN, in the control register 0, RCU_CTL0. This function should be enabled after the HXTAL start-up delay and disabled when the HXTAL is stopped. Once the HXTAL failure is detected, the HXTAL will be automatically disabled. The HXTAL clock stuck flag, CKMIF, in the Interrupt register, RCU_INT, will be set and the HXTAL failure event will be generated. This failure interrupt is connected to the Non-Maskable Interrupt, NMI, of the Cortex®-M23. If the HXTAL is selected as the clock source of CK_SYS or PLL, the HXTAL failure will force the CK_SYS source to IRC8M and the PLL will be disabled automatically

**Clock Output Capability**

The clock output capability is ranging from 32 kHz to 72 MHz. There are several clock signals can be selected via the CK_OUT clock source selection bits, CKOUTSEL, in the configuration register 0 (RCU_CFG0). The corresponding GPIO pin should be configured in the properly alternate function I/O (AFIO) mode to output the selected clock signal.

**Table 4-1. Clock source select**

| Clock Source Selection bits | Clock Source |
|---|---|
| 000 | No Clock |
| 001 | CK_IRC28M |
| 010 | CK_IRC40K |
| 011 | CK_LXTAL |
| 100 | CK_SYS |

| Clock Source Selection bits | Clock Source |
|---|---|
| 101 | CK_IRC8M |
| 110 | CK_HXTAL |
| 111 | CK_PLL or CK_PLL/2 |

The CK_OUT frequency can be reduced by a configurable binary divider, controlled by the CKOUTDIV[2:0] bits, in the configuration register 0(RCU_CFG0).

**Deep-sleep mode clock control**

When the MCU is in Deep-sleep mode, the USART0 can wake up the MCU, when their clock is provided by LXTAL clock and LXTAL clock is enable.

If the USART0 clock is selected IRC8M clock in Deep-sleep mode, they have capable of open IRC8M clock or close IRC8M clock, which used to the USART0 to wake up the Deep-sleep mode.

**Voltage control**

The core domain voltage in Deep-sleep mode can be controlled by DSLPVS[1:0] bits in the Deep-sleep mode voltage register (RCU_DSV).

**Table 4-2. Core domain voltage selected in Deep-sleep mode -**

| DSLPVS[1:0] | Deep-sleep mode voltage(V) |
|---|---|
| 00 | 1.0 |
| 01 | 0.9 |
| 10 | 0.8 |
| 11 | 1.2 |

The RCU_DSV register are protected by voltage key register (RCU_VKEY). Only after write 0x1A2B3C4D to the RCU_VKEY register, the RCU_DSV register can be write.

## 4.3. Register definition

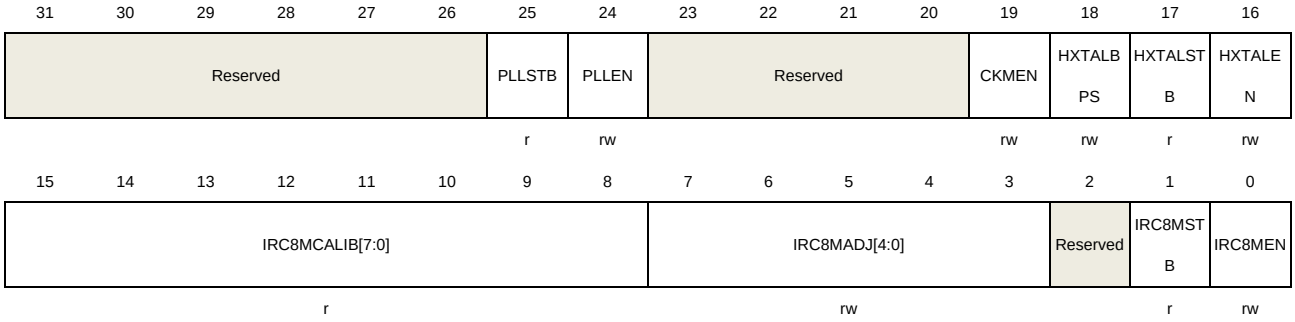
RCU base address: 0x4002 1000

### 4.3.1. Control register 0 (RCU_CTL0)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25 | PLLSTB | PLL clock stabilization flag<br>Set by hardware to indicate if the PLL output clock is stable and ready for use.<br>0: PLL is not stable<br>1: PLL is stable |
| 24 | PLLEN | PLL enable<br>Set and reset by software. This bit cannot be reset if the PLL clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL is switched off<br>1: PLL is switched on |
| 23:20 | Reserved | Must be kept at reset value. |
| 19 | CKMEN | HXTAL clock monitor enable<br>0: Disable the external 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor<br>1: Enable the external 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor<br>When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC8M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software.<br>**Note:** When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC8M internal RC oscillator regardless of the control bit, IRC8MEN, |

| | | |
|---|---|---|
| | | state. |
| 18 | HXTALBPS | External crystal oscillator (HXTAL) clock bypass mode enable<br>The HXTALBPS bit can be written only if the HXTALEN is 0.<br>0: Disable the HXTAL Bypass mode<br>1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock. |
| 17 | HXTALSTB | External crystal oscillator (HXTAL) clock stabilization flag<br>Set by hardware to indicate if the HXTAL oscillator is stable and ready for use.<br>0: HXTAL oscillator is not stable<br>1: HXTAL oscillator is stable |
| 16 | HXTALEN | External high speed oscillator enable<br>Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL input clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: External 4 ~ 32 MHz crystal oscillator disabled<br>1: External 4 ~ 32 MHz crystal oscillator enabled |
| 15:8 | IRC8MCALIB[7:0] | Internal 8M RC oscillator calibration value register<br>These bits are load automatically at power on. |
| 7:3 | IRC8MADJ[4:0] | Internal 8M RC oscillator clock trim adjust value<br>These bits are set by software. The trimming value is there bits (IRC8MADJ) added to the IRC8MCALIB[7:0] bits. The trimming value should trim the IRC8M to 8 MHz ± 1%. |
| 2 | Reserved | Must be kept at reset value. |
| 1 | IRC8MSTB | IRC8M high speed internal oscillator stabilization flag<br>Set by hardware to indicate if the IRC8M oscillator is stable and ready for use.<br>0: IRC8M oscillator is not stable<br>1: IRC8M oscillator is stable |
| 0 | IRC8MEN | Internal high speed oscillator enable<br>Set and reset by software. This bit cannot be reset if the IRC8M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when HXTALCKM is set.<br>0: Internal 8 MHz RC oscillator disabled<br>1: Internal 8 MHz RC oscillator enabled |

### 4.3.2. Configuration register 0 (RCU_CFG0)

Address offset: 0x04
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLDV | CKOUTDIV[2:0] | | | PLLMF[4] | CKOUTSEL[2:0] | | | Reserved | | PLLMF[3:0] | | | | PLLPRE DV | PLLSEL |
| rw | rw | | | rw | rw | | | | | rw | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCPSC[1:0] | | APB2PSC[2:0] | | | APB1PSC[2:0] | | | AHBPSC[3:0] | | | | SCSS[1:0] | | SCS[1:0] | |
| rw | | rw | | | rw | | | rw | | | | r | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLLDV | The CK_PLL divide by 1 or 2 for CK_OUT<br>0: CK_PLL divide by 2 for CK_OUT<br>1: CK_PLL divide by 1 for CK_OUT |
| 30:28 | CKOUTDIV[2:0] | The CK_OUT divider which the CK_OUT frequency can be reduced see bits 26:24 of RCU_CFG0 for CK_OUT.<br>000: The CK_OUT is divided by 1<br>001: The CK_OUT is divided by 2<br>010: The CK_OUT is divided by 4<br>011: The CK_OUT is divided by 8<br>100: The CK_OUT is divided by 16<br>101: The CK_OUT is divided by 32<br>110: The CK_OUT is divided by 64<br>111: The CK_OUT is divided by 128 |
| 27 | PLLMF[4] | Bit 4 of PLLMF register<br>see bits 21:18 of RCU_CFG0. |
| 26:24 | CKOUTSEL[2:0] | CK_OUT clock source selection<br>Set and reset by software.<br>000: No clock selected<br>001: Internal 28M RC oscillator clock selected<br>010: Internal 40K RC oscillator clock selected<br>011: External low speed oscillator clock selected<br>100: System clock selected<br>101: Internal 8MHz RC Oscillator clock selected<br>110: External High Speed oscillator clock selected<br>111: (CK_PLL / 2) or CK_PLL selected depend on PLLDV |
| 23:22 | Reserved | Must be kept at reset value |
| 21:18 | PLLMF[3:0] | PLL multiply factor<br>These bits and bit 27 of RCU_CFG0 are written by software to define the PLL multiplication factor.<br>00000: (PLL source clock x 2)<br>00001: (PLL source clock x 3) |

00010: (PLL source clock x 4)
00011: (PLL source clock x 5)
00100: (PLL source clock x 6)
00101: (PLL source clock x 7)
00110: (PLL source clock x 8)
00111: (PLL source clock x 9)
01000: (PLL source clock x 10)
01001: (PLL source clock x 11)
01010: (PLL source clock x 12)
01011: (PLL source clock x 13)
01100: (PLL source clock x 14)
01101: (PLL source clock x 15)
01110: (PLL source clock x 16)
01111: (PLL source clock x 16)
10000: (PLL source clock x 17)
10001: (PLL source clock x 18)
10010: (PLL source clock x 19)
10011: (PLL source clock x 20)
10100: (PLL source clock x 21)
10101: (PLL source clock x 22)
10110: (PLL source clock x 23)
10111: (PLL source clock x 24)
11000: (PLL source clock x 25)
11001: (PLL source clock x 26)
11010: (PLL source clock x 27)
11011: (PLL source clock x 28)
11100: (PLL source clock x 29)
11101: (PLL source clock x 30)
11110: (PLL source clock x 31)
11111: (PLL source clock x 32)
**Note:** The PLL output frequency must not exceed 72 MHz.

| | | |
|---|---|---|
| 17 | PLLPREDV | HXTAL divider for PLL source clock selection. This bit is the same bit as bit PREDV[0] from RCU_CFG1. Refer to RCU_CFG1 PREDV bits description.<br>Set and cleared by software to divide or not which is selected to PLL.<br>0: HXTAL clock selected<br>1: HXTAL / 2 clock selected |
| 16 | PLLSEL | PLL clock source selection<br>Set and reset by software to control the PLL clock source.<br>0: (IRC8M / 2) clock selected as source clock of PLL<br>1: HXTAL selected as source clock of PLL |
| 15:14 | ADCPSC[1:0] | ADC clock prescaler selection<br>These bits and bit 31 of RCU_CFG2 are written by software to define the ADC |

| | | |
|---|---|---|
| | | clock prescaler. Set and cleared by software.<br>000: (CK_APB2 / 2) selected<br>001: (CK_ APB2 / 4) selected<br>010: (CK_ APB2 / 6) selected<br>011: (CK_ APB2 / 8) selected<br>100: (CK_AHB / 3) selected<br>101: (CK_ AHB / 5) selected<br>110: (CK_ AHB / 7) selected<br>111: (CK_ AHB / 9) selected |
| 13:11 | APB2PSC[2:0] | APB2 prescaler selection<br>Set and reset by software to control the APB2 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 10:8 | APB1PSC[2:0] | APB1 prescaler selection<br>Set and reset by software to control the APB1 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 7:4 | AHBPSC[3:0] | AHB prescaler selection<br>Set and reset by software to control the AHB clock division ratio<br>0xxx: CK_SYS selected<br>1000: (CK_SYS / 2) selected<br>1001: (CK_SYS / 4) selected<br>1010: (CK_SYS / 8) selected<br>1011: (CK_SYS / 16) selected<br>1100: (CK_SYS / 64) selected<br>1101: (CK_SYS / 128) selected<br>1110: (CK_SYS / 256) selected<br>1111: (CK_SYS / 512) selected |
| 3:2 | SCSS[1:0] | System clock switch status<br>Set and reset by hardware to indicate the clock source of system clock.<br>00: Select CK_IRC8M as the CK_SYS source<br>01: Select CK_HXTAL as the CK_SYS source<br>10: Select CK_PLL as the CK_SYS source<br>11: Reserved |
| 1:0 | SCS[1:0] | System clock switch |

Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC8M when leaving Deep-sleep and Standby mode or by HXTAL clock monitor when the HXTAL failure is detected and the HXTAL is selected as the clock source of CK_SYS or PLL.

00: Select CK_IRC8M as the CK_SYS source

01: Select CK_HXTAL as the CK_SYS source

10: Select CK_PLL as the CK_SYS source

11: Reserved

### 4.3.3. Interrupt register (RCU_INT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CKMIC | Reserved | IRC28M STBIC | PLL STBIC | HXTAL STBIC | IRC8M STBIC | LXTAL STBIC | IRC40K STBIC |
| | | | | | | | | w | | w | w | w | w | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | IRC28M STBIE | PLL STBIE | HXTAL STBIE | IRC8M STBIE | LXTAL STBIE | IRC40K STBIE | CKMIF | Reserved | IRC28M STBIF | PLL STBIF | HXTAL STBIF | IRC8M STBIF | LXTAL STBIF | IRC40K STBIF |
| | | rw | rw | rw | rw | rw | rw | r | | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | CKMIC | HXTAL clock stuck interrupt clear<br>Write 1 by software to reset the CKMIF flag.<br>0: Not reset CKMIF flag<br>1: Reset CKMIF flag |
| 22 | Reserved | Must be kept at reset value. |
| 21 | IRC28MSTBIC | IRC28M stabilization interrupt clear<br>Write 1 by software to reset the IRC28MSTBIF flag.<br>0: Not reset IRC28MSTBIF flag<br>1: Reset IRC28MSTBIF flag |
| 20 | PLLSTBIC | PLL stabilization interrupt clear<br>Write 1 by software to reset the PLLSTBIF flag.<br>0: Not reset PLLSTBIF flag<br>1: Reset PLLSTBIF flag |

| | | |
|---|---|---|
| 19 | HXTALSTBIC | HXTAL stabilization interrupt clear<br>Write 1 by software to reset the HXTALSTBIF flag.<br>0: Not reset HXTALSTBIF flag<br>1: Reset HXTALSTBIF flag |
| 18 | IRC8MSTBIC | IRC8M stabilization interrupt clear<br>Write 1 by software to reset the IRC8MSTBIF flag.<br>0: Not reset IRC8MSTBIF flag<br>1: Reset IRC8MSTBIF flag |
| 17 | LXTALSTBIC | LXTAL stabilization interrupt clear<br>Write 1 by software to reset the LXTALSTBIF flag.<br>0: Not reset LXTALSTBIF flag<br>1: Reset LXTALSTBIF flag |
| 16 | IRC40KSTBIC | IRC40K stabilization interrupt clear<br>Write 1 by software to reset the IRC40KSTBIF flag.<br>0: Not reset IRC40KSTBIF flag<br>1: Reset IRC40KSTBIF flag |
| 15:14 | Reserved | Must be kept at reset value |
| 13 | IRC28MSTBIE | IRC28M stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC28M stabilization interrupt.<br>0: Disable the IRC28M stabilization interrupt<br>1: Enable the IRC28M stabilization interrupt |
| 12 | PLLSTBIE | PLL stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL stabilization interrupt.<br>0: Disable the PLL stabilization interrupt<br>1: Enable the PLL stabilization interrupt |
| 11 | HXTALSTBIE | HXTAL stabilization interrupt enable<br>Set and reset by software to enable/disable the HXTAL stabilization interrupt<br>0: Disable the HXTAL stabilization interrupt<br>1: Enable the HXTAL stabilization interrupt |
| 10 | IRC8MSTBIE | IRC8M stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC8M stabilization interrupt<br>0: Disable the IRC8M stabilization interrupt<br>1: Enable the IRC8M stabilization interrupt |
| 9 | LXTALSTBIE | LXTAL stabilization interrupt enable<br>LXTAL stabilization interrupt enable/disable control<br>0: Disable the LXTAL stabilization interrupt<br>1: Enable the LXTAL stabilization interrupt |
| 8 | IRC40KSTBIE | IRC40K stabilization interrupt enable |

| | | |
|---|---|---|
| | | IRC40K stabilization interrupt enable/disable control<br>0: Disable the IRC40K stabilization interrupt<br>1: Enable the IRC40K stabilization interrupt |
| 7 | CKMIF | HXTAL clock stuck interrupt flag<br>Set by hardware when the HXTAL clock is stuck.<br>Reset by software when setting the CKMIC bit.<br>0: Clock operating normally<br>1: HXTAL clock stuck |
| 6 | Reserved | Must be kept at reset value |
| 5 | IRC28MSTBIF | IRC28M stabilization interrupt flag<br>Set by hardware when the IRC28M is stable and the IRC28MSTBIE bit is set.<br>Reset by software when setting the IRC28MSTBIC bit.<br>0: No IRC28M stabilization interrupt generated<br>1: IRC28M stabilization interrupt generated |
| 4 | PLLSTBIF | PLL stabilization interrupt flag<br>Set by hardware when the PLL is stable and the PLLSTBIE bit is set.<br>Reset by software when setting the PLLSTBIC bit.<br>0: No PLL stabilization interrupt generated<br>1: PLL stabilization interrupt generated |
| 3 | HXTALSTBIF | HXTAL stabilization interrupt flag<br>Set by hardware when the External 4 ~ 32 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.<br>Reset by software when setting the HXTALSTBIC bit.<br>0: No HXTAL stabilization interrupt generated<br>1: HXTAL stabilization interrupt generated |
| 2 | IRC8MSTBIF | IRC8M stabilization interrupt flag<br>Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set.<br>Reset by software when setting the IRC8MSTBIC bit.<br>0: No IRC8M stabilization interrupt generated<br>1: IRC8M stabilization interrupt generated |
| 1 | LXTALSTBIF | LXTAL stabilization interrupt flag<br>Set by hardware when the external 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set.<br>Reset by software when setting the LXTALSTBIC bit.<br>0: No LXTAL stabilization interrupt generated<br>1: LXTAL stabilization interrupt generated |
| 0 | IRC40KSTBIF | IRC40K stabilization interrupt flag<br>Set by hardware when the Internal 40kHz RC oscillator clock is stable and the |

IRC40KSTBIE bit is set.

Reset by software when setting the IRC40KSTBIC bit.

0: No IRC40K stabilization clock ready interrupt generated

1: IRC40K stabilization interrupt generated

### 4.3.4. APB2 reset register (RCU_APB2RST)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | TIMER16 RST | TIMER15 RST | TIMER14 RST |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USART0 RST | Reserved. | SPI0 RST | TIMER0 RST | Reserved. | ADC RST | Reserved | | | | | | | | CFGCMP RST |
| | rw | | rw | rw | | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value |
| 18 | TIMER16RST | TIMER16 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER16 |
| 17 | TIMER15RST | TIMER15 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER15 |
| 16 | TIMER14RST | TIMER14 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER14 |
| 15 | Reserved | Must be kept at reset value |
| 14 | USART0RST | USART0 Reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART0 |
| 13 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 12 | SPI0RST | SPI0 Reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI0 |
| 11 | TIMER0RST | TIMER0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER0 |
| 10 | Reserved | Must be kept at reset value |
| 9 | ADCRST | ADC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the ADC |
| 8:1 | Reserved | Must be kept at reset value |
| 0 | CFGCMPRST | System configuration and comparator reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset system configuration and comparator |

### 4.3.5. APB1 reset register (RCU_APB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | PMURST | Reserved | | | | | I2C1RST | I2C0RST | Reserved | | | USART1RST | Reserve. |
| | | | rw | | | | | | rw | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SPI1RST | Reserved | | WWDGTRST | Reserved | | TIMER13RST | Reserved | | | TIMER5RST | Reserved | | TIMER2RST | Reserved |
| | rw | | | rw | | | rw | | | | rw | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value |
| 28 | PMURST | Power control reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset power control unit |

| | | |
|---|---|---|
| 27:23 | Reserved | Must be kept at reset value |
| 22 | I2C1RST | I2C1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset I2C1 |
| 21 | I2C0RST | I2C0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset I2C0 |
| 20:18 | Reserved | Must be kept at reset value |
| 17 | USART1RST | USART1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset USART1 |
| 16:15 | Reserved | Must be kept at reset value |
| 14 | SPI1RST | SPI1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset SPI1 |
| 13:12 | Reserved | Must be kept at reset value |
| 11 | WWDGTRST | Window watchdog timer reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset window watchdog timer |
| 10:9 | Reserved | Must be kept at reset value |
| 8 | TIMER13RST | TIMER13 timer reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER13 TIMER |
| 7:5 | Reserved | Must be kept at reset value |
| 4 | TIMER5RST | TIMER5 timer reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER5 TIMER |
| 3:2 | Reserved | Must be kept at reset value |
| 1 | TIMER2RST | TIMER2 timer reset |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER2 timer |
| 0 | Reserved | Must be kept at reset value |

### 4.3.6. AHB enable register (RCU_AHBEN)

Address offset: 0x14

Reset value: 0x0000 0014

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | PFEN | Reserved. | | PCEN | PBEN | PAEN | Reserved |
| | | | | | | | | | rw | | | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | CRCEN | Reserved | FMC SPEN | Reserved | SRAM SPEN | Reserved | DMAEN |
| | | | | | | | | | rw | | rw | | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22 | PFEN | GPIO port F clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port F clock<br>1: Enabled GPIO port F clock |
| 21:20 | Reserved | Must be kept at reset value |
| 19 | PCEN | GPIO port C clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port C clock<br>1: Enabled GPIO port C clock |
| 18 | PBEN | GPIO port B clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port B clock<br>1: Enabled GPIO port B clock |
| 17 | PAEN | GPIO port A clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port A clock<br>1: Enabled GPIO port A clock |
| 16:7 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 6 | CRCEN | CRC clock enable<br>This bit is set and reset by software.<br>0: Disabled CRC clock<br>1: Enabled CRC clock |
| 5 | Reserved | Must be kept at reset value |
| 4 | FMCSPEN | FMC clock enable<br>This bit is set and reset by software to enable/disable FMC clock during Sleep mode.<br>0: Disabled FMC clock during Sleep mode<br>1: Enabled FMC clock during Sleep mode |
| 3 | Reserved | Must be kept at reset value |
| 2 | SRAMSPEN | SRAM interface clock enable<br>This bit is set and reset by software to enable/disable SRAM interface clock during Sleep mode.<br>0: Disabled SRAM interface clock during Sleep mode.<br>1: Enabled SRAM interface clock during Sleep mode |
| 1 | Reserved | Must be kept at reset value |
| 0 | DMAEN | DMA clock enable<br>This bit is set and reset by software.<br>0: Disabled DMA clock<br>1: Enabled DMA clock |

### 4.3.7. APB2 enable register (RCU_APB2EN)

Address offset: 0x18
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | DBGMCUEN | Reserved | | | TIMER16EN | TIMER15EN | TIMER14EN |
| | | | | | | | | | rw | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USART0EN | Reserved | SPI0EN | TIMER0EN | Reserved | ADCEN | Reserved | | | | | | | | CFGCMPEN |
| | rw | | rw | rw | | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22 | DBGMCUEN | DBGMCU clock enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled DBGMCU clock<br>1: Enabled DBGMCU clock |
| 21:19 | Reserved | Must be kept at reset value |
| 18 | TIMER16EN | TIMER16 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER16 timer clock<br>1: Enabled TIMER16 timer clock |
| 17 | TIMER15EN | TIMER15 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER15 timer clock<br>1: Enabled TIMER15 timer clock |
| 16 | TIMER14EN | TIMER14 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER14 timer clock<br>1: Enabled TIMER14 timer clock |
| 15 | Reserved | Must be kept at reset value |
| 14 | USART0EN | USART0 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART0 clock<br>1: Enabled USART0 clock |
| 13 | Reserved | Must be kept at reset value |
| 12 | SPI0EN | SPI0 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI0 clock<br>1: Enabled SPI0 clock |
| 11 | TIMER0EN | TIMER0 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER0 timer clock<br>1: Enabled TIMER0 timer clock |
| 10 | Reserved | Must be kept at reset value |
| 9 | ADCEN | ADC interface clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC interface clock<br>1: Enabled ADC interface clock |
| 8:1 | Reserved | Must be kept at reset value |
| 0 | CFGCMPEN | System configuration and comparator clock enable |

This bit is set and reset by software.

0: Disabled System configuration and comparator clock

1: Enabled System configuration and comparator clock

## 4.3.8. APB1 enable register (RCU_APB1EN)

Address offset:0x1C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | PMUEN | Reserved | | | | | I2C1EN | I2C0EN | Reserved | | | USART1EN | Reserved |
| | | | rw | | | | | | rw | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SPI1EN | Reserved | | WWDGTEN | Reserved | | TIMER13EN | Reserved | | | TIMER5EN | Reserved | | TIMER2EN | Reserved |
| | rw | | | rw | | | rw | | | | rw | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value |
| 28 | PMUEN | Power interface clock enable<br>This bit is set and reset by software.<br>0: Disabled Power interface clock<br>1: Enabled Power interface clock |
| 27:23 | Reserved | Must be kept at reset value |
| 22 | I2C1EN | I2C1 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C1 clock<br>1: Enabled I2C1 clock |
| 21 | I2C0EN | I2C0 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C0 clock<br>1: Enabled I2C0 clock |
| 20:18 | Reserved | Must be kept at reset value |
| 17 | USART1EN | USART1 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART1 clock<br>1: Enabled USART1 clock |

| | | |
|---|---|---|
| 16:15 | Reserved | Must be kept at reset value |
| 14 | SPI1EN | SPI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI1 clock<br>1: Enabled SPI1 clock |
| 13:12 | Reserved | Must be kept at reset value |
| 11 | WWDGTEN | Window watchdog timer clock enable<br>This bit is set and reset by software.<br>0: Disabled Window watchdog timer clock<br>1: Enabled Window watchdog timer clock |
| 10:9 | Reserved | Must be kept at reset value |
| 8 | TIMER13EN | TIMER13 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER13 timer clock<br>1: Enabled TIMER13 timer clock |
| 7:5 | Reserved | Must be kept at reset value |
| 4 | TIMER5EN | TIMER5 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER5 timer clock<br>1: Enabled TIMER5 timer clock |
| 3:2 | Reserved | Must be kept at reset value |
| 1 | TIMER2EN | TIMER2 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER2 timer clock<br>1: Enabled TIMER2 timer clock |
| 0 | Reserved | Must be kept at reset value |

### 4.3.9. Backup domain control register (RCU_BDCTL)

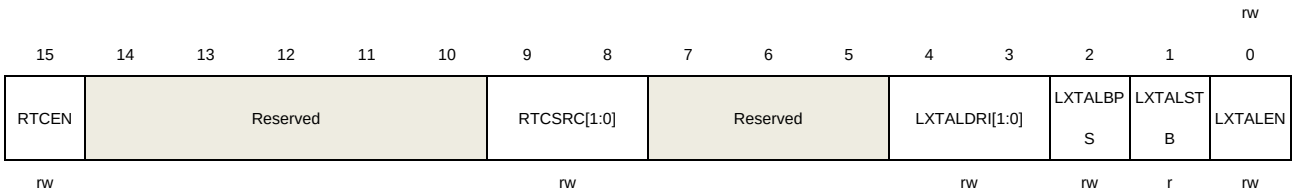
Address offset: 0x20

Reset value: 0x0000 0018, reset by backup domain reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

**Note:**The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the backup domain control register (BDCTL) are only reset after a backup domain reset. These bits can be modified only when the BKPWEN bit in the power control register (PMU_CTL) has to be set.

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



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTCEN | Reserved | | | | | RTCSRC[1:0] | | Reserved | | | LXTALDRI[1:0] | | LXTALBPS | LXTALSTB | LXTALEN |
| rw | | | | | | rw | | | | | rw | | rw | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |
| 16 | BKPRST | Backup domain reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Resets backup domain |
| 15 | RTCEN | RTC clock enable<br>This bit is set and reset by software.<br>0: Disabled RTC clock<br>1: Enabled RTC clock |
| 14:10 | Reserved | Must be kept at reset value |
| 9:8 | RTCSRC[1:0] | RTC clock entry selection<br>Set and reset by software to control the RTC clock source.<br>00: No clock selected<br>01: CK_LXTAL selected as RTC source clock<br>10: CK_IRC40K selected as RTC source clock<br>11: (CK_HXTAL / 32) selected as RTC source clock |
| 7:5 | Reserved | Must be kept at reset value |
| 4:3 | LXTALDRI[1:0] | LXTAL drive capability<br>Set and reset by software. Backup domain reset reset this value.<br>00: lower driving capability<br>01: medium low driving capability<br>10: medium high driving capability<br>11: higher driving capability (reset value)<br>**Note:** The LXTALDRI is not in bypass mode. |
| 2 | LXTALBPS | LXTAL bypass mode enable<br>Set and reset by software.<br>0: Disable the LXTAL Bypass mode<br>1: Enable the LXTAL Bypass mode |
| 1 | LXTALSTB | External low-speed oscillator stabilization<br>Set by hardware to indicate if the LXTAL output clock is stable and ready for use.<br>0: LXTAL is not stable |

| | | |
|---|---|---|
| | | 1: LXTAL is stable |
| 0 | LXTALEN | LXTAL enable<br>Set and reset by software.<br>0: Disable LXTAL<br>1: Enable LXTAL |

## 4.3.10. Reset source /clock register (RCU_RSTSCK)

Address offset: 0x24

Reset value: 0x0C00 0000, reset flags reset by power reset only, other reset by system reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LP RSTF | WWDGT RSTF | FWDGT RSTF | SW RSTF | POR RSTF | EP RSTF | OBL RSTF | RSTFC | V12 RSTF | Reserved | | | | | | |
| r | r | r | r | r | r | r | rw | r | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | IRC40K STB | IRC40K EN |
| | | | | | | | | | | | | | | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LPRSTF | Low-power reset flag<br>Set by hardware when Deep-sleep /standby reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No Low-power management reset generated<br>1: Low-power management reset generated |
| 30 | WWDGTRSTF | Window watchdog timer reset flag<br>Set by hardware when a window watchdog timer reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No window watchdog reset generated<br>1: Window watchdog reset generated |
| 29 | FWDGTRSTF | Free Watchdog timer reset flag<br>Set by hardware when a Free Watchdog timer generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No Free Watchdog timer reset generated<br>1: Free Watchdog timer reset generated |
| 28 | SWRSTF | Software reset flag<br>Set by hardware when a software reset generated.<br>Reset by writing 1 to the RSTFC bit. |

| | | |
|---|---|---|
| | | 0: No software reset generated<br>1: Software reset generated |
| 27 | PORRSTF | Power reset flag<br>Set by hardware when a power reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No power reset generated<br>1: Power reset generated |
| 26 | EPRSTF | External pin reset flag<br>Set by hardware when an external pin generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No external pin reset generated<br>1: External pin reset generated |
| 25 | OBLRSTF | Option byte loader reset flag<br>Set by hardware when an option byte loader generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No option byte loader reset generated<br>1: Option byte loader reset generated |
| 24 | RSTFC | Reset flag clear<br>This bit is set by software to clear all reset flags.<br>0: Not clear reset flags<br>1: Clear reset flags |
| 23 | V12RSTF | V12 domain Power reset flag<br>Set by hardware when a V12 domain power reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No V12 domain power reset generated<br>1: V12 domain power reset generated |
| 22:2 | Reserved | Must be kept at reset value |
| 1 | IRC40KSTB | IRC40K stabilization<br>Set by hardware to indicate if the IRC40K output clock is stable and ready for use.<br>0: IRC40K is not stable<br>1: IRC40K is stable |
| 0 | IRC40KEN | IRC40K enable<br>Set and reset by software.<br>0: Disable IRC40K<br>1: Enable IRC40K |

### 4.3.11. AHB reset register (RCU_AHBRST)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | PFRST | Reserved | | PCRST | PBRST | PARST | Reserved |
| | | | | | | | | | rw | | | rw | rw | rw | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22 | PFRST | GPIO port F reset<br>This bit is set and reset by software.<br>0: No reset GPIO port F<br>1: Reset GPIO port F |
| 21:20 | Reserved | Must be kept at reset value |
| 19 | PCRST | GPIO port C reset<br>This bit is set and reset by software.<br>0: No reset GPIO port C<br>1: Reset GPIO port C |
| 18 | PBRST | GPIO port B reset<br>This bit is set and reset by software.<br>0: No reset GPIO port B<br>1: Reset GPIO port B |
| 17 | PARST | GPIO port A reset<br>This bit is set and reset by software.<br>0: No reset GPIO port A<br>1: Reset GPIO port A |
| 16:0 | Reserved | Must be kept at reset value |

### 4.3.12. Configuration register 1 (RCU_CFG1)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 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 | PREDV[3:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:0 | PREDV[3:0] | CK_HXTAL divider previous PLL<br>This bit is set and reset by software. These bits can be written when PLL is disable<br>**Note:** The bit 0 of PREDV is same as bit 17 of RCU_CFG0, so modifying bit 17 of RCU_CFG0 also modifies bit 0 of RCU_CFG1.<br>The CK_HXTAL is divided by (PREDV + 1).<br>0000: input to PLL not divided<br>0001: input to PLL divided by 2<br>0010: input to PLL divided by 3<br>0011: input to PLL divided by 4<br>0100: input to PLL divided by 5<br>0101: input to PLL divided by 6<br>0110: input to PLL divided by 7<br>0111: input to PLL divided by 8<br>1000: input to PLL divided by 9<br>1001: input to PLL divided by 10<br>1010: input to PLL divided by 11<br>1011: input to PLL divided by 12<br>1100: input to PLL divided by 13<br>1101: input to PLL divided by 14<br>1110: input to PLL divided by 15<br>1111: input to PLL divided by 16 |

## 4.3.13. Configuration register 2 (RCU_CFG2)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCPSC[2] | | | | | | | Reserved | | | | | | | | IRC28MDIV |
| rw | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | ADCSEL | Reserved | | | | | | USART0SEL[1:0] | |
| | | | | | | | rw | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | ADCPSC[2] | Bit 2 of ADCPSC<br>see bits 15:14 of RCU_CFG0 |
| 30:17 | Reserved | Must be kept at reset value |
| 16 | IRC28MDIV | IRC28M divider or not<br>0: IRC28M /2 used as ADC clock<br>1: IRC28M used as ADC clock |
| 15:9 | Reserved | Must be kept at reset value |
| 8 | ADCSEL | CK_ADC clock source selection<br>This bit is set and reset by software.<br>0: CK_ADC select CK_IRC28M<br>1: CK_ADC select CK_APB2 which is divided by 2,4,6,8 or. CK_AHB which is divided by 3,5,7,9 |
| 7:2 | Reserved | Must be kept at reset value |
| 1:0 | USART0SEL[1:0] | CK_USART0 clock source selection<br>This bit is set and reset by software.<br>00: CK_USART0 select CK_APB2<br>01: CK_USART0 select CK_SYS<br>10: CK_USART0 select CK_LXTAL<br>11: CK_USART0 select CK_IRC8M |

### 4.3.14. Control register 1 (RCU_CTL1)

Address offset: 0x34

Reset value: 0x0000 XX80 where X is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC28MCALIB[7:0] | | | | | | | | IRC28MADJ[4:0] | | | | | Reserved | IRC28MSTB | IRC28MEN |
| r | | | | | | | | rw | | | | | | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | IRC28MCALIB[7:0] | Internal 28M RC oscillator calibration value register<br>These bits are load automatically at power on. |
| 7:3 | IRC28MADJ[4:0] | Internal 28M RC oscillator clock trim adjust value<br>These bits are set by software. The trimming value is there bits (IRC28MADJ) |

| | | |
|---|---|---|
| | | added to the IRC28MCALIB[7:0] bits. The trimming value should trim the IRC28M to 28MHz ± 1%. |
| 2 | Reserved | Must be kept at reset value |
| 1 | IRC28MSTB | IRC28M Internal 28M RC oscillator stabilization Flag<br>Set by hardware to indicate if the IRC28M oscillator is stable and ready for use.<br>0: IRC28M oscillator is not stable<br>1: IRC28M oscillator is stable |
| 0 | IRC28MEN | IRC28M Internal 28M RC oscillator enable<br>Set and reset by software.<br>0: Internal 28 MHz RC oscillator disabled<br>1: Internal 28 MHz RC oscillator enabled |

### 4.3.15. Voltage key register (RCU_VKEY)

Address offset: 0x100

Reset value: 0x0000 0000.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | KEY[31:0] | The key of RCU_DSV register<br>These bits are written only by software and read as 0. Only after write 0x1A2B3C4D to the RCU_VKEY, the RCU_DSV register can be written. |

### 4.3.16. Deep-sleep mode voltage register (RCU_DSV)

Offset: 0x134

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 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 | | | | | | | | | | | | | | DSLPVS[1:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1:0 | DSLPVS[1:0] | Deep-sleep mode voltage select<br>These bits is set and reset by software<br>00 : The core voltage is 1.0V in Deep-sleep mode<br>01 : The core voltage is 0.9V in Deep-sleep mode<br>10 : The core voltage is 0.8V in Deep-sleep mode<br>11 : The core voltage is 1.2V in Deep-sleep mode |

# 5. Interrupt / event controller (EXTI)

## 5.1. Overview

Cortex®-M23 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and power management controls. It's tightly coupled to the processer core. You can read the Technical Reference Manual of Cortex®-M23 for more details about NVIC.

EXTI (interrupt/event controller) contains up to 21 independent edge detectors and generates interrupt requests or events to the processer. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

## 5.2. Characteristics

- Cortex®-M23 system exception
- Up to 28 maskable peripheral interrupts for GD32E23x series
- 2 bits interrupt priority configuration - 4 priority levels
- Efficient interrupt processing
- Support exception pre-emption and tail-chaining
- Wake up system from power saving mode
- Up to 21 independent edge detectors in EXTI
- Three trigger types: rising, falling and both edges
- Software interrupt or event trigger
- Trigger sources configurable

## 5.3. Interrupts function overview

The Arm Cortex®-M23 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).

The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. The following tables list all exception

types.

**Table 5-1. NVIC exception types in Cortex®-M23**

| Exception type | Vector number | Priority (a) | Vector address | Description |
|---|---|---|---|---|
| - | 0 | - | 0x0000_0000 | Reserved |
| Reset | 1 | -3 | 0x0000_0004 | Reset |
| NMI | 2 | -2 | 0x0000_0008 | Non maskable interrupt |
| HardFault | 3 | -1 | 0x0000_000C | All class of fault |
| - | 4-10 | - | 0x0000_0010 -0x0000_002B | Reserved |
| SVCall | 11 | Programmable | 0x0000_002C | System service call via SWI instruction |
| - | 12-13 | - | 0x0000_0030 -0x0000_0034 | Reserved |
| PendSV | 14 | Programmable | 0x0000_0038 | Pendable request for system service |
| SysTick | 15 | Programmable | 0x0000_003C | System tick timer |

The SysTick calibration value is 9000 and SysTick clock frequency is fixed to HCLK*0.125. So this will give a 1ms SysTick interrupt if HCLK is configured to 72MHz.

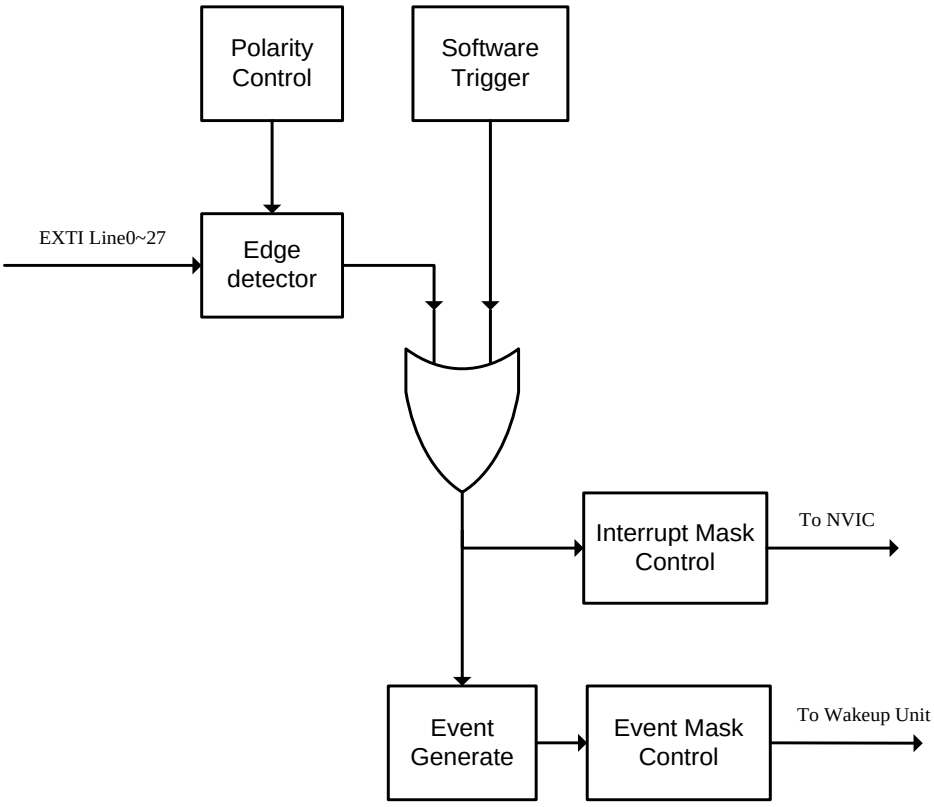
**Table 5-2. Interrupt vector table**

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| IRQ 0 | 16 | WWDGT interrupt | 0x0000_0040 |
| IRQ 1 | 17 | LVD from EXTI interrupt | 0x0000_0044 |
| IRQ 2 | 18 | RTC global interrupt | 0x0000_0048 |
| IRQ 3 | 19 | FMC global interrupt | 0x0000_004C |
| IRQ 4 | 20 | RCU global interrupt | 0x0000_0050 |
| IRQ 5 | 21 | EXTI line0-1 interrupt | 0x0000_0054 |
| IRQ 6 | 22 | EXTI line2-3 interrupt | 0x0000_0058 |
| IRQ 7 | 23 | EXTI line4-15 interrupt | 0x0000_005C |
| IRQ 8 | 24 | Reserved | 0x0000_0060 |
| IRQ 9 | 25 | DMA channel0 global interrupt | 0x0000_0064 |
| IRQ 10 | 26 | DMA channel1-2 global interrupt | 0x0000_0068 |
| IRQ 11 | 27 | DMA channel3-4 global interrupt | 0x0000_006C |
| IRQ 12 | 28 | ADC and CMP interrupt | 0x0000_0070 |
| IRQ 13 | 29 | TIMER0 break, update, trigger and commutation interrupts | 0x0000_0074 |
| IRQ 14 | 30 | TIMER0 capture compare interrupt | 0x0000_0078 |
| IRQ 15 | 31 | Reserved | 0x0000_007C |
| IRQ 16 | 32 | TIMER2 global interrupt | 0x0000_0080 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 17** | 33 | TIMER5 interrupt | 0x0000_0084 |
| **IRQ 18** | 34 | Reserved | 0x0000_0088 |
| **IRQ 19** | 35 | TIMER13 global interrupt | 0x0000_008C |
| **IRQ 20** | 36 | TIMER14 global interrupt | 0x0000_0090 |
| **IRQ 21** | 37 | TIMER15 global interrupt | 0x0000_0094 |
| **IRQ 22** | 38 | TIMER16 global interrupt | 0x0000_0098 |
| **IRQ 23** | 39 | I2C0 event interrupt | 0x0000_009C |
| **IRQ 24** | 40 | I2C1 event interrupt | 0x0000_00A0 |
| **IRQ 25** | 41 | SPI0 global interrupt | 0x0000_00A4 |
| **IRQ 26** | 42 | SPI1 global interrupt | 0x0000_00A8 |
| **IRQ 27** | 43 | USART0 global interrupt | 0x0000_00AC |
| **IRQ 28** | 44 | USART1 global interrupt | 0x0000_00B0 |
| **IRQ 29** | 45 | Reserved | 0x0000_00B4 |
| **IRQ 30** | 46 | Reserved | 0x0000_00B8 |
| **IRQ 31** | 47 | Reserved | 0x0000_00BC |
| **IRQ 32** | 48 | I2C0 error interrupt | 0x0000_00C0 |
| **IRQ 33** | 49 | Reserved | 0x0000_00C4 |
| **IRQ 34** | 50 | I2C1 error interrupt | 0x0000_00C8 |
| **IRQ 35** | 51 | Reserved | 0x0000_00CC |
| **IRQ 36** | 52 | Reserved | 0x0000_00D0 |
| **IRQ 37** | 53 | Reserved | 0x0000_00D4 |
| **IRQ 38** | 54 | Reserved | 0x0000_00D8 |
| **IRQ 39-41** | 55-57 | Reserved | 0x0000_00DC-0x0000_00E4 |
| **IRQ 42** | 58 | Reserved | 0x0000_00E8 |
| **IRQ 43-47** | 59-63 | Reserved | 0x0000_00EC-0x0000_00FC |
| **IRQ 48** | 64 | Reserved | 0x0000_0100 |
| **IRQ 49-50** | 65-66 | Reserved | 0x0000_0104-0x0000_0108 |
| **IRQ 51** | 67 | Reserved | 0x0000_010C |
| **IRQ52-66** | 68-82 | Reserved | 0x0000_0110-0x0000_0148 |
| **IRQ67** | 83 | Reserved | 0x0000_014C |

## 5.4. External interrupt and event block diagram

**Figure 5-1. Block diagram of EXTI**




## 5.5. External interrupt and event function overview

The EXTI contains up to 21 independent edge detectors and generates interrupts request or event to the processer. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

The EXTI trigger source includes 16 external lines from GPIO pins and 5 lines from internal modules which refer to ***Table 5-3. EXTI source*** for GD32E23x series. All GPIO pins can be selected as an EXTI trigger source by configuring SYSCFG_EXTISSx registers in SYSCFG module (please refer to ***System configuration registers (SYSCFG)*** section for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex®-M23 processor fully implements the Wait For Interrupt (WFI), Wait For Event (WFE) and the Send Event (SEV) instructions. The Wake-up Interrupt Controller (WIC) enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and event. EXTI can be used to wake up processor and the whole system when some expected event occurs, such as a special GPIO pin toggling or RTC alarm.

**Table 5-3. EXTI source**

| EXTI line number | Source |
|---|---|
| 0 | PA0 / PB0 / PF0 |
| 1 | PA1 / PB1 / PF1 |
| 2 | PA2 / PB2 |
| 3 | PA3 / PB3 |
| 4 | PA4 / PB4 |
| 5 | PA5 / PB5 |
| 6 | PA6 / PB6 / PF6 |
| 7 | PA7 / PB7 / PF7 |
| 8 | PA8 / PB8 |
| 9 | PA9 / PB9 |
| 10 | PA10 / PB10 |
| 11 | PA11 / PB11 |
| 12 | PA12 / PB12 |
| 13 | PA13 / PB13 / PC13 |
| 14 | PA14 / PB14 / PC14 |
| 15 | PA15 / PB15 / PC15 |
| 16 | LVD |
| 17 | RTC alarm |
| 18 | Reserved |
| 19 | RTC tamper and timestamp |
| 20 | Reserved |
| 21 | CMP output |
| 22 | Reserved |
| 23 | Reserved |
| 24 | Reserved |
| 25 | USART0 wakeup |
| 26 | Reserved |
| 27 | Reserved |

# 5.6. Register definition

EXTI base address: 0x4001 0400

## 5.6.1. Interrupt enable register (EXTI_INTEN)

Address offset: 0x00

Reset value: 0x0F94 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | INTEN27 | INTEN26 | INTEN25 | INTEN24 | INTEN23 | INTEN22 | INTEN21 | INTEN20 | INTEN19 | INTEN18 | INTEN17 | INTEN16 |
| | | | | 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 |
| INTEN15 | INTEN14 | INTEN13 | INTEN12 | INTEN11 | INTEN10 | INTEN9 | INTEN8 | INTEN7 | INTEN6 | INTEN5 | INTEN4 | INTEN3 | INTEN2 | INTEN1 | INTEN0 |
| rw | rw | rw | rw | rw | rw | rw | Rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27: 0 | INTENx | Interrupt enable bit x(x=0..27)<br>0: Interrupt from linex is disabled<br>1: Interrupt from linex is enabled |

## 5.6.2. Event enable register (EXTI_EVEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | EVEN27 | EVEN26 | EVEN25 | EVEN24 | EVEN23 | EVEN22 | EVEN21 | EVEN20 | EVEN19 | EVEN18 | EVEN17 | EVEN16 |
| | | | | 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 |
| EVEN15 | EVEN14 | EVEN13 | EVEN12 | EVEN11 | EVEN10 | EVEN9 | EVEN8 | EVEN7 | EVEN6 | EVEN5 | EVEN4 | EVEN3 | EVEN2 | EVEN1 | EVEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27: 0 | EVENx | Event enable bit x(x=0..27)<br>0: Event from linex is disabled<br>1: Event from linex is enabled |

### 5.6.3. Rising edge trigger enable register (EXTI_RTEN)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | RTEN21 | Reserved | RTEN19 | Reserved | RTEN17 | RTEN16 |
| | | | | | | | | | | rw | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTEN15 | RTEN14 | RTEN13 | RTEN12 | RTEN11 | RTEN10 | RTEN9 | RTEN8 | RTEN7 | RTEN6 | RTEN5 | RTEN4 | RTEN3 | RTEN2 | RTEN1 | RTEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value |
| 21 | RTENx | Rising edge trigger enable (x=21)<br>0: Rising edge of linex is invalid<br>1: Rising edge of linex is valid as an interrupt / event request |
| 20 | Reserved | Must be kept at reset value |
| 19 | RTENx | Rising edge trigger enable (x=19)<br>0: Rising edge of linex is invalid<br>1: Rising edge of linex is valid as an interrupt / event request |
| 18 | Reserved | Must be kept at reset value |
| 17:0 | RTENx | Rising edge trigger enable (x=0..17)<br>0: Rising edge of linex is invalid<br>1: Rising edge of linex is valid as an interrupt / event request |

### 5.6.4. Falling edge trigger enable register (EXTI_FTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | FTEN21 | Reserved | FTEN19 | Reserved | FTEN17 | FTEN16 |
| | | | | | | | | | | rw | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FTEN15 | FTEN14 | FTEN13 | FTEN12 | FTEN11 | FTEN10 | FTEN9 | FTEN8 | FTEN7 | FTEN6 | FTEN5 | FTEN4 | FTEN3 | FTEN2 | FTEN1 | FTEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31: 22 | Reserved | Must be kept at reset value |
| 21 | FTENx | Falling edge trigger enable (x=21)<br>0: Falling edge of linex is invalid<br>1: Falling edge of linex is valid as an interrupt / event request |
| 20 | Reserved | Must be kept at reset value |
| 19 | FTENx | Falling edge trigger enable (x=19)<br>0: Falling edge of linex is invalid<br>1: Falling edge of linex is valid as an interrupt / event request |
| 18 | Reserved | Must be kept at reset value |
| 17: 0 | FTENx | Falling edge trigger enable (x=0..17)<br>0: Falling edge of linex is invalid<br>1: Falling edge of linex is valid as an interrupt / event request |

### 5.6.5. Software interrupt event register (EXTI_SWIEV)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | SWIEV21 | Reserved | SWIEV19 | Reserved | SWIEV17 | SWIEV16 |
| | | | | | | | | | | rw | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWIEV15 | SWIEV14 | SWIEV13 | SWIEV12 | SWIEV11 | SWIEV10 | SWIEV9 | SWIEV8 | SWIEV7 | SWIEV6 | SWIEV5 | SWIEV4 | SWIEV3 | SWIEV2 | SWIEV1 | SWIEV0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value |
| 21 | SWIEVx | Interrupt / event software trigger (x=21)<br>0: Deactivate the EXTIx software interrupt / event request<br>1: Activate the EXTIx software interrupt / event request |
| 20 | Reserved | Must be kept at reset value |
| 19 | SWIEVx | Interrupt / event software trigger (x=19)<br>0: Deactivate the EXTIx software interrupt / event request<br>1: Activate the EXTIx software interrupt / event request |
| 18 | Reserved | Must be kept at reset value |
| 17: 0 | SWIEVx | Interrupt / event software trigger (x=0..17) |

0: Deactivate the EXTIx software interrupt / event request

1: Activate the EXTIx software interrupt / event request

### 5.6.6. Pending register (EXTI_PD)

Address offset: 0x14

Reset value: undefined

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | PD21 | Reserved | PD19 | Reserved | PD17 | PD16 |
| | | | | | | | | | | rc_w1 | | rc_w1 | | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PD15 | PD14 | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31: 22 | Reserved | Must be kept at reset value |
| 21 | PDx | Interrupt pending status (x=21)<br>0: EXTI linex is not triggered<br>1: EXTI linex is triggered<br>This bit is cleared to 0 by writing 1 to it. |
| 20 | Reserved | Must be kept at reset value |
| 19 | PDx | Interrupt pending status (x=19)<br>0: EXTI Linex is not triggered<br>1: EXTI Linex is triggered<br>This bit is cleared to 0 by writing 1 to it. |
| 18 | Reserved | Must be kept at reset value |
| 17: 0 | PDx | Interrupt pending status (x=0..17)<br>0: EXTI Linex is not triggered<br>1: EXTI Linex is triggered<br>This bit is cleared to 0 by writing 1 to it. |

# 6. General-purpose and alternate-function I/Os (GPIO and AFIO)

## 6.1. Overview

There are up to 39 general purpose I/O pins, (GPIO), named PA0 ~ PA15 and PB0 ~ PB15, PC13 ~ PC15, PF0 ~ PF1, PF6 ~ PF7 for the GD32E23x device to implement logic input/output functions. Each GPIO port has related control and configuration registers to satisfy the requirements of specific applications.

The GPIO ports are pin-shared with other alternative functions (AFs) to obtain maximum flexibility on the package pins. The GPIO pins can be used as alternative functional pins by configuring the corresponding registers such as the AF input or output pins.

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), input, peripheral alternate function or analog mode. Each GPIO pin can be configured as pull-up, pull-down or floating. All GPIOs are high-current capable except for analog mode.

## 6.2. Characteristics

- Input/output direction control
- Schmitt trigger input function enable control
- Each pin weak pull-up/pull-down function
- Output push-pull/open-drain enable control
- Output set/reset control
- Output drive speed selection
- Analog input/output configuration
- Alternate function input/output configuration
- Port configuration lock
- Single cycle toggle output capability

## 6.3. Function overview

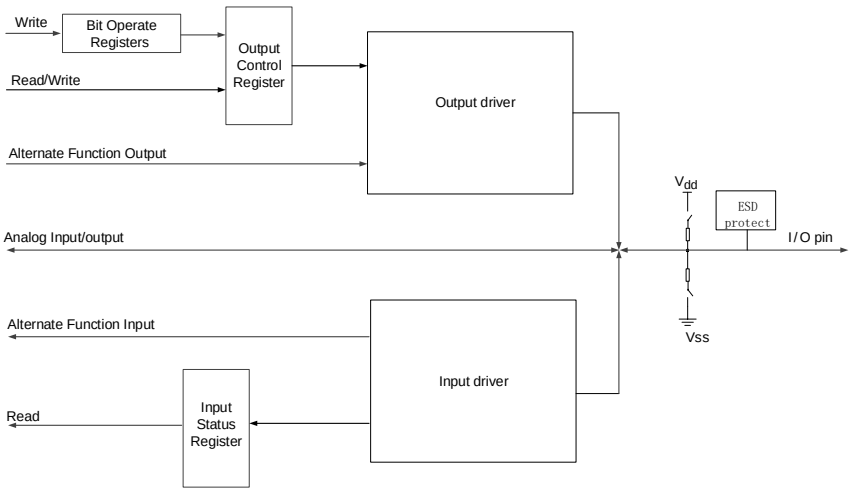
Each of the general-purpose I/O ports can be configured as GPIO inputs, GPIO outputs, AF function or analog mode by GPIO 32-bit control register (GPIOx_CTL). AFIO input or output direction is decided by AFIO function after AFIO enable. When the port is output (GPIO output or AFIO output), it can be configured as push-pull or open drain mode by GPIO output mode registers (GPIOx_OMODE). And the port max speed can be configured by GPIO output speed registers (GPIOx_OSPD). Each port can be configured as floating (no pull-up and pull-down), pull-up or pull-down function by GPIO pull-up/pull-down registers (GPIOx_PUD).

**Table 6-1. GPIO configuration table**

| PAD TYPE | | | CTLn | OMn | PUDn |
|---|---|---|---|---|---|
| GPIO INPUT | X | Floating | 00 | X | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| GPIO OUTPUT | Push-pull | Floating | 01 | 0 | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| | Open-drain | Floating | | 1 | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| AFIO INPUT | X | Floating | 10 | X | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| AFIO OUTPUT | Push-pull | Floating | 10 | 0 | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| | Open-drain | Floating | | 1 | 00 |
| | | Pull-up | | | 01 |
| | | Pull-down | | | 10 |
| ANALOG | X | X | 11 | X | XX |

*Figure 6-1. Basic structure of of a general-pupose I/O* shows the basic structure of an I/O Port bit.

**Figure 6-1. Basic structure of of a general-pupose I/O**




## 6.3.1. GPIO pin configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured as the input floating mode that input disabled without pull-up(PU)/pull-down(PD) resistors. But the Serial-Wired Debug pins are configured as AF

PU/PD mode after reset:

PA14: SWCLK in AF pull-down mode

PA13: SWDIO in AF pull-up mode

The GPIO pins can be configured as inputs or outputs. When the GPIO pins are configured as input pins, all GPIO pins have an internal weak pull-up and weak pull-down which can be chosen. When the GPIO pins are configured as input pins, the data on the external pads can be captured at every AHB clock cycle to the port input status register (GPIOx_ISTAT).

When the GPIO pins are configured as output pins, the user can configure the speed of the ports and chooses the output driver mode: push-pull or open-drain mode. The value of the port output control register (GPIOx_OCTL) is output on the I/O pin.

There is no need to read-then-write when programming the GPIOx_OCTL at the bit level, the user can modify only one bit or several bits in a single atomic AHB write access by programming '1' to the bit operate register (GPIOx_BOP, or for clearing only GPIOx_BC, or for toggle only GPIOx_TG). The other bits will not be affected.

### 6.3.2. Alternate functions (AF)

When the port is configured as AFIO (set CTLy bits to "0b10", which is in GPIOx_CTL registers), the port is used as peripheral alternate functions. Each port has sixteen alternate functions can be configured by GPIO alternate functions select registers (GPIOx_AFSELy(y=0,1)). The detail alternate function assignments for each port are described in the device datasheet.

### 6.3.3. Additional functions

Some pins have additional functions, which have priority over the configuration in the standard GPIO registers. When for ADC additional functions, the port must be configured as analog mode. When for RTC, WKUPx and oscillators additional functions, the port type is set automatically by related RTC, PMU and RCU registers. These ports can be used as normal GPIO when the additional functions disabled.

### 6.3.4. Input configuration

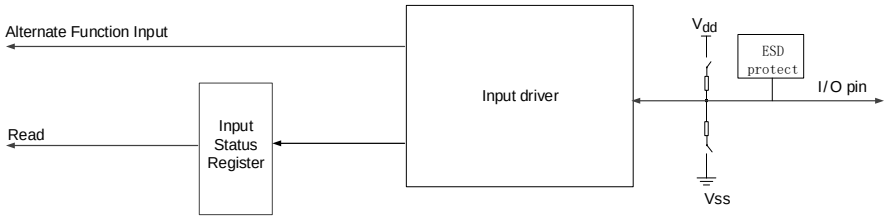
When GPIO pin is configured as input:

- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- Every AHB clock cycle the data present on the I/O pad is got to the port input status register.
- Disable the output buffer.

***Figure 6-2. Basic structure of Input configuration*** shows the input configuration of the

GPIO pin.

**Figure 6-2. Basic structure of Input configuration**



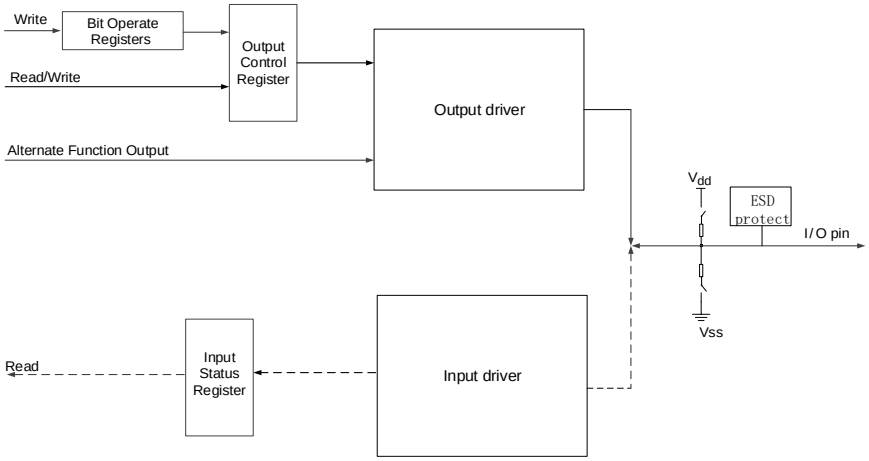

## 6.3.5. Output configuration

When GPIO pin is configured as output:

- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- The output buffer is enabled.
- Open-Drain mode: The pad outputs “0” when a “0” in the output control register; while the pad leaves Hi-Z when a “1” in the output control register.
- Push-Pull mode: The pad outputs “0” when a “0” in the output control register; while the pad outputs “1” when a “1” in the output control register.
- A read access to the port output control register gets the last written value.
- A read access to the port input status register gets the I/O state.

***Figure 6-3. Basic structure of Output configuration*** shows the output configuration of the GPIO pin.

**Figure 6-3. Basic structure of Output configuration**




## 6.3.6. Analog configuration

When GPIO pin is used as analog configuration:

- The weak pull-up and pull-down resistors are disabled.
- The output buffer is disabled.

- The schmitt trigger input is de-activated.
- Read access to the port input status register gets the value “0”.

*Figure 6-4. Basic structure of Analog configuration* shows the analog configuration of the GPIO pin.

**Figure 6-4. Basic structure of Analog configuration**




### 6.3.7. Alternate function (AF) configuration

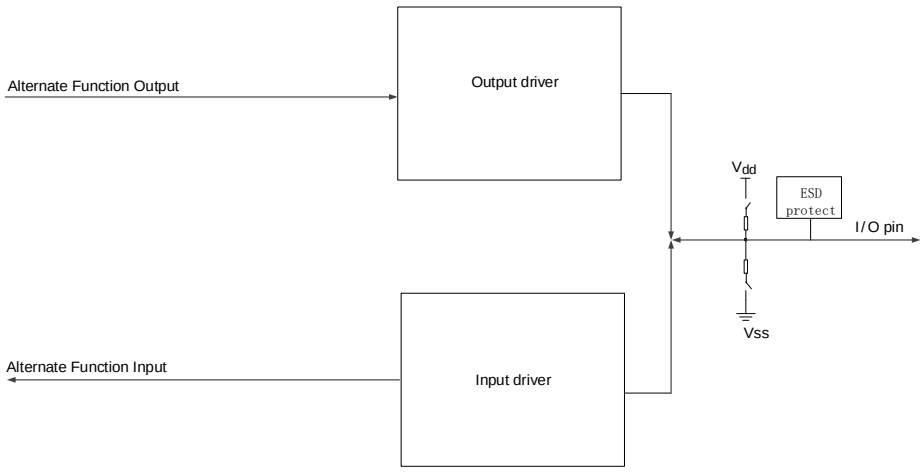
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

When be configured as alternate function:

- The output buffer is enabled in open-drain or push-pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- The data present on the I/O pin is sampled into the port input status register every AHB clock cycle.
- A read access to the port input status register gets the I/O state in Open-Drain mode.
- A read access to the port output control register gets the last written value in Push-Pull mode.

*Figure 6-5. Basic structure of Alternate function configuration* shows the alternate function configuration of the GPIO pin.

**Figure 6-5. Basic structure of Alternate function configuration**



### 6.3.8. GPIO locking function

The locking mechanism allows the IO configuration to be protected.

The protected registers are GPIOx_CTL, GPIOx_OMODE, GPIOx_OSPD, GPIOx_PUD, GPIOx_AFSELy(y=0,1). It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx_LOCK). When the special LOCK sequence has been applied on a port bit, it is no longer able to modify the value of the port bit until the next reset. It should be recommended to be used in the configuration of driving a power module.

### 6.3.9. GPIO single cycle toggle function

GPIO could toggle the I/O output level in single AHB cycle by writing 1 to the corresponding bit of GPIOx_TG register. The output signal frequency could up to the half of the AHB clock.

## 6.4. Register definition

GPIOA base address: 0x4800 0000
GPIOB base address: 0x4800 0400
GPIOC base address: 0x4800 0800
GPIOF base address: 0x4800 1400

### 6.4.1. Port control register (GPIOx_CTL, x=A..C,F)

Address offset: 0x00
Reset value: 0x2800 0000 for port A; 0x0000 0000 for others.

This register has to be accessed by word (32-bit)

| 31 30 | 29 28 | 27 26 | 25 24 | 23 22 | 21 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|
| CTL15[1:0] | CTL14[1:0] | CTL13[1:0] | CTL12[1:0] | CTL11[1:0] | CTL10[1:0] | CTL9[1:0] | CTL8[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|
| CTL7[1:0] | CTL6[1:0] | CTL5[1:0] | CTL4[1:0] | CTL3[1:0] | CTL2[1:0] | CTL1[1:0] | CTL0[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | CTL15[1:0] | Pin 15 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 29:28 | CTL14[1:0] | Pin 14 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 27:26 | CTL13[1:0] | Pin 13 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 25:24 | CTL12[1:0] | Pin 12 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 23:22 | CTL11[1:0] | Pin 11 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 21:20 | CTL10[1:0] | Pin 10 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 19:18 | CTL9[1:0] | Pin 9 configuration bits |

| | | |
|---|---|---|
| | | These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 17:16 | CTL8[1:0] | Pin 8 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 15:14 | CTL7[1:0] | Pin 7 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 13:12 | CTL6[1:0] | Pin 6 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 11:10 | CTL5[1:0] | Pin 5 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 9:8 | CTL4[1:0] | Pin 4 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 7:6 | CTL3[1:0] | Pin 3 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 5:4 | CTL2[1:0] | Pin 2 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 3:2 | CTL1[1:0] | Pin 1 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 1:0 | CTL0[1:0] | Pin 0 configuration bits<br>These bits are set and cleared by software.<br>00: Input mode (reset value)<br>01: GPIO output mode<br>10: Alternate function mode<br>11: Analog mode |

### 6.4.2. Port output mode register (GPIOx_OMODE, x=A..C,F)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OM15 | OM14 | OM13 | OM12 | OM11 | OM10 | OM9 | OM8 | OM7 | OM6 | OM5 | OM4 | OM3 | OM2 | OM1 | OM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | OM15 | Pin 15 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 14 | OM14 | Pin 14 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 13 | OM13 | Pin 13 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 12 | OM12 | Pin 12 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 11 | OM11 | Pin 11 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 10 | OM10 | Pin 10 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 9 | OM9 | Pin 9 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 8 | OM8 | Pin 8 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 7 | OM7 | Pin 7 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 6 | OM6 | Pin 6 output mode bit<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to OM0 description |
| 5 | OM5 | Pin 5 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 4 | OM4 | Pin 4 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 3 | OM3 | Pin 3 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 2 | OM2 | Pin 2 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 1 | OM1 | Pin 1 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 0 | OM0 | Pin 0 output mode bit<br>These bits are set and cleared by software.<br>0: Output push-pull mode (reset value)<br>1: Output open-drain mode |

### 6.4.3. Port output speed register (GPIOx_OSPD, x=A..C,F)

Address offset: 0x08

Reset value: 0x0C00 0000 for port A; 0x0000 0000 for others.

This register has to be accessed by word (32-bit)

| 31 30 | 29 28 | 27 26 | 25 24 | 23 22 | 21 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|
| OSPD15[1:0] | OSPD14[1:0] | OSPD13[1:0] | OSPD12[1:0] | OSPD11[1:0] | OSPD10[1:0] | OSPD9[1:0] | OSPD8[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|
| OSPD7[1:0] | OSPD6[1:0] | OSPD5[1:0] | OSPD4[1:0] | OSPD3[1:0] | OSPD2[1:0] | OSPD1[1:0] | OSPD0[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | OSPD15[1:0] | Pin 15 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 29:28 | OSPD14[1:0] | Pin 14 output max speed bits<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to OSPD0[1:0] description |
| 27:26 | OSPD13[1:0] | Pin 13 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 25:24 | OSPD12[1:0] | Pin 12 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 23:22 | OSPD11[1:0] | Pin 11 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 21:20 | OSPD10[1:0] | Pin 10 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 19:18 | OSPD9[1:0] | Pin 9 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 17:16 | OSPD8[1:0] | Pin 8 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 15:14 | OSPD7[1:0] | Pin 7 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 13:12 | OSPD6[1:0] | Pin 6 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 11:10 | OSPD5[1:0] | Pin 5 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 9:8 | OSPD4[1:0] | Pin 4 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 7:6 | OSPD3[1:0] | Pin 3 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 5:4 | OSPD2[1:0] | Pin 2 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |

| | | |
|---|---|---|
| 3:2 | OSPD1[1:0] | Pin 1 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 1:0 | OSPD0[1:0] | Pin 0 output max speed bits<br>These bits are set and cleared by software.<br>x0: Output max speed 2M (reset value)<br>01: Output max speed 10M<br>11: Output max speed 50M |

### 6.4.4. Port pull-up/down register (GPIOx_PUD, x=A..C,F)

Address offset: 0x0C

Reset value: 0x2400 0000 for port A; 0x0000 0000 for others.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUD15[1:0] | | PUD14[1:0] | | PUD13[1:0] | | PUD12[1:0] | | PUD11[1:0] | | PUD10[1:0] | | PUD9[1:0] | | PUD8[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PUD7[1:0] | | PUD6[1:0] | | PUD5[1:0] | | PUD4[1:0] | | PUD3[1:0] | | PUD2[1:0] | | PUD1[1:0] | | PUD0[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | PUD15[1:0] | Pin 15 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 29:28 | PUD14[1:0] | Pin 14 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 27:26 | PUD13[1:0] | Pin 13 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 25:24 | PUD12[1:0] | Pin 12 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 23:22 | PUD11[1:0] | Pin 11 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 21:20 | PUD10[1:0] | Pin 10 pull-up or pull-down bits<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to PUD0[1:0] description |
| 19:18 | PUD9[1:0] | Pin 9 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 17:16 | PUD8[1:0] | Pin 8 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 15:14 | PUD7[1:0] | Pin 7 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 13:12 | PUD6[1:0] | Pin 6 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 11:10 | PUD5[1:0] | Pin 5 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 9:8 | PUD4[1:0] | Pin 4 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 7:6 | PUD3[1:0] | Pin 3 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 5:4 | PUD2[1:0] | Pin 2 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 3:2 | PUD1[1:0] | Pin 1 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 1:0 | PUD0[1:0] | Pin 0 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>00: Floating mode, no pull-up and pull-down (reset value)<br>01: With pull-up mode<br>10: With pull-down mode<br>11: Reserved |

### 6.4.5. Port input status register (GPIOx_ISTAT, x=A..C,F)

Address offset: 0x10

Reset value: 0x0000 XXXX

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISTAT15 | ISTAT14 | ISTAT13 | ISTAT12 | ISTAT11 | ISTAT10 | ISTAT 9 | ISTAT 8 | ISTAT 7 | ISTAT 6 | ISTAT 5 | ISTAT 4 | ISTAT 3 | ISTAT 2 | ISTAT 1 | ISTAT 0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | ISTATy | Port input status (y=0..15)<br>These bits are set and cleared by hardware.<br>0: Input signal low<br>1: Input signal high |

## 6.4.6. Port output control register (GPIOx_OCTL, x=A..C,F)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OCTL15 | OCTL14 | OCTL13 | OCTL12 | OCTL11 | OCTL10 | OCTL9 | OCTL8 | OCTL7 | OCTL6 | OCTL5 | OCTL4 | OCTL3 | OCTL2 | OCTL1 | OCTL0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | OCTLy | Port output control (y=0..15)<br>These bits are set and cleared by software.<br>0: Pin output low<br>1: Pin output high |

## 6.4.7. Port bit operate register (GPIOx_BOP, x=A..C,F)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

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

| CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| BOP15 | BOP14 | BOP13 | BOP12 | BOP11 | BOP10 | BOP9 | BOP8 | BOP7 | BOP6 | BOP5 | BOP4 | BOP3 | BOP2 | BOP1 | BOP0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Cry | Port clear bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Clear the corresponding OCTLy bit |
| 15:0 | BOPy | Port set bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Set the corresponding OCTLy bit |

## 6.4.8. Port configuration lock register (GPIOx_LOCK, x=A,B)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | LKK |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LK15 | LK14 | LK13 | LK12 | LK11 | LK10 | LK9 | LK8 | LK7 | LK6 | LK5 | LK4 | LK3 | LK2 | LK1 | LK0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |
| 16 | LKK | Lock key<br>It can only be set by using the lock key writing sequence. And is always readable.<br>0: GPIOx_LOCK register and the port configuration are not locked<br>1: GPIOx_LOCK register is locked until an MCU reset<br>LOCK key writing sequence:<br>Write 1→Write 0→Write 1→ Read 0→ Read 1<br>**Note:** The value of LKy(y=0..15) must be held during the LOCK Key writing sequence. |
| 15:0 | LKy | Port lock bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: Port configuration not locked |

1: Port configuration locked

### 6.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x=A,B,F)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEL7[3:0] | | | | SEL6[3:0] | | | | SEL5[3:0] | | | | SEL4[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEL3[3:0] | | | | SEL2[3:0] | | | | SEL1[3:0] | | | | SEL0[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | SEL7[3:0] | Pin 7 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 27:24 | SEL6[3:0] | Pin 6 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 23:20 | SEL5[3:0] | Pin 5 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 19:16 | SEL4[3:0] | Pin 4 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 15:12 | SEL3[3:0] | Pin 3 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 11:8 | SEL2[3:0] | Pin 2 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 7:4 | SEL1[3:0] | Pin 1 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 3:0 | SEL0[3:0] | Pin 0 alternate function selected<br>These bits are set and cleared by software.<br>0000: AF0 selected (reset value) |

0001: AF1 selected

0010: AF2 selected

0011: AF3 selected

0100: AF4 selected (Port A,B only)

0101: AF5 selected (Port A,B only)

0110: AF6 selected (Port A,B only)

0111: AF7 selected (Port A,B only)

1000 ~ 1111: Reserved

### 6.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x=A,B,F)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEL15[3:0] | | | | SEL14[3:0] | | | | SEL13[3:0] | | | | SEL12[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEL11[3:0] | | | | SEL10[3:0] | | | | SEL9[3:0] | | | | SEL8[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | SEL15[3:0] | Pin 15 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 27:24 | SEL14[3:0] | Pin 14 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 23:20 | SEL13[3:0] | Pin 13 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 19:16 | SEL12[3:0] | Pin 12 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 15:12 | SEL11[3:0] | Pin 1 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 11:8 | SEL10[3:0] | Pin 10 alternate function selected<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to SEL8[3:0] description |
| 7:4 | SEL9[3:0] | Pin 9 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 3:0 | SEL8[3:0] | Pin 8 alternate function selected<br>These bits are set and cleared by software.<br>0000: AF0 selected (reset value)<br>0001: AF1 selected<br>0010: AF2 selected<br>0011: AF3 selected<br>0100: AF4 selected (Port A,B only)<br>0101: AF5 selected (Port A,B only)<br>0110: AF6 selected (Port A,B only)<br>0111: AF7 selected (Port A,B only)<br><br>1000 ~ 1111: Reserved |

### 6.4.11. Bit clear register (GPIOx_BC, x=A..C,F)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CRy | Port clear bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Clear the corresponding OCTLy bit |

### 6.4.12. Port bit toggle register (GPIOx_TG, x=A..C,F)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TG15 | TG14 | TG13 | TG12 | TG11 | TG10 | TG9 | TG8 | TG7 | TG6 | TG5 | TG4 | TG3 | TG2 | TG1 | TG0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | TGy | Port toggle bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Toggle the corresponding OCTLy bit |

# 7. Cyclic redundancy checks management unit (CRC)

## 7.1. Overview

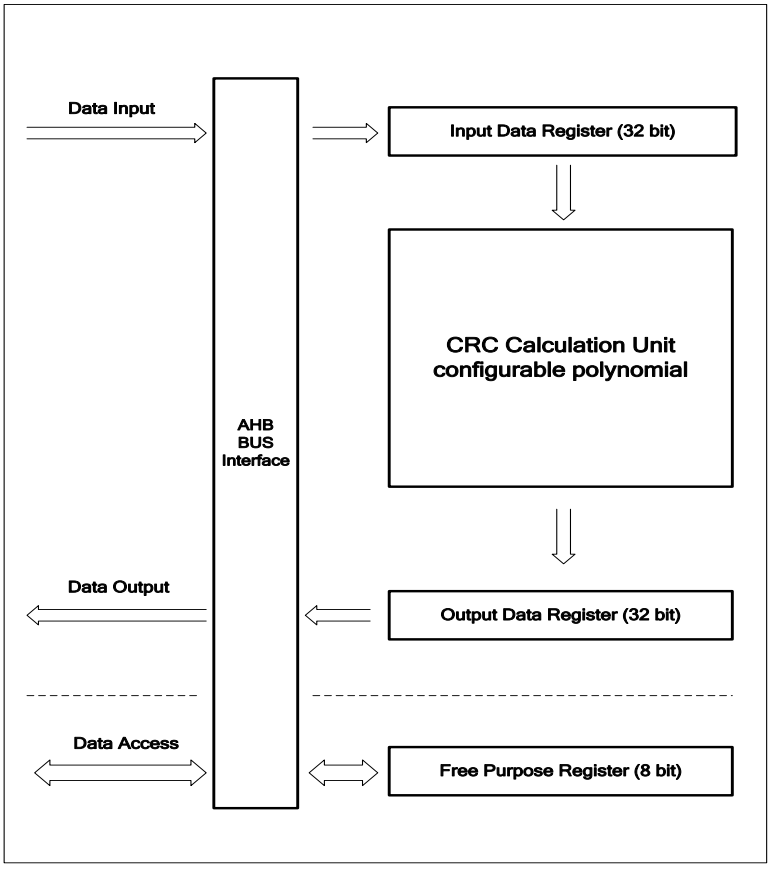
A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

This CRC management unit can be used to calculate 7/8/16/32 bit CRC code within user configurable polynomial.

## 7.2. Characteristics

- Input data supports 7/8/16/32 size bit.
- Different input size for different calculation time.1/2/4 cycle for 7(8)/16/32 bits.
- Input and output data can be reversed.
- User configurable polynomial size.
- User configurable initial value after CRC reset.
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices.

**Figure 7-1. Block diagram of CRC calculation unit**



## 7.3. Function overview

- CRC management unit is used to calculate the 32-bit raw data, and CRC_DATA register will receive the raw data and store the calculation result.

  If the CRC_DATA register has not been cleared by software setting the CRC_CTL register, the new input raw data will be calculated based on the result of previous value of CRC_DATA.

  CRC calculation will spend 4/2/1 AHB clock cycles for 32/16/8(7) bit data size, during this period AHB will not be hanged because of the existence of the 32bit input buffer.

- This module supplies an 8-bit free register CRC_FDATA.

  CRC_FDATA is unrelated to the CRC calculation, any value you write in will be read out at anytime.

- Reversible function can reverse the input data and output data.

  For input data, 3 reverse types can be selected.

  Original data is 0x3456CDEF:

  1)byte reverse:

  32-bit data is divided into 4 groups and reverse implement in group inside. Reversed data: 0x2C6AB3F7

  2)half-word reverse:

  32-bit data is divided into 2 groups and reverse implement in group inside. Reversed data: 0x6A2CF7B3

  3)word reverse:

  32-bit data is divided into 1 groups and reverse implement in group inside. Reversed data: 0xF7B36A2C

  For output data, reverse type is word reverse.

  For example: when REV_O=1, calculation result 0x3344CCDD will be converted to 0xBB3322CC.

- User configurable initial calculation data is available.

  When RST bit is set or write operation to CRC_IDATA register, the CRC_DATA register will be automatically initialized to the value in CRC_IDATA.

- User configurable polynomial.

  Depends on PS [1:0] bits, the valid polynomial and output bit width can be selected by user. If the polynomial is less than 32 bit, the high bits of the input data and output data

is unavailable. It is strongly recommend resetting the CRC management unit after change the PS[1:0] bits or polynomial.

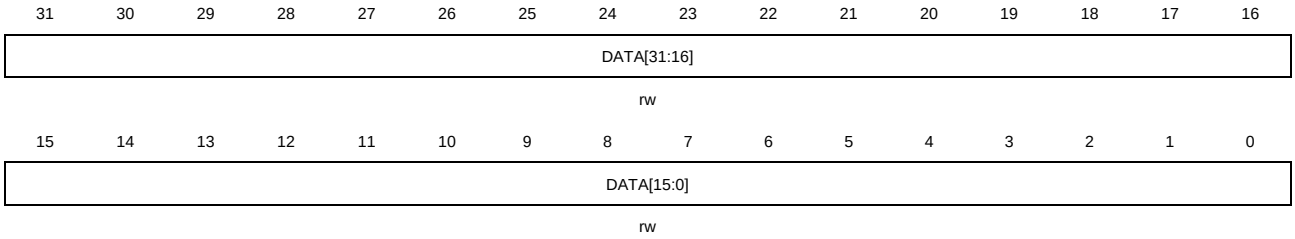
# 7.4. Register definition

CRC base address: 0x4002 3000

## 7.4.1. Data register (CRC_DATA)

Address offset: 0x00
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit)

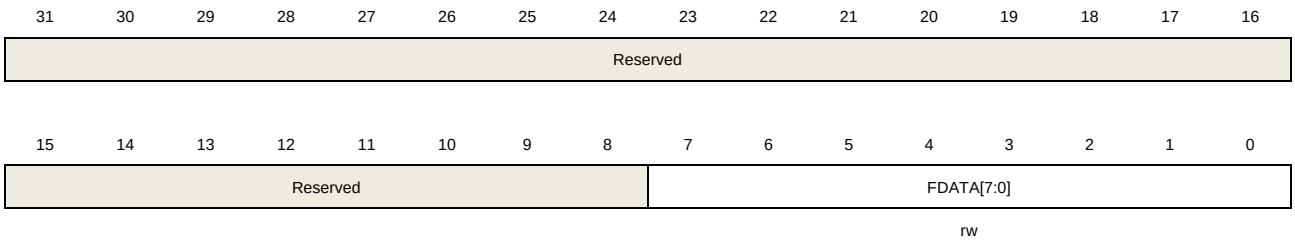



| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | CRC calculation result bits<br>Software writes and reads.<br>This register is used to calculate new data, and the register can be written the new data directly. Write value cannot be read because the read value is the previous CRC calculation result. |

## 7.4.2. Free data register (CRC_FDATA)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | FDATA[7:0] | Free Data Register bits |

Software writes and reads.

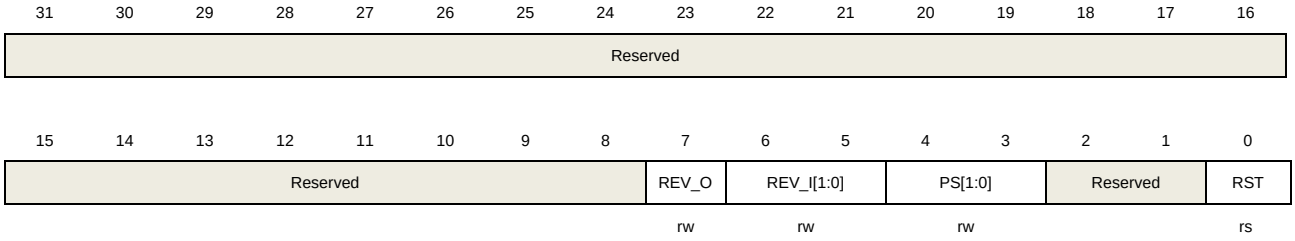
These bits are unrelated with CRC calculation. This byte can be used for any goal by any other peripheral. The CRC_CTL register will generate no effect to the byte.

### 7.4.3. Control register (CRC_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | REV_O | Reverse output data value in bit order<br>0: Not bit reversed for output data<br>1: Bit reversed for output data |
| 6:5 | REV_I[1:0] | Reverse type for input data<br>0: Dot not use reverse for input data<br>1: Reverse input data with every 8-bit length<br>2: Reverse input data with every 16-bit length<br>3: Reverse input data with whole 32-bit length |
| 4:3 | PS[1:0] | Size of polynomial<br>0: 32 bit<br>1: 16 bit (POLY [15:0] is used for calculation)<br>2: 8 bit (POLY [7:0] is used for calculation)<br>3: 7 bit (POLY [6:0] is used for calculation) |
| 2:1 | Reserved | Must be kept at reset value |
| 0 | RST | Set this bit can reset the CRC_DATA register to the value in CRC_IDATA then automatically cleared itself to 0 by hardware. This bit will take no effect to CRC_FDATA.<br>Software writes and reads. |

### 7.4.4. Initialization data register (CRC_IDATA)

Address offset: 0x10

Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit)

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IDATA[31:0] | Configurable initial CRC data value<br>When RST bit in CRC_CTL asserted, CRC_DATA will be programmed to this value. |

### 7.4.5. Polynomial register (CRC_POLY)

Address offset: 0x14

Reset value: 0x04C1 1DB7

This register has to be accessed by word (32-bit)

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | POLY[31:0] | User configurable polynomial value<br>This value is used together with PS[1:0] bits. |

# 8. Direct memory access controller (DMA)

## 8.1. Overview

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Data can be quickly moved by DMA between peripherals and memory as well as memory and memory without any CPU actions. There are 5 channels in the DMA controller. Each channel is dedicated to manage memory access requests from one or more peripherals. An arbiter is implemented inside to handle the priority among DMA requests.

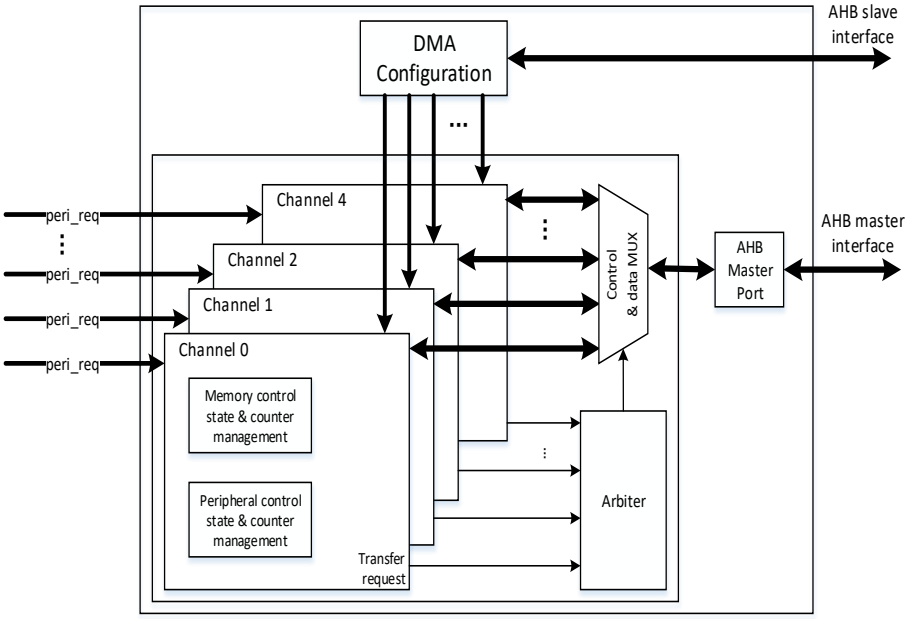
The system bus is shared by the DMA controller and the Cortex®-M23 core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system bus bandwidth for the CPU.

## 8.2. Characteristics

- Programmable length of data to be transferred, max to 65536
- 5 channels and each channel are configurable
- AHB and APB peripherals, FLASH, SRAM can be accessed as source and destination
- Each channel is connected to fixed hardware DMA request
- Software DMA channel priority (low, medium, high, ultra high) and hardware DMA channel priority (DMA channel 0 has the highest priority and DMA channel 4 has the lowest priority)
- Support independent 8, 16, 32-bit memory and peripheral transfer
- Support independent fixed and increasing address generation algorithm of memory and peripheral
- Support circular transfer mode
- Support peripheral to memory, memory to peripheral, and memory to memory transfers
- One separate interrupt per channel with three types of event flags
- Support interrupt enable and clear

## 8.3. Block diagram

**Figure 8-1. Block diagram of DMA**




As shown in ***Figure 8-1. Block diagram of DMA***, a DMA controller consists of four main parts:

- DMA configuration through AHB slave interface
- Data transmission through two AHB master interfaces for memory access and peripheral access
- An arbiter inside to manage multiple peripheral requests coming at the same time
- Channel management to control address/data selection and data counting

## 8.4. Function overview

### 8.4.1. DMA operation

Each DMA transfer consists of two operations, including the loading of data from the source and the storage of the loaded data to the destination. The source and destination addresses are computed by the DMA controller based on the programmed values in the DMA_CHxPADDR, DMA_CHxMADDR, and DMA_CHxCTL registers. The DMA_CHxCNT register controls how many transfers to be transmitted on the channel. The PWIDTH and MWIDTH bits in the DMA_CHxCTL register determine how many bytes to be transmitted in a transfer.

Suppose DMA_CHxCNT is 4, and both PNAGA and MNAGA are set. The DMA transfer operations for each combination of PWIDTH and MWIDTH are shown in the ***Table 8-1. DMA transfer operation***.

**Table 8-1. DMA transfer operation**

| Transfer size | | Transfer operations | |
|---|---|---|---|
| **Source** | **Destination** | **Source** | **Destination** |
| 32 bits | 32 bits | 1: Read B3B2B1B0[31:0] @0x0<br>2: Read B7B6B5B4[31:0] @0x4<br>3: Read BBBAB9B8[31:0] @0x8<br>4: Read BFBEBDBC[31:0] @0xC | 1: Write B3B2B1B0[31:0] @0x0<br>2: Write B7B6B5B4[31:0] @0x4<br>3: Write BBBAB9B8[31:0] @0x8<br>4: Write BFBEBDBC[31:0] @0xC |
| 32 bits | 16 bits | 1: Read B3B2B1B0[31:0] @0x0<br>2: Read B7B6B5B4[31:0] @0x4<br>3: Read BBBAB9B8[31:0] @0x8<br>4: Read BFBEBDBC[31:0] @0xC | 1: Write B1B0[7:0] @0x0<br>2: Write B5B4[7:0] @0x2<br>3: Write B9B8[7:0] @0x4<br>4: Write BDBC[7:0] @0x6 |
| 32 bits | 8 bits | 1: Read B3B2B1B0[31:0] @0x0<br>2: Read B7B6B5B4[31:0] @0x4<br>3: Read BBBAB9B8[31:0] @0x8<br>4: Read BFBEBDBC[31:0] @0xC | 1: Write B0[7:0] @0x0<br>2: Write B4[7:0] @0x1<br>3: Write B8[7:0] @0x2<br>4: Write BC[7:0] @0x3 |
| 16 bits | 32 bits | 1: Read B1B0[15:0] @0x0<br>2: Read B3B2[15:0] @0x2<br>3: Read B5B4[15:0] @0x4<br>4: Read B7B6[15:0] @0x6 | 1: Write 0000B1B0[31:0] @0x0<br>2: Write 0000B3B2[31:0] @0x4<br>3: Write 0000B5B4[31:0] @0x8<br>4: Write 0000B7B6[31:0] @0xC |
| 16 bits | 16 bits | 1: Read B1B0[15:0] @0x0<br>2: Read B3B2[15:0] @0x2<br>3: Read B5B4[15:0] @0x4<br>4: Read B7B6[15:0] @0x6 | 1: Write B1B0[15:0] @0x0<br>2: Write B3B2[15:0] @0x2<br>3: Write B5B4[15:0] @0x4<br>4: Write B7B6[15:0] @0x6 |
| 16 bits | 8 bits | 1: Read B1B0[15:0] @0x0<br>2: Read B3B2[15:0] @0x2<br>3: Read B5B4[15:0] @0x4<br>4: Read B7B6[15:0] @0x6 | 1: Write B0[7:0] @0x0<br>2: Write B2[7:0] @0x1<br>3: Write B4[7:0] @0x2<br>4: Write B6[7:0] @0x3 |
| 8 bits | 32 bits | 1: Read B0[7:0] @0x0<br>2: Read B1[7:0] @0x1<br>3: Read B2[7:0] @0x2<br>4: Read B3[7:0] @0x3 | 1: Write 000000B0[31:0] @0x0<br>2: Write 000000B1[31:0] @0x4<br>3: Write 000000B2[31:0] @0x8<br>4: Write 000000B3[31:0] @0xC |
| 8 bits | 16 bits | 1: Read B0[7:0] @0x0<br>2: Read B1[7:0] @0x1<br>3: Read B2[7:0] @0x2<br>4: Read B3[7:0] @0x3 | 1, Write 00B0[15:0] @0x0<br>2, Write 00B1[15:0] @0x2<br>3, Write 00B2[15:0] @0x4<br>4, Write 00B3[15:0] @0x6 |
| 8 bits | 8 bits | 1: Read B0[7:0] @0x0<br>2: Read B1[7:0] @0x1<br>3: Read B2[7:0] @0x2<br>4: Read B3[7:0] @0x3 | 1, Write B0[7:0] @0x0<br>2, Write B1[7:0] @0x1<br>3, Write B2[7:0] @0x2<br>4, Write B3[7:0] @0x3 |

The CNT bits in the DMA_CHxCNT register control how many data to be transmitted on the channel and must be configured before enable the CHEN bit in the register. During the transmission, the CNT bits indicate the remaining number of data items to be transferred.

The DMA transmission is disabled by clearing the CHEN bit in the DMA_CHxCTL register.

- If the DMA transmission is not completed when the CHEN bit is cleared, two situations may be occurred when restart this DMA channel:
  - If no register configuration operations of the channel occurs before restart the DMA channel, the DMA will continue to complete the rest of the transmission.
  - If any register configuration operations to DMA_CHxCNT, DMA_CHxPADDR or DMA_CHxMADDR of corresponding channel occur, the DMA will restart a new transmission.
- If the DMA transmission has been finished when clearing the CHEN bit, enable the DMA channel without any register configuration operation to DMA_CHxCNT, DMA_CHxPADDR or DMA_CHxMADDR of corresponding channel will not launch any DMA transfer.

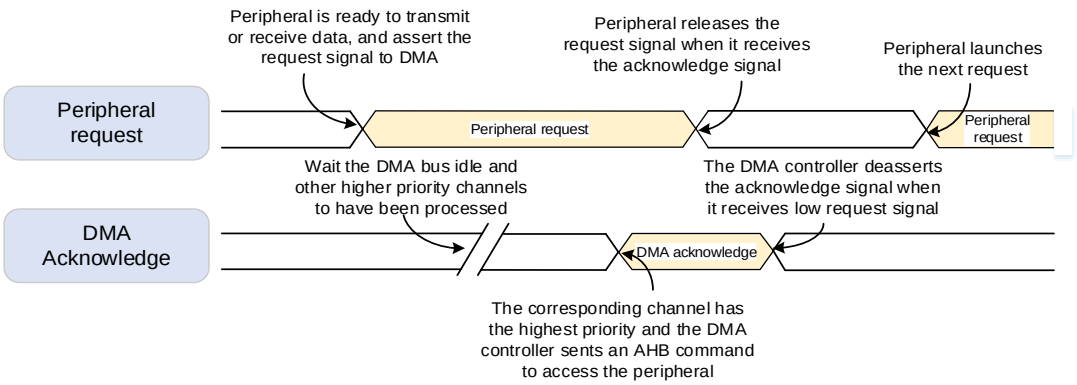
### 8.4.2. Peripheral handshake

To ensure a well-organized and efficient data transfer, a handshake mechanism is introduced between the DMA and peripherals, including a request signal and a acknowledge signal:

- Request signal asserted by peripheral to DMA controller, indicating that the peripheral is ready to transmit or receive data
- Acknowledge signal responded by DMA to peripheral, indicating that the DMA controller has initiated an AHB command to access the peripheral

*Figure 8-2. Handshake mechanism* shows how the handshake mechanism works between the DMA controller and peripherals.

**Figure 8-2. Handshake mechanism**



### 8.4.3. Arbitration

When two or more requests are received at the same time, the arbiter determines which request is served based on the priorities of channels. There are two-stage priorities, including the software priority and the hardware priority. The arbiter determines which channel is selected to respond according to the following priority rules:

- Software priority: Four levels, including low, medium, high and ultra-high by configuring the PRIO bits in the DMA_CHxCTL register.
- For channels with equal software priority level, priority is given to the channel with lower channel number.

### 8.4.4. Address generation

Two kinds of address generation algorithm are implemented independently for memory and peripheral, including the fixed mode and the increased mode. The PNAGA and MNAGA bit in the DMA_CHxCTL register are used to configure the next address generation algorithm of peripheral and memory.

In the fixed mode, the next address is always equal to the base address configured in the base address registers (DMA_CHxPADDR, DMA_CHxMADDR).

In the increasing mode, the next address is equal to the current address plus 1 or 2 or 4, depending on the transfer data width.

### 8.4.5. Circular mode

Circular mode is implemented to handle continue peripheral requests (for example, ADC scan mode). The circular mode is enabled by setting the CMEN bit in the DMA_CHxCTL register.

In circular mode, the CNT bits are automatically reloaded with the pre-programmed value and the full transfer finish flag is asserted at the end of every DMA transfer. DMA can always responds the peripheral request until the CHEN bit in the DMA_CHxCTL register is cleared.

### 8.4.6. Memory to memory mode

The memory to memory mode is enabled by setting the M2M bit in the DMA_CHxCTL register. In this mode, the DMA channel can also work without being triggered by a request from a peripheral. The DMA channel starts transferring as soon as it is enabled by setting the CHEN bit in the DMA_CHxCTL register, and completed when the DMA_CHxCNT register reaches zero.

### 8.4.7. Channel configuration

When starting a new DMA transfer, it is recommended to respect the following steps:

1. Read the CHEN bit and judge whether the channel is enabled or not. If the channel is enabled, clear the CHEN bit by software. When the CHEN bit is read as '0', configuring and starting a new DMA transfer is allowed.
2. Configure the M2M bit and DIR bit in the DMA_CHxCTL register to set the transfer mode.
3. Configure the CMEN bit in the DMA_CHxCTL register to enable/disable the circular mode.
4. Configure the PRIO bits in the DMA_CHxCTL register to set the channel software priority.
5. Configure the memory and peripheral transfer width, memory and peripheral address generation algorithm in the DMA_CHxCTL register.
6. Configure the enable bit for full transfer finish interrupt, half transfer finish interrupt, transfer error interrupt in the DMA_CHxCTL register.
7. Configure the DMA_CHxPADDR register for setting the peripheral base address.
8. Configure the DMA_CHxMADDR register for setting the memory base address.
9. Configure the DMA_CHxCNT register to set the total transfer data number.
10. Configure the CHEN bit with '1' in the DMA_CHxCTL register to enable the channel.

### 8.4.8. Interrupt

Each DMA channel has a dedicated interrupt. There are three types of interrupt event, including full transfer finish, half transfer finish, and transfer error.

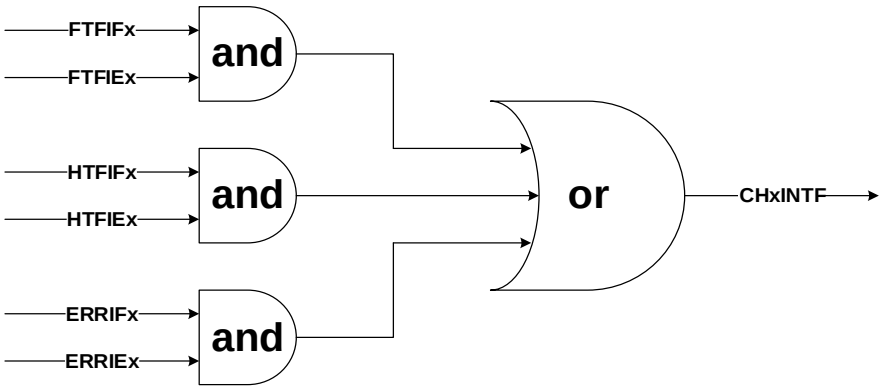
Each interrupt event has a dedicated flag bit in the DMA_INTF register, a dedicated clear bit in the DMA_INTC register, and a dedicated enable bit in the DMA_CHxCTL register. The relationship is described in the ***Table 8-2. interrupt events***.

**Table 8-2. interrupt events**

| Interrupt event | Flag bit | Clear bit | Enable bit |
|---|---|---|---|
| | DMA_INTF | DMA_INTC | DMA_CHxCTL |
| Full transfer finish | FTFIF | FTFIFC | FTFIE |
| Half transfer finish | HTFIF | HTFIFC | HTFIE |
| Transfer error | ERRIF | ERRIFC | ERRIE |

The DMA interrupt logic is shown in the ***Figure 8-3. DMA interrupt logic***, an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

**Figure 8-3. DMA interrupt logic**




**Note:** "x" indicates channel number (x=0…4).

## 8.4.9. DMA request mapping

Several requests from peripherals may be mapped to one DMA channel. They are logically ORed before entering the DMA. For details, see the ***Figure 8-4. DMA request mapping***. The request of each peripheral can be independently enabled or disabled by programming the registers of the corresponding peripheral. The user has to ensure that only one request is enabled at a time on one channel. ***Table 8-3. DMA requests for each channel*** lists the support request from peripheral for each channel of DMA.

**Figure 8-4. DMA request mapping**

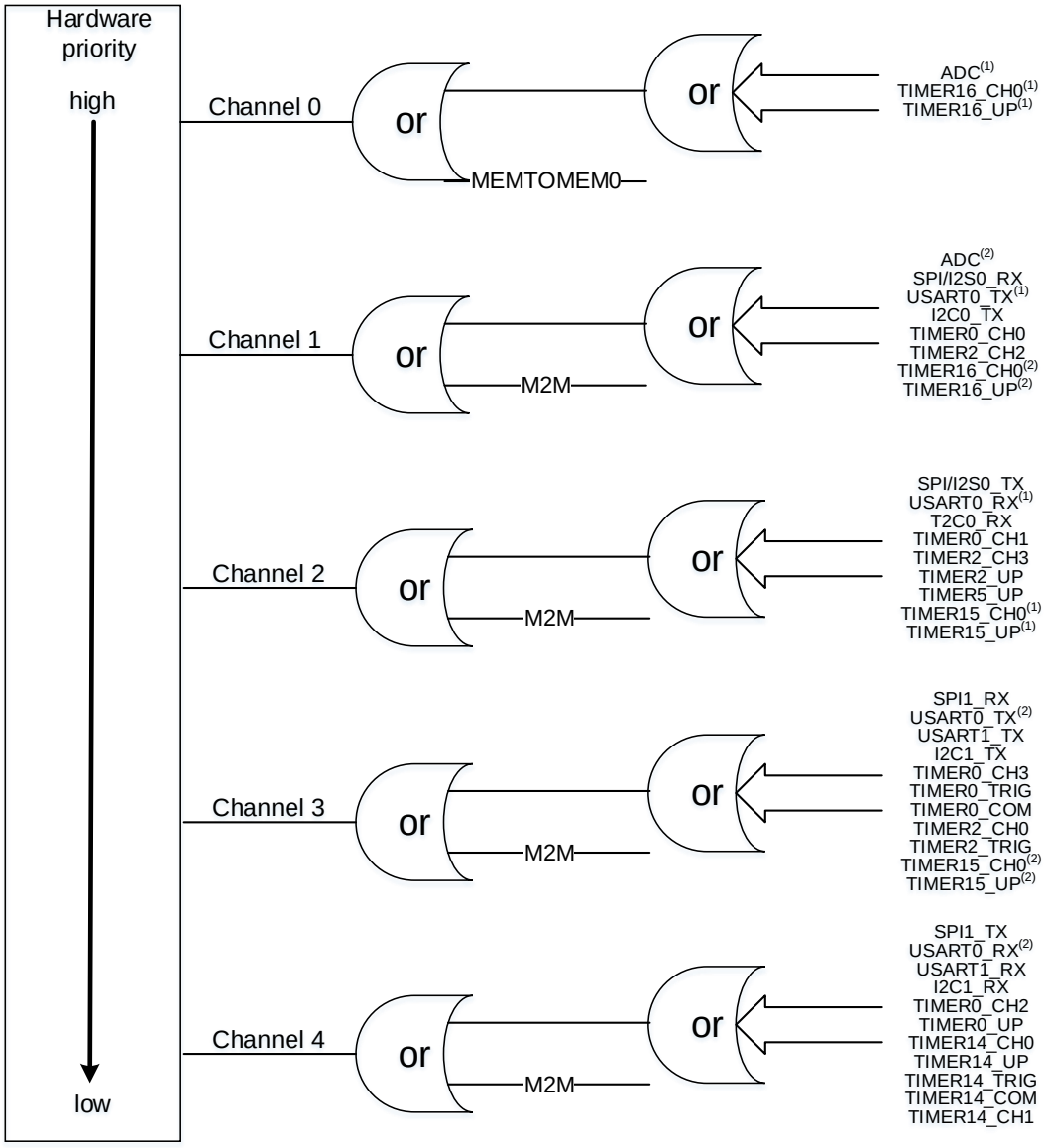



**Table 8-3. DMA requests for each channel**

| Peripheral | Channel 0 | Channel 1 | Channel 2 | Channel 3 | Channel 4 |
|---|---|---|---|---|---|
| ADC | ADC(1) | ADC(2) | • | • | • |
| SPI/I2S | • | SPI/I2S0_RX | SPI/I2S0_TX | SPI1_RX | SPI1_TX |
| USART | • | USART0_TX(1) | USART0_RX(1) | USART0_TX(2)<br>USART1_TX | USART0_RX(2)<br>USART1_RX |
| I2C | • | I2C0_TX | I2C0_RX | I2C1_TX | I2C1_RX |
| TIMER0 | • | TIMER0_CH0 | TIMER0_CH1 | TIMER0_CH3<br>TIMER0_TRIG<br>TIMER0_COM | TIMER0_CH2<br>TIMER0_UP |

| | | | | | |
|---|---|---|---|---|---|
| TIMER2 | • | TIMER2_CH2 | TIMER2_CH3<br>TIMER2_UP | TIMER2_CH0<br>TIMER2_TRIG | • |
| TIMER5 | • | • | TIMER5_UP | • | • |
| TIMER14 | • | • | • | • | TIMER14_CH0<br>TIMER14_UP<br>TIMER14_TRIG<br>TIMER14_COM<br>TIMER14_CH1 |
| TIMER15 | • | • | TIMER15_CH0(1)<br>TIMER15_UP(1) | TIMER15_CH0(2)<br>TIMER15_UP(2) | • |
| TIMER16 | TIMER16_CH0(1)<br>TIMER16_UP(1) | TIMER16_CH0(2)<br>TIMER16_UP(2) | • | • | • |

1. When the corresponding remapping bit in the SYSCFG_CFG0 register is cleared, the request is mapped on the channel.
2. When the corresponding remapping bit in the SYSCFG_CFG0 register is set, the request is mapped on the channel.

# 8.5. Register definition

DMA base address: 0x4002 0000

## 8.5.1. Interrupt flag register (DMA_INTF)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | ERRIF4 | HTFIF4 | FTFIF4 | GIF4 |
| | | | | | | | | | | | | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRIF3 | HTFIF3 | FTFIF3 | GIF3 | ERRIF2 | HTFIF2 | FTFIF2 | GIF2 | ERRIF1 | HTFIF1 | FTFIF1 | GIF1 | ERRIF0 | HTFIF0 | FTFIF0 | GIF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value |
| 19/15/11/7/3 | ERRIFx | Error flag of channel x (x=0…4)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Transfer error has not occurred on channel x<br>1: Transfer error has occurred on channel x |
| 18/14/10/6/2 | HTFIFx | Half transfer finish flag of channel x (x=0…4)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Half number of transfer has not finished on channel x<br>1: Half number of transfer has finished on channel x |
| 17/13/9/5/1 | FTFIFx | Full Transfer finish flag of channel x (x=0…4)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Transfer has not finished on channel x<br>1: Transfer has finished on channel x |
| 16/12/8/4/0 | GIFx | Global interrupt flag of channel x (x=0…4)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: None of ERRIF, HTFIF or FTFIF occurs on channel x<br>1: At least one of ERRIF, HTFIF or FTFIF occurs on channel x |

## 8.5.2. Interrupt flag clear register (DMA_INTC)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | ERRIFC4 | HTFIFC4 | FTFIFC4 | GIFC4 |
| | | | | | | | | | | | | w | w | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRIFC3 | HTFIFC3 | FTFIFC3 | GIFC3 | ERRIFC2 | HTFIC2 | FTFIFC2 | GIFC2 | ERRIFC1 | HTFIFC1 | FTFIFC1 | GIFC1 | ERRIFC0 | HTFIFC0 | FTFIFC0 | GIFC0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value |
| 19/15/11/7/3 | ERRIFCx | Clear bit for error flag of channel x (x=0…4)<br>0: No effect<br>1: Clear error flag |
| 18/14/10/6/2 | HTFIFCx | Clear bit for half transfer finish flag of channel x (x=0…4)<br>0: No effect<br>1: Clear half transfer finish flag |
| 17/13/9/5/1 | FTFIFCx | Clear bit for full transfer finish flag of channel x (x=0…4)<br>0: No effect<br>1: Clear full transfer finish flag |
| 16/12/8/4/0 | GIFCx | Clear global interrupt flag of channel x (x=0…4)<br>0: No effect<br>1: Clear GIFx, ERRIFx, HTFIFx and FTFIFx bits in the DMA_INTF register |

### 8.5.3. Channel x control register (DMA_CHxCTL)

x = 0...4, where x is a channel number

Address offset: 0x08 + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | M2M | PRIO[1:0] | | MWIDTH[1:0] | | PWIDTH[1:0] | | MNAGA | PNAGA | CMEN | DIR | ERRIE | HTFIE | FTFIE | CHEN |
| | rw | rw | | rw | | rw | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value |
| 14 | M2M | Memory to Memory Mode |

| | | |
|---|---|---|
| | | Software set and cleared<br>0: Disable Memory to Memory Mode<br>1: Enable Memory to Memory mode<br>This bit can not be written when CHEN is ‘1’. |
| 13:12 | PRIO[1:0] | Priority level<br>Software set and cleared<br>00: Low<br>01: Medium<br>10: High<br>11: Ultra high<br>These bits can not be written when CHEN is ‘1’. |
| 11:10 | MWIDTH[1:0] | Transfer data size of memory<br>Software set and cleared<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved<br>These bits can not be written when CHEN is ‘1’. |
| 9:8 | PWIDTH[1:0] | Transfer data size of peripheral<br>Software set and cleared<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved<br>These bits can not be written when CHEN is ‘1’. |
| 7 | MNAGA | Next address generation algorithm of memory<br>Software set and cleared<br>0: Fixed address mode<br>1: Increasing address mode<br>This bit can not be written when CHEN is ‘1’. |
| 6 | PNAGA | Next address generation algorithm of peripheral<br>Software set and cleared<br>0: Fixed address mode<br>1: Increasing address mode<br>This bit can not be written when CHEN is ‘1’. |
| 5 | CMEN | Circular mode enable<br>Software set and cleared<br>0: Disable circular mode<br>1: Enable circular mode<br>This bit can not be written when CHEN is ‘1’. |

| | | |
|---|---|---|
| 4 | DIR | Transfer direction<br>Software set and cleared<br>0: Read from peripheral and write to memory<br>1: Read from memory and write to peripheral<br>This bit can not be written when CHEN is ‘1’. |
| 3 | ERRIE | Enable bit for channel error interrupt<br>Software set and cleared<br>0: Disable the channel error interrupt<br>1: Enable the channel error interrupt |
| 2 | HTFIE | Enable bit for channel half transfer finish interrupt<br>Software set and cleared<br>0:Disable channel half transfer finish interrupt<br>1:Enable channel half transfer finish interrupt |
| 1 | FTFIE | Enable bit for channel full transfer finish interrupt<br>Software set and cleared<br>0:Disable channel full transfer finish interrupt<br>1:Enable channel full transfer finish interrupt |
| 0 | CHEN | Channel enable<br>Software set and cleared<br>0:Disable channel<br>1:Enable channel |

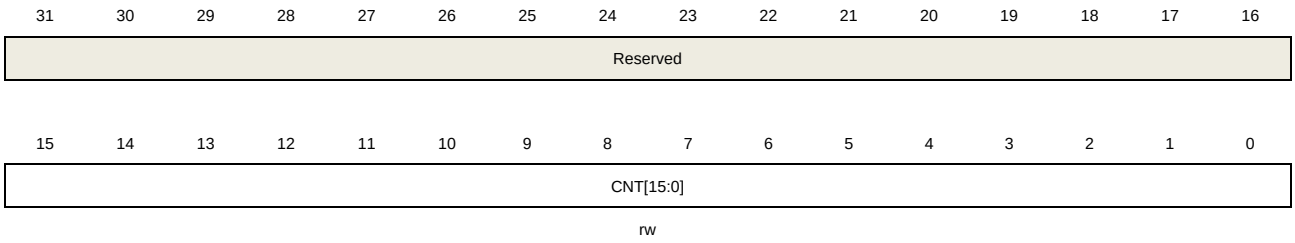
### 8.5.4. Channel x counter register (DMA_CHxCNT)

x = 0...4, where x is a channel number

Address offset: 0x0C + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | Transfer counter<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’.<br>This register indicates how many transfers remain. Once the channel is enabled, it |

is read-only, and decreases after each DMA transfer. If the register is zero, no transaction can be issued whether the channel is enabled or not. Once the transmission of the channel is complete, the register can be reloaded automatically by the previously programmed value if the channel is configured in circular mode.

### 8.5.5. Channel x peripheral base address register (DMA_CHxPADDR)

x = 0...4, where x is a channel number

Address offset: 0x10 + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PADDR[31:0] | Peripheral base address<br>These bits can not be written when CHEN in the DMA_CHxCTL register is '1'.<br>When PWIDTH is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address.<br>When PWIDTH is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address. |

### 8.5.6. Channel x memory base address register (DMA_CHxMADDR)

x = 0...4, where x is a channel number

Address offset: 0x14 + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MADDR[31:0] | Memory base address |

These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’.
When MWIDTH in the DMA_CHxCTL register is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address.
When MWIDTH in the DMA_CHxCTL register is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.

# 9. Debug (DBG)

## 9.1. Overview

The GD32E23x series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the Arm CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the Arm Cortex®-M23. The debug system supports serial wire debug (SWD) and trace functions. The debug and trace functions refer to the following documents:

- Cortex®-M23 Technical Reference Manual
- Arm Debug Interface v5 Architecture Specification

The DBG hold unit helps debugger to debug power saving mode, TIMER, I2C, RTC, WWDGT, and FWDGT. When corresponding bit is set, it provides a clock in power saving mode or holds the state for TIMER, I2C, RTC, WWDGT and FWDGT.

## 9.2. SW function overview

Debug capabilities can be accessed by a debug tool via Serial Wire (SW - Debug Port).

### 9.2.1. Pin assignment

The synchronous serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK).

The pin assignment is as following:

PA14 : SWCLK

PA13 : SWDIO

If SWD not used, all 2-pin can be released to other GPIO functions. Please refer to ***GPIO pin configuration***.

## 9.3. Debug hold function overview

### 9.3.1. Debug support for power saving mode

When the STB_HOLD bit in DBG control register 0 (DBG_CTL0) is set, and entering the standby mode, the clock of AHB bus and system clock are provided by CK_IRC8M, and the debugger can debug in standby mode. When exiting the standby mode, a system reset

generated.

When the DSLP_HOLD bit in DBG control register 0 (DBG_CTL0) is set, and entering the Deep-sleep mode, the clock of AHB bus and system clock are provided by CK_IRC8M, and the debugger can debug in Deep-sleep mode.

When the SLP_HOLD bit in DBG control register 0 (DBG_CTL0) is set, and entering the sleep mode, the clock of AHB bus for CPU is not closed, and the debugger can debug in sleep mode.

### 9.3.2. Debug support for TIMER, I2C, RTC, WWDGT and FWDGT

When the core is halted and the corresponding bit in DBG control register 0 or DBG control register 1 (DBG_CTL0 or DBG_CTL1) is set, the following events occur.

For TIMER, the timer counters are stopped and held for debugging.

For I2C, SMBUS timeout is held for debugging.

For RTC, the counter is stopped for debugging.

For WWDGT or FWDGT, the counter clock is stopped for debugging.

# 9.4. Register definition

DBG base address: 0x4001 5800

## 9.4.1. ID code register (DBG_ID)

Address offset: 0x00

Read only

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_CODE[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_CODE[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ID_CODE[31:0] | DBG ID code register<br>These bits can only be read by software, These bits are unchanged constant |

## 9.4.2. Control register 0 (DBG_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved. | | | | TIMER13_HOLD | Reserved | | | | | | | TIMER5_HOLD | Reserved | | I2C1_HOLD |
| | | | | rw | | | | | | | | rw | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2C0_HOLD | Reserved | | TIMER2_HOLD | Reserved | TIMER0_HOLD | WWDGT_HOLD | FWDGT_HOLD | Reserved | | | | | STB_HOLD | DSLP_HOLD | SLP_HOLD |
| rw | | | rw | | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27 | TIMER13_HOLD | TIMER 13 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 13 counter for debugging when the core is halted |
| 26:20 | Reserved | Must be kept at reset value |
| 19 | TIMER5_HOLD | TIMER 5 hold bit |

| | | |
|---|---|---|
| | | This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 5 counter for debugging when the core is halted |
| 18:17 | Reserved | Must be kept at reset value |
| 16 | I2C1_HOLD | I2C1 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the I2C1 SMBUS timeout for debugging when the core is halted |
| 15 | I2C0_HOLD | I2C0 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the I2C0 status to avoid SMBUS timeout for debugging when the core is halted |
| 14:13 | Reserved | Must be kept at reset value |
| 12 | TIMER2_HOLD | TIMER 2 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 2 counter for debugging when the core is halted |
| 11 | Reserved | Must be kept at reset value |
| 10 | TIMER0_HOLD | TIMER 0 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 0 counter for debugging when the core is halted |
| 9 | WWDGT_HOLD | WWDGT hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the WWDGT counter clock for debugging when the core is halted |
| 8 | FWDGT_HOLD | FWDGT hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the FWDGT counter clock for debugging when the core is halted |
| 7:3 | Reserved | Must be kept at reset value |
| 2 | STB_HOLD | Standby mode hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: In the standby mode, the clock of AHB bus and system clock are provided by CK_IRC8M, a system reset generated when exiting standby mode |

| | | |
|---|---|---|
| 1 | DSLP_HOLD | Deep-sleep mode hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: In the Deep-sleep mode, the clock of AHB bus and system clock are provided by CK_IRC8M |
| 0 | SLP_HOLD | Sleep mode hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: In the sleep mode, the clock of AHB is on. |

### 9.4.3. Control register 1 (DBG_CTL1)

Address offset: 0x08

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved. | | | | | | | | | | | | | TIMER16_HOLD | TIMER15_HOLD | TIMER14_HOLD |
| | | | | | | | | | | | | | rw | rw | rw |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value |
| 18 | TIMER16_HOLD | TIMER 16 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 16 counter for debugging when the core is halted |
| 17 | TIMER15_HOLD | TIMER 15 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 15 counter for debugging when the core is halted |
| 16 | TIMER14_HOLD | TIMER 14 hold bit<br>This bit is set and reset by software<br>0: no effect<br>1: hold the TIMER 14 counter for debugging when the core is halted |
| 15:11 | Reserved | Must be kept at reset value |
| 10 | RTC_HOLD | RTC hold bit<br>This bit is set and reset by software<br>0: no effect |

| | | |
|---|---|---|
| | | 1: hold the RTC counter for debugging when the core is halted |
| 9:0 | Reserved | Must be kept at reset value |

# 10. Analog to digital converter (ADC)

## 10.1. Overview

A 12-bit successive approximation analog-to-digital converter module(ADC) is integrated on the MCU chip, which can sample analog signals from 10 external channels and 2 internal channels. The 12 ADC sampling channels all support a variety of operation modes. After sampling and conversion, the conversion results can be stored in the corresponding data registers according to the least significant bit alignment or the most significant bit alignment. An on-chip hardware oversample scheme improves performances and reduces the computational burden of MCU.

## 10.2. Characteristics

- High performance:
  - ADC sampling rsolution: 12-bit, 10-bit,8-bit, or 6-bit.
  - Foreground calibration function.
  - Programmable sampling time.
  - Data storage mode: the most significant bit and the least significant bit.
  - DMA support.
- Dual clock domain architecture (APB clock and ADC clock).
- Analog input channels:
  - 10 external analog inputs.
  - 1 channel for internal temperature sensor ($V_{SENSE}$).
  - 1 channel for internal reference voltage ($V_{REFINT}$).
- Start-of-conversion can be initiated:
  - By software.
  - By hardware triggers.
- Operation modes:
  - Converts a single channel or scans a sequence of channels.
  - Single operation mode converts the selected inputs once for per trigger.
  - Continuous operation mode converts selected inputs continuously.
  - Discontinuous operation mode.
- Interrupt generation.
  - At the end of routine conversions.
  - Analog watchdog event.
- Conversion result threshold monitor function: analog watchdog.
- Oversampler.
  - 16-bit data register.
  - Oversampling ratio adjustable from 2x to 256x.
  - Programmable data shift up to 8-bits.

- Channel input range: $V_{SSA} \leq V_{IN} \leq V_{DDA}$.

## 10.3. Pins and internal signals

*Figure 10-1. ADC module block diagram* shows the ADC block diagram. *Table 10-1. ADC internal input signals* and *Table 10-2. ADC input pins definition* give the ADC internal signals and pins description.

**Table 10-1. ADC internal input signals**

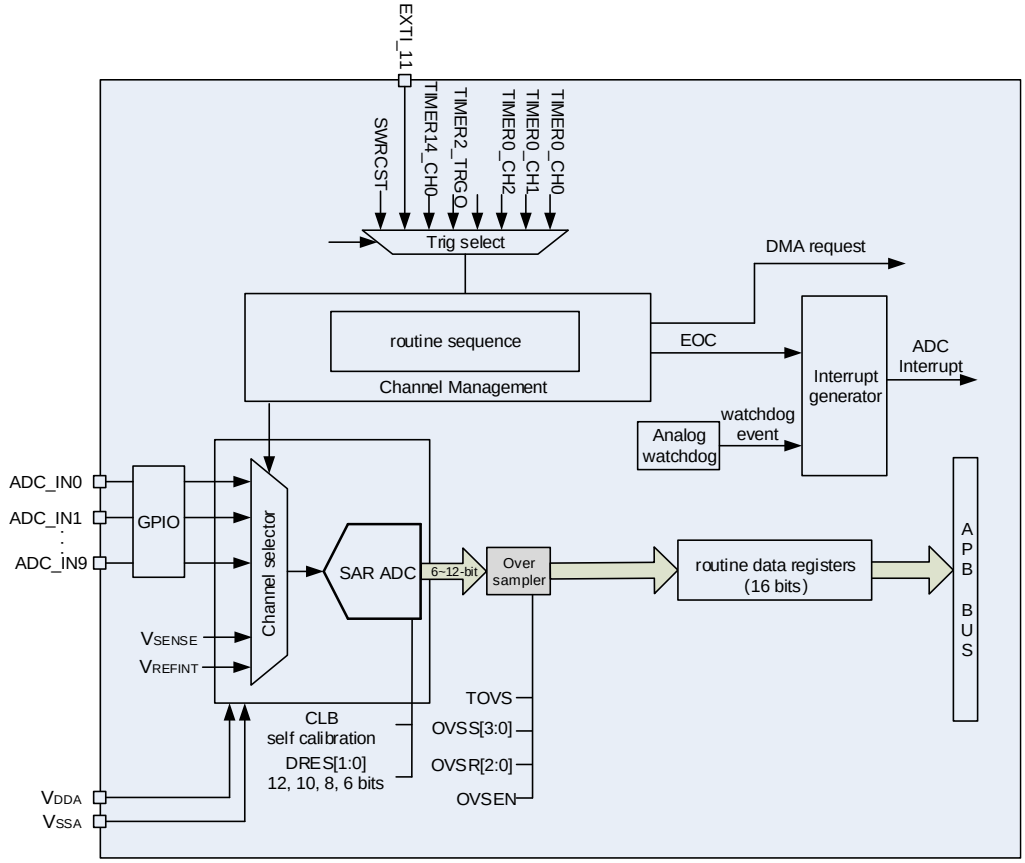
| Internal signal name | Description |
|---|---|
| $V_{SENSE}$ | Internal temperature sensor output voltage |
| $V_{REFINT}$ | Internal voltage reference output voltage |

**Table 10-2. ADC input pins definition**

| Name | Description |
|---|---|
| $V_{DDA}$ | Analog power supply equals to $V_{DD}$ |
| $V_{SSA}$ | Ground for analog power supply equals to $V_{SS}$ |
| ADCx_IN [9:0] | Up to 10 external channels |

## 10.4. Function overview

**Figure 10-1. ADC module block diagram**




### 10.4.1. Foreground calibration function

During the foreground calibration procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application can not use the ADC until the calibration is completed. The calibration should be performed before starting A/D conversion. The calibration is initiated by setting the CLB bit to 1. The CLB bit stays at 1 during the calibration sequence. It is then cleared by hardware as soon as the calibration is completed.

When the ADC operating conditions change (such as supply power voltage $V_{DDA}$, temperature and so on), it is recommended to re-run a calibration cycle.

The internal analog calibration can be reset by setting the RSTCLB bit in ADC_CTL1 register.

Calibration procedure by software:

1. Ensure ADCON=1.
2. Delay 14 CK_ADC to wait for ADC stability.
3. Set RSTCLB (optional).

4. Set CLB=1.
5. Wait for CLB =0.

### 10.4.2. Dual clock domain architecture

The ADC sub-module, with exception of the APB interface block, is feed by an ADC clock, which can be asynchronous and independent from the APB clock.

Application can reduce PLCK frequency for low power operation while still keeping optimum ADC performance.

Refer to RCU Section ***4.2.1*** for more information on generating this clock source.

### 10.4.3. ADC enable

The ADCON bit on the ADC_CTL1 register is the enable switch of the ADC module. The ADC module will keep in reset state if this bit is 0. For power saving, when this bit is reset, the analog sub-module will be put into power off mode. After ADC is enabled, you need delay $t_{su}$ time for sampling, the value of $t_{su}$ please refer to the chip datasheet.

### 10.4.4. Routine sequence

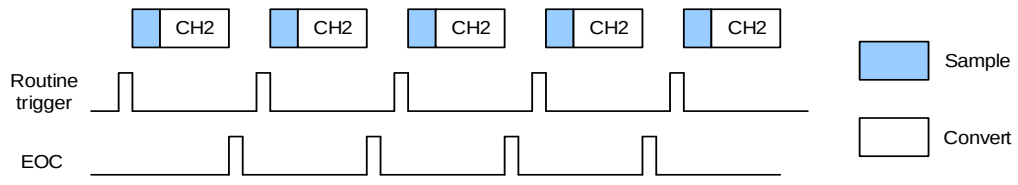
The channel management circuit can organize the sampling conversion channels into a sequence: routine sequence. The routine sequence supports up to 16 channels, and each channel is called routine channel. The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length. The ADC_RSQ0~ADC_RSQ2 registers specify the selected channels of the routine sequence operation modes

### 10.4.5. Operation mode

#### Single operation mode

In the single operation mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits in ADC_RSQ2 at a routine trigger. When the ADCON is 1, the ADC samples and converts a single channel, once the corresponding software trigger or external trigger is active.

**Figure 10-2. Single operation mode**




After the conversion of a single routine channel, the conversion data will be stored in the ADC_RDATA register, the EOC will be set. An interrupt will be generated if the EOCIE bit is

set.

Software procedure for single operation mode of a routine channel:

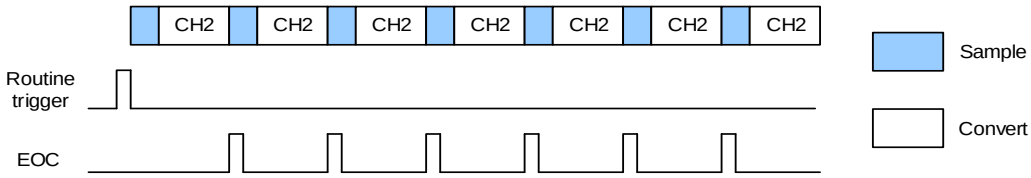
1. Make sure the DISRC, SM bits in the ADC_CTL0 register and CTN bit in the ADC_CTL1 register are reset;
2. Configure RSQ0 with the analog channel number.
3. Configure the ADC_SAMPTx register.
4. Configure the ETERC and ETSRC bits in the ADC_CTL1 register if in need.
5. Set the SWRCST bit, or generate an external trigger for the routine sequence.
6. Wait for the EOC flag to be set.
7. After a delay of one CK_ADC, read the converted date in the ADC_RDATA register.
8. Clear the EOC flag by writing 0.

**Note:** After EOC is set, a delay of one CK_ADC is required before reading the ADC conversion result.

### Continuous operation mode

The continuous operation mode will be enabled when the CTN bit in the ADC_CTL1 register is set. In this mode, the ADC performs conversion on the channel specified in the RSQ0[4:0]. When the ADCON has been set high, the ADC samples and converts specified channel, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA register.

**Figure 10-3. Continuous operation mode**




Software procedure for continuous operation mode on a routine channel:

1. Set the CTN bit in the ADC_CTL1 register.
2. Configure the RSQ0 with the analog channel number.
3. Configure the ADC_SAMPTx register.
4. Configure the ETERC and ETSRC bits in the ADC_CTL1 register if it is needed.
5. Set the SWRCST bit, or generate an external trigger for the routine sequence.
6. Wait the EOC flag to be set.
7. After a delay of one CK_ADC, read the converted date in the ADC_RDATA register.
8. Clear the EOC flag by writing 0 to it.
9. Repeat steps 6~8 as soon as the conversion is in need.

**Note:** After EOC is set, a delay of one CK_ADC is required before reading the ADC conversion result.

To get rid of checking, DMA can be used to transfer the converted data:

1. Set the CTN and DMA bits in the ADC_CTL1 register.

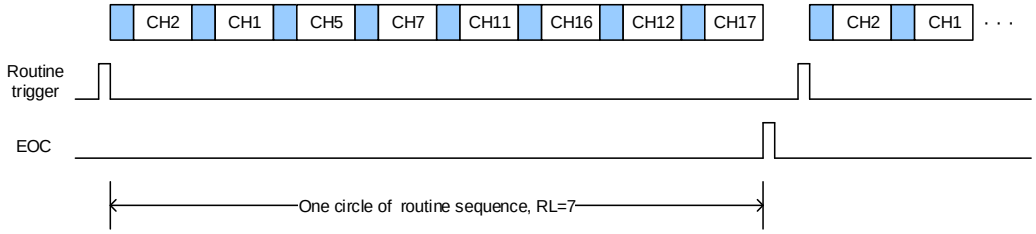
2. Configure the RSQ0 with the analog channel number.
3. Configure the ADC_SAMPTx register.
4. Configure the ETERC and ETSRC bits in the ADC_CTL1 register if it is needed.
5. Prepare the DMA module to transfer data from the ADC_RDATA.
6. Set the SWRCST bit, or generate an external trigger for the routine sequence.

### Scan operation mode

The scan operation mode will be enabled when the SM bit in the ADC_CTL0 register is set. In this mode, the ADC performs conversion on all channels with a specific routine sequence specified in the ADC_RSQ0~ADC_RSQ2 registers. When the ADCON has been set high, the ADC samples and converts specified channels one by one in the routine sequence till the end of the sequence, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA register. After conversion of the routine sequence, the EOC will be set. An interrupt will be generated if the EOCIE bit is set. The DMA bit in ADC_CTL1 register must be set when the routine sequence works in scan mode.

After conversion of a routine sequence, the conversion can be restarted automatically if the CTN bit in the ADC_CTL1 register is set.

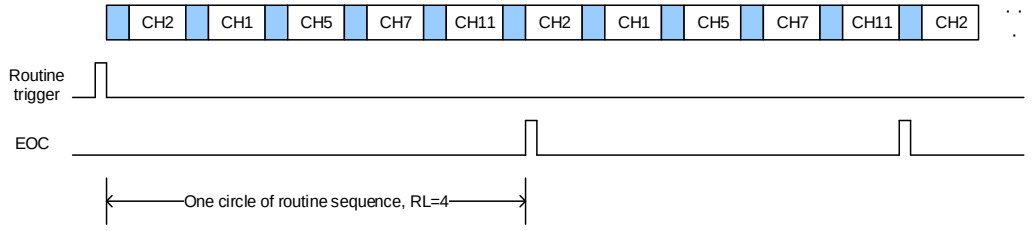
**Figure 10-4. Scan operation mode, continuous disable**




Software procedure for scan operation mode on a routine sequence:

1. Set the SM bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register;
2. Configure the ADC_RSQx and ADC_SAMPTx registers;
3. Configure the ETERC and ETSRC bits in the ADC_CTL1 register if it is needed;
4. Prepare the DMA module to transfer data from the ADC_RDATA;
5. Set the SWRCST bit, or generate an external trigger for the routine sequence;
6. Wait for the EOC flag to be set;
7. Clear the EOC flag by writing 0.

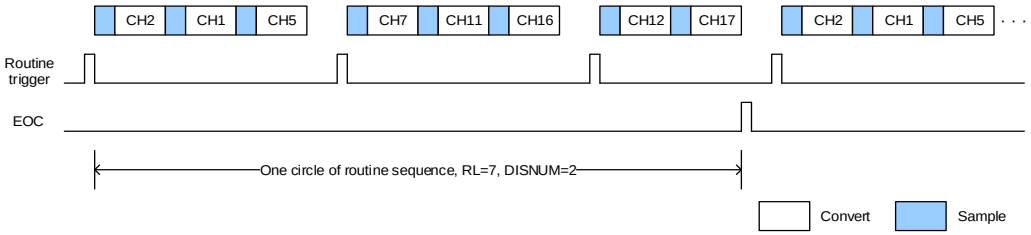
**Figure 10-5. Scan operation mode, continuous enable**



### Discontinuous operation mode

The discontinuous operation mode will be enabled when the DISRC bit in the ADC_CTL0 register is set. In this mode, the ADC performs a short sequence of n conversions (n does not exceed 8) which is part of the sequence of conversions selected in the ADC_RSQ0~ADC_RSQ2 registers. The value of n is configured by the DISNUM[2:0] bits in the ADC_CTL0 register. When the corresponding software trigger or external trigger is active, the ADC samples and converts the next n channels configured in the ADC_RSQ0~ADC_RSQ2 registers until all the channels of routine sequence are done. The EOC will be set after every circle of the routine sequence. An interrupt will be generated if the EOCIE bit is set.

**Figure 10-6. Discontinuous operation mode**




Software procedure for discontinuous operation mode on a routine sequence:

1. Set the DISRC bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register.
2. Configure the DISNUM [2:0] bits in the ADC_CTL0 register.
3. Configure the ADC_RSQx and ADC_SAMPTx registers.
4. Configure the ETERC and ETSRC bits in the ADC_CTL1 register if it is needed.
5. Prepare the DMA module to transfer data from the ADC_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an external trigger for the routine sequence.
7. Repeat step6 if in need.
8. Wait the EOC flag to be set.
9. Clear the EOC flag by writing 0 to it.

## 10.4.6. Conversion result threshold monitor function

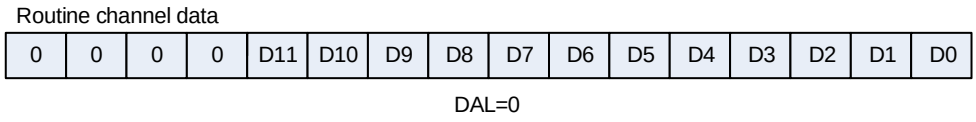
The analog watchdog is enabled when the RWDEN bit in the ADC_CTL0 register is set for routine sequence. This function is used to monitor whether the conversion result exceeds the set thresholds, and the WDE bit in ADC_STAT register will be set. An interrupt will be generated if the WDEIE bit is set. The ADC_WDHT and ADC_WDLT registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold values are independent of the alignment, which is specified by the DAL bit in the ADC_CTL1 register. One or more channels, which are selected by the RWDEN, WDSC and WDCHSEL [4:0] bits in ADC_CTL0 register, can be monitored by the analog watchdog.

### 10.4.7. Data storage mode

The alignment of data stored after conversion can be specified by DAL bit in the ADC_CTL1 register.

When in the most significant bit alignment, the 12/10/8-bit data are aligned on a half-word, while the 6-bit data are aligned on a byte basis as shown blew ***Figure 10-7. Data storage mode of 12-bit resolution***, ***Figure 10-8. Data storage mode of 10-bit resolution***, ***Figure 10-9. Data storage mode of 8-bit resolution*** and ***Figure 10-10. Data storage mode of 6-bit resolution***.

**Figure 10-7. Data storage mode of 12-bit resolution**



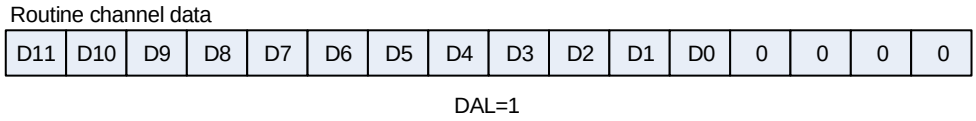
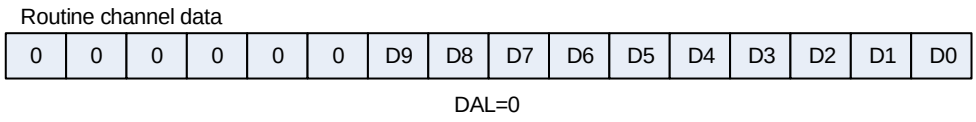




**Figure 10-8. Data storage mode of 10-bit resolution**



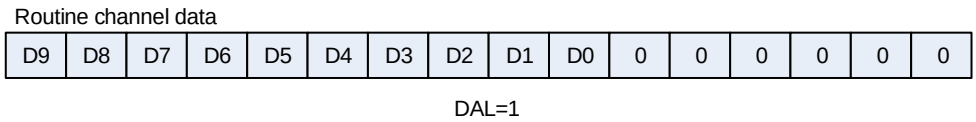
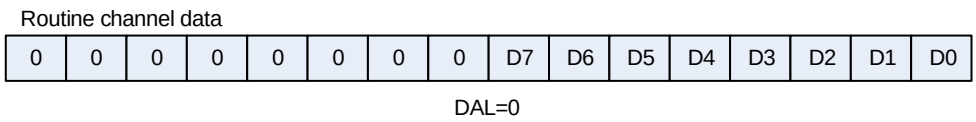




**Figure 10-9. Data storage mode of 8-bit resolution**



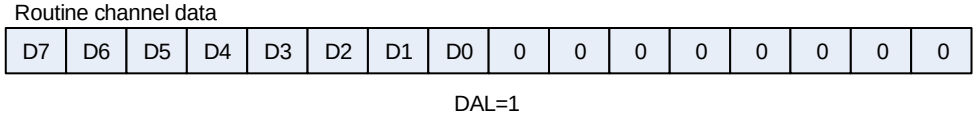
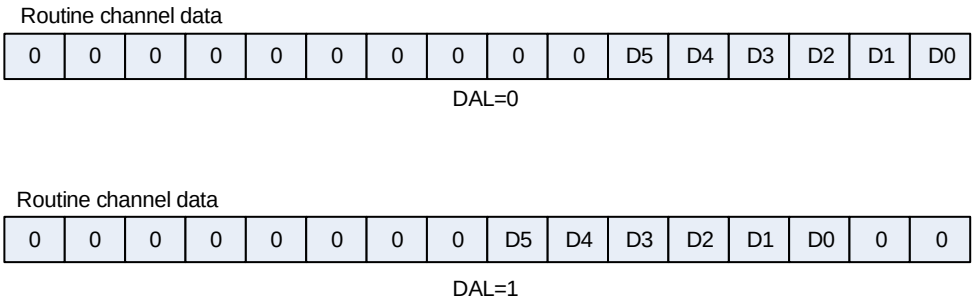



**Figure 10-10. Data storage mode of 6-bit resolution**




### 10.4.8. Sample time configuration

The number of CK_ADC cycles which is used to sample the input voltage can be specified by the SPTn [2:0] bits in the ADC_SAMPT0 and ADC_SAMPT1 registers. A different sampling time can be specified for each channel. For 12-bit resolution, the total sampling and conversion time is “sampling time + 12.5” CK_ADC cycles.

Example:

CK_ADC = 28MHz and sampling time is 1.5 cycles, the total conversion time is “1.5+12.5” CK_ADC cycles, that means 0.500us.

### 10.4.9. External trigger configuration

The conversion of routine sequence can be triggered by rising edge of external trigger inputs. The external trigger source of routine sequence is controlled by the ETSRC[2:0] bits in the ADC_CTL1 register.

**Table 10-3. External trigger source for ADC**

| ETSRC[2:0] | Trigger Source | Trigger Type |
|---|---|---|
| 000 | TIMER0_CH0 | Hardware trigger |
| 001 | TIMER0_CH1 | |
| 010 | TIMER0_CH2 | |
| 011 | reserved | |
| 100 | TIMER2_TRGO | |
| 101 | TIMER14_CH0 | |
| 110 | EXTI_11 | |
| 111 | SWRCST | Software trigger |

### 10.4.10. DMA request

The DMA request, which is enabled by the DMA bit in ADC_CTL1 register, is used to transfer data of routine sequence for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a routine channel. When this request is received, the DMA will transfer the converted data from the ADC_RDATA register to the destination

which is specified by the user.

### 10.4.11. ADC internal channels

When the TSVREN bit in ADC_CTL1 register is set, the temperature sensor channel (ADC_IN16) and $V_{REFINT}$ channel (ADC_IN17) are enabled. The temperature sensor can be used to measure the ambient temperature of the device. The sensor output voltage can be converted into a digital value by ADC. The sampling time for the temperature sensor is recommended to be set to at least 17.1μs. When this sensor is not in use, it can be set in power down mode by resetting the TSVREN bit.

The output voltage of the temperature sensor changes linearly with temperature. Because there is an offset, which is up to 45°C and varies from chip to chip due to the chip production process variation, the internal temperature sensor is more appropriate to detect temperature variations than absolute temperature. When it is used to detect accurate temperature, an external temperature sensor part should be used to calibrate the offset error.

The internal reference voltage ($V_{REFINT}$) provides a stable (bandgap) voltage output for the ADC and comparators. $V_{REFINT}$ is internally connected to the ADC_IN17 input channel.

To use the temperature sensor:

1. Configure the conversion sequence (ADC_IN16) and the sampling time(17.1μs) for the channel.
2. Enable the temperature sensor by setting the TSVREN bit in the ADC control register 1 (ADC_CTL1).
3. Start the ADC conversion by setting the ADCON bit or by the triggers.
4. Read the internal temperature sensor output voltage($V_{temperature}$), and get the temperature with the following equation:

$$\text{Temperature (°C)} = \{(V_{25} - V_{temperature}) / \text{Avg_Slope}\} + 25.$$

$V_{25}$: internal temperature sensor output voltage at 25°C, the typical value please refer to the datasheet.

Avg_Slope: average slope for curve between Temperature vs. internal temperature sensor output voltage, the typical value please refer to the datasheet.

### 10.4.12. Programmable resolution (DRES) - fast conversion mode

It is possible to obtain faster conversion time ($t_{ADC}$) by reducing the ADC resolution.

The resolution is configured by programming the DRES[1:0] bits in the ADC_CTL0 register. For applications that do not require high data accuracy, lower resolution allows faster conversion time. The DRES [1:0] bits must only be changed when the ADCON bit is reset. Lower resolution reduces the conversion time needed for the successive approximation steps as shown in ***Table 10-4. tCONV timings depending on resolution***.

**Table 10-4. $t_{CONV}$ timings depending on resolution**

| DRES[1:0] bits | $t_{CONV}$ (ADC clock cycles) | $t_{CONV}$(ns) at $f_{ADC}$=28MHz | $t_{SMPL}$(min) (ADC clock cycles) | $t_{ADC}$(ADC clock cycles) | $t_{ADC}$(ns) at $f_{ADC}$=28MHz |
|---|---|---|---|---|---|
| 12 | 12.5 | 446ns | 1.5 | 14 | 500ns |
| 10 | 10.5 | 375ns | 1.5 | 12 | 429ns |
| 8 | 8.5 | 304ns | 1.5 | 10 | 357ns |
| 6 | 6.5 | 232ns | 1.5 | 8 | 286ns |

## 10.4.13. On-chip hardware oversampling

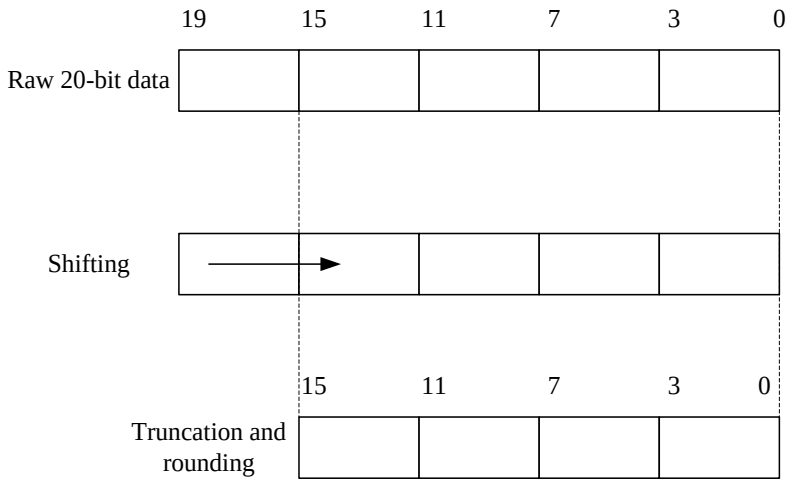
The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit. The on-chip hardware oversampling circuit is enabled by OVSEN bit in the ADC_OVSAMPCTL register.It provides a result with the following form, where N and M can be adjusted, and $D_{out}(n)$ is the n-th output digital signal of the ADC:

$$\text{Result} = \frac{1}{\text{M}} * \sum_{\text{n=0}}^{\text{n=N-1}} D_{OUT}(n) \quad (10\text{-}1)$$

The on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting.The oversampling ratio N is defined by the OVSR[2:0] bits in the ADC_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M means bit right shifting up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_OVSAMPCTL register.

Summation units can produce up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the data register.

**Figure 10-11. 20-bit to 16-bit result truncation**



**Note**: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

*Figure 10-12. A numerical example with 5-bit shifting and rounding* shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 10-12. A numerical example with 5-bit shifting and rounding**

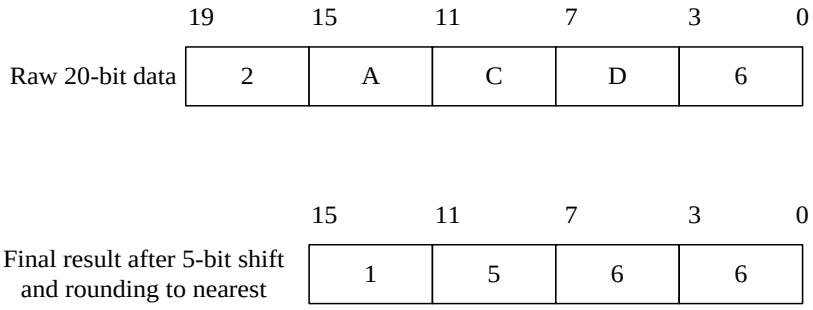



*Table 10-5. Maximum output results for N and M combimations (grayed values indicates truncation)* below gives the data format for the various N and M combinations, and the raw conversion data equals 0xFFF.

**Table 10-5. Maximum output results for N and M combimations (grayed values indicates truncation)**

| Oversampling ratio | Max Raw data | No-shift OVSS=0000 | 1-bit shift OVSS=0001 | 2-bit shift OVSS=0010 | 3-bit shift OVSS=0011 | 4-bit shift OVSS=0100 | 5-bit shift OVSS=0101 | 6-bit shift OVSS=0110 | 7-bit shift OVSS=0111 | 8-bit shift OVSS=1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2x | 0x1FFE | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F | 0x003F | 0x001F |
| 4x | 0x3FFC | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F | 0x003F |
| 8x | 0x7FF8 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F |
| 16x | 0xFFF0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF |
| 32x | 0x1FFE0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF |
| 64x | 0x3FFC0 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF |
| 128x | 0x7FF80 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF |
| 256x | 0xFFF00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF |

When compared to standard conversion mode, the conversion timings of oversampling mode do not change, and the sampling time remains equal throughout the oversampling sequence. New data is supplied every N conversions, and the equivalent delay is equal to:

$$N \times t_{ADC} = N \times (t_{SMPL} + \mathbf{t_{CONV}}) \quad (10\text{-}2)$$

### Oversampling work with ADC modes

Most of the ADC work modes are available when oversampling is enabled.

- Routine channels.
- ADC started by software or external triggers.
- Single or scan, continuous or discontinuous operation modes.
- Programmable sample time.
- Analog watchdog.

The oversampling configuration can only be changed when ADCON is reset. Make sure configuring the oversampling before setting ADCON to 1.

### 10.4.14. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for routine sequence.
- The analog watchdog event.

Separate interrupt enable bits are available for flexibility.

## 10.5. Register definition

ADC base address: 0x4001 2400

### 10.5.1. Status register (ADC_STAT)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 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 | | | | | | | | | | | STRC | Reserved | | EOC | WDE |
| | | | | | | | | | | | rc_w0 | | | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value |
| 4 | STRC | Start flag of routine sequence conversion<br>0: Conversion is not started<br>1: Conversion is started<br>Set by hardware when routine sequence conversion starts. Cleared by software writing 0 to it. |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | EOC | End flag of routine sequence conversion<br>0: No end of routine sequence conversion<br>1: End of routine sequence conversion<br>Set by hardware at the end of a routine sequence conversion.<br>Cleared by software writing 0 to it or by reading the ADC_RDATA register. |
| 0 | WDE | Analog watchdog event flag<br>0: No analog watchdog event<br>1: Analog watchdog event<br>Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT and ADC_WDHT registers.<br>Cleared by software writing 0 to it. |

### 10.5.2. Control register 0 (ADC_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | DRES [1:0] | | RWDEN | Reserved | | | | | | |
| | | | | | | rw | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISNUM [2:0] | | | Reserved | DISRC | Reserved | WDSC | SM | Reserved | WDEIE | EOCIE | WDCHSEL [4:0] | | | | |
| rw | | | | rw | | rw | rw | | rw | rw | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value |
| 25:24 | DRES[1:0] | ADC resolution<br>00: 12bit<br>01: 10bit<br>10: 8bit<br>11: 6bit |
| 23 | RWDEN | Routine channel analog watchdog enable<br>0: Analog watchdog disable<br>1: Analog watchdog enable |
| 22:16 | Reserved | Must be kept at reset value |
| 15:13 | DISNUM[2:0] | Number of conversions in discontinuous mode<br>The number of channels to be converted after a trigger will be DISNUM [2:0] +1 |
| 12 | Reserved | Must be kept at reset value |
| 11 | DISRC | Discontinuous mode on routine sequence<br>0: Discontinuous operation mode disable<br>1: Discontinuous operation mode enable |
| 10 | Reserved | Must be kept at reset value. |
| 9 | WDSC | When in scan mode, analog watchdog is effective on a single channel<br>0: All channels have analog watchdog function<br>1: A single channel has analog watchdog function |
| 8 | SM | Scan mode<br>0: Scan operation mode disable<br>1: Scan operation mode enable |
| 7 | Reserved | Must be kept at reset value. |
| 6 | WDEIE | Interrupt enable for WDE<br>0: Interrupt disable<br>1: Interrupt enable |
| 5 | EOCIE | Interrupt enable for EOC |

| | | |
|---|---|---|
| | | 0: Interrupt disable<br>1: Interrupt enable |
| 4:0 | WDCHSEL[4:0] | Analog watchdog channel select<br>00000: ADC channel 0<br>00001: ADC channel 1<br>00010: ADC channel 2<br>……<br>01000: ADC channel 8<br>01001: ADC channel9<br>01010: ADC channel 16<br>01011: ADC channel 17<br>Other values are reserved.<br>**Note:** ADC analog inputs Channel 16 and Channel 17 are internally connected to the temperature sensor and $V_{REFINT}$ analog inputs. |

### 10.5.3. Control register 1 (ADC_CTL1)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TSVREN | SWRCST | Reserved | ETERC | ETSRC [2:0] | | | Reserved |
| | | | | | | | | rw | rw | | rw | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DAL | Reserved | | DMA | Reserved | | | | RSTCLB | CLB | CTN | ADCON |
| | | | | rw | | | rw | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23 | TSVREN | Channel 16 and 17 enable of ADC.<br>0: Channel 16 and 17 of ADC disable<br>1: Channel 16 and 17 of ADC enable |
| 22 | SWRCST | Software start on conversion of routine sequence.<br>Set 1 on this bit starts the conversion of a routine sequence if ETSRC is 111. It is set by software and cleared by software or by hardware after the conversion starts. |
| 21 | Reserved | Must be kept at reset value. |
| 20 | ETERC | External trigger enable for routine sequence<br>0: External trigger for routine sequence disable<br>1: External trigger for routine sequence enable |
| 19:17 | ETSRC[2:0] | External trigger select for routine sequence |

| | | |
|---|---|---|
| | | 000: TIMER0 CH0<br>001: TIMER0 CH1<br>010: TIMER0 CH2<br>011: reserved<br>100: TIMER2 TRGO<br>101: TIMER14 CH0<br>110: EXTI line 11<br>111: SWRCST |
| 16:12 | Reserved | Must be kept at reset value |
| 11 | DAL | Data alignment<br>0: LSB alignment<br>1: MSB alignment |
| 10:9 | Reserved | Must be kept at reset value |
| 8 | DMA | DMA request enable.<br>0: DMA request disable<br>1: DMA request enable |
| 7:4 | Reserved | Must be kept at reset value |
| 3 | RSTCLB | Reset calibration<br>This bit is set by software and cleared by hardware after the calibration registers are initialized.<br>0: Calibration register initialization done.<br>1: Calibration register initialization starts |
| 2 | CLB | ADC calibration<br>0: Calibration done<br>1: Calibration start |
| 1 | CTN | Continuous mode<br>0: Continuous operation mode disable<br>1: Continuous operation mode enable |
| 0 | ADCON | ADC ON. The ADC will be waked up when this bit is changed from low to high and take a stabilization time. When this bit is high and “1” is written to it with other bits of this register unchanged, the conversion will start.<br>0: ADC disable and power down<br>1: ADC enable |

### 10.5.4. Sample time register 0 (ADC_SAMPT0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | SPT17[2:0] | | | SPT16[2:0] | | | Reserved | |
| | | | | | | | | | rw | | | rw | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:21 | SPT17[2:0] | Refer to SPT16[2:0] description |
| 20:18 | SPT16[2:0] | Channel sampling time<br>000: channel sampling time is 1.5 cycles<br>001: channel sampling time is 7.5 cycles<br>010: channel sampling time is 13.5 cycles<br>011: channel sampling time is 28.5 cycles<br>100: channel sampling time is 41.5 cycles<br>101: channel sampling time is 55.5 cycles<br>110: channel sampling time is 71.5 cycles<br>111: channel sampling time is 239.5 cycles |
| 17:0 | Reserved | Must be kept at reset value |

## 10.5.5. Sample time register 1 (ADC_SAMPT1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SPT9[2:0] | | | SPT8[2:0] | | | SPT7[2:0] | | | SPT6[2:0] | | | SPT5[2:1] | |
| | | rw | | | rw | | | rw | | | rw | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPT5[0] | | SPT4[2:0] | | | SPT3[2:0] | | | SPT2[2:0] | | | SPT1[2:0] | | | SPT0[2:0] | |
| rw | | rw | | | rw | | | rw | | | rw | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:27 | SPT9[2:0] | Refer to SPT0[2:0] description |
| 26:24 | SPT8[2:0] | Refer to SPT0[2:0] description |
| 23:21 | SPT7[2:0] | Refer to SPT0[2:0] description |

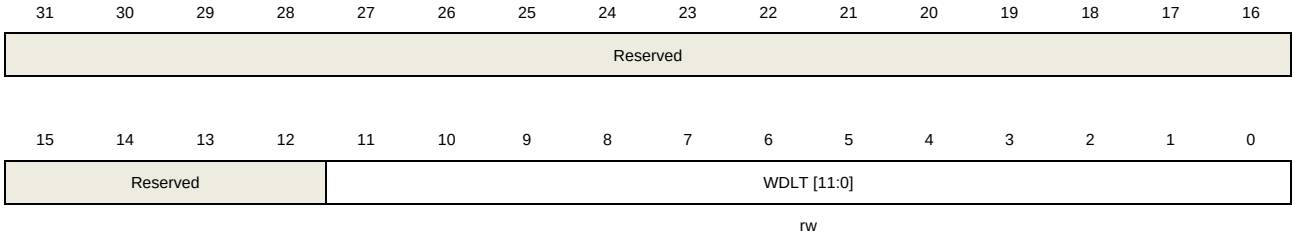
| | | |
|---|---|---|
| 20:18 | SPT6[2:0] | Refer to SPT0[2:0] description |
| 17:15 | SPT5[2:0] | Refer to SPT0[2:0] description |
| 14:12 | SPT4[2:0] | Refer to SPT0[2:0] description |
| 11:9 | SPT3[2:0] | Refer to SPT0[2:0] description |
| 8:6 | SPT2[2:0] | Refer to SPT0[2:0] description |
| 5:3 | SPT1[2:0] | Refer to SPT0[2:0] description |
| 2:0 | SPT0[2:0] | Channel sampling time<br>000: channel sampling time is 1.5 cycles<br>001: channel sampling time is 7.5 cycles<br>010: channel sampling time is 13.5 cycles<br>011: channel sampling time is 28.5 cycles<br>100: channel sampling time is 41.5 cycles<br>101: channel sampling time is 55.5 cycles<br>110: channel sampling time is 71.5 cycles<br>111: channel sampling time is 239.5 cycles |

### 10.5.6. Watchdog low threshold register (ADC_WDLT)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | WDLT[11:0] | Low threshold for analog watchdog<br>These bits define the low threshold for the analog watchdog. |

### 10.5.7. Routine sequence register 0 (ADC_RSQ0)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | RL [3:0] | | | | RSQ15[4:1] | | | |
| | | | | | | | | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSQ15[0] | RSQ14[4:0] | | | | | RSQ13[4:0] | | | | | RSQ12[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:20 | RL[3:0] | Routine sequence length<br>The total number of conversion in routine sequence equals to RL[3:0] +1. |
| 19:15 | RSQ15[4:0] | Refer to RSQ0[4:0] description |
| 14:10 | RSQ14[4:0] | Refer to RSQ0[4:0] description |
| 9:5 | RSQ13[4:0] | Refer to RSQ0[4:0] description |
| 4:0 | RSQ12[4:0] | Refer to RSQ0[4:0] description |

## 10.5.8. Routine sequence register 1 (ADC_RSQ1)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | RSQ11[4:0] | | | | | RSQ10[4:0] | | | | | RSQ9[4:1] | | | |
| | | rw | | | | | rw | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSQ9[0] | RSQ8[4:0] | | | | | RSQ7[4:0] | | | | | RSQ6[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

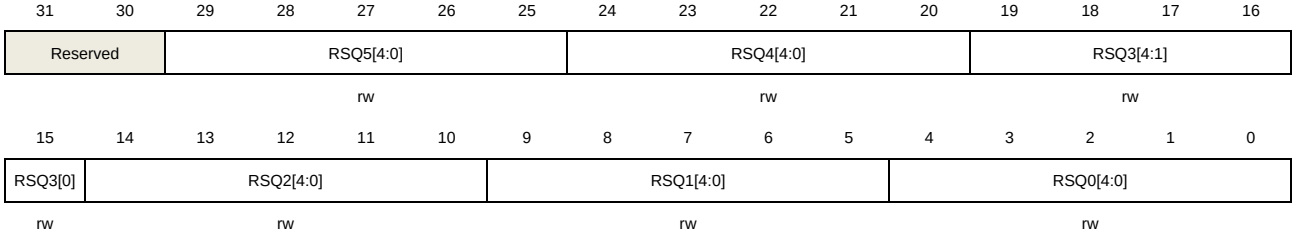
| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:25 | RSQ11[4:0] | Refer to RSQ0[4:0] description |
| 24:20 | RSQ10[4:0] | Refer to RSQ0[4:0] description |
| 19:15 | RSQ9[4:0] | Refer to RSQ0[4:0] description |
| 14:10 | RSQ8[4:0] | Refer to RSQ0[4:0] description |
| 9:5 | RSQ7[4:0] | Refer to RSQ0[4:0] description |
| 4:0 | RSQ6[4:0] | Refer to RSQ0[4:0] description |

## 10.5.9. Routine sequence register 2 (ADC_RSQ2)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



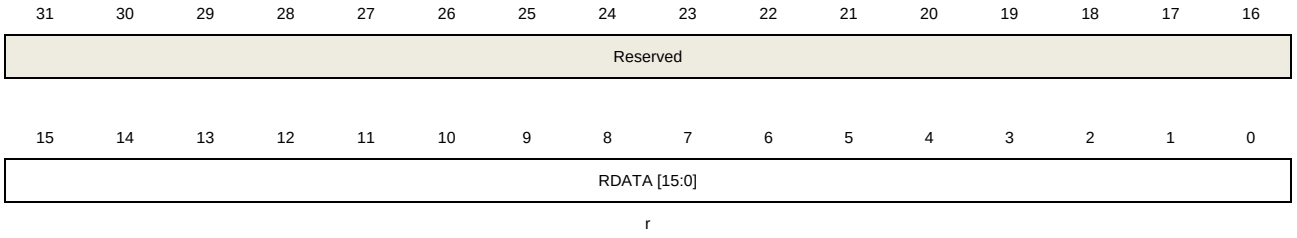

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:25 | RSQ5[4:0] | Refer to RSQ0[4:0] description |
| 24:20 | RSQ4[4:0] | Refer to RSQ0[4:0] description |
| 19:15 | RSQ3[4:0] | Refer to RSQ0[4:0] description |
| 14:10 | RSQ2[4:0] | Refer to RSQ0[4:0] description |
| 9:5 | RSQ1[4:0] | Refer to RSQ0[4:0] description |
| 4:0 | RSQ0[4:0] | The channel number (0..9, 16, 17) is written to these bits to select a channel as the nth conversion in the routine sequence. |

## 10.5.10. Routine data register (ADC_RDATA)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | RDATA[15:0] | Routine channel data<br>These bits contain the conversion result from routine channel, which is read only. |

### 10.5.11. Oversampling control register (ADC_OVSAMPCTL)

Address offset: 0x80

Reset value: 0x0000_0000

This register has to be accessed by word(32-bit).

| 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 | | | | | | TOVS | OVSS[3:0] | | | | OVSR[2:0] | | | Reserved | OVSEN |
| | | | | | | rw | rw | | | | rw | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9 | TOVS | Triggered Oversampling<br>This bit is set and cleared by software.<br>0: All oversampling conversions for a channel are done consecutively after a trigger<br>1: Each conversion needs a trigger for a oversampled channel and the number of triggers is determined by the oversampling ratio(OVSR[2:0]).<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 8:5 | OVSS [3:0] | Oversampling shift<br>These bits are set and cleared by software.<br>0000: No shift<br>0001: Shift 1 bit<br>0010: Shift 2 bits<br>0011: Shift 3 bits<br>0100: Shift 4 bits<br>0101: Shift 5 bits<br>0110: Shift 6 bits<br>0111: Shift 7 bits<br>1000: Shift 8 bits<br>Other: reserved<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 4:2 | OVSR [2:0] | Oversampling ratio<br>This bit filed defines the number of oversampling ratio.<br>000: 2x<br>001: 4x<br>010: 8x<br>011: 16x<br>100: 32x |

| | | |
|---|---|---|
| | | 101: 64x<br>110: 128x<br>111: 256x<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 1 | Reserved | Must be kept at reset value |
| 0 | OVSEN | Oversampling enable<br>This bit is set and cleared by software.<br>0: Oversampling disabled<br>1: Oversampling enabled<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |

# 11. Comparator (CMP)

## 11.1. Overview

The general purpose CMP can work either standalone (all terminal are available on I / Os) or together with the timers.

It can be used to wake up the MCU from low-power mode by an analog signal, provide a trigger source when an analog signal is in a certain condition.

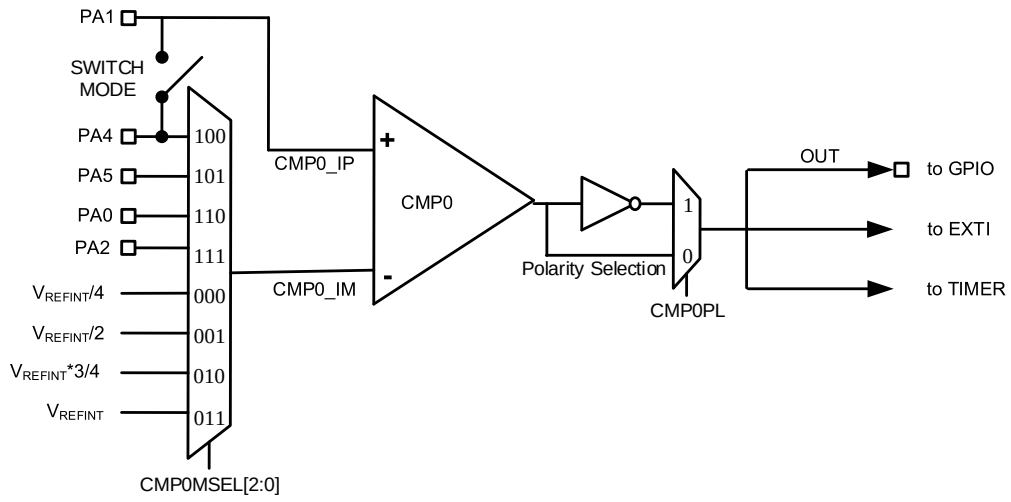
## 11.2. Characteristics

- Rail-to-rail comparators.
- Configurable hysteresis.
- Configurable speed and consumption.
- Configurable analog input source.
  - Multiplexed I / O pins.
  - The whole or sub-multiple values of internal reference voltage.
- Outputs to I / O.
- Outputs to timers for triggering.
- Outputs to EXTI.

## 11.3. Function overview

The block diagram of CMP is shown below.

**Figure 11-1. CMP block diagram**




**Note**: $V_{REFINT}$ is 1.2V.

### 11.3.1. CMP clock

The clock of the CMP which is connected to APB bus, is synchronous with PCLK. It shares the common reset and clock enable bits with SYSCFG.

### 11.3.2. CMP I / O configuration

These I / Os must be configured in analog mode in the GPIOs registers before they are selected as CMP inputs.

The CMP output can be redirected internally and externally simultaneously.

Refer to pin definitions in datasheet, and the CMP output can be connected to the corresponding I/O port via the alternate function of the GPIO.

CMP output internally connect to the TIMER and the connections between them are as follows:

- CMP output to the TIMER input channel.
- CMP output to the TIMER break.
- CMP output to the TIMER OCPRE_CLR.

In order to work even in Deep-sleep mode, the polarity selection logic and the output redirection to the port work independently from PCLK.

*Table 11-1. CMP inputs and outputs summary* details the inputs and outputs of the CMP.

**Table 11-1. CMP inputs and outputs summary**

| | CMP |
|---|---|
| **CMP non inverting inputs connected to I / Os** | PA1<br>PA4 |
| **CMP inverting inputs connected to I / Os** | PA0<br>PA2<br>PA4<br>PA5 |
| **CMP inverting inputs connected to internal signals** | $V_{REFINT}$/4<br>$V_{REFINT}$/2<br>$V_{REFINT}$*3/4<br>$V_{REFINT}$ |
| **CMP outputs connected to I / Os** | PA0<br>PA6<br>PA11 |
| **CMP outputs connected to EXTI** | • |

| | CMP |
|---|---|
| **CMP outputs connected to internal signals** | TIMER0_CH0<br>TIMER2_CH0<br>TIMER0 OCPRE_CLR<br>TIMER2 OCPRE_CLR |
| **CMP outputs(motor control protection)** | TIMER0 BREAK |

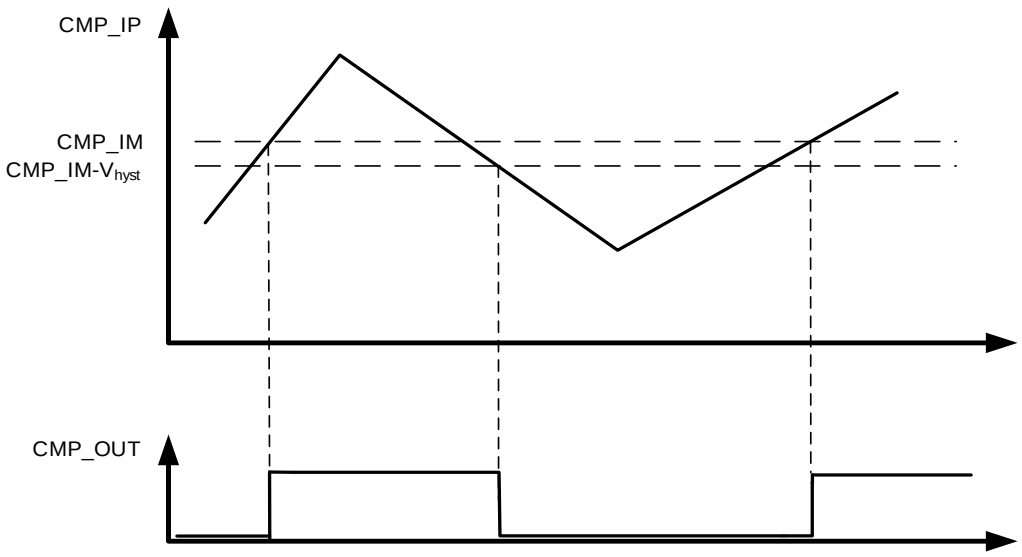
### 11.3.3. CMP operating mode

For a given application, there is a trade-off between the CMP power consumption versus propagation delay, which is adjusted by configuring bits CMPxM[1:0] in CMPx_CS register. The CMP works fastest with highest power consumption when CMPxM[1:0] = 2'b00, while works slowest with lowest power consumption when CMPxM[1:0] = 2'b11.

### 11.3.4. CMP hysteresis

In order to avoid spurious output transitions that caused by the noise signal, a programmable hysteresis is designed to force the hysteresis value by configuring CMPx_CS register. This function could be shut down if it is unnecessary.

**Figure 11-2. CMP hysteresis**




### 11.3.5. CMP register write protection

The CMP control and status register (CMPx_CS) can be protected from writing by setting CMPxLK bit to 1. The CMPx_CS register, including the CMPxLK bit will be read-only, and

can only be reset by the MCU reset.

### 11.3.6. CMP interrupt

The CMP output is connected to the EXTI and the EXTI line is exclusive to CMP. With this function, CMP can generate either interrupt or event which could be used to exit from low-power mode.

## 11.4. Register definition

CMP base address：0x4001 001C

### 11.4.1. CMP Control / status register (CMPx_CS)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP0LK | CMP0O | CMP0HST[1:0] | | CMP0PL | CMP0OSEL[2:0] | | | Reserved | CMP0MSEL[2:0] | | | CMP0M[1:0] | | CMP0SW | CMP0EN |
| rwo | r | rw | | rw | rw | | | | rw | | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | CMP0LK | CMP0 lock<br>This bit can set all control bits of CMP0 as read-only. It can only be set once by software and cleared by a system reset.<br>0: CMPx_CS[15:0] bits are read-write<br>1: CMPx_CS[15:0] bits are read-only |
| 14 | CMP0O | CMP0 output state<br>This bit is a copy of CMP0 output state, which is read only.<br>0: Non-inverting input below inverting input and the output is low<br>1: Non-inverting input above inverting input and the output is high |
| 13:12 | CMP0HST[1:0] | CMP0 hysteresis<br>These bits are used to control the hysteresis level.<br>00: No hysteresis<br>01: Low hysteresis<br>10: Medium hysteresis<br>11: High hysteresis |
| 11 | CMP0PL | Polarity of CMP0 output<br>This bit is used to select the polarity of CMP0 output.<br>0: Output is not inverted<br>1 : Output is inverted |
| 10:8 | CMP0OSEL[2:0] | CMP0 output selection<br>These bits are used to select the destination of the CMP0 output. |

| | | |
|---|---|---|
| | | 000: No selection<br>001: TIMER0 break input<br>010: TIMER0 CH0 input capture<br>011: TIMER0 OCPRE_CLR input<br>100: Reserved<br>101: Reserved<br>110: TIMER2 CH0 input capture<br>111: TIMER2 OCPRE_CLR input<br><br>**Note:** It is recommended to enable CMP first, and then configure the timer channel, when using TIMER to capture the output signal of the comparator. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CMP0MSEL[2:0] | CMP0_IM input selection<br>These bits are used to select the source connected to the CMP0_IM input of the CMP0.<br>000: $V_{REFINT}/4$<br>001: $V_{REFINT}/2$<br>010: $V_{REFINT}*3/4$<br>011: $V_{REFINT}$<br>100: PA4<br>101: PA5<br>110: PA0<br>111: PA2 |
| 3:2 | CMP0M[1:0] | CMP0 mode<br>These bits are used to control the operating mode.<br>00: High speed / full power<br>01: Medium speed / medium power<br>10: Low speed / low power<br>11: Very-low speed / ultra-low power |
| 1 | CMP0SW | CMP0 switch<br>This bit is used to close a switch between CMP0 non-inverting input on PA1 and PA4 I / O.<br>0: Switch open<br>1: Switch closed |
| 0 | CMP0EN | CMP0 enable<br>0: CMP0 disabled<br>1: CMP0 enabled |

# 12. Watchdog timer (WDGT)

The watchdog timer (WDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. There are two watchdog timer peripherals in the chip: free watchdog timer (FWDGT) and window watchdog timer (WWDGT). They offer a combination of a high safety level, flexibility of use and timing accuracy. Both watchdog timers are offered to resolve malfunctions of software.

The watchdog timer will generate a reset when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

## 12.1. Free watchdog timer (FWDGT)

### 12.1.1. Overview

The Free watchdog timer (FWDGT) has free clock source (IRC40K). Thereupon the FWDGT can operate even if the main clock fails. It's suitable for the situation that requires an independent environment and lower timing accuracy.

The free watchdog timer causes a reset when the internal down counter reaches 0 or the counter is refreshed when the value of the counter is greater than the window register value. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

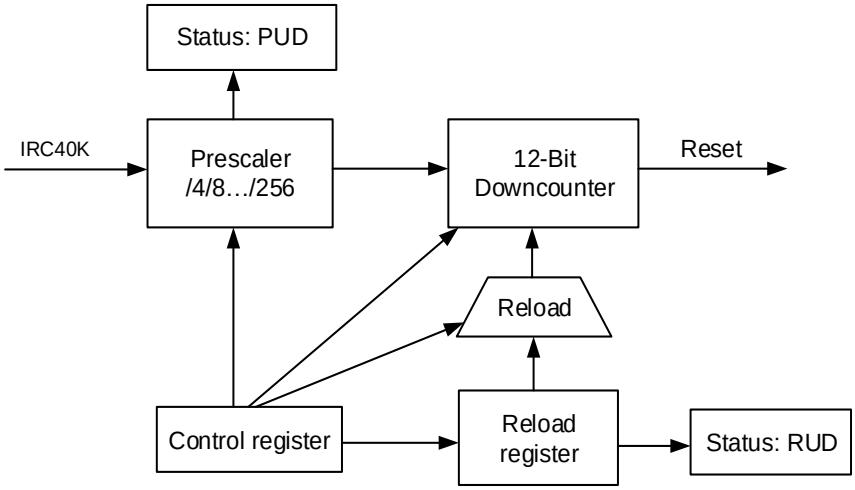
### 12.1.2. Characteristics

- Free-running 12-bit down counter.
- Generate reset in two conditions when FWDGT is enabled:
  - Reset when the counter reached 0.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Free clock source，FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog bit, automatically start the FWDGT or not when power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

### 12.1.3. Function overview

The free watchdog consists of an 8-stage prescaler and a 12-bit down counter. ***Figure 12-1. Free watchdog block diagram*** shows the functional block of the free watchdog module

**Figure 12-1. Free watchdog block diagram**




The free watchdog is enabled by writing the value (0xCCCC) to the control register (FWDGT_CTL), then counter starts counting down. When the counter reaches the value (0x000), there will be a reset.

The counter can be reloaded by writing the value (0xAAAA) to the FWDGT_CTL register at any time. The reload value comes from the FWDGT_RLD register. The software can prevent the watchdog reset by reloading the counter before the counter reaches the value (0x000).

By setting the appropriate window in the FWDGT_WND register, the FWDGT can also work as a window watchdog timer. A reset will occur if the reload operation is performed while the counter is greater than the value stored in the window register (FWDGT_WND). The default value of the FWDGT_WND is 0x0000 0FFF, so if it is not updated, the window option is disabled. A reload operation is performed in order to reset the downcounter to the FWDGT_RLD value and the prescaler counter to generate the next reload, as soon as the window value is changed.

The free watchdog can automatically start at power on when the hardware free watchdog bit in the device option bits is set. To avoid reset, the software should reload the counter before the counter reaches 0x000.

The FWDGT_PSC register and the FWDGT_RLD register are write protected. Before writing these registers, the software should write the value (0x5555) to the FWDGT_CTL register. These registers will be protected again by writing any other value to the FWDGT_CTL register. When an update operation of the prescaler register (FWDGT_PSC) or the reload value register (FWDGT_RLD) is ongoing, the status bits in the FWDGT_STAT register are set.

If the FWDGT_HOLD bit in DBG module is cleared, the FWDGT continues to work even the Cortex®-M23 core halted (Debug mode). The FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

**Table 12-1. Min/max FWDGT timeout period at 40 kHz (IRC40K)**

| Prescaler divider | PSC[2:0] bits | Min timeout (ms) RL[11:0]= 0x000 | Max timeout (ms) RL[11:0]= 0xFFF |
|---|---|---|---|
| 1 / 4 | 000 | 0.025 | 409.525 |
| 1 / 8 | 001 | 0.025 | 819.025 |
| 1 / 16 | 010 | 0.025 | 1638.025 |
| 1 / 32 | 011 | 0.025 | 3276.025 |
| 1 / 64 | 100 | 0.025 | 6552.025 |
| 1 / 128 | 101 | 0.025 | 13104.025 |
| 1 / 256 | 110 or 111 | 0.025 | 26208.025 |

The FWDGT timeout can be more accurately by calibrating the IRC40K.

**Note:** 1. When after the execution of watchdog reload operation, if the MCU needs enter the deepsleep / standby mode immediately, (more than 3) IRC40K clock intervals must be inserted in the middle of reload and deepsleep / standby mode commands by software setting. 2. There must be an interval of 7 or more IRC32K clock cycles between two sonsecutive watchdog reload operations.

## 12.1.4. Register definition

FWDGT base address: 0x4000 3000

### Control register (FWDGT_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit) 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMD[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CMD[15:0] | Write only. Several different functions are realized by writing these bits with different values.<br>0x5555: Disable the FWDGT_PSC, FWDGT_RLD and FWDGT_WND write protection.<br>0xCCCC: Start the free watchdog timer counter. When the counter reduces to 0, the free watchdog generates a reset.<br>0xAAAA: Reload the counter. |

### Prescaler register (FWDGT_PSC)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit) 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 | | | | | | | | | | | | | PSC[2:0] | | |
| | | | | | | | | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2:0 | PSC[2:0] | Free watchdog timer prescaler selection. Write 0x5555 in the FWDGT_CTL register before writing these bits. During a write operation to this register, the PUD |

bit in the FWDGT_STAT register is set and the value read from this register is invalid.

000: 1 / 4

001: 1 / 8

010: 1 / 16

011: 1 / 32

100: 1 / 64

101: 1 / 128

110: 1 / 256

111: 1 / 256

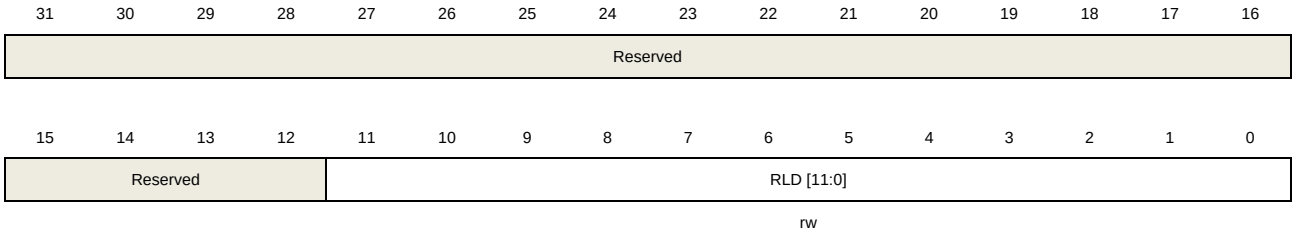
If several prescaler values are used by the application, it is mandatory to wait until PUD bit has been reset before changing the prescaler value. If the prescaler value has been updated, it is not necessary to wait until PUD has been reset before continuing code execution (Before entering low-power mode, it is necessary to wait until PUD is reset).

### Reload register (FWDGT_RLD)

Address offset: 0x08

Reset value: 0x0000 0FFF

This register can be accessed by half-word(16-bit) or word(32-bit) access.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | RLD[11:0] | Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT conter with the RLD value.<br>These bits are write-protected. Write 0X5555 to the FWDGT_CTL register before writing these bits. During a write operation to this register, the RUD bit in the FWDGT_STAT register is set and the value read from this register is invalid.<br>If several reload values are used by the application, it is mandatory to wait until RUD bit has been reset before changing the reload value. If the reload value has been updated, it is not necessary to wait until RUD has been reset before continuing code execution (Before entering low-power mode, it is necessary to wait until PUD is reset). |

### Status register (FWDGT_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit) 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 | | | | | | | | | | | | | WUD | RUD | PUD |
| | | | | | | | | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | WUD | Watchdog counter window value update.<br>When a write operation to FWDGT_WND register ongoing, this bit is set and the value read from FWDGT_WND register is invalid. |
| 1 | RUD | Free watchdog timer counter reload value update.<br>During a write operation to FWDGT_RLD register, this bit is set and the value read from FWDGT_RLD register is invalid. |
| 0 | PUD | Free watchdog timer prescaler value update.<br>During a write operation to FWDGT_PSC register, this bit is set and the value read from FWDGT_PSC register is invalid. |

### Window register (FWDGT_WND)

Address offset: 0x10

Reset value: 0x0000 0FFF

This register can be accessed by half-word(16-bit) or word(32-bit) 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 | | | | WND[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | WND[11:0] | Watchdog counter window value. These bits are used to contain the high limit of the window value to be compared to the downcounter. A reset will occur if the reload operation is performed while the counter is greater than the value stored in |

this register. The WUD bit in the FWDGT_STAT register must be reset in order to be able to change the reload value.

These bits are write protected. Write 5555h in the FWDGT_CTL register before writing these bits.

If several window values are used by the application, it is mandatory to wait until WUD bit has been reset before changing the window value. However, after updating the window value it is not necessary to wait until WUD is reset before continuing code execution except in case of low-power mode entry (Before entering low-power mode, it is necessary to wait until PUD is reset).

## 12.2. Window watchdog timer (WWDGT)

### 12.2.1. Overview

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. After the window watchdog timer starts, the value of down counter reduces progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit has been cleared). The watchdog timer also causes a reset when the counter is refreshed before the counter reached the window register value. So the software should refresh the counter in a limited window. The window watchdog timer generates an early wakeup status flag when the counter reaches 0x40 or refreshes before the counter reaches the window value. Interrupt occurs if it is enabled.

The window watchdog timer clock is prescaled from the APB1 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

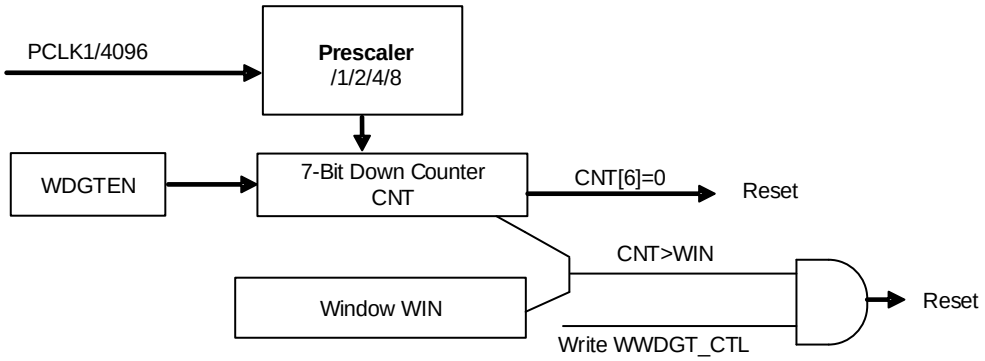
### 12.2.2. Characteristics

- Programmable free-running 7-bit down counter.
- Generate reset in two conditions when WWDGT is enabled:
  - Reset when the counter reached 0x3F.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Early wakeup interrupt (EWI): if the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40 or refreshes before it reaches the window value.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

### 12.2.3. Function overview

If the window watchdog timer is enabled (set the WDGTEN bit in the WWDGT_CTL), the watchdog timer cause a reset when the counter reaches 0x3F (the CNT[6] bit has been cleared), or the counter is refreshed before the counter reaches the window register value.

**Figure 12-2. Window watchdog timer block diagram**



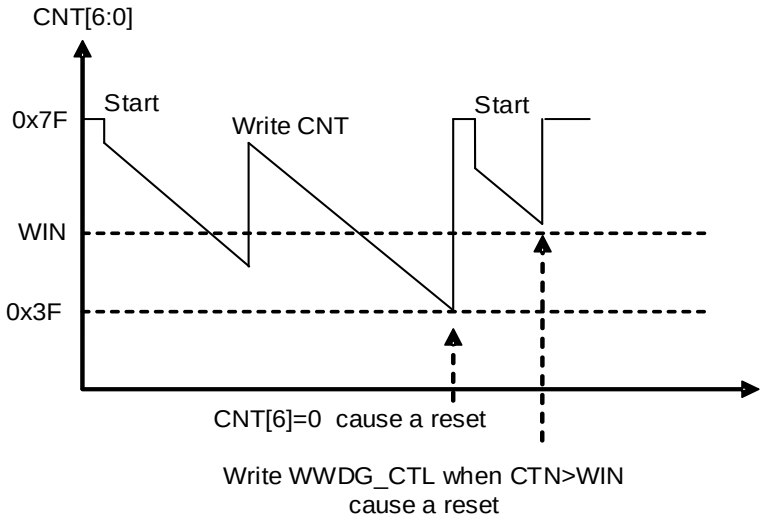

The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDGTEN bit in the WWDGT_CTL register. When window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F(it implies that the CNT[6] bit should be set). The CNT[5:0] determine the maximum time interval between two reloading. The count down speed depends on the APB1 clock and the prescaler (PSC[1:0] bits in the WWDGT_CFG register).

The WIN[6:0] bits in the configuration register (WWDGT_CFG) specifies the window value. The software can prevent the reset event by reloading the down counter. The counter value is less than the window value and greater than 0x3F, otherwise the watchdog causes a reset.

The early wakeup interrupt (EWI) is enabled by setting the EWIE bit in the WWDGT_CFG register, and the interrupt will be generated when the counter reaches 0x40 or the counter is refreshed before it reaches the window value. The software can do something such as communication or data logging in the interrupt service routine (ISR) in order to analyse the reason of software malfunctions or save the important data before resetting the device. Moreover the software can reload the counter in ISR to manage a software system check and so on. In this case, the WWDGT will never generate a WWDGT reset but can be used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDGT_STAT register.

**Figure 12-3. Window watchdog timing diagram**




Calculate the WWDGT timeout by using the formula below.

$$t_{WWDGT} = t_{PCLK1} \times 4096 \times 2^{PSC} \times (CNT[5:0] + 1) \quad (ms) \tag{12-1}$$

where:

$t_{WWDGT}$: WWDGT timeout

$t_{PCLK1}$: APB1 clock period measured in ms

The table below shows the minimum and maximum values of the $t_{WWDGT}$.

**Table 12-2. Min-max timeout value at 72 MHz ($f_{PCLK1}$)**

| Prescaler divider | PSC[1:0] | Min timeout value CNT[6:0] =0x40 | Max timeout value CNT[6:0]=0x7F |
|---|---|---|---|
| 1/1 | 00 | 56 µs | 3.64 ms |
| 1/2 | 01 | 113 µs | 7.28 ms |
| 1/4 | 10 | 227 µs | 14.56 ms |
| 1/8 | 11 | 455 µs | 29.12 ms |

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex®-M23 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

## 12.2.4. Register definition

WWDGT base address: 0x4000 2C00

### Control register (WWDGT_CTL)

Address offset: 0x00
Reset value: 0x0000 007F

This register can be accessed by half-word(16-bit) or word(32-bit)

| 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 | | | | | | | | WDGTEN | CNT[6:0] | | | | | | |
| | | | | | | | | rs | rw | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | WDGTEN | Start the Window watchdog timer. Cleared by a hardware reset. Writing 0 has no effect.<br>0: Window watchdog timer disabled<br>1: Window watchdog timer enabled |
| 6:0 | CNT[6:0] | The value of the watchdog timer counter. A reset occur when the value of this counter decreases from 0x40 to 0x3F. When the value of this counter is greater than the window value, writing this counter also causes a reset. |

### Configuration register (WWDGT_CFG)

Address offset: 0x04
Reset value: 0x0000 007F

This register can be accessed by half-word(16-bit) or word(32-bit)

| 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 | | | | | | EWIE | PSC[1:0] | | WIN[6:0] | | | | | | |
| | | | | | | rs | rw | | rw | | | | | | |

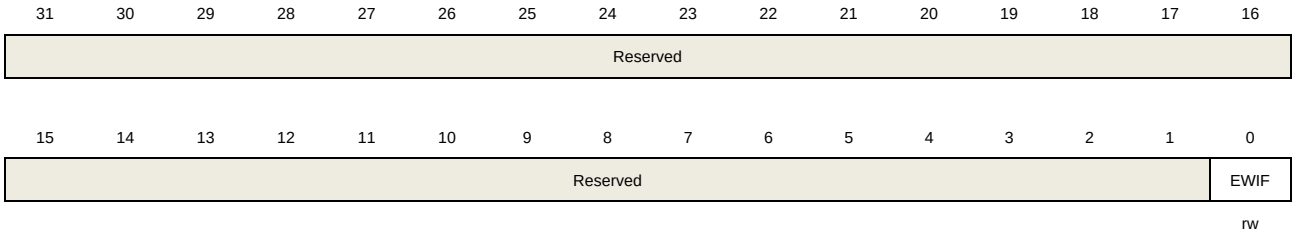
| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | EWIE | Early wakeup interrupt enable. If the bit is set, an interrupt occurs when the counter |

| | | |
|---|---|---|
| | | reaches 0x40. It can be cleared by a hardware reset or software clock reset (refer to 4.3.5. APB1 reset register ) . A write operation of 0 has no effect. |
| 8:7 | PSC[1:0] | Prescaler. The time base of the watchdog counter<br>00: (PCLK1 / 4096) / 1<br>01: (PCLK1 / 4096) / 2<br>10: (PCLK1 / 4096) / 4<br>11: (PCLK1 / 4096) / 8 |
| 6:0 | WIN[6:0] | The Window value. A reset occur if the watchdog counter (CNT bits in WWDGT_CTL) is written when the value of the watchdog counter is greater than the Window value. |

### Status register (WWDGT_STAT)

Address offset: 0x08
Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | EWIF | Early wakeup interrupt flag. When the counter reaches 0x40 or refreshes before it reaches the window value, this bit is set by hardware even the interrupt is not enabled (EWIE in WWDGT_CFG is cleared). This bit is cleared by writing 0. There is no effect when writing 1. |

# 13. Real-time clock(RTC)

## 13.1. Overview

The RTC provides a time which includes hour/minute/second/sub-second and a date including year/month/day/week day. The time and date are expressed in BCD code except sub-second. Sub-second is expressed in binary code. Hour adjustment for daylight saving time. Working in power saving mode and smart wakeup is software configurable. Support improving the calendar accuracy using extern accurate low frequency clock.

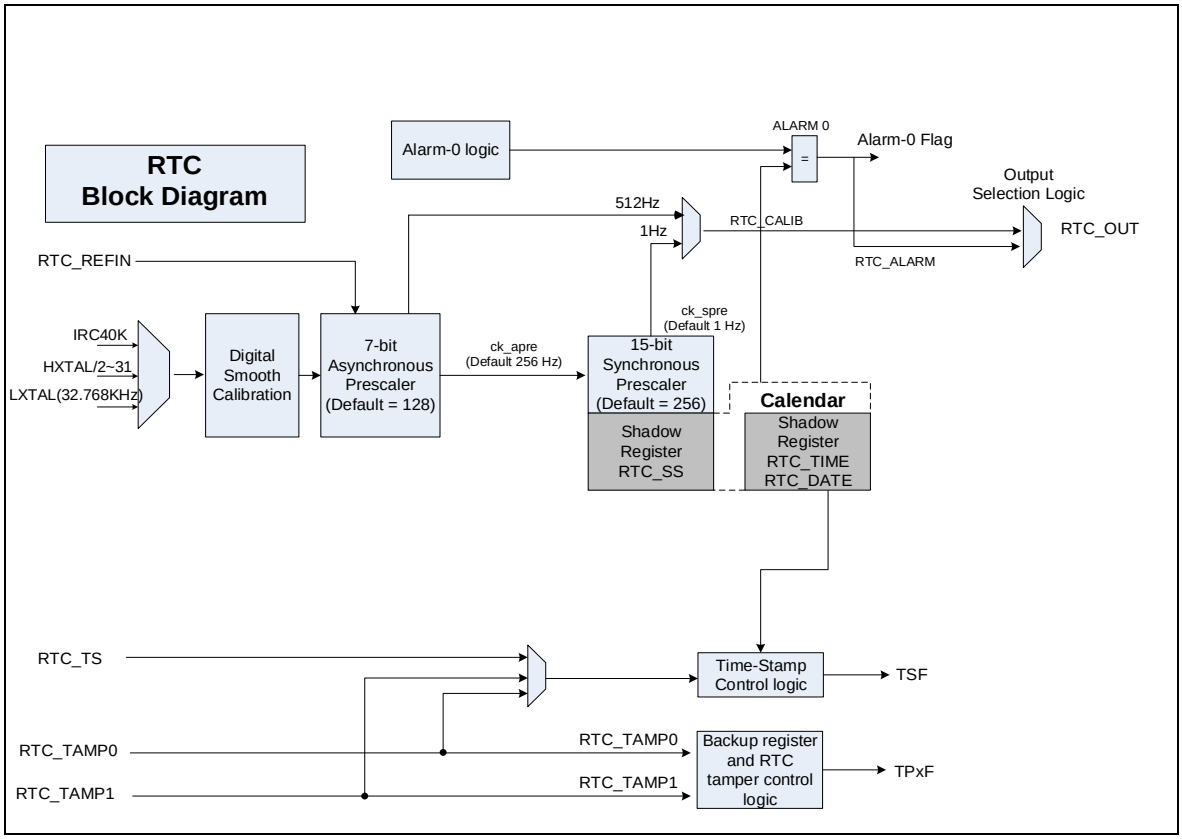
## 13.2. Characteristics

- Daylight saving compensation supported by software.
- External high-accurate low frequency (50Hz or 60Hz) clock used to achieve higher calendar accuracy performed by reference clock detection option function.
- Atomic clock adjustment (max adjustment accuracy is 0.95PPM) for calendar calibration performed by digital calibration function.
- Sub-second adjustment by shift function.
- Time-stamp function for saving event time.
- Two Tamper sources can be chosen and tamper type is configurable.
- Programmable calendar and one field maskable alarms.
- Maskable interrupt source:
  - Alarm 0
  - Time-stamp detection
  - Tamper detection
- Five 32-bit (20 bytes total) universal backup registers which can keep data under power saving mode. Backup register will be reset if tamper event detected.

# 13.3. Function overview

## 13.3.1. Block diagram

**Figure 13-1. Block diagram of RTC**




The RTC unit includes:

- Alarm event/interrupt
- Tamper event/interrupt
- 32-bit backup registers
- Optional RTC output function
  - 512Hz (default prescale):RTC_OUT
  - 1Hz(default prescale):RTC_OUT
  - Alarm event(polarity is configurable):RTC_OUT
- Optional RTC input function:
  - time stamp event detection: RTC_TS
  - tamper 0 event detection: RTC_TAMP0
  - tamper 1 event detection: RTC_TAMP1
  - reference clock input: RTC_REFIN(50 or 60 Hz)

### 13.3.2. Clock source and prescalers

RTC unit has three independent clock sources: LXTAL, IRC40K and HXTAL divided by 32.

In the RTC unit, there are two prescalers used for implementing the calendar and other functions. One prescaler is a 7-bit asynchronous prescaler and the other is a 15-bit synchronous prescaler. Asynchronous prescaler is mainly used for reducing power consumption. The asynchronous prescaler is recommended to set as high as possible if both prescalers are used.

The frequency formula of two prescalers is shown as below:

$$f_{ck_apre} = \frac{f_{rtcclk}}{FACTOR_A + 1} \quad (13\text{-}1)$$

$$f_{ck_spre} = \frac{f_{ck_apre}}{FACTOR_S + 1} = \frac{f_{rtcclk}}{(FACTOR_A + 1)*(FACTOR_S + 1)} \quad (13\text{-}2)$$

The ck_apre clock is used to driven the RTC_SS down counter which stands for the time left to next second in binary format and when it reaches 0 it will automatically reload FACTOR_S value. The ck_spre clock is used to driven the calendar registers. Each clock will make second plus one.

### 13.3.3. Shadow registers introduction

BPSHAD control bit decides the location when APB bus accesses the RTC calendar register RTC_DATE, RTC_TIME and RTC_SS. By default, the BPSHAD is cleared, and APB bus accesses the shadow calendar registers. Shadow calendar registers is updated with the value of real calendar registers every two RTC clock and at the same time RSYNF bit will be set once. This update mechanism is not performed in Deep-Sleep mode and Standby mode. When exiting these modes, software must clear RSYNF bit and wait it is asserted (the max wait time is 2 RTC clock) before reading calendar register under BPSHAD=0 situation.

**Note**: When reading calendar registers (RTC_SS, RTC_TIME, RTC_DATE) under BPSHAD=0, the frequency of the APB clock ($f_{apb}$) must be at least 7 times the frequency of the RTC clock ($f_{rtcclk}$).

System reset will reset the shadow calendar registers.

### 13.3.4. Configurable and field maskable alarm

RTC alarm function is divided into some fields and each has a maskable bit.

RTC alarm function can be enabled or disabled by ALRMxEN(x=0)bit in RTC_CTL. If all the alarm fields value match the corresponding calendar value when ALRMxEN=1(x=0), the Alarm flag will be set.

**Note:** FACTOR_S in the RTC_PSC register must be larger than 3 if MSKS bit reset in RTC_ALRMxTD(x=0).

If a field is masked, the field is considered as matched in logic. If all the fields have been

masked, the Alarm Flag will assert 3 RTC clock later after ALRMxEN (x=0) is set.

### 13.3.5. RTC initialization and configuration

#### RTC register write protection

BKPWEN bit in the PMU_CTL register is cleared in default, so writing to RTC registers needs setting BKPWEN bit ahead of time.

After power-on reset, most of RTC registers are write protected. Unlocking this protection is the first step before writing to them.

Following steps below will unlock the write protection:

1. Write '0xCA' into the RTC_WPK register
2. Write '0x53' into the RTC_WPK register

Writing a wrong value to RTC_WPK will make write protection valid again. The state of write protection is not affected by system reset. Following registers are writing protected but others are not:

RTC_TIME, RTC_DATE, RTC_CTL, RTC_STAT, RTC_PSC, RTC_ALRMxTD, RTC_SHIFTCTL, RTC_HRFC, RTC_ALRMxSS

#### Calendar initialization and configuration

The prescaler and calendar value can be programmed by the following steps:

1. Enter initialization mode (by setting INITM=1) and polling INITF bit until INITF=1.
2. Program both the asynchronous and synchronous prescaler factors in RTC_PSC register.
3. Write the initial calendar values into the shadow calendar registers (RTC_TIME and RTC_DATE), and use the CS bit in the RTC_CTL register to configure the time format (12 or 24 hours).
4. Exit the initialization mode (by setting INITM=0).

About 4 RTC clock cycles later, real calendar registers will load from shadow registers and calendar counter restarts.

**Note:** Reading calendar register (BPSHAD=0) after initialization, software should confirm the RSYNF bit to 1.

YCM flag indicates whether the calendar has been initialized by checking the year field of calendar.

#### Daylight saving Time

RTC unit supports daylight saving time adjustment through S1H, A1H and DSM bit.

S1H and A1H can subtract or add 1 hour to the calendar when the calendar is running.S1H

and A1H operation can be tautologically set and DSM bit can be used to recording this adjustment operation. After setting the S1H/A1H, subtracting/adding 1 hour will perform when next second comes.

#### Alarm function operation process

To avoid unexpected alarm assertion and metastable state, alarm function has an operation flow:

1. Disable Alarm (by resetting ALRMxEN (x=0) in RTC_CTL)
2. Set the Alarm registers needed(RTC_ALRMxTD/RTC_ALRMxSS)
3. Enable Alarm function (by setting ALRMxENin the RTC_CTL)

### 13.3.6. Calendar reading

#### Reading calendar registers under BPSHAD=0

When BPSHAD=0, calendar value is read from shadow registers. For the existence of synchronization mechanism, a basic request has to meet: the APB1 bus clock frequency must be equal to or greater than 7 times the RTC clock frequency. APB1 bus clock frequency lower than RTC clock frequency is not allowed in any case.

When APB1 bus clock frequency is not equal to or greater than 7 times the RTC clock frequency, the calendar reading flow should be obeyed:

1. reading calendar time register and date register twice
2. if the two values are equal, the value can be seen as the correct value
3. if the two values are not equal, a third reading should performed
4. the third value can be seen as the correct value

RSYNF is asserted once every 2 RTC clock and at this time point, the shadow registers will be updated to current time and date.

To ensure consistency of the 3 values (RTC_SS, RTC_TIME, and RTC_DATE), below consistency mechanism is used in hardware:

1. reading RTC_SS will lock the updating of RTC_TIME and RTC_DATE
2. reading RTC_TIME will lock the updating of RTC_DATE
3. reading RTC_DATE will unlock updating of RTC_TIME and RTC_DATE

If the software wants to read calendar in a short time interval(smaller than 2 RTCCLK periods), RSYNF must be cleared by software after the first calendar read, and then the software must wait until RSYNF is set again before next reading.

In below situations, software should wait RSYNF bit asserted before reading calendar registers (RTC_SS, RTC_TIME, and RTC_DATE):

1. after a system reset

2. after an initialization
3. after shift function

Especially that software must clear RSYNF bit and wait it asserted before reading calendar register after wakeup from power saving mode.

#### Reading calendar registers under BPSHAD=1

When BPSHAD=1, RSYNF is cleared and maintains as 0 by hardware so reading calendar registers does not care about RSYNF bit. Current calendar value is read from real-time calendar counter directly. The benefit of this configuration is that software can get the real current time without any delay after wakeup from power saving mode (Deep-sleep /Standby Mode).

Because of no RSYNF bit periodic assertion, the results of the different calendar registers (RTC_SS/RTC_TIME/RTC_DATE) might not be coherent with each other when clock ck_apre edge occurs between two reading calendar registers.

In addition, if current calendar register is changing and at the same time the APB bus reading calendar register is also performing, the value of the calendar register read out might be not correct.

To ensure the correctness and consistency of the calendar value, software must perform reading operation as this: read all calendar registers continuously, if the last two values are the same, the data is coherent and correct.

### 13.3.7. Resetting the RTC

There are two reset sources used in RTC unit: system reset and backup domain reset.

System reset will affect calendar shadow registers and some bits of the RTC_STAT. When system reset is valid, the bits or registers mentioned before are reset to the default value.

Backup domain reset will affect the following registers and system reset will not affect them:
- RTC current real-time calendar registers
- RTC Control register (RTC_CTL)
- RTC Prescaler register (RTC_PSC)
- RTC High resolution frequency compensation register (RTC_HRFC)
- RTC Shift control register (RTC_SHIFTCTL)
- RTC Time stamp registers (RTC_SSTS/RTC_TTS/RTC_DTS)
- RTC Tamper register (RTC_TAMP)
- RTC Backup registers (RTC_BKPx)
- RTC Alarm registers (RTC_ALRMxSS/RTC_ALRMxTD)(x=0)

The RTC unit will go on running when system reset occurs or enter power saving mode, but if backup domain reset occurs, RTC will stop counting and all registers will reset.

### 13.3.8. RTC shift function

When there is a remote clock with higher degree of precision and RTC 1Hz

clock(ck_spre)has an offset (in a fraction of a second) with the remote clock, RTC unit provides a function named shift function to remove this offset and thus make second precision higher.

RTC_SS register indicates the fraction of a second in binary format and is down counting when RTC is running. Therefore by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] or by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] and at the same time set A1S bit can delay or advance the time when next second arrives.

The maximal RTC_SS value depends on the FACTOR_S value in RTC_PSC. The higher FACTOR_S, the higher adjustment precision.

Because of the 1Hz clock(ck_spre) is generated by FACTOR_A and FACTOR_S, the higher FACTOR_S means the lower FACTOR_A, then more power consuming.

**Note:** Before using shift function, the software must check the MSB of SSC in RTC_SS (SSC[15]) and confirm it is 0.

After writing RTC_SHIFTCTL register, the SOPF bit in RTC_STAT will be set at once. When shift operation is complete, SOPF bit is cleared by hardware. System reset does not affect SOPF bit.

Shift operation only works correctly when REFEN=0.

Software must not write to RTC_SHIFTCTL if REFEN=1.

### 13.3.9. RTC reference clock detection

RTC reference clock detection is another way to increase the precision of RTC second. To enable this function, you should have an external clock source (50Hz or 60 Hz) which is more precise than LXTAL clock source.

After enabling this function (REFEN=1), each 1Hz clock(ck_spre)edge is compared to the nearest RTC_REFIN clock edge. In most cases, the two clock edges are aligned every time. But when two clock edges are misaligned for the reason of LXTAL poor precision, the RTC reference clock detection function will shift the 1Hz clock edge a little to make next 1Hz clock edge aligned to reference clock edge.

When REFEN=1, a time window is applied at every second update time. Different detection state will use different window period.

7 ck_apre window is used when detecting the first reference clock edge and 3 ck_apre window is used for the edge aligned operation.

Whatever window used, the asynchronous prescaler counter will be forced to reload when the reference clock is detected in the window. When the two clock (ck_spre and reference clock) edges are aligned, this reload operation has no effect for 1Hz clock. But when the two clock edge are not aligned, this reload operation will shift ck_spre clock edge a bit to make

the ck_spre(1Hz) clock edge aligned to the reference clock edge.

When reference detection function is running while the external reference clock is removed (no reference clock edge found in 3 ck_apre window), the calendar updating still can be performed by LXTAL clock only. If the reference clock is recovered later, detection function will use 7 ck_apre window to identify the reference clock and use 3 ck_apre window to adjust the 1Hz clock (ck_spre) edge.

**Note:** Software must configure the FACTOR_A=0x7F and FACTOR_S=0xFF before enabling reference detection function (REFEN=1)

Reference detection function does not work in Standby Mode.

### 13.3.10. RTC smooth digital calibration

RTC smooth calibration function is a way to calibrate the RTC frequency based on RTC clock in a configurable calibration period time.

This calibration is equally executed in a period time and the cycle number of the RTC clock in the period time will be added or subtracted. The resolution of the calibration is about 0.954PPM with the range from -487.1PPM to +488.5PPM.

The calibration period time can be configured to the $2^{20}/2^{19}/2^{18}$ RTC clock cycles which stands for 32/16/8 seconds if RTC input frequency is 32.768 KHz.

The High resolution frequency compensation register (RTC_HRFC) specifies the number of RTCCLK clock cycles to be calibrated during the period time:

So using CMSK can mask clock cycles from 0 to 511 and thus the RTC frequency can be reduced by up to 487.1PPM.

To increase the RTC frequency the FREQI bit can be set. If FREQI bit is set, there will be 512 additional cycles to be added during period time which means every $2^{11}/2^{10}/2^{9}$(32/16/8 seconds) RTC clock insert one cycle.

So using FREQI can increase the RTC frequency by 488.5PPM.

The combined using of CMSK and FREQI can adjust the RTC cycles from -511 to +512 cycles in the period time which means the calibration range is -487.1PPM to +488.5PPM with a resolution of about 0.954PPM.

When calibration function is running, the output frequency of calibration is calculated by the following formula:

$$f_{cal} = f_{rtcclk} \times \left(1 + \frac{FREQI \times 512 - CMSK}{2^N + CMSK - FREQI \times 512}\right) \quad (13\text{-}3)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

#### Calibration when FACTOR_A < 3

When asynchronous prescaler value (FACTOR_A) is set to less than 3, software should not

set FREQI bit to 1 when using calibration function. FREQI setting will be ignored when FACTOR_A<3.

When the FACTOR_A is less than 3, the FACTOR_S value should be set to a value less than the nominal value. Assuming that RTC clock frequency is nominal 32.768 KHz, the corresponding FACTOR_S should be set as following rule:

FACTOR_A = 2: 2 less than nominal FACTOR_S (8189 with 32.768 KHz)

FACTOR_A = 1: 4 less than nominal FACTOR_S (16379 with 32.768 KHz)

FACTOR_A = 0: 8 less than nominal FACTOR_S (32759 with 32.768 KHz)

When the FACTOR_A is less than 3, CMSK is 0x100，the formula of calibration frequency is as follows:

$$f_{cal} = f_{rtcclk} \times (1 + \frac{256-CMSK}{2^N+CMSK-256}) \quad (13\text{-}4)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

### Verifying the RTC calibration

Calibration 1Hz output is provided to assist software to measure and verify the RTC precision.

Up to 2 RTC clock cycles measurement error may occur when measuring the RTC frequency over a limited measurement period. To eliminate this measurement error the measurement period should be the same as the calibration period.

- When the calibration period is 32 seconds(this is default configuration)

Using exactly 32s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 0.477PPM (0.5 RTCCLK cycles over 32s)

- When the calibration period is 16 seconds(by setting CWND16 bit)

In this configuration, CMSK[0] is fixed to 0 by hardware. Using exactly 16s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 0.954PPM (0.5 RTCCLK cycles over 16s)

- When the calibration period is 8 seconds(by setting CWND8 bit)

In this configuration, CMSK[1:0] is fixed to 0 by hardware. Using exactly 8s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 1.907PPM (0.5 RTCCLK cycles over 8s)

### Re-calibration on-the-fly

When the INITF bit is 0, software can update the value of RTC_HRFC using following steps:

1. Wait the SCPF=0
2. Write the new value into RTC_HRFC register
3. After 3 ck_apre clocks, the new calibration settings take effect

### 13.3.11. Time-stamp function

Time-stamp function is performed on RTC_TS pin and is enabled by control bit TSEN.

When a time-stamp event occurs on RTC_TS pin, the calendar value will be saved in time-stamp registers (RTC_DTS/RTC_TTS/RTC_SSTS) and the time-stamp flag (TSF) is set to 1 by hardware. Time-stamp event can generate an interrupt if time-stamp interrupt enable (TSIE) is set.

Time-stamp registers only record the calendar at the first time time-stamp event occurs which means that time-stamp registers will not change when TSF=1.

To extend the time-stamp event source, one optional feature is provided: tamper function can also be considered as time-stamp function if TPTS is set.

**Note:** When the time-stamp event occurs, TSF is set 2 ck_apre cycles delay because of synchronization mechanism.

### 13.3.12. Tamper detection

The RTC_TAMPx pin input can be used for tamper event detection under edge detection mode or level detection mode with configurable filtering setting.

#### RTC backup registers (RTC_BKPx)

The RTC backup registers are located in the $V_{DD}$ backup domain. The wake up action from Standby Mode or System Reset does not affect these registers.

These registers are only reset by detected tamper event and backup domain reset.

#### Tamper detection function initialization

RTC tamper detection function can be independently enabled on tamper input pin by setting corresponding TPxEN bit. Tamper detection configuration is set before enable TPxEN bit. When the tamper event is detected, the corresponding flag (TPxF) will assert. Tamper event can generate an interrupt if tamper interrupt enable (TPIE) is set. Any tamper event will reset all backup registers (RTC_BKPx).

#### Timestamp on tamper event

The TPTS bit can control whether the tamper detection function is used as time-stamp function. If the bit is set to 1, the TSF bit will be set when the tamper event detected. If enable the time-stamp function. Whatever the TPTS bit is, the TPxF will assert when tamper event detected.

#### Edge detection mode on tamper input detection

When FLT bit is set to 0x0, the tamper detection is set to edge detection mode and TPxEG bit determines the rising edge or falling edge is the detecting edge. When tamper detection is under edge detection mode, the internal pull-up resistors on the tamper detection input pin are deactivated.

Because of detecting the tamper event will reset the backup registers (RTC_BKPx), writing to the backup register should ensure that the tamper event reset and the writing operation will not occur at the same time, a recommend way to avoid this situation is disable the tamper detection before writing to the backup register and re-enable tamper detection after finish writing.

**Note:** Tamper detection is still running when $V_{DD}$ power is switched off if tamper is enabled.

#### Level detection mode with configurable filtering on tamper input detection

When FLT bit is not reset to 0x0, the tamper detection is set to level detection mode and FLT bit determines the consecutive number of samples (2, 4 or 8) needed for valid level. When DISPU is set to 0x0(this is default), the internal pull-up resistance will pre-charge the tamper input pin before each sampling and thus larger capacitance is allowed to connect to the tamper input pin. The pre-charge duration is configured through PRCH bit. Higher capacitance needs long pre-charge time.

The time interval between each sampling is also configurable. Through adjusting the sampling frequency (FREQ), software can balance between the power consuming and tamper detection latency.

### 13.3.13. Calibration clock output

Calibration clock can be output on the RTC_OUT if COEN bit is set to 1.

When the COS bit is set to 0(this is default) and asynchronous prescaler is set to 0x7F(FACTOR_A), the frequency of RTC_CALIB is $f_{rtcclk}/64$. When the RTCCLK is 32.768KHz, RTC_CALIB output is corresponding to 512Hz.It's recommend to using rising edge of RTC_CALIB output because there may be a light jitter on falling edge.

When the COS bit is set to 1, the RTC_CALIB frequency is:

$$f_{rtc_calib} = \frac{f_{rtcclk}}{(FACTOR_A+1)\times(FACTOR_S+1)} \quad (13\text{-}5)$$

When the RTCCLK is 32.768 KHz, RTC_CALIB output is corresponding to 1Hz if prescaler are default values.

### 13.3.14. Alarm output

When OS control bits are not reset, RTC_ALARM alternate function output is enabled. This

function will directly output the content of alarm flag in RTC_STAT.

The OPOL bit in RTC_CTL can configure the polarity of the alarm output which means that the RTC_ALARM output is the opposite of the corresponding flag bit or not.

### 13.3.15. RTC power saving mode management

**Table 13-1. RTC power saving mode management**

| Mode | Active in Mode | Exit Mode |
|---|---|---|
| Sleep | Yes | RTC Interrupts |
| Deep-Sleep | Yes: if clock source is LXTAL or IRC40K | RTC Alarm/ Tamper Event/ Timestamp Event |
| Standby | Yes: if clock source is LXTAL or IRC40K | RTC Alarm/ Tamper Event/ Timestamp Event |

### 13.3.16. RTC interrupts

All RTC interrupts are connected to the EXTI controller.

Below steps should be followed if you want to use the RTC alarm/tamper/timestamp:

1. Configure and enable the corresponding interrupt line to RTC alarm/tamper/timestamp event of EXTI and set the rising edge for triggering
2. Configure and enable the RTC alarm/tamper/timestamp global interrupt
3. Configure and enable the RTC alarm/tamper/timestamp function

**Table 13-2. RTC interrupts control**

| Interrupt | Event flag | Control Bit | Exit Sleep | Exit Deep-sleep | Exit Standby |
|---|---|---|---|---|---|
| Alarm 0 | ALRM0F | ALRM0IE | Y | Y(*) | Y(*) |
| Timestamp | TSF | TSIE | Y | Y(*) | Y(*) |
| Tamper 0 | TP0F | TPIE | Y | Y(*) | Y(*) |
| Tamper 1 | TP1F | TPIE | Y | Y(*) | Y(*) |

* Only active when RTC clock source is LXTAL or IRC40K.

## 13.4. Register definition

RTC base address: 0x4000 2800

### 13.4.1. Time register (RTC_TIME)

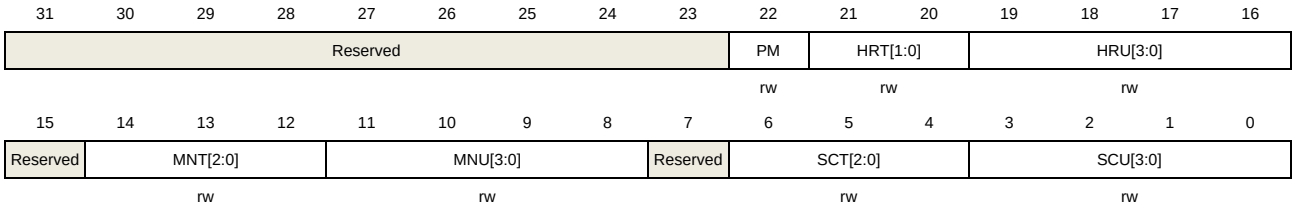
Address offset: 0x00

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22 | PM | AM/PM mark<br>0: AM or 24-hour format<br>1: PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | Reserved | Must be kept at reset value |
| 14:12 | MNT[2:0] | Minute tens in BCD code |
| 11:8 | MNU[3:0] | Minute units in BCD code |
| 7 | Reserved | Must be kept at reset value |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 13.4.2. Date register (RTC_DATE)

Address offset: 0x04

System reset value: 0x0000 2101 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | YRT[3:0] | | | | YRU[3:0] | | | |
| | | | | | | | | | | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DOW[2:0] | | | MONT | MONU[3:0] | | | | Reserved | | DAYT[1:0] | | DAYU[3:0] | | | |
| rw | | | rw | rw | | | | | | rw | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:20 | YRT[3:0] | Year tens in BCD code |
| 19:16 | YRU[3:0] | Year units in BCD code |
| 15:13 | DOW[2:0] | Days of the week<br>0x0: Reserved<br>0x1: Monday<br>…<br>0x7: Sunday |
| 12 | MONT | Month tens in BCD code |
| 11:8 | MONU[3:0] | Month units in BCD code |
| 7:6 | Reserved | Must be kept at reset value |
| 5:4 | DAYT[1:0] | Day tens in BCD code |
| 3:0 | DAYU[3:0] | Day units in BCD code |

### 13.4.3. Control register (RTC_CTL)

Address offset: 0x08

System reset: not affected

Backup domain reset value: 0x0000 0000

This register is writing protected

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | COEN | OS[1:0] | | OPOL | COS | DSM | S1H | A1H |
| | | | | | | | | rw | rw | | rw | rw | rw | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TSIE | Reserved | | ALRM0IE | TSEN | Reserved | | ALRM0EN | Reserved | CS | BPSHAD | REFEN | TSEG | Reserved | | |
| rw | | | rw | rw | | | rw | | rw | rw | rw | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 23 | COEN | Calibration output enable<br>0: Disable calibration output<br>1: Enable calibration output |
| 22:21 | OS[1:0] | Output selection<br>This bit is used for selecting flag source to output<br>0x0: Disable output RTC_ALARM<br>0x1: Enable alarm0 flag output<br>0x2: Reserved<br>0x3: Reserved |
| 20 | OPOL | Output polarity<br>This bit is used to invert output RTC_ALARM<br>0: Disable invert output RTC_ALARM<br>1: Enable invert output RTC_ALARM |
| 19 | COS | Calibration output selection<br>Valid only when COEN=1 and prescalers are at default values<br>0: Calibration output is 512 Hz<br>1: Calibration output is 1Hz |
| 18 | DSM | Daylight saving mark<br>This bit is flexible used by software. Often can be used to recording the daylight saving hour adjustment. |
| 17 | S1H | Subtract 1 hour(winter time change)<br>One hour will be subtracted from current time if it is not 0<br>0: No effect<br>1: 1 hour will be subtracted at next second change time. |
| 16 | A1H | Add 1 hour(summer time change)<br>One hour will be added from current time<br>0: No effect<br>1: 1 hour will be added at next second change time |
| 15 | TSIE | Time-stamp interrupt enable<br>0: Disable time-stamp interrupt<br>1: Enable time-stamp interrupt |
| 14:13 | Reserved | Must be kept at reset value |
| 12 | ALRM0IE | RTC alarm-0 interrupt enable<br>0: Disable alarm interrupt<br>1: Enable alarm interrupt |
| 11 | TSEN | Time-stamp function enable<br>0: Disable time-stamp function<br>1: Enable time-stamp function |

| | | |
|---|---|---|
| 10:9 | Reserved | Must be kept at reset value |
| 8 | ALRM0EN | Alarm-0 function enable<br>0: Disable alarm function<br>1: Enable alarm function |
| 7 | Reserved | Must be kept at reset value |
| 6 | CS | Clock System<br>0: 24-hour format<br>1: 12-hour format<br>**Note:** Can only be written in initialization state |
| 5 | BPSHAD | Shadow registers bypass control<br>0: Reading calendar from shadow registers<br>1: Reading calendar from current real-time calendar<br>**Note:** If frequency of APB1 clock is less than seven times the frequency of RTCCLK, this bit must set to 1. |
| 4 | REFEN | Reference clock detection function enable<br>0: Disable reference clock detection function<br>1: Enable reference clock detection function<br>**Note:** Can only be written in initialization state and FACTOR_S must be 0x00FF |
| 3 | TSEG | Valid event edge of time-stamp<br>0: rising edge is valid event edge for time-stamp event<br>1: falling edge is valid event edge for time-stamp event |
| 2:0 | Reserved | Must be kept at reset value |

### 13.4.4. Status register (RTC_STAT)

Address offset: 0x0C

System reset: Only INITM, INITF and RSYNF bits are set to 0. Others are not affected

Backup domain reset value: 0x0000 0007

This register is writing protected except RTC_STAT[14:8].

This register has to be accessed by word(32-bit)

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TP1F | TP0F | TSOVRF | TSF | Reserved | | ALRM0F | INITM | INITF | RSYNF | YCM | SOPF | Reserved | | ALRM0WF |
| | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rw | r | rc_w0 | r | r | | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |
| 16 | SCPF | Smooth calibration pending flag |

| | | |
|---|---|---|
| | | Set to 1 by hardware when software writes to RTC_HRFC and set to 0 by hardware when smooth calibration window is completed. |
| 15 | Reserved | Must be kept at reset value |
| 14 | TP1F | RTC_TAMP1 detected flag<br>Set to 1 by hardware when tamper detection is found on tamper1 input pin. Software can clear this bit by writing 0 into this bit. |
| 13 | TP0F | RTC_TAMP0 detected flag<br>Set to 1 by hardware when tamper detection is found on tamper0 input pin. Software can clear this bit by writing 0 into this bit. |
| 12 | TSOVRF | Time-stamp overflow flag<br>This bit is set by hardware when a time-stamp event is detected if TSF bit is set before.<br>Cleared by software writing 0. |
| 11 | TSF | Time-stamp flag<br>Set by hardware when time-stamp event is detected.<br>Cleared by software writing 0. |
| 10:9 | Reserved | Must be kept at reset value |
| 8 | ALRM0F | Alarm-0 occurs flag<br>Set to 1 by hardware when current time/date matches the time/date of alarm 0 setting value.<br>Cleared by software writing 0. |
| 7 | INITM | Enter initialization mode<br>0: Free running mode<br>1: Enter initialization mode for setting calendar time/date and prescaler. Counter will stop under this mode. |
| 6 | INITF | Initialization state flag<br>Set to 1 by hardware, calendar registers and prescaler can be programmed in this state.<br>0:Calendar registers and prescaler register cannot be changed<br>1:Calendar registers and prescaler register can be changed |
| 5 | RSYNF | Register synchronization flag<br>Set to 1 by hardware every 2 RTCCLK which will copy current calendar time/date into shadow register. Initialization mode(INITM), shift operation pending flag(SOPF) or bypass mode(BPSHAD) will clear this bit. This bit is also can be cleared by software writing 0.<br>0:Shadow register are not yet synchronized<br>1:Shadow register are synchronized |
| 4 | YCM | Year configuration mark |

| | | |
|---|---|---|
| | | Set by hardware if the year field of calendar date register is not the default value 0.<br>0:Calendar has not been initialized<br>1:Calendar has been initialized |
| 3 | SOPF | Shift function operation pending flag<br>0:No shift operation is pending<br>1:Shift function operation is pending |
| 2:1 | Reserved | Must be kept at reset value |
| 0 | ALRM0WF | Alarm 0 configuration can be write flag<br>Set by hardware if alarm register can be written after ALRM0EN bit has reset.<br>0:Alarm registers programming is not allowed<br>1:Alarm registers programming is allowed |

### 13.4.5. Prescaler register (RTC_PSC)

Address offset: 0x10
System reset: not effected
Backup domain reset value: 0x007F 00FF
This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FACTOR_A[6:0] | | | | | | |
| | | | | | | | | | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | FACTOR_S[14:0] | | | | | | | | | | | | | | |
| | rw | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22:16 | FACTOR_A[6:0] | Asynchronous prescaler factor<br>ck_apre frequency = RTCCLK frequency/(FACTOR_A+1) |
| 15 | Reserved | Must be kept at reset value |
| 14:0 | FACTOR_S[14:0] | Synchronous prescaler factor<br>ck_spre frequency = ck_apre frequency/(FACTOR_S+1) |

### 13.4.6. Alarm 0 time and date register (RTC_ALRM0TD)

Address offset: 0x1C
System reset: not effect
Backup domain reset value: 0x0000 0000
This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSKD | DOWS | DAYT[1:0] | | DAYU[3:0] | | | | MSKH | PM | HRT[1:0] | | HRU[3:0] | | | |
| rw | rw | rw | | rw | | | | rw | rw | rw | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSKM | MNT[2:0] | | | MNU[3:0] | | | | MSKS | SCT[2:0] | | | SCU[3:0] | | | |
| rw | rw | | | rw | | | | rw | rw | | | rw | | | |

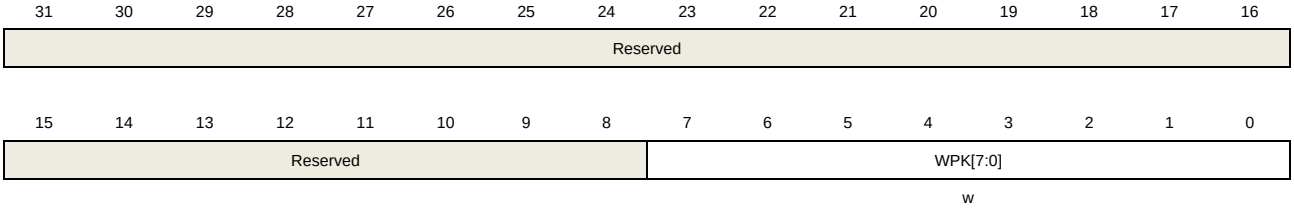
| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MSKD | Alarm date mask bit<br>0:Not mask date/day field<br>1:Mask date/day field |
| 30 | DOWS | Day of the week selected<br>0:DAYU[3:0] indicates the date units<br>1: DAYU[3:0] indicates the week day and DAYT[1:0] has no means. |
| 29:28 | DAYT[1:0] | Date tens in BCD code |
| 27:24 | DAYU[3:0] | Date units or week day in BCD code |
| 23 | MSKH | Alarm hour mask bit<br>0:Not mask hour field<br>1:Mask hour field |
| 22 | PM | AM/PM flag<br>0:AM or 24-hour format<br>1:PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | MSKM | Alarm minutes mask bit<br>0:Not mask minutes field<br>1:Mask minutes field |
| 14:12 | MNT[2:0] | Minutes tens in BCD code |
| 11:8 | MNU[3:0] | Minutes units in BCD code |
| 7 | MSKS | Alarm second mask bit<br>0:Not mask second field<br>1:Mask second field |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 13.4.7. Write protection key register (RTC_WPK)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | WPK[7:0] | Key for write protection |

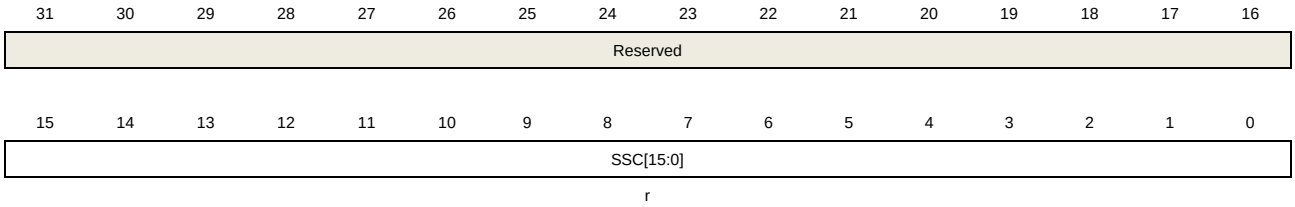
### 13.4.8. Sub second register (RTC_SS)

Address offset: 0x28

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | SSC[15:0] | Sub second value<br>This value is the counter value of synchronous prescaler. Second fraction value is calculated by the below formula:<br>Second fraction = ( FACTOR_S - SSC ) / ( FACTOR_S + 1 ) |

### 13.4.9. Shift function control register (RTC_SHIFTCTL)

Address offset: 0x2C

System reset: not effect

Backup Reset value: 0x0000 0000

This register is writing protected and can only be wrote when SOPF=0

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A1S | Reserved | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SFS[14:0] | | | | | | | | | | | | | | |
| | w | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | A1S | One second add<br>0:Not add 1 second<br>1:Add 1 second to the clock/calendar.<br>This bit is jointly used with SFS field to add a fraction of a second to the clock. |
| 30:15 | Reserved | Must be kept at reset value |
| 14:0 | SFS[14:0] | Subtract a fraction of a second<br>The value of this bit will add to the counter of synchronous prescaler.<br>When only using SFS, the clock will delay because the synchronous prescaler is a down counter:<br>Delay (seconds) = SFS / ( FACTOR_S + 1 )<br>When jointly using A1S and SFS, the clock will advance:<br>Advance (seconds) = ( 1 - ( SFS / ( FACTOR_S + 1 ) ) ) |

**Note:** Writing to this register will cause RSYNF bit to be cleared.

## 13.4.10. Time of time stamp register (RTC_TTS)

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: no effect

This register will record the calendar time when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | PM | HRT[1:0] | | HRU[3:0] | | | |
| | | | | | | | | | r | r | | r | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | MNT[2:0] | | | MNU[3:0] | | | | Reserved | SCT[2:0] | | | SCU[3:0] | | | |
| | r | | | r | | | | | r | | | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22 | PM | AM/PM mark<br>0:AM or 24-hour format |

| | | |
|---|---|---|
| | | 1:PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | Reserved | Must be kept at reset value |
| 14:12 | MNT[2:0] | Minute tens in BCD code |
| 11:8 | MNU[3:0] | Minute units in BCD code |
| 7 | Reserved | Must be kept at reset value |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 13.4.11. Date of time stamp register (RTC_DTS)

Address offset: 0x34

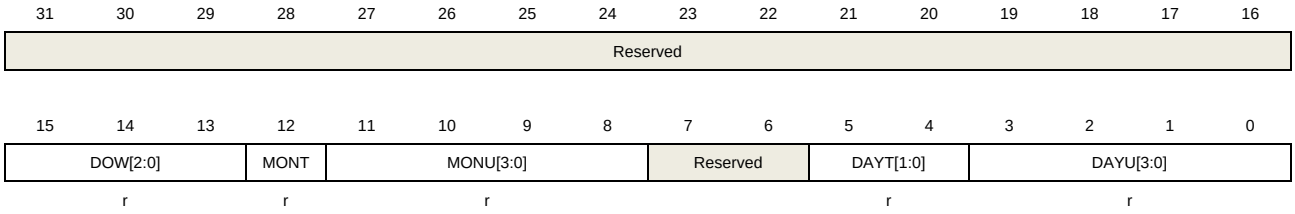
Backup domain reset value: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:13 | DOW[2:0] | Days of the week |
| 12 | MONT | Month tens in BCD code |
| 11:8 | MONU[3:0] | Month units in BCD code |
| 7:6 | Reserved | Must be kept at reset value |
| 5:4 | DAYT[1:0] | Day tens in BCD code |
| 3:0 | DAYU[3:0] | Day units in BCD code |

### 13.4.12. Sub second of time stamp register (RTC_SSTS)

Address offset: 0x38

Backup domain reset: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | SSC[15:0] | Sub second value<br>This value is the counter value of synchronous prescaler when TSF is set to 1. |

### 13.4.13. High resolution frequency compensation register (RTC_HRFC)

Address offset: 0x3C

Backup domain reset: 0x0000 0000

System Reset: no effect

This register is write protected.

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FREQI | CWND8 | CWND16 | Reserved | | | | | | | | CMSK[8:0] | | | | |
| rw | rw | rw | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | FREQI | Increase RTC frequency by 488.5PPM<br>0: No effect<br>1: One RTCCLK pulse is inserted every $2^{11}$ pulses.<br>This bit should be used in conjunction with CMSK bit. If the input clock frequency is 32.768KHz, the number of RTCCLK pulses added during 32s calibration window is (512 * FREQI) - CMSK |

| | | |
|---|---|---|
| 14 | CWND8 | Frequency compensation window 8 second selected<br>0:No effect<br>1:Calibration window is 8 second<br>**Note:** When CWND8=1, CMSK[1:0] are stuck at “00”. |
| 13 | CWND16 | Frequency compensation window 16 second selected<br>0:No effect<br>1:Calibration window is 16 second<br>**Note:** When CWND16=1, CMSK[0] are stuck at “0”. |
| 12:9 | Reserved | Must be kept at reset value |
| 8:0 | CMSK[8:0] | Calibration mask number<br>The number of mask pulse out of $2^{20}$ RTCCLK pulse.<br>This feature will decrease the frequency of calendar with a resolution of 0.9537 PPM. |

## 13.4.14. Tamper register (RTC_TAMP)

Address offset: 0x40

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PC15MDE | PC15VAL | PC14MDE | PC14VAL | PC13MDE | PC13VAL | Reserved | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISPU | PRCH[1:0] | | FLT[1:0] | | FREQ[2:0] | | | TPTS | Reserved | | TP1EG | TP1EN | TPIE | TP0EG | TP0EN |
| rw | rw | | rw | | rw | | | rw | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23 | PC15MDE | PC15 Mode<br>0:No effect<br>1:Force PC15 to push-pull output if LXTAL is disable |
| 22 | PC15VAL | PC15 Value<br>Only valid when LXTAL is disabled and PC15MDE=1,PC15 output this bit data. |
| 21 | PC14MDE | PC14 Mode<br>0:No effect<br>1:Force PC14 to push-pull output if LXTAL is disable |
| 20 | PC14VAL | PC14 Value<br>Only valid when LXTAL is disabled and PC14MDE=1,PC14 output this bit data. |

| | | |
|---|---|---|
| 19 | PC13MDE | PC13 Mode<br>0:No effect<br>1:Force PC13 to push-pull output if all RTC alternate functions are disabled. |
| 18 | PC13VAL | PC13 value or alarm output type value<br>When PC13 is used to output alarm:<br>0:PC13 is in open-drain output type<br>1:PC13 is in push-pull output type<br>When all RTC alternate functions are disabled and PC13MDE=1:<br>0:PC13 output 0<br>1:PC13 output 1 |
| 17:16 | Reserved | Must be kept at reset value |
| 15 | DISPU | RTC_TAMPx pull up disable bit<br>0:Enable inner pull-up before sampling for pre-charge RTC_TAMPx pin<br>1:Disable pre-charge duration |
| 14:13 | PRCH[1:0] | Pre-charge duration time of RTC_TAMPx<br>This setting determines the pre-charge time before each sampling.<br>0x0:1 RTC clock<br>0x1:2 RTC clock<br>0x2:4 RTC clock<br>0x3:8 RTC clock |
| 12:11 | FLT[1:0] | RTC_TAMPx filter count setting<br>This bit determines the tamper sampling type and the number of consecutive sample.<br>0x0: Detecting tamper event using edge mode. Pre-charge duration is disabled automatically<br>0x1: Detecting tamper event using level mode.2 consecutive valid level samples will make an effective tamper event<br>0x2:Detecting tamper event using level mode.4 consecutive valid level samples will make an effective tamper event<br>0x3:Detecting tamper event using level mode.8 consecutive valid level samples will make an effective tamper event |
| 10:8 | FREQ[2:0] | Sampling frequency of tamper event detection<br>0x0: Sample once every 32768 RTCCLK(1Hz if RTCCLK=32.768KHz)<br>0x1: Sample once every 16384 RTCCLK(2Hz if RTCCLK=32.768KHz)<br>0x2: Sample once every 8192 RTCCLK(4Hz if RTCCLK=32.768KHz)<br>0x3: Sample once every 4096 RTCCLK(8Hz if RTCCLK=32.768KHz)<br>0x4: Sample once every 2048 RTCCLK(16Hz if RTCCLK=32.768KHz)<br>0x5: Sample once every 1024 RTCCLK(32Hz if RTCCLK=32.768KHz)<br>0x6: Sample once every 512 RTCCLK(64Hz if RTCCLK=32.768KHz)<br>0x7: Sample once every 256 RTCCLK(128Hz if RTCCLK=32.768KHz) |

| | | |
|---|---|---|
| 7 | TPTS | Make tamper function used for timestamp function<br>0:No effect<br>1:TSF is set when tamper event detected even TSEN=0 |
| 6:5 | Reserved | Must be kept at reset value |
| 4 | TP1EG | Tamper 1 event trigger edge<br>If tamper detection is in edge mode(FLT =0):<br>0: Rising edge triggers a tamper detection event<br>1: Falling edge triggers a tamper detection event<br>If tamper detection is in level mode(FLT !=0):<br>0: Low level triggers a tamper detection event<br>1: High level triggers a tamper detection event |
| 3 | TP1EN | Tamper 1 detection enable<br>0:Disable tamper 1 detection function<br>1:Enable tamper 1 detection function |
| 2 | TPIE | Tamper detection interrupt enable<br>0:Disable tamper interrupt<br>1:Enable tamper interrupt |
| 1 | TP0EG | Tamper 0 event trigger edge<br>If tamper detection is in edge mode(FLT =0):<br>0: Rising edge triggers a tamper detection event<br>1: Falling edge triggers a tamper detection event<br>If tamper detection is in level mode(FLT !=0):<br>0: Low level triggers a tamper detection event<br>1: High level triggers a tamper detection event |
| 0 | TP0EN | Tamper 0 detection enable<br>0:Disable tamper 0 detection function<br>1:Enable tamper 0 detection function |

**Note:** It's strongly recommended that reset the TPxEN before change the tamper configuration.

## 13.4.15. Alarm 0 sub second register (RTC_ALRM0SS)

Address offset: 0x44

Backup domain reset: 0x0000 0000

System reset: no effect

This register is write protected and can only be wrote when ALRM0EN=0 or INITM=1

This register has to be accessed by word(32-bit)

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

| Reserved | SSC[14:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27:24 | MSKSSC[3:0] | Mask control bit of SSC<br>0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all the rest alarm fields are matched.<br>0x1: SSC[0] is to be compared and all others are ignored<br>0x2: SSC[1:0] is to be compared and all others are ignored<br>0x3: SSC[2:0] is to be compared and all others are ignored<br>0x4: SSC[3:0] is to be compared and all others are ignored<br>0x5: SSC[4:0] is to be compared and all others are ignored<br>0x6: SSC[5:0] is to be compared and all others are ignored<br>0x7: SSC[6:0] is to be compared and all others are ignored<br>0x8: SSC[7:0] is to be compared and all others are ignored<br>0x9: SSC[8:0] is to be compared and all others are ignored<br>0xA: SSC[9:0] is to be compared and all others are ignored<br>0xB: SSC[10:0] is to be compared and all others are ignored<br>0xC: SSC[11:0] is to be compared and all others are ignored<br>0xD: SSC[12:0] is to be compared and all others are ignored<br>0xE: SSC[13:0] is to be compared and all others are ignored<br>0xF: SSC[14:0] is to be compared and all others are ignored<br>**Note:** The bit 15 of synchronous counter (SSC[15] in RTC_SS) is never compared. |
| 23:15 | Reserved | Must be kept at reset value |
| 14:0 | SSC[14:0] | Alarm sub second value<br>This value is the alarm sub second value which is to be compared with synchronous prescaler counter SSC. Bit number is controlled by MSKSSC bits. |

## 13.4.16. Backup registers (RTC_BKPx) (x=0..4)

Address offset: 0x50~0x60

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | Data<br>These registers can be wrote or read by software. The content remains valid even in power saving mode. Tamper detection flag TPxF assertion will reset these registers. Also when the FMC readout protection disables will reset these registers. |

# 14. Timer (TIMERx)

**Table 14-1. Timers (TIMERx) are devided into six sorts**

| TIMER | TIMER0 | TIMER2 | TIMER13 | TIMER14 | TIMER15/16 | TIMER5 |
|---|---|---|---|---|---|---|
| **TYPE** | Advanced | General-L0 | General-L2 | General-L3 | General-L4 | Basic |
| **Prescaler** | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit |
| **Counter** | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit |
| **Count mode** | UP,DOWN, Center-aligned | UP,DOWN, Center-aligned | UP ONLY | UP ONLY | UP ONLY | UP ONLY |
| **Repetition** | ● | × | × | ● | ● | × |
| **CH Capture/ Compare** | 4 | 4 | 1 | 2 | 1 | 0 |
| **Complementary & Dead-time** | ● | × | × | ● | ● | × |
| **Break** | ● | × | × | ● | ● | × |
| **Single Pulse** | ● | ● | × | ● | ● | ● |
| **Quadrature Decoder** | ● | ● | × | × | × | × |
| **Master-slave management** | ● | ● | × | ● | × | × |
| **Inter connection** | ●(1) | ●(2) | × | ●(3) | × | × |
| **DMA** | ● | ● | × | ● | ● | ●(4) |
| **Debug Mode** | ● | ● | ● | ● | ● | ● |

(1) TIMER0 **ITI0:** TIMER14_TRGO **ITI1:** 0 **ITI2:** TIMER2_TRGO **ITI3:** 0

(2) TIMER2 **ITI0:** TIMER0_TRGO **ITI1:** 0 **ITI2:** TIMER14_TRGO **ITI3:** 0

(3) TIMER14 **ITI0:** 0 **ITI1:** TIMER2_TRGO **ITI2:** 0 **ITI3:** 0

(4) Only update events will generate a DMA request. TIMER5 do not have DMAS bit (DMA request source selection).

# 14.1. Advanced timer (TIMERx, x=0)

## 14.1.1. Overview

The advanced timer module (TIMER0) is a four-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The advanced timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the advanced timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time insertion module which is suitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counter value increasing in unison.

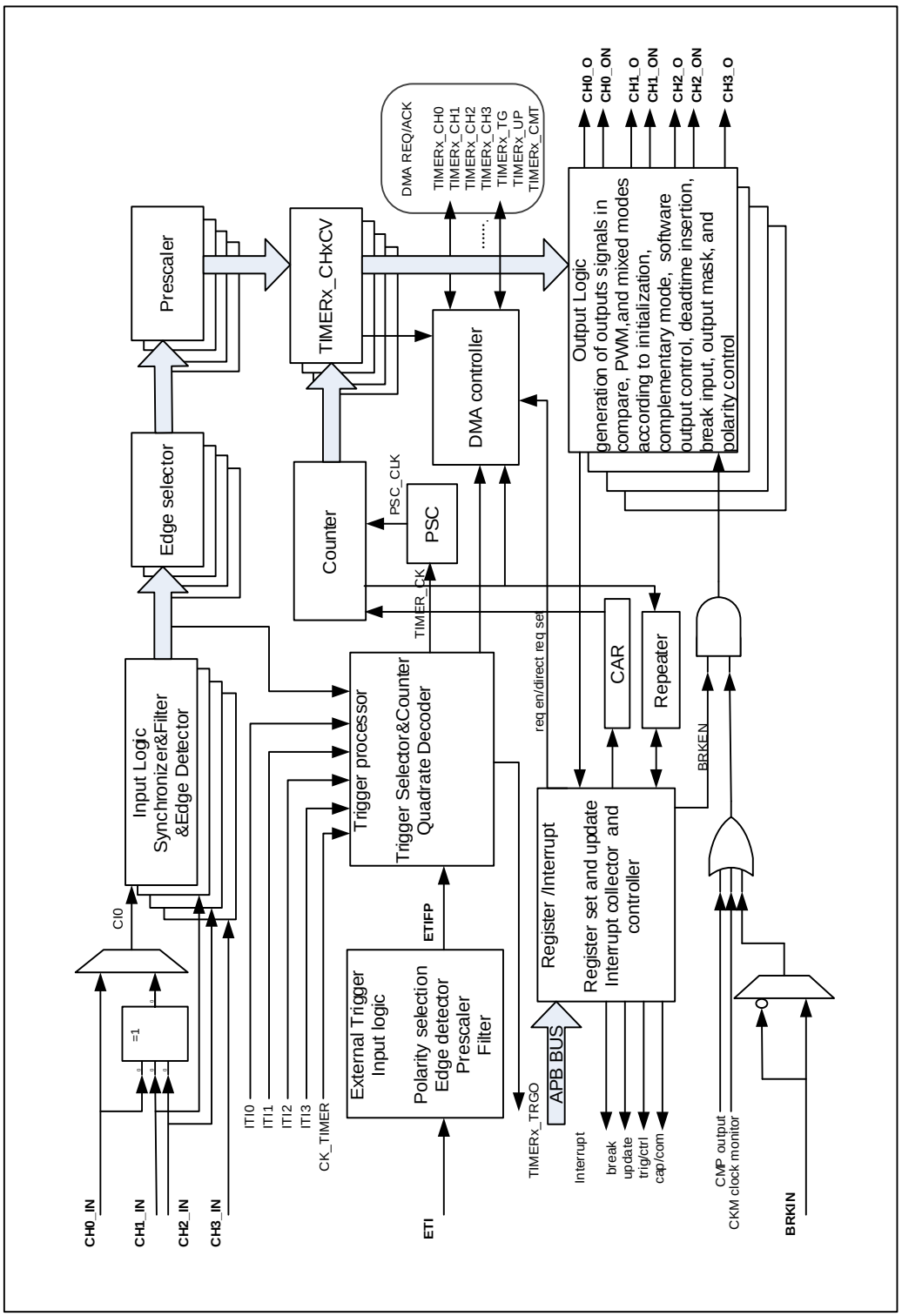
## 14.1.2. Characteristics

- Total channel num: 4.
- Counter width: 16 bits.
- Clock source of timer is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16 bits. The factor can be changed ongoing.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode and single pulse mode.
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request: update event, trigger event, compare/capture event commutation event and break input.
- Daisy chaining of timer module allows a single timer to start multiple timers.
- Timer synchronization allows the selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 14.1.3. Block diagram

*Figure 14-1. Advanced timer block diagram* provides details of the internal configuration of the advanced timer.

**Figure 14-1. Advanced timer block diagram**




**Note:** Only TIMER0 supports CMP output as the break source input.

## 14.1.4. Function overview

### Clock source configuration

The advanced timer has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

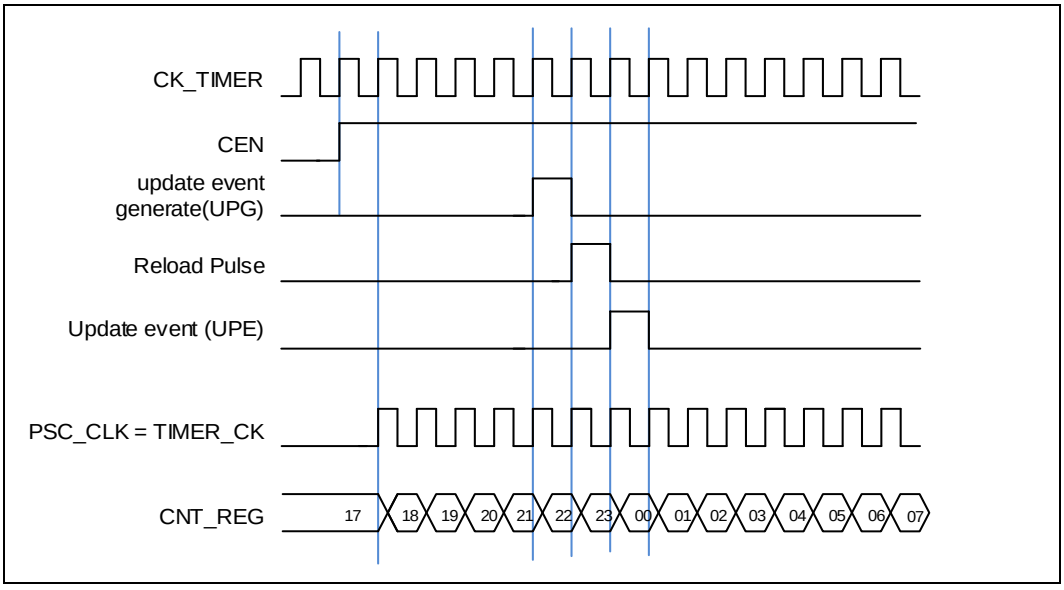
- SMC [2:0] == 3'b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when the SMC [2:0] == 3'b000. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, which drives counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register, details as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

**Figure 14-2. Timing chart of internal clock divided by 1**




- SMC [2:0] == 3'b111 (external clock mode 0). External input pin is selected as timer clock source

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0,

0x1, 0x2 or 0x3.

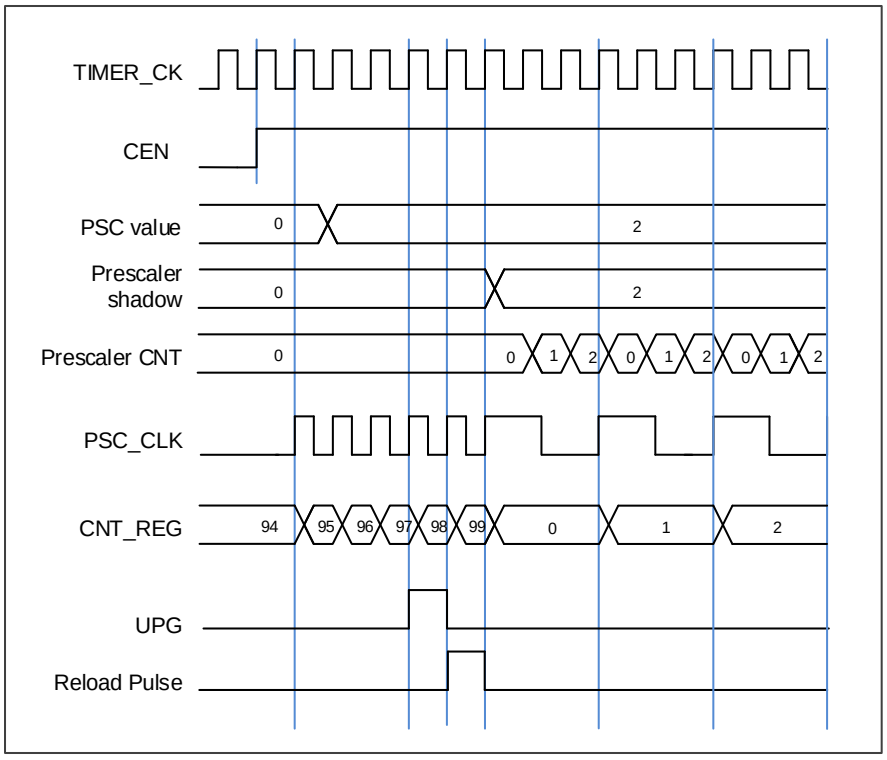
- SMC1== 1'b1 (external clock mode 1). External input ETI is selected as timer clock source (ETI)

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

### Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-3. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again and an overflow event

will be generated. In addition, the update events will be generated after (TIMERx_CREP+1) times of overflow events. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the shadow registers (repetition counter, counter auto reload register, prescaler register) are updated.

*Figure 14-4. Timing chart of up counting mode, PSC=0/2* and *Figure 14-5. Timing chart of up counting mode, change TIMERx_CAR on the go* show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-4. Timing chart of up counting mode, PSC=0/2**

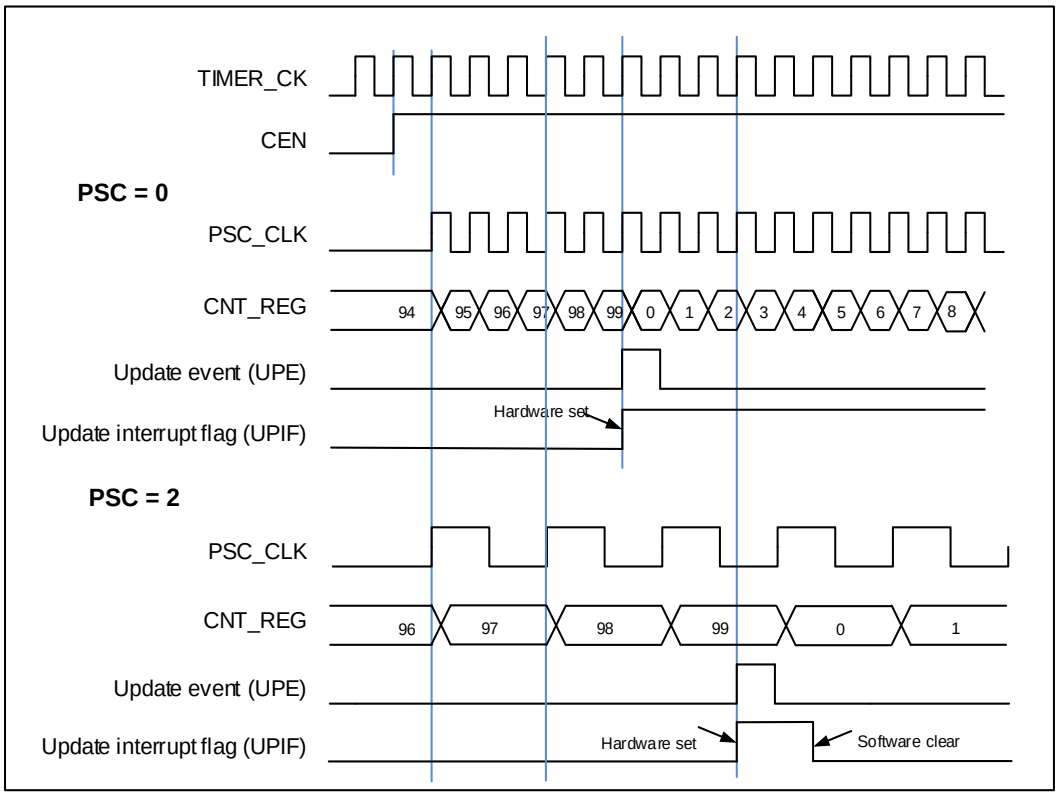
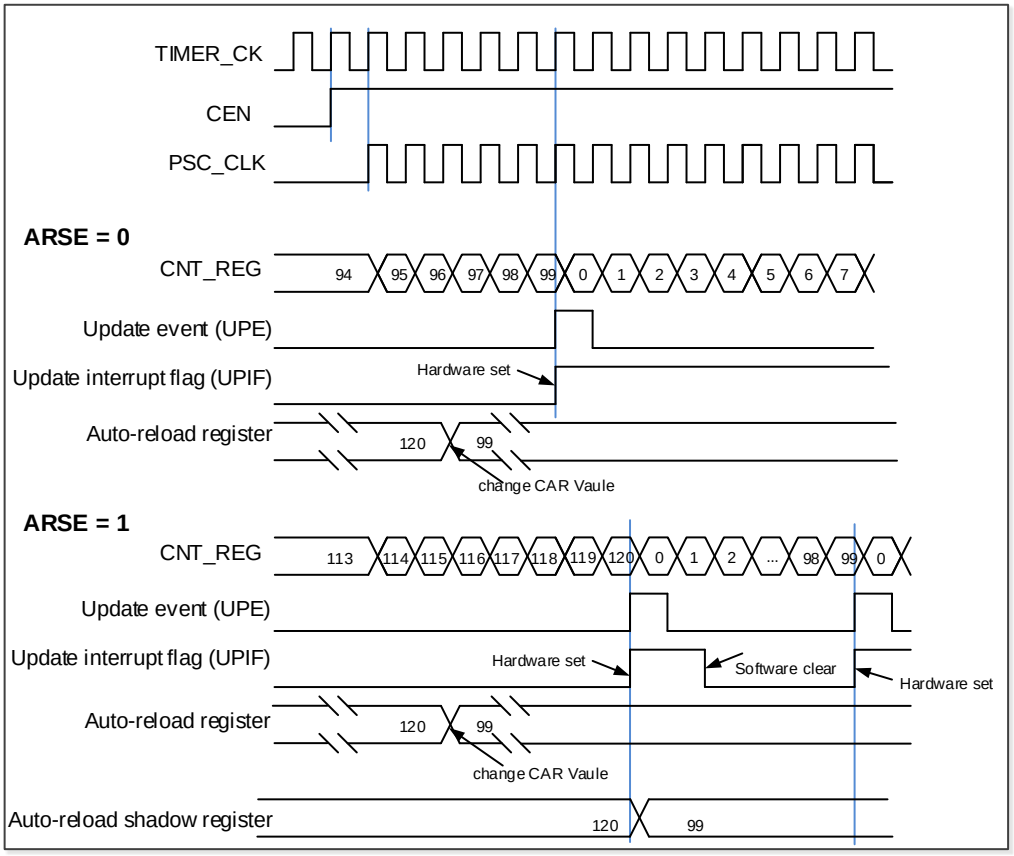


**Figure 14-5. Timing chart of up counting mode, change TIMERx_CAR on the go**




## Counter down counting

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter the counter will start counting down from the counter-reload value again and an underflow event will be generated. In addition, the update event will be generated after (TIMERx_CREP+1) times of underflow. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down-counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to the counter-reload value and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the shadow registers (repetition counter, counter auto reload register, prescaler register) are updated.

*Figure 14-6. Timing chart of down counting mode, PSC=0/2* and *Figure 14-7. Timing chart of down counting mode, change TIMERx_CAR on the go* show some examples of the counter behavior in different clock frequencies when TIMERx_CAR=0x99.

**Figure 14-6. Timing chart of down counting mode, PSC=0/2**

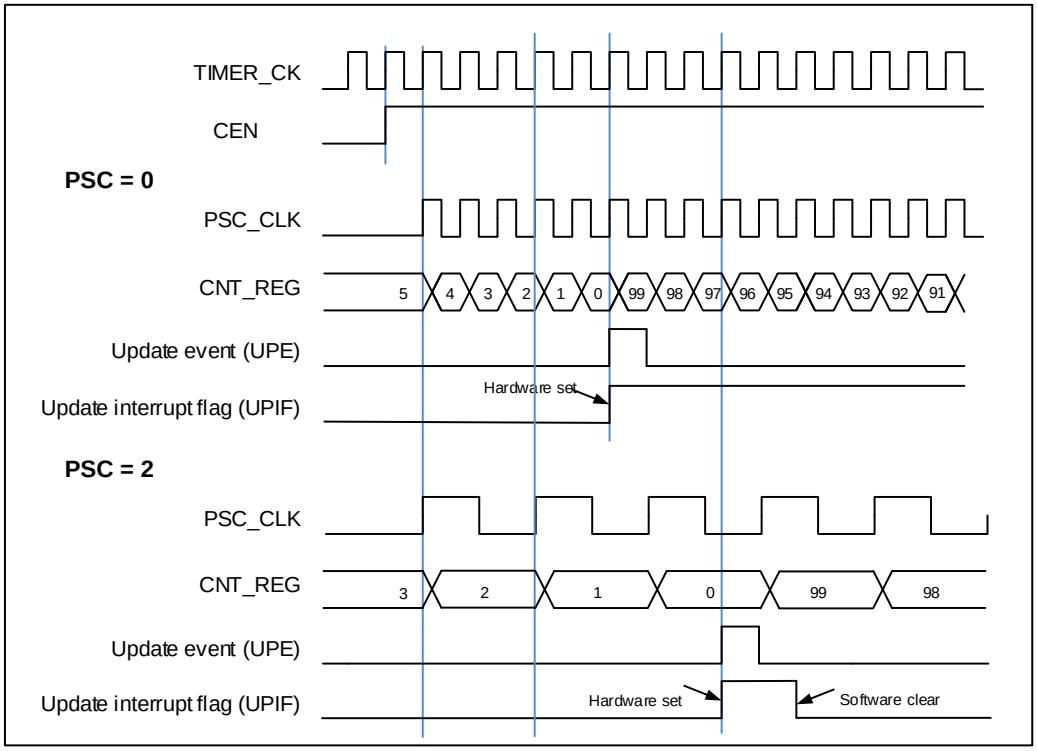
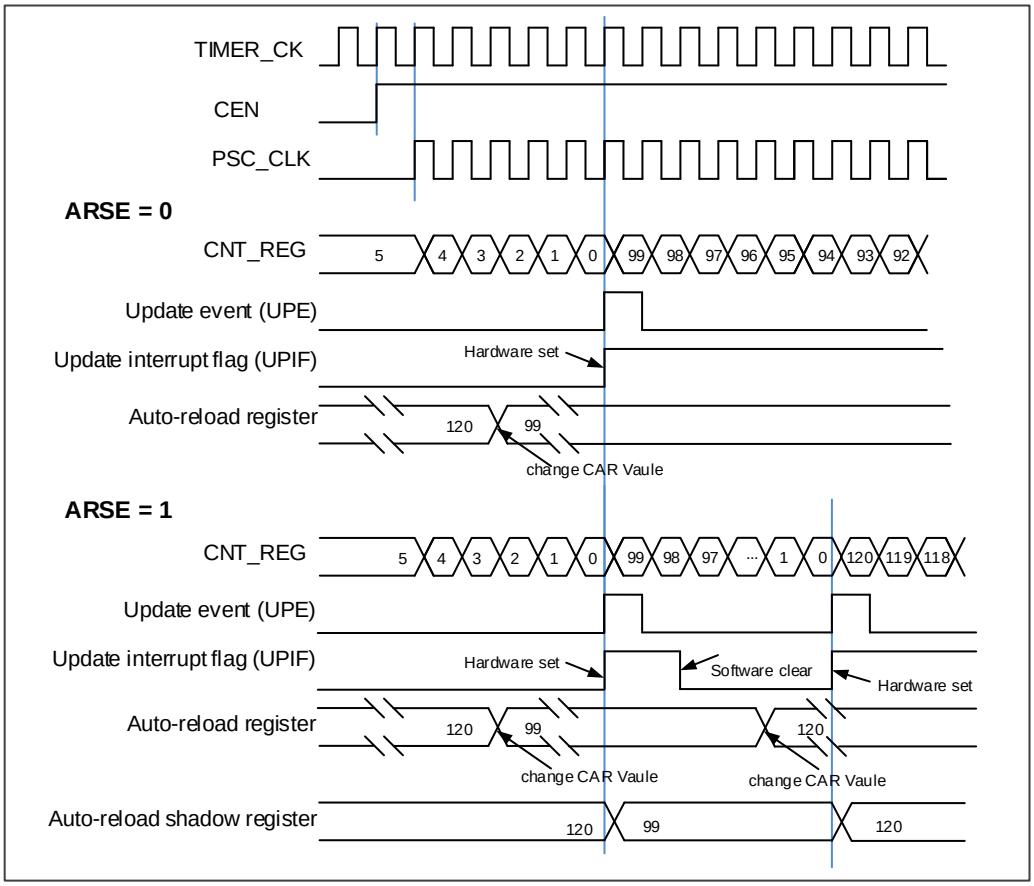



**Figure 14-7. Timing chart of down counting mode, change TIMERx_CAR on the go**



### Counter center-aligned counting

In this mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The Timer module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

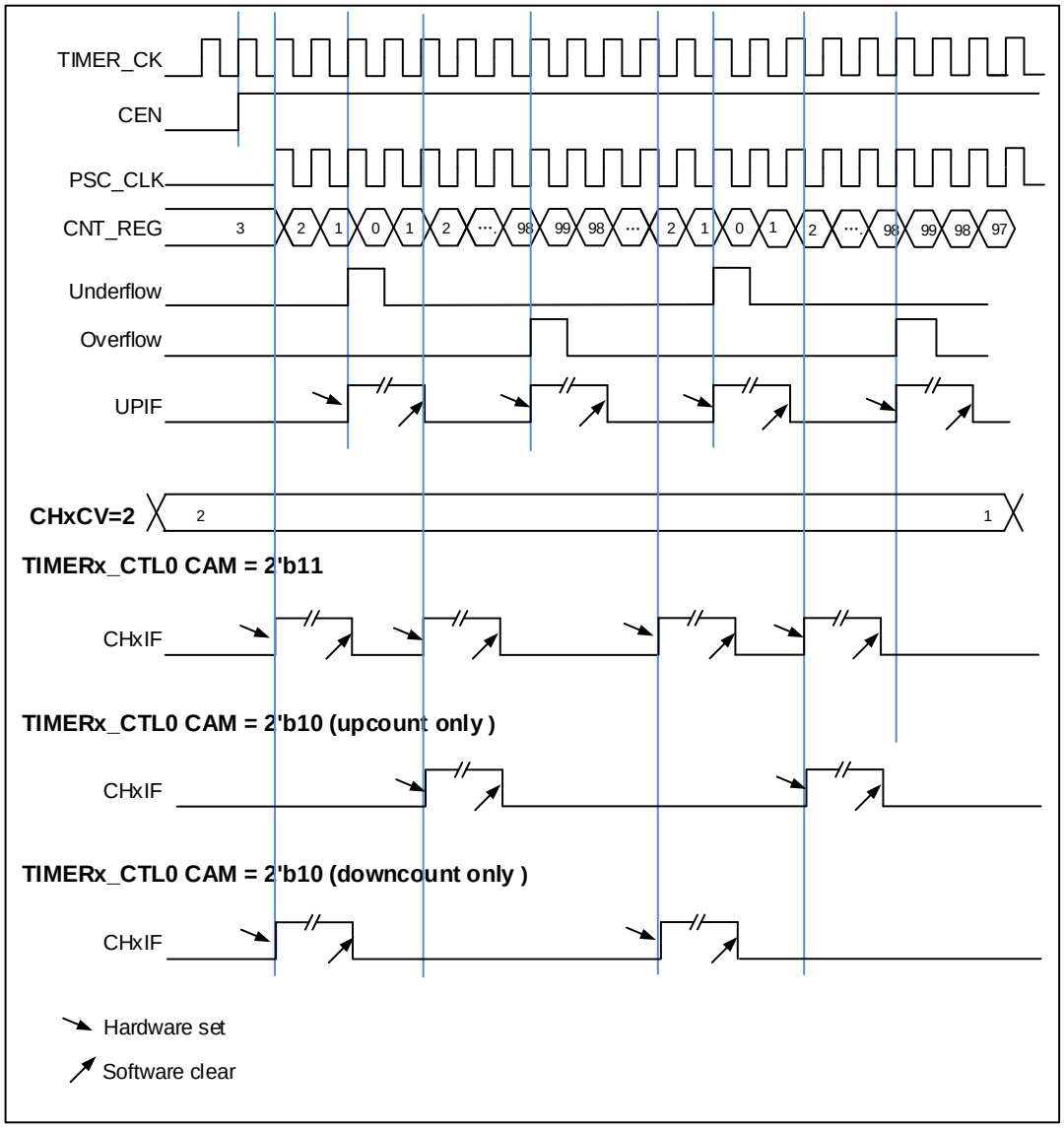
The UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 14-8. Timing chart of center-aligned counting mode.***

If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the shadow registers (repetition counter, counter auto-reload register, prescaler register) are updated.

***Figure 14-8. Timing chart of center-aligned counting mode*** show some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0

**Figure 14-8. Timing chart of center-aligned counting mode**




### Update event (from overflow/underflow) rate configuration

The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx_CREP register. Counter repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in TIMERx_CREP register. The repetition counter is decremented at each counter overflow (does not exist in down counting mode) and underflow (does not exist in up counting mode).

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP in TIMERx_CREP register and generator an update event.

The new written CREP value will not take effect until the next update event. When the value of CREP is odd, and the counter is counting in center-aligned mode, the update event is generated (on overflow or underflow) depending on when the written CREP value takes

effect. If an update event is generated by software after writing an odd number to CREP, the update events will be generated on the underflow. If the next update event occurs on overflow after writing an odd number to CREP, then the subsequent update events will be generated on the overflow.

**Figure 14-9. Repetition counter timing chart of center-aligned counting mode**

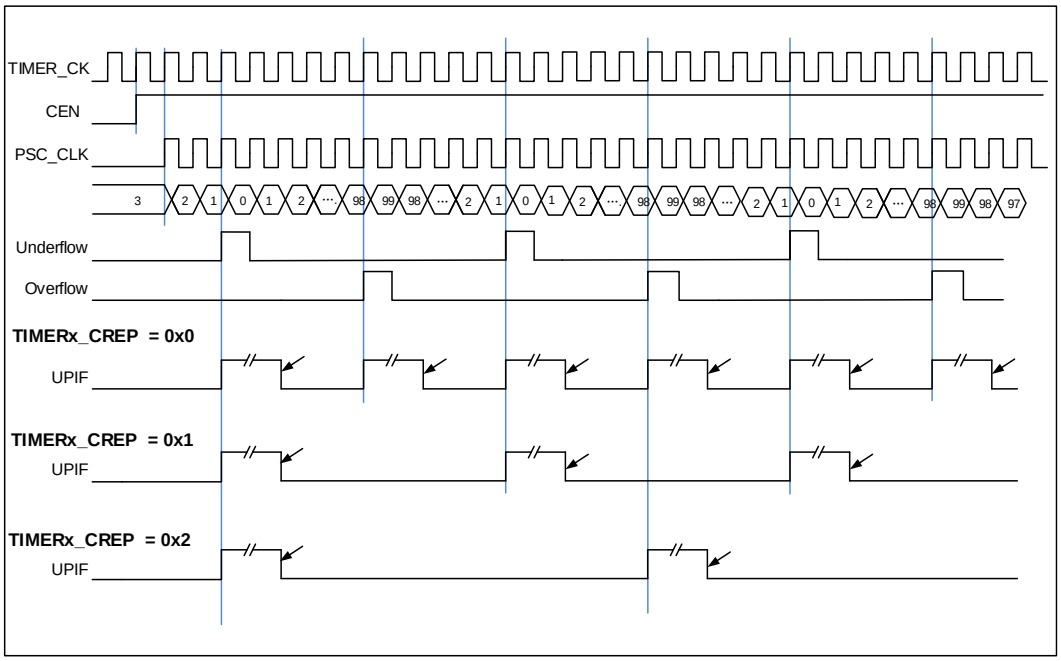



**Figure 14-10. Repetition counter timing chart of up counting mode**

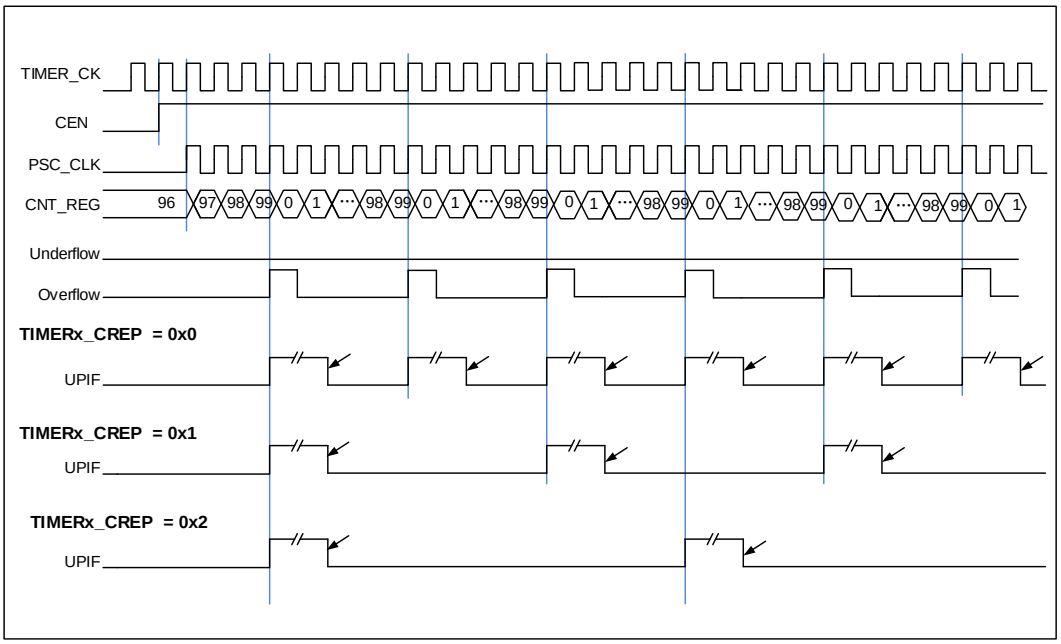
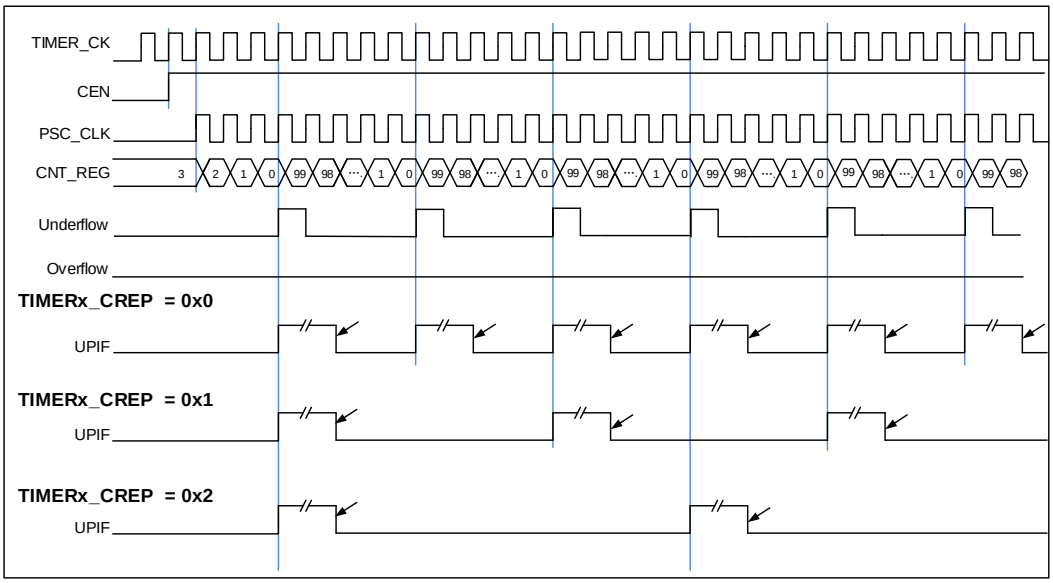


**Figure 14-11. Repetition counter timing chart of down counting mode**



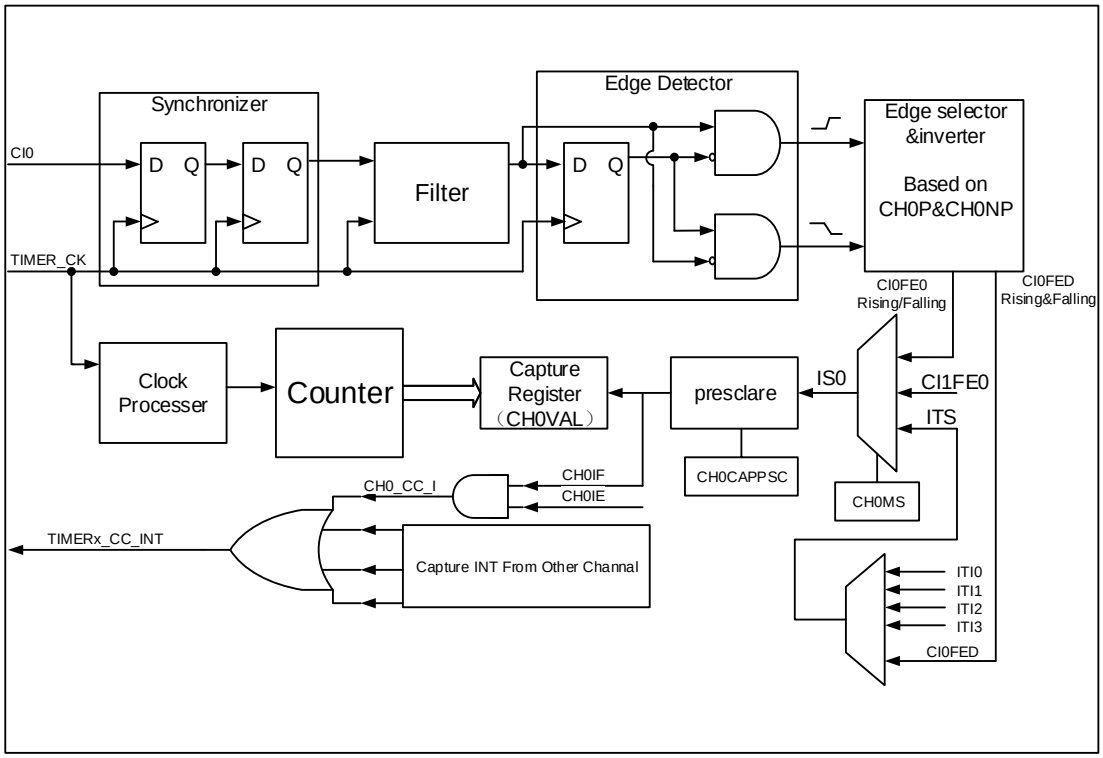

## Input capture and output compare channels

The advanced timer has four independent channels which can be used as capture inputs or compare outputs. Each channel is built around a channel capture compare register including an input stage, a channel controller and an output stage.

- **Channel input capture function**

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if it is enabled when CHxIE=1.

**Figure 14-12. Channel input capture principle**




The input signals of channelx (CIx) can be the TIMERx_CHx signal or the XOR signal of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the input signal of channel (CIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT in TIMERx_CHCTL0).

Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT.

**Step2**: Edge selection (CHxP/CHxNP in TIMERx_CHCTL2).

Rising edge or falling edge, choose one by configuring CHxP/CHxNP bits.

**Step3**: Capture source selection (CHxMS in TIMERx_CHCTL0).

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN in TIMERx_DMAINTEN).

Enable the related interrupt to get the interrupt and DMA request.

**Step5**: Capture enable (CHxEN in TIMERx_CHCTL2).

**Result**: When the wanted input signal is captured, TIMERx_CHxCV will be set by counter's value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

The channel input capture function can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty cycle.

- **Channel output compare function**

**Figure 14-13. Channel output compare principle (with complementary output, x=0,1,2)**

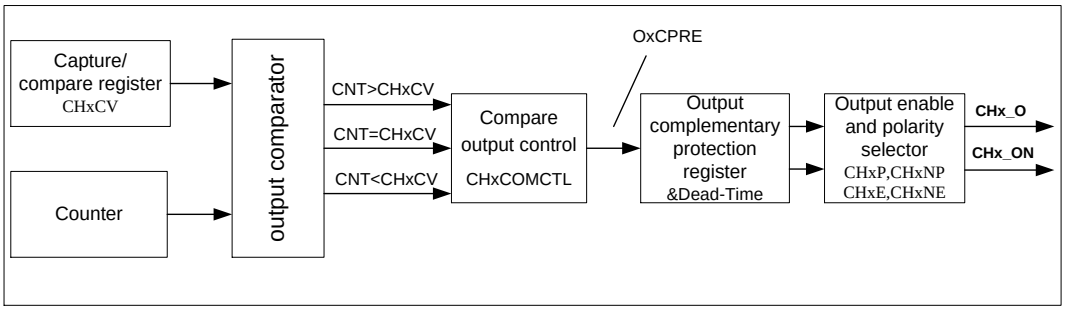



**Figure 14-14. Channel output compare principle (CH3_O)**

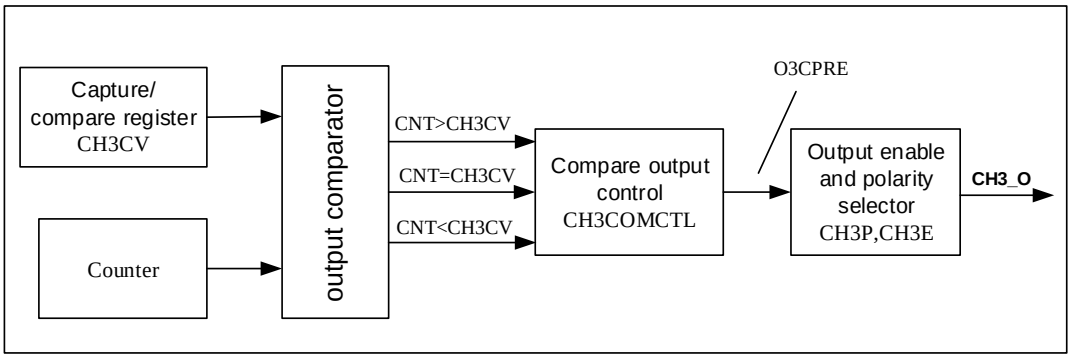



*Figure 14-13. Channel output compare principle (with complementary output, x=0,1,2)*and *Figure 14-14. Channel output compare principle (CH3_O)* show the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O/CHx_ON and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O/CH0_ON depends on OxCPRE signal, CHxP/CHxNP bit and CH0E/CH0NE bit (please refer to the TIMERx_CHCTL2 register for more details). For examples,

1) Configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxE=1 (the output of CHx_O is enabled),
   If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
   If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

2) Configure CHxNP=0 (the active level of CHx_ON is low, contrary to OxCPRE), CHxNE=1 (the output of CHx_ON is enabled),
   If the output of OxCPRE is active(high) level, the output of CHx_O is active(low) level;
   If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(high) level.

When CH0_O and CH0_ON are output at the same time, the specific outputs of CH0_O and CH0_ON are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register..

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CxCDE=1.

So, the process can be divided into several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL.
- Select the active polarity by CHxP/CHxNP.
- Enable the output by CHxEN.

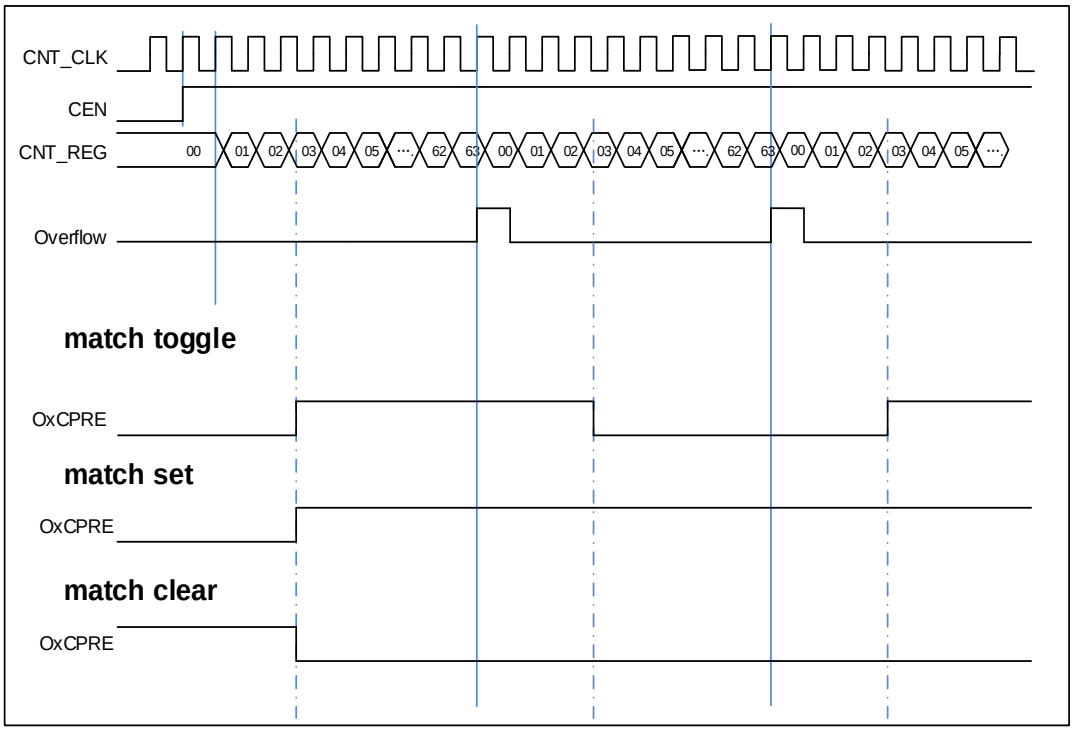
**Step3:** Interrupt/DMA request enable configuration by CHxIE/CxCDE.

**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.
The TIMERx_CHxCV can be changed onging to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

***Figure 14-15. Output-compare in three modes*** shows the three compare modes: toggle/set/clear. CAR=0x63, CHxVAL=0x3.

**Figure 14-15. Output-compare in three modes**




## Output PWM function

In the output PWM function (by setting the CHxCOMCTL bit to 3'b110 (PWM mode 0) or to 3'b 111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined by TIMERx_CHxCV. ***Figure 14-16. Timing chart of EAPWM*** shows the EAPWM output and interrupts waveform.

The CAPWM's period is determined by 2*TIMERx_CAR, and the duty cycle is determined by 2*TIMERx_CHxCV. ***Figure 14-17. Timing chart of CAPWM*** shows the CAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 0 (CHxCOMCTL=3'b110). And if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 1 (CHxCOMCTL=3'b111).

**Figure 14-16. Timing chart of EAPWM**

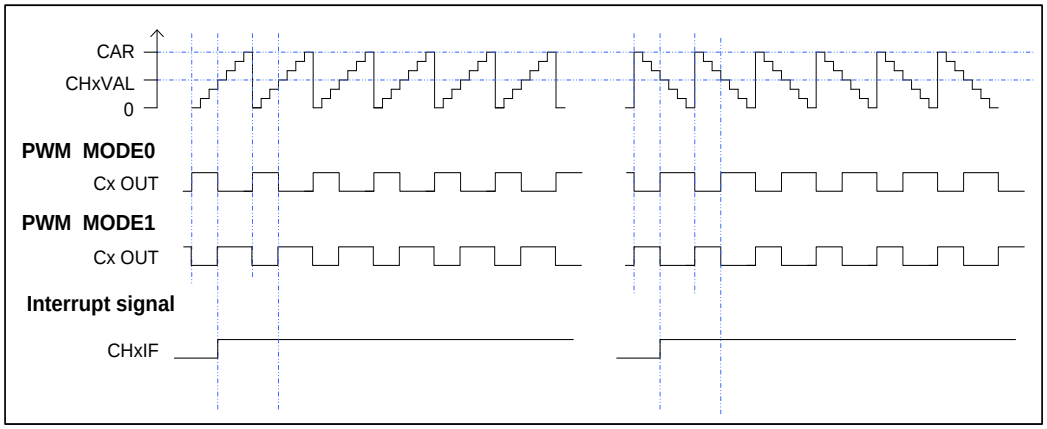
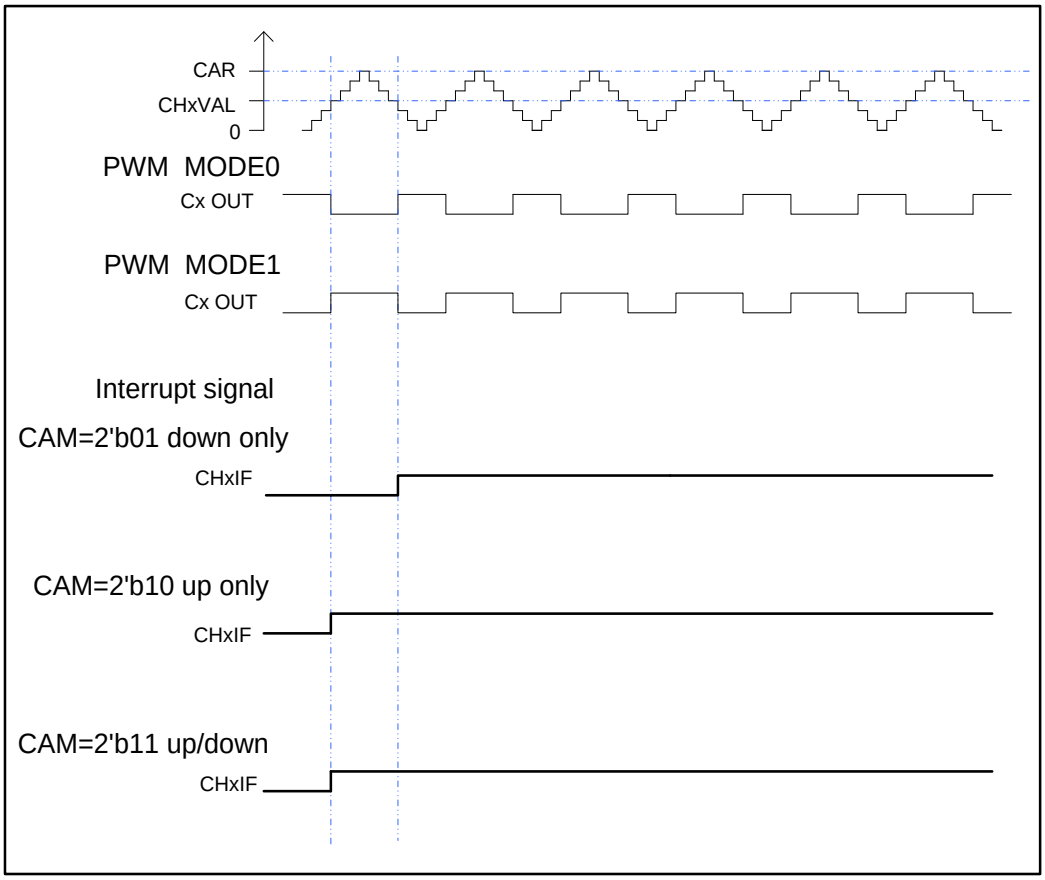



**Figure 14-17. Timing chart of CAPWM**




### Channel output prepare signal

As is shown in ***Figure 14-13. Channel output compare principle (with complementary output, x=0,1,2)*** when TIMERx is configured in compare match output mode,a middle signal which is OxCPRE signal (Channel x output prepare signal) will be generated before the channel outputs signal. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit. The OxCPRE signal has several types of output function. These include keeping the

original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is a forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Channel output complementary PWM

Function of complementary is for a pair of channels, CHx_O and CHx_ON, the two output signals cannot be active at the same time. The TIMERx has 4 channels, but only the first three channels have this function. The complementary signals CHx_O and CHx_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register.

**Table 14-2. Complementary outputs controlled by parameters**

<table>
<tr><th colspan="5">Complementary Parameters</th><th colspan="2">Output Status</th></tr>
<tr><th>POEN</th><th>ROS</th><th>IOS</th><th>CHxEN</th><th>CHxNEN</th><th>CHx_O</th><th>CHx_ON</th></tr>
<tr><td rowspan="5">0</td><td rowspan="5">0/1</td><td rowspan="3">0</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1).</td></tr>
<tr><td>1</td><td colspan="2" rowspan="3">CHx_O/ CHx_ON output “off-state” (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3)</td></tr>
<tr><td rowspan="2">1</td><td>0</td></tr>
<tr><td>1</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td colspan="2">CHx_O/ CHx_ON output “off-state”:<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN.</td></tr>
<tr><td>1</td><td>0</td><td>0/1</td><td>0</td><td>0</td><td colspan="2">CHx_O/CHx_ON = LOW<br>CHx_O/CHx_ON output disable.</td></tr>
</table>

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | 1 | CHx_O = LOW<br>CHx_O output disable. | CHx_ON =OxCPRE⊕<br>(4)CHxNP<br>CHx_ON output enable. |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | CHx_ON = LOW<br>CHx_ON output disable. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | CHx_ON =(!OxCPRE)(5)⊕<br>CHxNP.<br>CHx_ON output enable. |
| | 1 | | 0 | 0 | CHx_O = CHxP<br>CHx_O output “off-state”. | CHx_ON = CHxNP<br>CHx_ON output “off-state”. |
| | | | | 1 | CHx_O = CHxP<br>CHx_O output “off-state” | CHx_ON =OxCPRE⊕CHxNP<br>CHx_ON output enable |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON = CHxNP<br>CHx_ON output “off-state”. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON =(!OxCPRE)⊕<br>CHxNP<br>CHx_ON output enable. |

**Note:**

(1) Output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “Off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (!OxCPRE)：the complementary output of the OxCPRE signal.

### Insertion dead time for complementary PWM

The dead time insertion is enabled when both CHxEN and CHxNEN are configured to 1’b1, it is also necessary to configure POEN to 1. The field named DTCFG defines the dead time delay that can be used for all channels except channel 3. Refer to the TIMERx_CCHP register for details about the delay time.

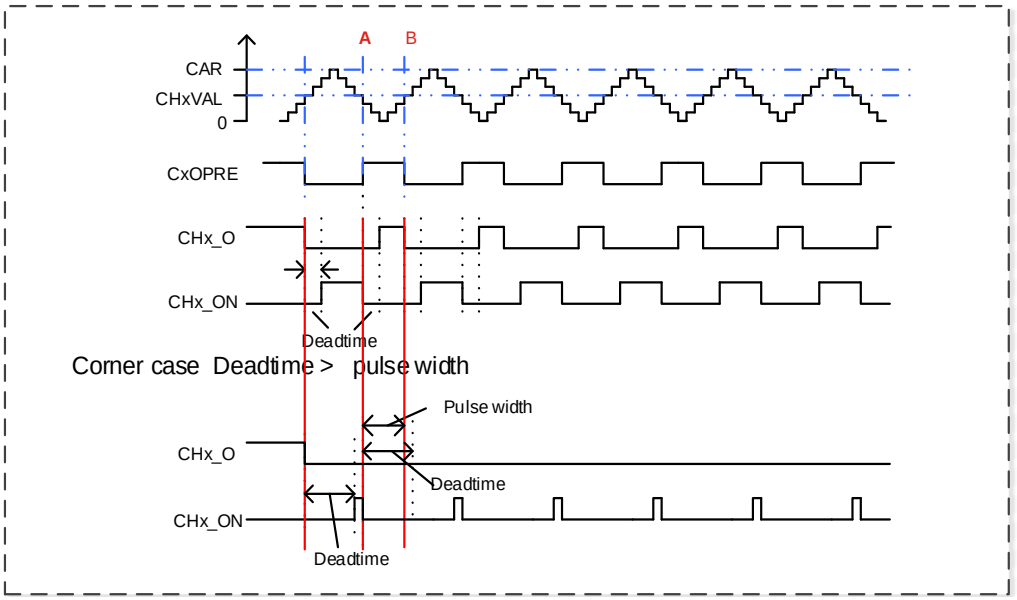
The dead time delay insertion ensures that two complementary signals are not active at the same time.

When the channelx match event (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled in PWM mode 0. At point A in ***Figure 14-18. Complementary output with dead time insertion***, CHx_O signal remains at the low level until the end of the dead time delay, while CHx_ON signal will be cleared at once. Similarly, at point B when the channelx match event (TIMERx counter = CHxVAL) occurs again, OxCPRE is cleared, and CHx_O signal will be cleared at once, while CHx_ON signal remains at the low level until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example: the dead

time delay is greater than or equal to the duty cycle of the CHx_O signal, then the CHx_O signal is always inactive (As shown in ***Figure 14-18. Complementary output with dead time insertion***).

**Figure 14-18. Complementary output with dead time insertion**



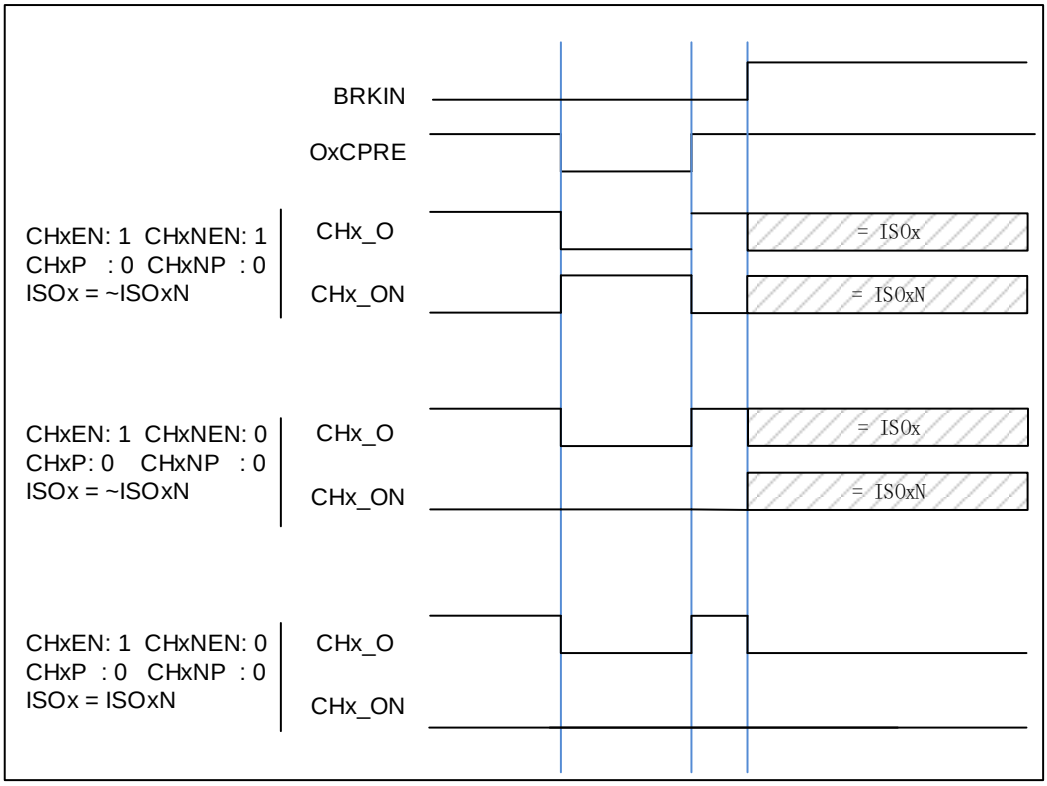

## Break mode

In this mode, CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. In any case, CHx_O and CHx_ON signals cannot be set to active level at the same time. The break sources are input break pin, CMP output (only TIMER0 support) and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx_CCHP register. The polarity of BRKIN break input is setting by the BRKP bit in TIMERx_CCHP. It should be noted that the break input from CMP is valid at a high level and there is no polarity selection in the TIMER module.

When a break occurs, the POEN bit is cleared asynchronously. As soon as POEN is 0, the level of the CHx_O and CHx_ON outputs are determined by the ISOx and ISOxN bits in the TIMERx_CTL1 register. If IOS is 0, the timer releases the enable output, otherwise, the enable output remains high. The complementary outputs are first in the reset state, and then the dead time generator is reactivated to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead time.

When a break occurs, the BRKIF bit in the TIMERx_INTF register will be set. If BRKIE is 1, an interrupt will be generated.

**Figure 14-19. Output behavior of the channel in response to a break (the break high active)**




## Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0FE0 and CI1FE1 derived from the TIMERx_CH0 and TIMERx_CH1 pins respectively to interact to control the counter value. The DIR bit is modified during each input source transition. The counter can be changed by the edges of CI0FE0 only, CI1FE1 only or both CI0FE0 and CI1FE1, the selection mode by setting the SMC[2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in ***Table 14-3. Counting direction in different quadrature decoder mode***. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-period value. Therefore, TIMERx_CAR register must be configured before the counter starts to count.

**Table 14-3. Counting direction in different quadrature decoder mode**

| Counting mode | Level | CI0FE0 | | CI1FE1 | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |
| Quadrature decoder mode 1 SMC[2:0]=3'b010 | CI1FE1=1 | Down | Up | - | - |
| | CI1FE1=0 | Up | Down | - | - |
| Quadrature decoder mode 0 SMC [2:0]=3'b001 | CI0FE0=1 | - | - | Up | Down |
| | CI0FE0=0 | - | - | Down | Up |
| Quadrature decoder mode 2 | CI1FE1=1 | Down | Up | X | X |

| | | | | | |
|---|---|---|---|---|---|
| SMC [2:0]=3'b011 | CI1FE1=0 | Up | Down | X | X |
| | CI0FE0=1 | X | X | Up | Down |
| | CI0FE0=0 | X | X | Down | Up |

**Note:** "-" means "no counting"; "X" means impossible. "0" means "low level", "1" means "high level".

**Figure 14-20. Counter behavior with CI0FE0 polarity non-inverted in mode 2**

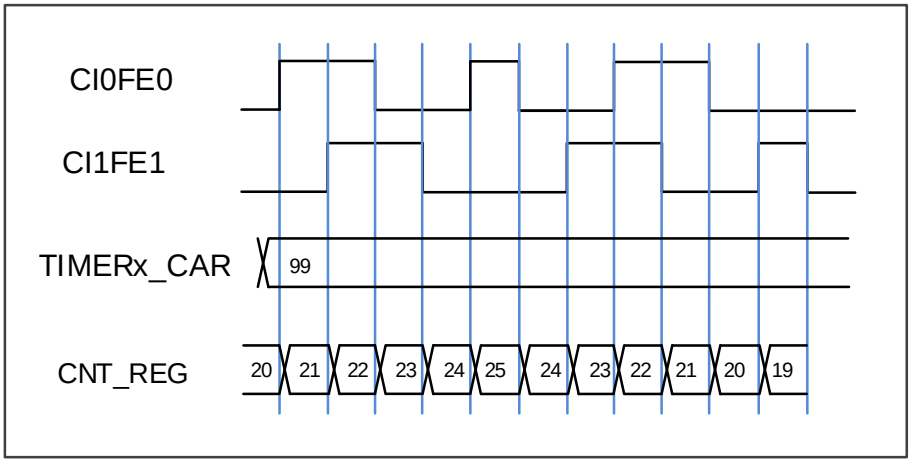
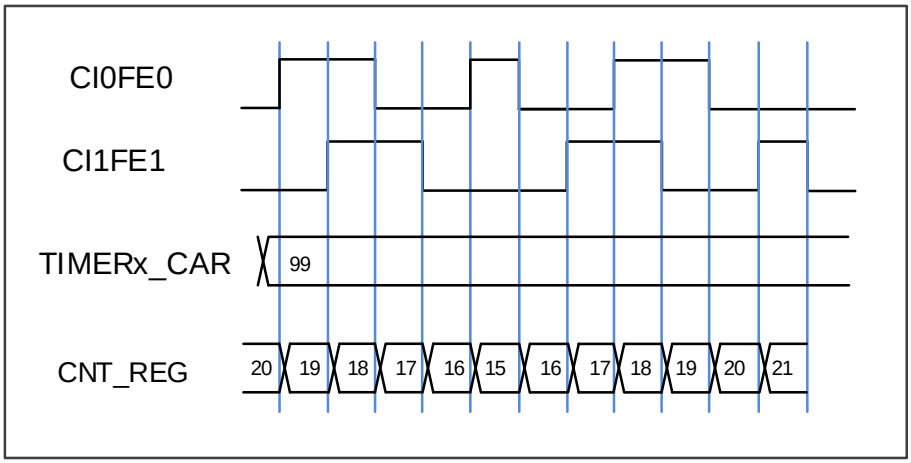



**Figure 14-21. Counter behavior with CI0FE0 polarity inverted in mode 2**




### Hall sensor function

Hall sensor is generally used to control BLDC Motor; the timers can support this function.

*Figure 14-22. Hall sensor is used to BLDC motor* show how to connect. And we can see we need two timers. First TIMER_in (Advanced/GeneralL0 TIMER) should accept three HALL sensor signals.

Each of the three input of HALL sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

By the internal connection such as TRGO-ITIx, TIMER_in and TIMER_out can be connected.

TIMER_out will generate PWM signal to control BLDC motor's speed based on the ITRx. Then, the feedback circuit is finished, also you change configuration to fit your request.

About the TIMER_in, it need have input XOR function, so you can choose from Advanced/GeneralL0 TIMER.

And TIMER_out need have functions of complementary and Dead-time, so only advanced timer can be chosen. Else, based on the timers' internal connection relationship, pair's timers can be selected.

TIMER_in (TIMER2) -> TIMER_out (TIMER0 ITI2)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI0S, then, each of input signal change will make the CI0 toggle. CH0VAL will record the value of counter at that moment.
- Enable ITIx connected to commutation function directly by setting CCUC and CCSE.
- Configuration PWM parameter based on your request.

**Figure 14-22. Hall sensor is used to BLDC motor**

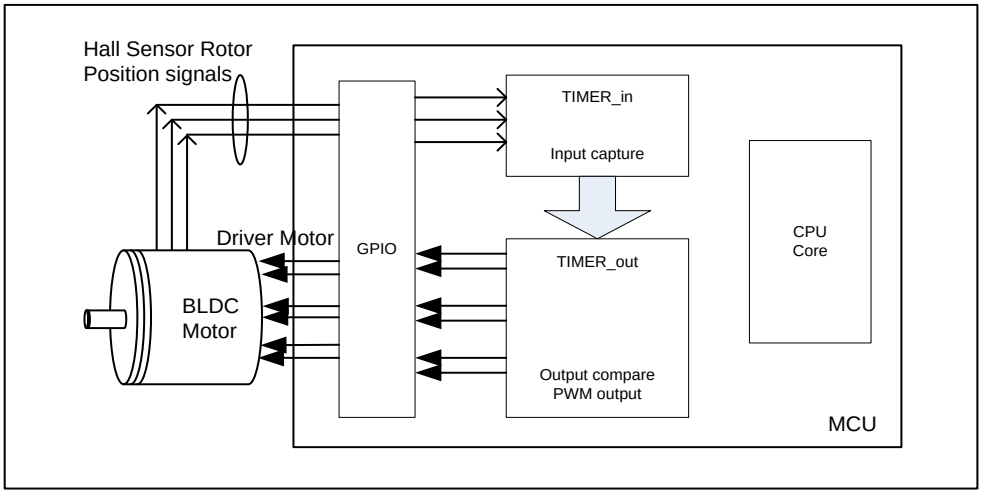
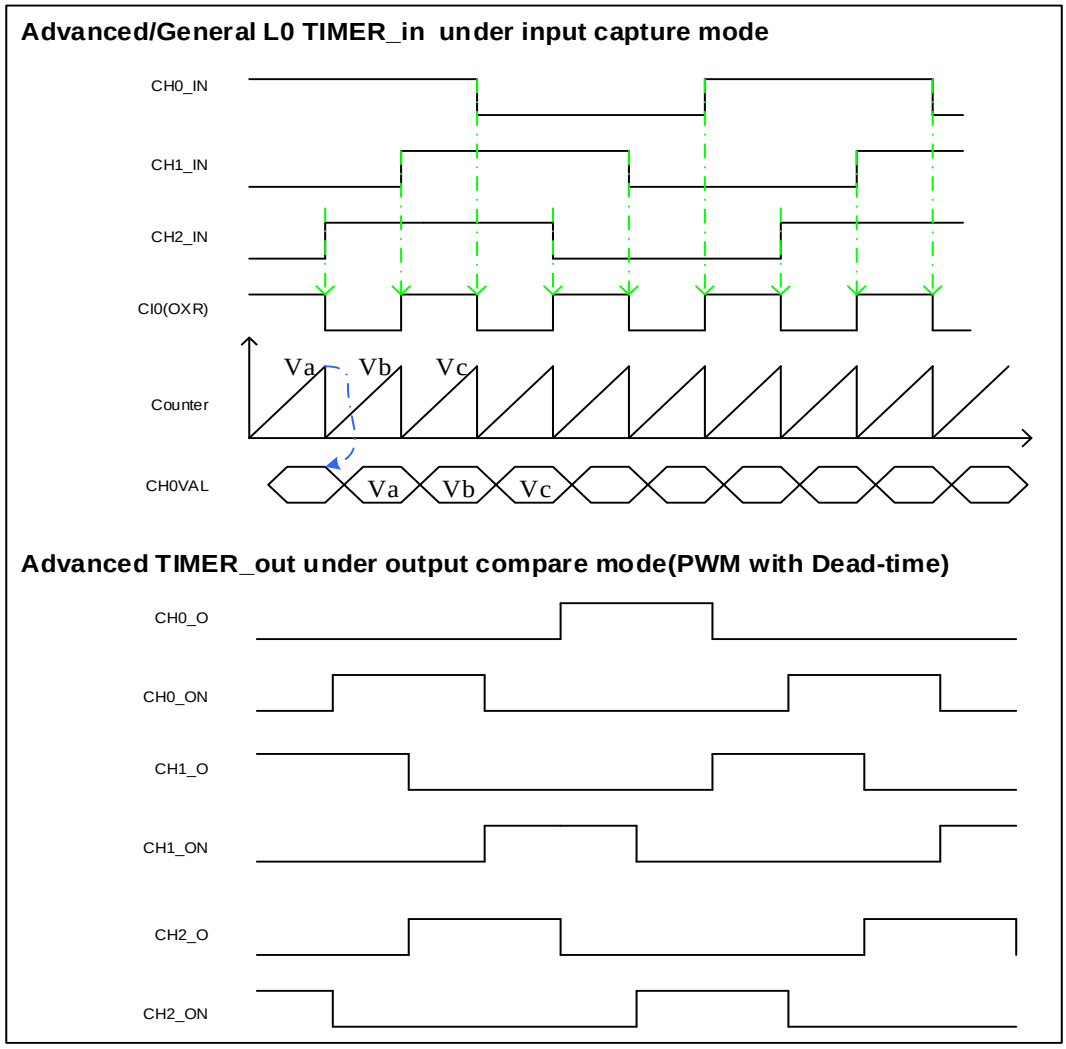


**Figure 14-23. Hall sensor timing between two timers**

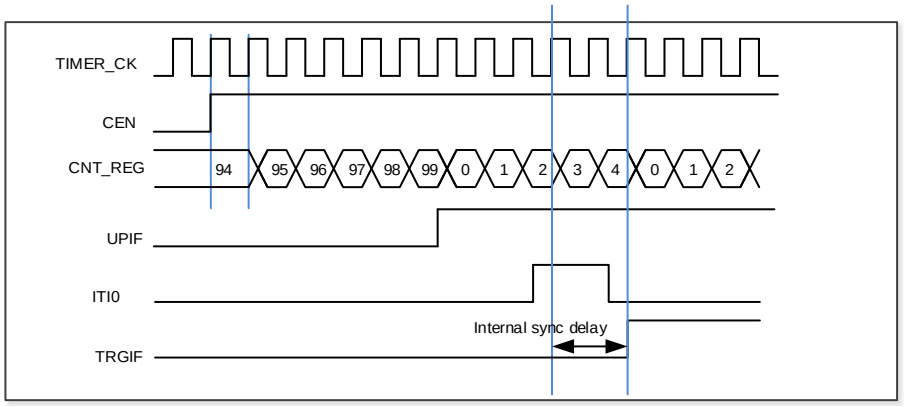
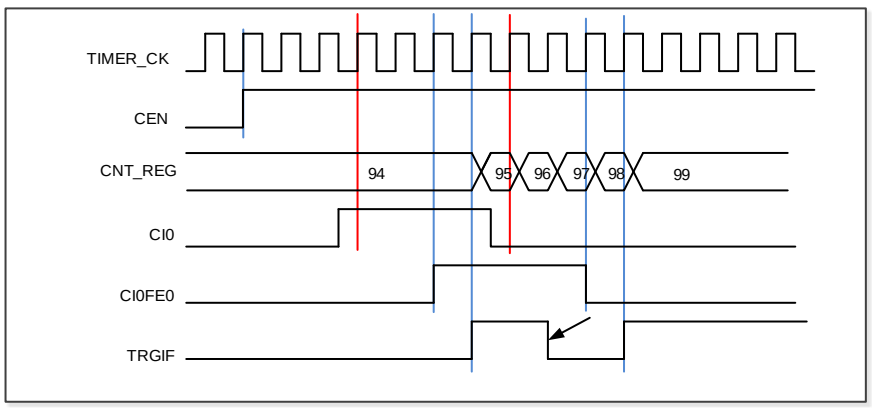



## Master-slave management

The TIMERx can be synchronized with a trigger in several modes including restart mode, pause mode and event mode which is selected by the SMC[2:0] bits in the TIMERx_SMCFG register. The input trigger of these modes can be selected by the TRGS[2:0] bits in the TIMERx_SMCFG register.

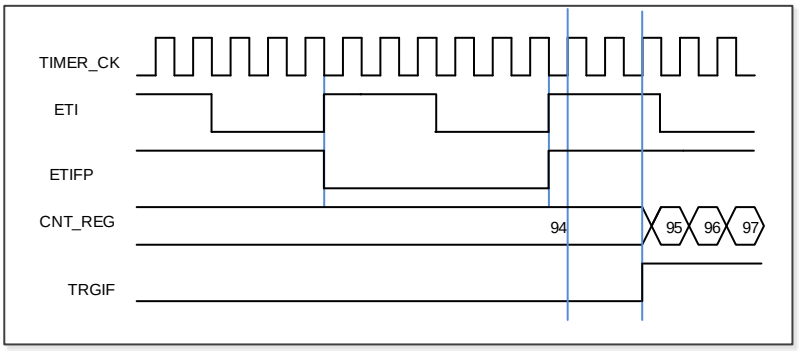
**Table 14-4. Examples of slave mode**

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| **LIST** | SMC[2:0]<br>3'b100 (restart mode)<br>3'b101 (pause mode)<br>3'b110 (event mode) | TRGS[2:0]<br>000: ITI0<br>001: ITI1<br>010: ITI2<br>011: ITI3<br>100: CI0F_ED<br>101: CI0FE0 | If CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion.<br>If ETIFP is selected as | For the ITIx, no filter and prescaler can be used.<br>For the CIx, filter can be used by configuring CHxCAPFLT, no prescaler can be used. |

<table>
<tr><th></th><th>Mode Selection</th><th>Source Selection</th><th>Polarity Selection</th><th>Filter and Prescaler</th></tr>
<tr><td></td><td></td><td>110: CI1FE1<br>111: ETIFP</td><td>the trigger source, configure the ETP for polarity selection and inversion.</td><td>For the ETIFP, filter can be used by configuring ETFC and prescaler can be used by configuring ETPSC.</td></tr>
<tr><td rowspan="2">Exam1</td><td>Restart mode<br>The counter will be cleared and restart when a rising edge of trigger input comes.</td><td>TRGS[2:0] = 3’b000<br>ITI0 is selected.</td><td>For ITI0, no polarity selector can be used.</td><td>For the ITI0, no filter and prescaler can be used.</td></tr>
<tr><td colspan="4">Figure 14-24. Restart mode<br>
</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode<br>The counter will be paused when the trigger input is low, and it will start when the trigger input is high.</td><td>TRGS[2:0]=3’b101<br>CI0FE0 is selected.</td><td>TI0S=0 (Non-xor) [CH0NP=0, CH0P=0] CI0FE0 does not invert. The capture event will occur on the rising edge only.</td><td>Filter is bypassed in this example.</td></tr>
<tr><td colspan="4">Figure 14-25. Pause mode<br>
</td></tr>
</table>

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| Exam3 | **Event mode**<br>The counter will start to count when a rising edge of trigger input comes. | TRGS[2:0] =3'b111<br>ETIFP is selected. | ETP = 0, the polarity of ETI does not change. | ETPSC = 1, ETI is divided by 2.<br>ETFC = 0, ETI does not filter. |

**Figure 14-26. Event mode**




### Single pulse mode

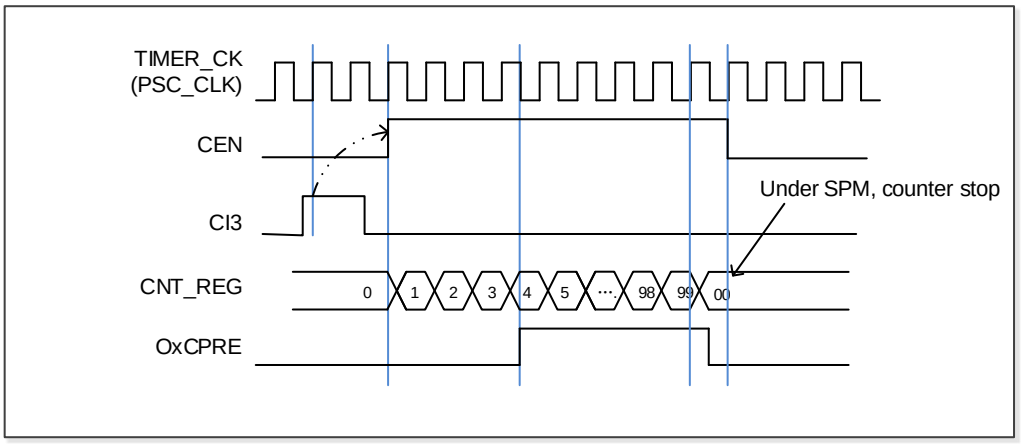
Single pulse mode is enabled by setting SPM in TIMERx_CTL0. If SPM is set, the counter will be cleared and stopped when the next update event occurs. In order to get a pulse waveform, the TIMERx is configured to PWM mode or compare mode by CHxCOMCTL.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 by software, the counter will be stopped and its value will be held.

In the single pulse mode, the active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in TIMERx_CHCTL0/1 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The CHxCOMFEN bit is available only when the output channel is configured to the PWM mode 0 or PWM mode 1 and the trigger source is derived from the trigger signal.

*Figure 14-27. Single pulse mode TIMERx_CHxCV=4, TIMERx_CAR=99* shows an example.

**Figure 14-27. Single pulse mode TIMERx_CHxCV=4, TIMERx_CAR=99**

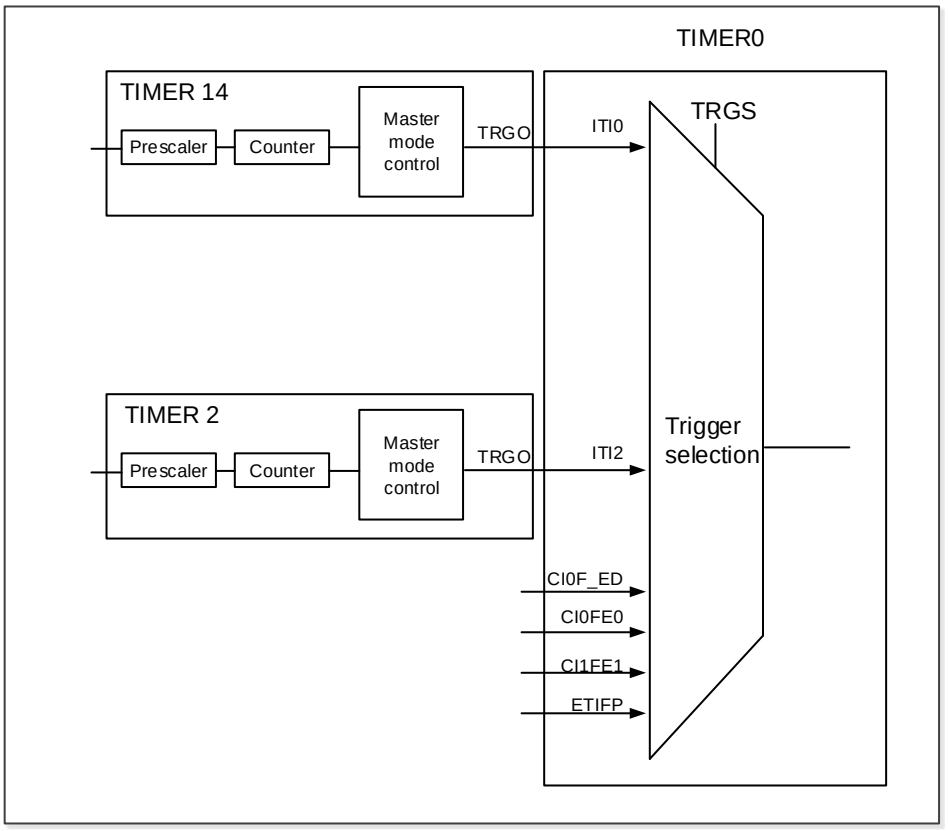



## Timers interconnection

Timer can be configured as interconnection, that is, one timer which operate in the master mode outputs TRGO signal to control another timer which operate in the slave mode, TRGO include reset evevt, start evevt, update evevt, capture/compare pulse evevt, compare evevt. slave timer received the ITIx and performs the corresponding mode, include internal clock mode, quadrature decoder mode, restart mode, pause mode, event mode, external clock mode.

*Figure 14-28. TIMER0 Master/Slave mode timer example* shows the timer0 trigger selection when it is configured in slave mode.

**Figure 14-28. TIMER0 Master/Slave mode timer example**




Other interconnection examples:

- TIMER2 as prescaler for TIMER0

We configure TIMER2 as a prescaler for TIMER0. Refer to ***Figure 14-28. TIMER0 Master/Slave mode timer example*** for connections. Do as follow:

1. Configure TIMER2 in master mode and select its Update Event (UPE) as trigger output (MMC=010 in the TIMER2_CTL1 register). Then TIMER2 drives a periodic signal on each counter overflow.
2. Configure the TIMER2 period (TIMER2_CAR registers).
3. Select the TIMER0 input trigger source from TIMER2 (TRGS=010 in the TIMER0_SMCFG register).
4. Configure TIMER0 in external clock mode 1 (SMC=111 in TIMER0_SMCFG register).
5. Start TIMER0 by writing '1 in the CEN bit (TIMER0_CTL0 register).
6. Start TIMER2 by writing '1 in the CEN bit (TIMER2_CTL0 register).

- Start Timer0 with Timer2's Enable/Update signal

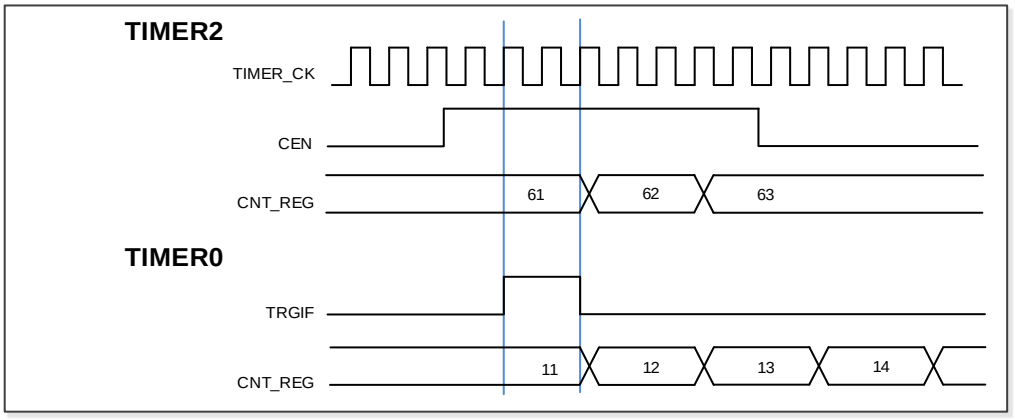
In this example, we enable Timer0 with the enable output of Timer2. Refer to ***Figure 14-29. Triggering TIMER0 with enable signal of TIMER2***. Timer0 starts counting from its current value on the divided internal clock after trigger by Timer2 enable output.

When Timer0 receives the trigger signal, its CEN bit is set and the counter counts until we disable timer0. In this example, both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT_CLK}$ = $f_{TIMER_CK}$/3). Timer0's SMC is set as event

mode, so Timer0 can not be disabled by Timer2's disable signal. Do as follow:

1. Configure Timer2 master mode to send its enable signal as trigger output (MMC=3'b001 in the TIMER2_CTL1 register)
2. Configure Timer0 to select the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
3. Configure Timer0 in event mode (SMC=3'b 110 in TIMERx_SMCFG register).
4. Start Timer2 by writing 1 in the CEN bit (TIMER2_CTL0 register).

**Figure 14-29. Triggering TIMER0 with enable signal of TIMER2**



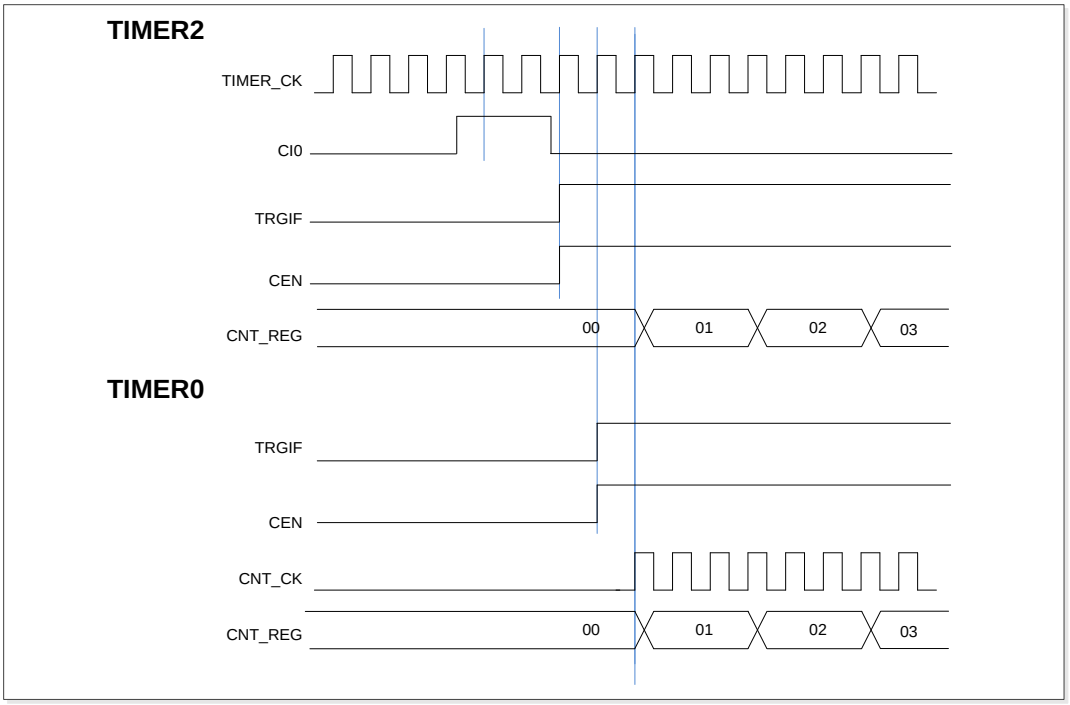

■ Using an external trigger to start 2 timers synchronously

We configure the start of TIMER0 is triggered by the enable of TIMER2, and TIMER2 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, TIMER2 must be configured in Master/Slave mode. Do as follow:

1. Configure TIMER2 slave mode to get the input trigger from CI0 (TRGS=100 in the TIMER2_SMCFG register).
2. Configure TIMER2 in event mode (SMC=110 in the TIMER2_SMCFG register).
3. Configure the TIMER2 in Master/Slave mode by writing MSM=1 (TIMER2_SMCFG register).
4. Configure TIMER0 to get the input trigger from TIMER2 (TRGS=010 in the TIMER0_SMCFG register).
5. Configure TIMER0 in event mode (SMC=110 in the TIMER0_SMCFG register).

When a rising edge occurs on TIMER2's CI0, two timer counters starts counting synchronously on the internal clock and both TRGIF flags are set.

**Figure 14-30. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input**




## Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx_DMATB. In fact, TIMERx_DMATB register is only a buffer, timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0 (1 transfer), the timer sends only one DMA request. While if TIMERx_DMATC is not 0, such as 3 (4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8 and DMATA+0xC at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

## Timer debug mode

When the Cortex®-M23 is halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register is set to 1, the TIMERx counter stops.

### 14.1.5. TIMERx registers(x=0)

TIMER0 base address: 0x4001 2C00

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CKDIV[1:0] | | ARSE | CAM[1:0] | | DIR | SPM | UPS | UPDIS | CEN |
| | | | | | | rw | | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS).<br>00: $f_{DTS}=f_{CK_TIMER}$<br>01: $f_{DTS}= f_{CK_TIMER}$ /2<br>10: $f_{DTS}= f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:5 | CAM[1:0] | Counter aligns mode selection<br>00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.<br>01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting down, CHxF bit can be set.<br>10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting up, CHxF bit can be set.<br>11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when counting up and counting down, CHxF bit can be set. |

| | | |
|---|---|---|
| | | After the counter is enabled, cannot be switched from 0x00 to non 0x00. |
| 4 | DIR | Direction<br>0: Count up<br>1: Count down<br>If the timer work in center-aligned mode or quadrature decoder mode, this bit is read only. |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode disable. The counter continues after update event.<br>1: Single pulse mode enable. The counter counts until the next update event occurs. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: This event generates update interrupts or DMA requests:<br>The counter generates an overflow or underflow event |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: Update event disable.<br>**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decoder mode. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | ISO3 | ISO2N | ISO2 | ISO1N | ISO1 | ISO0N | ISO0 | TI0S | MMC[2:0] | | | DMAS | CCUC | Reserved | CCSE |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value |
| 14 | ISO3 | Idle state of channel 3 output<br>Refer to ISO0 bit |
| 13 | ISO2N | Idle state of channel 2 complementary output<br>Refer to ISO0N bit |
| 12 | ISO2 | Idle state of channel 2 output<br>Refer to ISO0 bit |
| 11 | ISO1N | Idle state of channel 1 complementary output<br>Refer to ISO0N bit |
| 10 | ISO1 | Idle state of channel 1 output<br>Refer to ISO0 bit |
| 9 | ISO0N | Idle state of channel 0 complementary output<br>0: When POEN bit is reset, CH0_ON is set low.<br>1: When POEN bit is reset, CH0_ON is set high<br>This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output<br>0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high<br>The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 7 | TI0S | Channel 0 trigger input selection<br>0: The TIMERx_CH0 pin input is selected as channel 0 trigger input.<br>1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input. |
| 6:4 | MMC[2:0] | Master mode control<br>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.<br>000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source:<br>Master timer generate a reset |

| | | |
|---|---|---|
| | | the UPG bit in the TIMERx_SWEVG register is set<br>001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source :<br>CEN control bit is set<br>The trigger input in pause mode is high<br>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.<br>011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.<br>100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.<br>101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE.<br>110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE.<br>111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE. |
| 3 | DMAS | DMA request source selection<br>0: When capture or compare event occurs, the DMA request of channel x is sent<br>1: When update event occurs, the DMA request of channel x is sent. |
| 2 | CCUC | Commutation control shadow register update control<br>When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:<br>0: The shadow registers update by when CMTG bit is set.<br>1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.<br>When a channel does not have a complementary output, this bit has no effect. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled.<br>1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled.<br>After these bits have been written, they are updated based when commutation event coming.<br>When a channel does not have a complementary output, this bit has no effect. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETP | SMC1 | ETPSC[1:0] | | ETFC[3:0] | | | | MSM | TRGS[2:0] | | | OCRC | SMC[2:0] | | |
| rw | rw | rw | | rw | | | | rw | rw | | | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal<br>0: ETI is active at rising edge or high level .<br>1: ETI is active at falling edge or low level . |
| 14 | SMC1 | Part of SMC for enable External clock mode1<br>In external clock mode 1, the counter is clocked by any active edge on the ETIF signal.<br>0: External clock mode 1 disabled<br>1: External clock mode 1 enabled.<br>When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case.<br>The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time.<br>**Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed. |
| 13:12 | ETPSC[1:0] | The prescaler of external trigger<br>The frequency of external trigger signal ETIFP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETIFP frequency.<br>00: Prescaler disable.<br>01: The prescaler is 2.<br>10: The prescaler is 4.<br>11: The prescaler is 8. |
| 11:8 | ETFC[3:0] | External trigger filter control<br>The external trigger can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the external trigger signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| EXTFC[3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |

| | | |
|---|---|---|
| 4’b0001 | 2 | $f_{CK_TIMER}$ |
| 4’b0010 | 4 | |
| 4’b0011 | 8 | |
| 4’b0100 | 6 | $f_{DTS_CK}/2$ |
| 4’b0101 | 8 | |
| 4’b0110 | 6 | $f_{DTS_CK}/4$ |
| 4’b0111 | 8 | |
| 4’b1000 | 6 | $f_{DTS_CK}/8$ |
| 4’b1001 | 8 | |
| 4’b1010 | 5 | $f_{DTS_CK}/16$ |
| 4’b1011 | 6 | |
| 4’b1100 | 8 | |
| 4’b1101 | 5 | $f_{DTS_CK}/32$ |
| 4’b1110 | 6 | |
| 4’b1111 | 8 | |

7 MSM

Master-slave mode

This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.

0: Master-slave mode disable

1: Master-slave mode enable

6:4 TRGS[2:0]

Trigger selection

This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.

000: ITI0

001: ITI1

010: ITI2

011: ITI3

100: CI0F_ED

101: CI0FE0

110: CI1FE1

111: ETIFP

These bits must not be changed when slave mode is enabled.

3 OCRC

OCPRE clear source selection

0: OCPRE clear source is connected to the OCPRE_CLR input

1: OCPRE clear source is connected to ETIFP

2:0 SMC[2:0]

Slave mode control

000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high.

001: Quadrature decoder mode 0.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level.

010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level.
011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other.
100: Restart Mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.
101: Pause Mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.
110: Event Mode. A rising edge of the trigger input enables the counter.
111: External Clock Mode 0. The counter counts on the rising edges of the selected trigger.
Because CI0F_ED outputs 1 pulse for each transition on CI0F, and the pause mode checks the level of the trigger signal, when CI0F_ED is selected as the trigger input, the pause mode must not be used.

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | TRGDEN | CMTDEN | CH3DEN | CH2DEN | CH1DEN | CH0DEN | UPDEN | BRKIE | TRGIE | CMTIE | CH3IE | CH2IE | CH1IE | CH0IE | UPIE |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: disabled<br>1: enabled |
| 13 | CMTDEN | Commutation DMA request enable<br>0: disabled<br>1: enabled |
| 12 | CH3DEN | Channel 3 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 11 | CH2DEN | Channel 2 capture/compare DMA request enable<br>0: disabled |

| | | |
|---|---|---|
| | | 1: enabled |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 8 | UPDEN | Update DMA request enable<br>0: disabled<br>1: enabled |
| 7 | BRKIE | Break interrupt enable<br>0: disabled<br>1: enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: disabled<br>1: enabled |
| 5 | CMTIE | commutation interrupt enable<br>0: disabled<br>1: enabled |
| 4 | CH3IE | Channel 3 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 3 | CH2IE | Channel 2 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | CH3OF | CH2OF | CH1OF | CH0OF | Reserved | BRKIF | TRGIF | CMTIF | CH3IF | CH2IF | CH1IF | CH0IF | UPIF |
| | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | . | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12 | CH3OF | Channel 3 over capture flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 over capture flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |
| 7 | BRKIF | Break interrupt flag<br>When the break input is inactive, the bit is set by hardware.<br>When the break input is inactive, the bit can be cleared by software.<br>0: No active level break has been detected.<br>1: An active level has been detected. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event.<br>0: No trigger event occurred.<br>1: Trigger interrupt occurred. |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when channel's commutation event occurs, and cleared by software |

| | | |
|---|---|---|
| | | 0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4 | CH3IF | Channel 3 ‘s capture/compare interrupt flag<br>Refer to CH0IF description |
| 3 | CH2IF | Channel 2 ‘s capture/compare interrupt flag<br>Refer to CH0IF description |
| 2 | CH1IF | Channel 1 ‘s capture/compare interrupt flag<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 ‘s capture/compare interrupt flag<br>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No Channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | BRKG | TRGG | CMTG | CH3G | CH2G | CH1G | CH0G | UPG |
| | | | | | | | | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | BRKG | Break event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a break event<br>1: Generate a break event |

| | | |
|---|---|---|
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event<br>1: Generate a trigger event |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel's c/c control update event |
| 4 | CH3G | Channel 3's capture or compare event generation<br>Refer to CH0G description |
| 3 | CH2G | Channel 2's capture or compare event generation<br>Refer to CH0G description |
| 2 | CH1G | Channel 1's capture or compare event generation<br>Refer to CH0G description |
| 1 | CH0G | Channel 0's capture or compare event generation<br>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 1 capture or compare event<br>1: Generate a channel 1 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMCEN | CH1COMCTL[2:0] | | | CH1COMSEN | CH1COMFEN | CH1MS[1:0] | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| CH1CAPFLT[3:0] | | | | CH1CAPPSC[1:0] | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | CH1COMFEN | Channel 1 output compare fast enable<br>Refer to CH0COMFEN description |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is programmed as output mode<br>01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1<br>10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1<br>11: Channel 1 is programmed as input mode, IS1 is connected to ITS.<br>**Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, if the OCPRE clear source signal is detected as high level, the O0CPRE signal will be cleared. The OCPRE clear source signal is selected by OCRC bit in TIMERx_SMCFG register.<br>0: Channel 0 output compare clear disable<br>1: Channel 0 output compare clear enable |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O |

| | | |
|---|---|---|
| | | and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced to low level.<br>101: Force high. O0CPRE is forced to high level.<br>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.<br>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.<br>0: Channel 0 output compare shadow disable<br>1: Channel 0 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH0COMFEN | Channel 0 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.<br>0: Channel 0 output quickly compare disable.<br>1: Channel 0 output quickly compare enable. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in |

TIMERx_CHCTL2 register is reset).).
00: Channel 0 is programmed as output mode
01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0
10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0
11: Channel 0 is programmed as input mode, IS0 is connected to ITS
**Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>Same as Output compare mode |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| CH0CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}/32$ |
| 4'b1110 | 6 | |

| | | |
|---|---|---|
| 4'b1111 | 8 | |

| | | |
|---|---|---|
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.<br>00: Prescaler disable, input capture occurs on every channel input edge<br>01: The input capture occurs on every 2 channel input edges<br>10: The input capture occurs on every 4 channel input edges<br>11: The input capture occurs on every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 mode selection<br>Same as Output compare mode |

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMCEN | CH3COMCTL[2:0] | | | CH3COMSEN | CH3COMFEN | CH3MS[1:0] | | CH2COMCEN | CH2COMCTL[2:0] | | | CH2COMSEN | CH2COMFEN | CH2MS[1:0] | |
| CH3CAPFLT[3:0] | | | | CH3CAPPSC[1:0] | | | | CH2CAPFLT[3:0] | | | | CH2CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | CH3COMCEN | Channel 3 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH3COMCTL[2:0] | Channel 3 compare output control<br>Refer to CH0COMCTL description |
| 11 | CH3COMSEN | Channel 3 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | CH3COMFEN | Channel 3 output compare fast enable<br>Refer to CH0COMFEN description |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH3EN bit in TIMERx_CHCTL2 register is reset). |

| | | |
|---|---|---|
| | | 00: Channel 3 is programmed as output mode<br>01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3<br>10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3<br>11: Channel 3 is programmed as input mode, IS3 is connected to ITS.<br>**Note:** When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable.<br>When this bit is set, if the OCPRE clear source signal is detected as high level, the O0CPRE signal will be cleared. The OCPRE clear source signal is selected by OCRC bit in TIMERx_SMCFG register.<br>0: Channel 2 output compare clear disable<br>1: Channel 2 output compare clear enable |
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT.<br>001: Set the channel output. O2CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH2CV.<br>010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV.<br>011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV.<br>100: Force low. O2CPRE is forced to low level.<br>101: Force high. O2CPRE is forced to high level.<br>110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise.<br>111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise.<br>If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE). |
| 3 | CH2COMSEN | Channel 2 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled.<br>0: Channel 2 output compare shadow disable |

| | | |
|---|---|---|
| | | 1: Channel 2 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH2COMFEN | Channel 2 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH2_O is set to the compare level independently from the result of the comparison.<br>0: Channel 2 output quickly compare disable.<br>1: Channel 2 output quickly compare enable. |
| 1:0 | CH2MS[1:0] | Channel 2 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH2EN bit in TIMERx_CHCTL2 register is reset).).<br>00: Channel 2 is programmed as output mode<br>01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2<br>10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2<br>11: Channel 2 is programmed as input mode, IS2 is connected to ITS.<br>**Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | CH3CAPFLT[3:0] | Channel 3 input capture filter control<br>Refer to CH0CAPFLT description |
| 11:10 | CH3CAPPSC[1:0] | Channel 3 input capture prescaler<br>Refer to CH0CAPPSC description |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>Same as Output compare mode |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI2 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| CH2CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}/32$ |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

3:2 CH2CAPPSC[1:0] Channel 2 input capture prescaler

This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 register is clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH2MS[1:0] Channel 2 mode selection

Same as Output compare mode

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | CH3P | CH3EN | CH2NP | CH2NEN | CH2P | CH2EN | CH1NP | CH1NEN | CH1P | CH1EN | CH0NP | CH0NEN | CH0P | CH0EN |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13 | CH3P | Channel 3 capture/compare function polarity |

| | | |
|---|---|---|
| | | Refer to CH0P description |
| 12 | CH3EN | Channel 3 capture/compare function enable<br>Refer to CH0EN description |
| 11 | CH2NP | Channel 2 complementary output polarity<br>Refer to CH0NP description |
| 10 | CH2NEN | Channel 2 complementary output enable<br>Refer to CH0NEN description |
| 9 | CH2P | Channel 2 capture/compare function polarity<br>Refer to CH0P description |
| 8 | CH2EN | Channel 2 capture/compare function enable<br>Refer to CH0EN description |
| 7 | CH1NP | Channel 1 complementary output polarity<br>Refer to CH0NP description |
| 6 | CH1NEN | Channel 1 complementary output enable<br>Refer to CH0NEN description |
| 5 | CH1P | Channel 1 capture/compare function polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 capture/compare function enable<br>Refer to CH0EN description |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.<br>0: Channel 0 complementary output high level is active level<br>1: Channel 0 complementary output low level is active level<br>When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of input signal.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 2 | CH0NEN | Channel 0 complementary output enable<br>When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 capture/compare function polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 high level is active level |

| | | |
|---|---|---|
| | | 1: Channel 0 low level is active level<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: Reserved.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter.<br>**Note:** When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value. |

### Counter repetition register (TIMERx_CREP)

Address offset: 0x30
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | CREP[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP[7:0] | Counter repetition value<br>This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel0<br>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH1VAL[15:0] | Capture or compare value of channel1<br>When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH2VAL[15:0] | Capture or compare value of channel 2<br>When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH3VAL[15:0] | Capture or compare value of channel 3<br>When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POEN | OAEN | BRKP | BRKEN | ROS | IOS | PROT[1:0] | | DTCFG[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | POEN | Primary output enable<br>The bit can be set to 1 by:<br>- Write 1 to this bit<br>- If OAEN is set to 1, this bit is set to 1 at the next update event.<br>The bit can be cleared to 0 by:<br>- Write 0 to this bit<br>- Valid fault input (asynchronous).<br>When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set. |

| | | |
|---|---|---|
| | | 0: Disable channel outputs (CHxO or CHxON).<br>1: Enabled channel outputs (CHxO or CHxON).<br>**Note:** This bit is only valid when CHxMS=2'b00. |
| 14 | OAEN | Output automatic enable<br>0: The POEN bit can only be set by software.<br>1: POEN can be set at the next update event, if the break input is not active.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 13 | BRKP | Break polarity<br>This bit specifies the polarity of the BRKIN input signal.<br>0: BRKIN input active low<br>1: BRKIN input active high |
| 12 | BRKEN | Break enable<br>This bit can be set to enable the BRKIN and CKM clock failure event inputs.<br>0: Break inputs disabled<br>1: Break inputs enabled<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode "off-state" enable<br>When POEN bit is set (Run mode), this bit can be set to enable the "off-state" for the channels which has been configured in output mode. Please refer to ***Table 14-2. Complementary outputs controlled by parameters***.<br>0: "off-state" disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.<br>1: "off-state" enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is "off-state".<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode "off-state" enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the "off-state" for the channels which has been configured in output mode. Please refer to ***Table 14-2. Complementary outputs controlled by parameters***.<br>0: "off-state" disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.<br>1: "off-state" enabled. No matter the CHxEN/CHxNEN bits, the channels are "off-state".<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 9:8 | PROT[1:0] | Complementary register protect control<br>This bit-filed specifies the write protection property of registers. |

00: protect disable. No write protection.

01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.

10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected.

11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is configured in output) are writing protected.

This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.

7:0 DTCFG[7:0] Dead time configure

The relationship between DTVAL value and the duration of dead-time is as follow:

| DTCFG[7:5] | The duration of dead-time |
|---|---|
| 3'b0xx | DTCFG[7:0] * $t_{DTS_CK}$ |
| 3'b10x | (64+ DTCFG[5:0]) * $t_{DTS_CK}$ *2 |
| 3'b110 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *8 |
| 3'b111 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *16 |

**Note:**

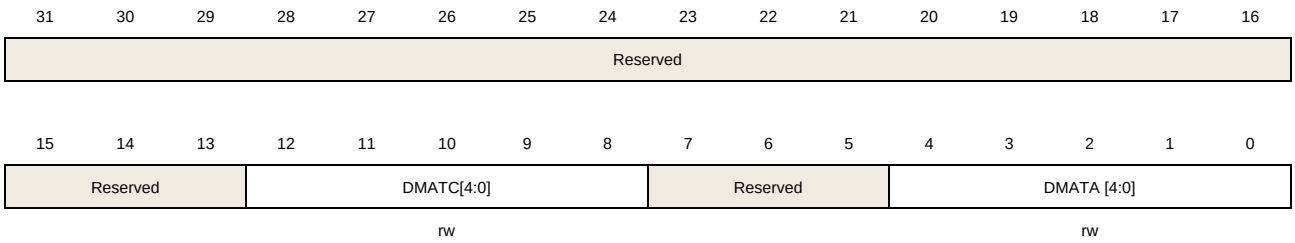
1. $t_{DTS_CK}$ is the period of DTS_CK which is configured by CKDIV[1:0] in TIMERx_CTL0.
2. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:8 | DMATC [4:0] | DMA transfer count<br>This filed defines the number(n) of the register that DMA will access(R/W), n = |

| | | |
|---|---|---|
| | | (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | DMATA [4:0] | DMA transfer access start address<br>This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4. |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | DMATB[15:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.<br>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CHVSEL | OUTSEL |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CHVSEL | Write CHxVAL register selection<br>This bit-field set and reset by software.<br>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored<br>0: No effect |
| 0 | OUTSEL | The output value selection<br>This bit-field set and reset by software<br>1: If POEN and IOS is 0, the output disabled<br>0: No effect |

## 14.2. General level0 timer (TIMERx, x=2)

### 14.2.1. Overview

The general level0 timer module (TIMER2) is a four-channel timer that supports input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level0 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counter value increasing in unison.

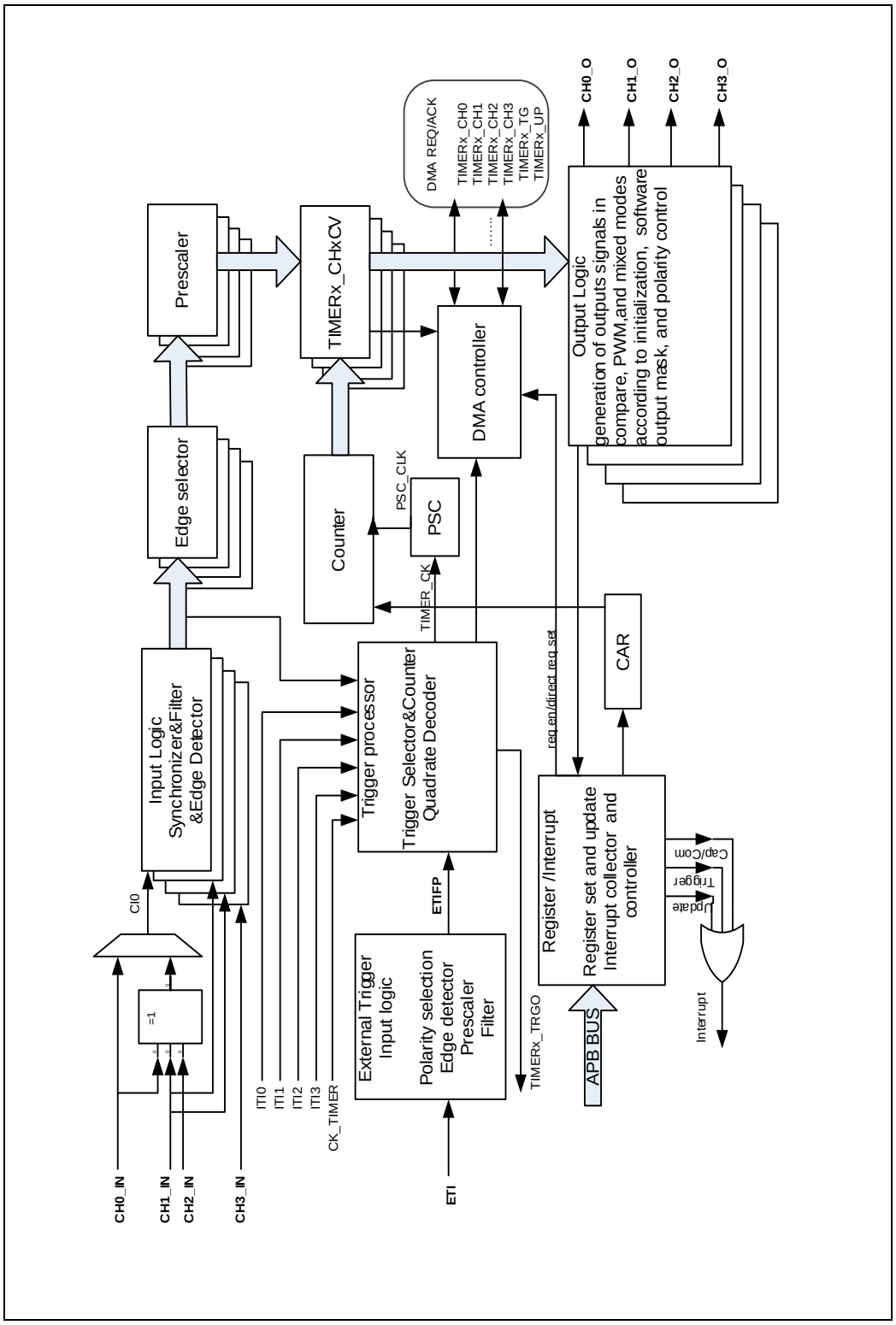
### 14.2.2. Characteristics

- Total channel num: 4.
- Counter width: 16 bits.
- Clock source of timer is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16 bits. The factor can be changed ongoing.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode and single pulse mode.
- Auto reload function.
- Interrupt output or DMA request: update event, trigger event and compare/capture event.
- Daisy chaining of timer module allows a single timer to start multiple timers.
- Timer synchronization allows the selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 14.2.3. Block diagram

***Figure 14-31. General Level 0 timer block diagram*** provides details on the internal configuration of the general level0 timer.

**Figure 14-31. General Level 0 timer block diagram**




## 14.2.4. Function overview

### Clock source configuration

The general level0 TIMER has the capability of being clocked by either the CK_TIMER or an

alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

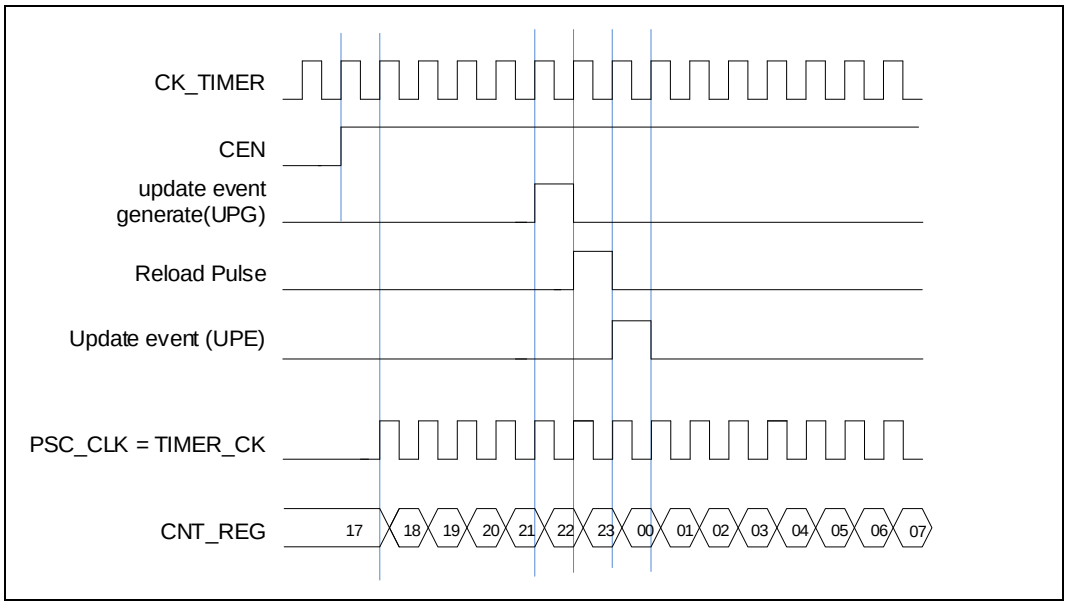
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when theSMC [2:0] == 3'b000. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register and described as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

**Figure 14-32. Timing chart of internal clock divided by 1**




- SMC [2:0] == 3'b111(external clock mode 0). External input pin source

The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

- SMC1== 1'b1(external clock mode 1). External input pin source (ETI)

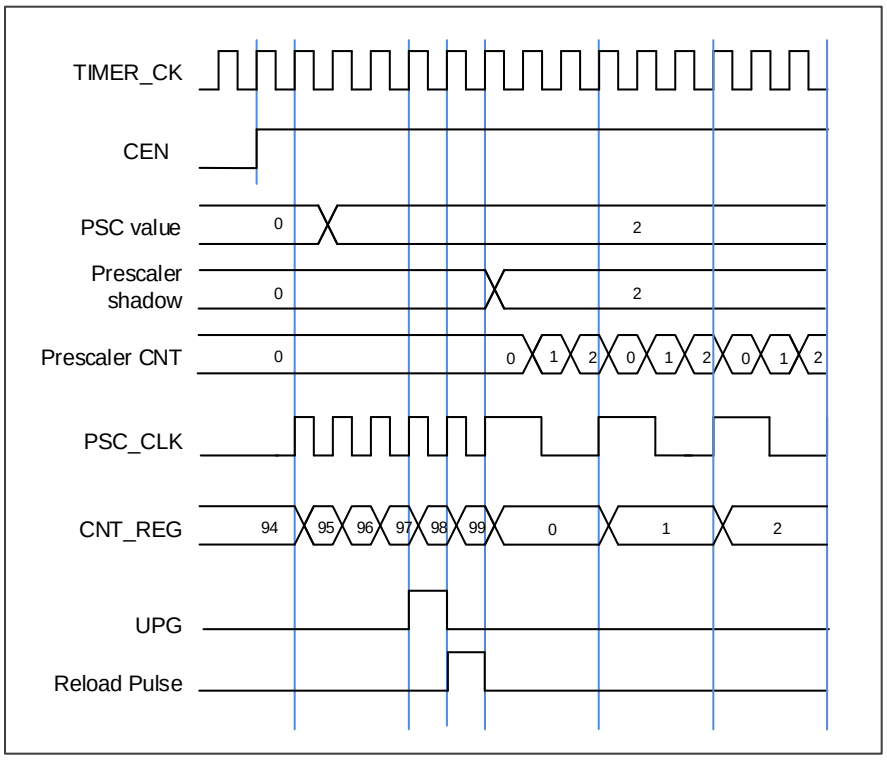
The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source

is set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the clock source is selected to come from the ETI signal, the trigger controller including the edge detection circuitry will generate a clock pulse during each ETI signal rising edge to clock the counter prescaler.

### Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-33. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. The update event is generated each time when counter overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter auto reload register, prescaler register) are updated.

*Figure 14-34. Timing chart of up counting mode, PSC=0/2* and *Figure 14-35. Timing chart of up counting mode, change TIMERx_CAR on the go*. show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-34. Timing chart of up counting mode, PSC=0/2**

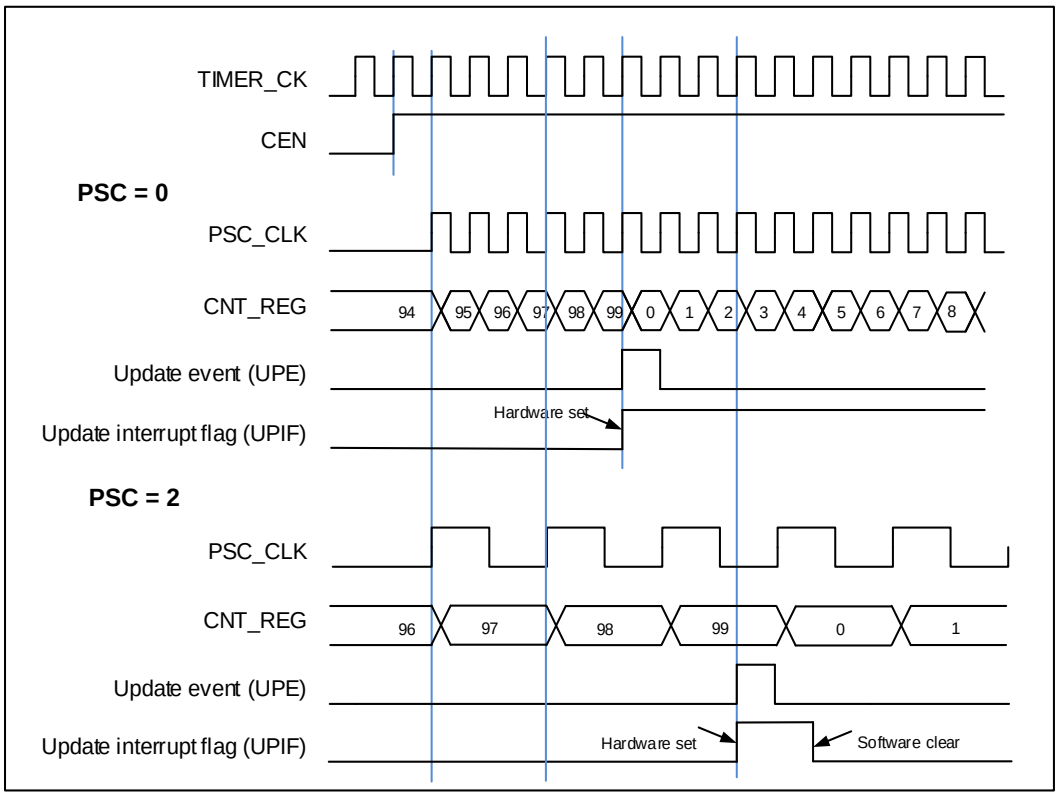
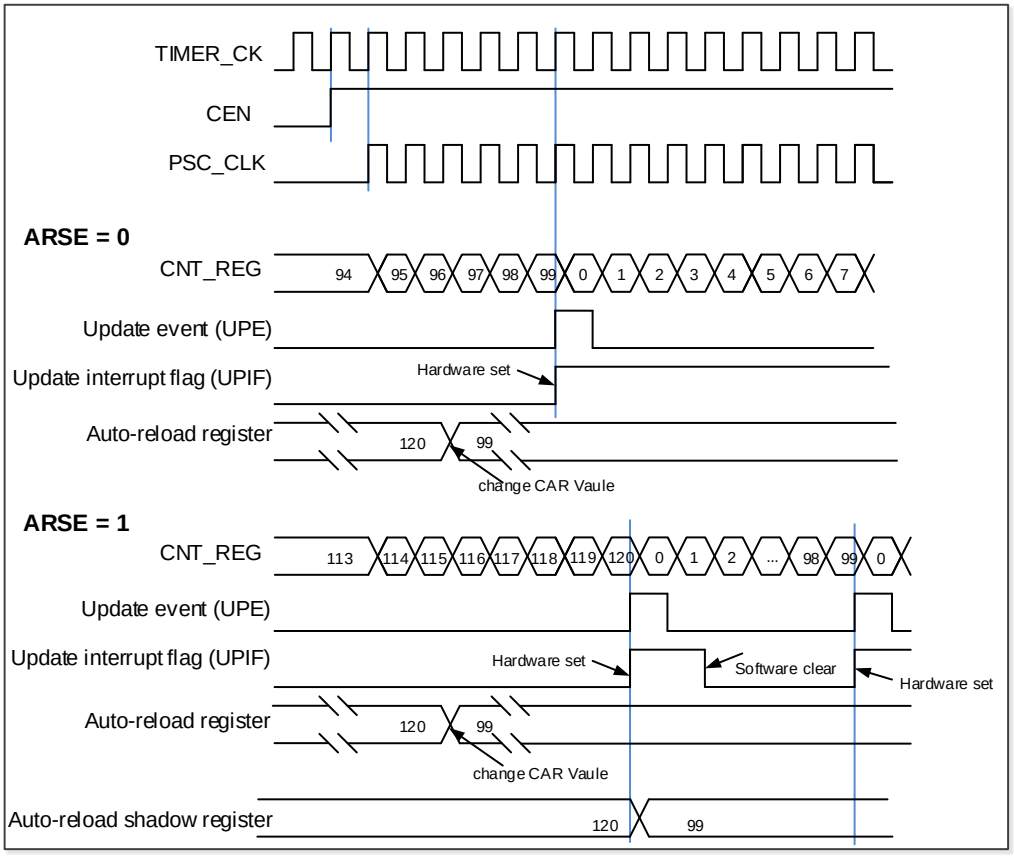


**Figure 14-35. Timing chart of up counting mode, change TIMERx_CAR on the go.**




## Counter down counting

In this mode, the counter counts down continuously from the counter reload value, which is defined in the TIMERx_CAR register, in a count-down direction. Once the counter reaches to 0, the counter will start counting down from the counter-reload value The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to the counter reload value and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter auto reload register, prescaler register) are updated.

***Figure 14-36. Timing chart of down counting mode, PSC=0/2*** and ***Figure 14-37. Timing chart of down counting mode, change TIMERx_CAR on the go.*** show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x99.

**Figure 14-36. Timing chart of down counting mode, PSC=0/2**

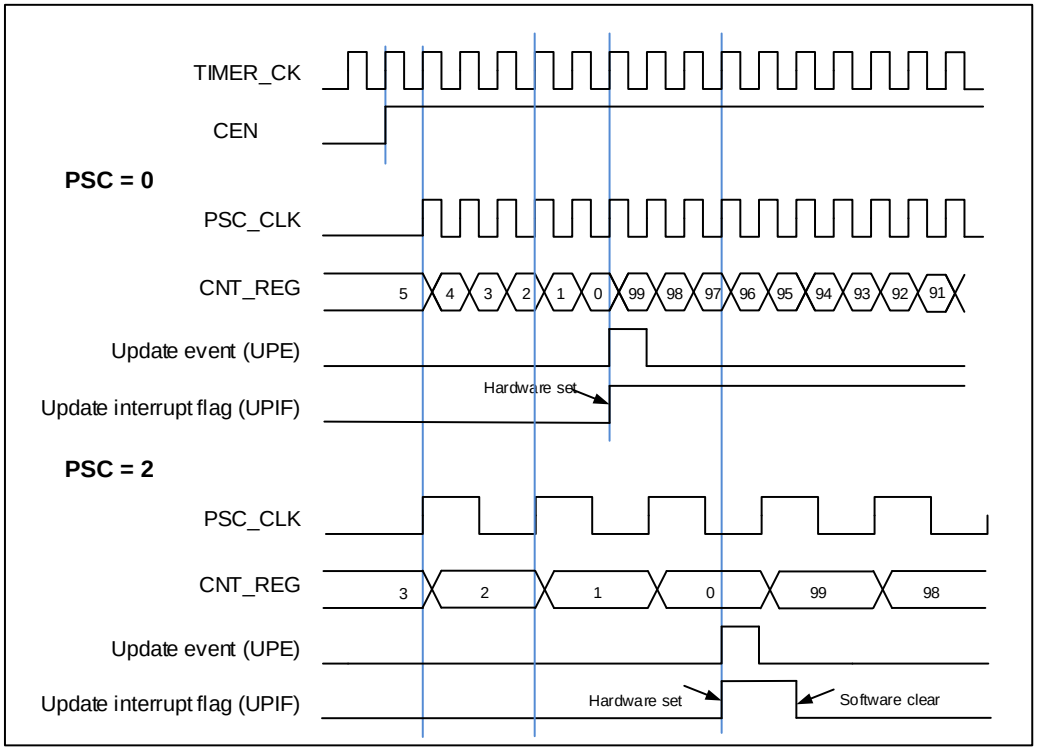
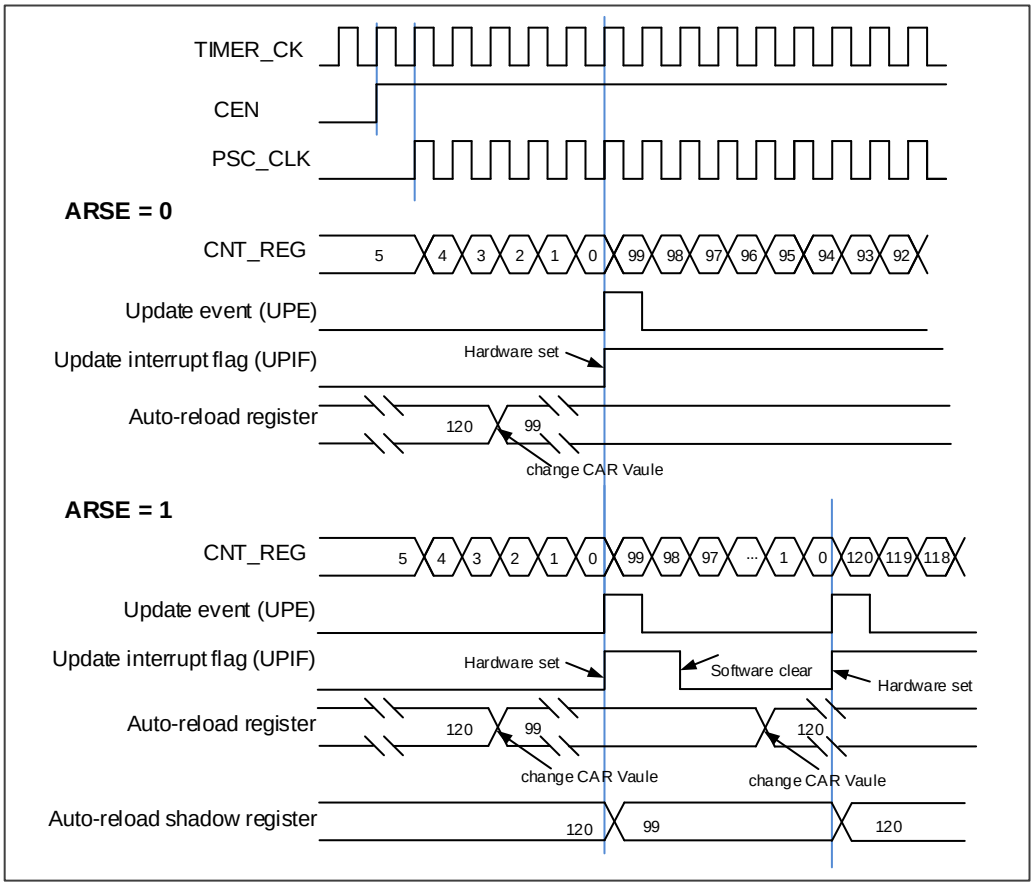



**Figure 14-37. Timing chart of down counting mode, change TIMERx_CAR on the go.**



### Counter center-aligned counting

In the center-aligned counting mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The TIMER module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting mode and generates an underflow event when the counter counts to 1 in the down-counting mode. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 and generate an update event irrespective of whether the counter is counting up or down in the center-aligned counting mode.

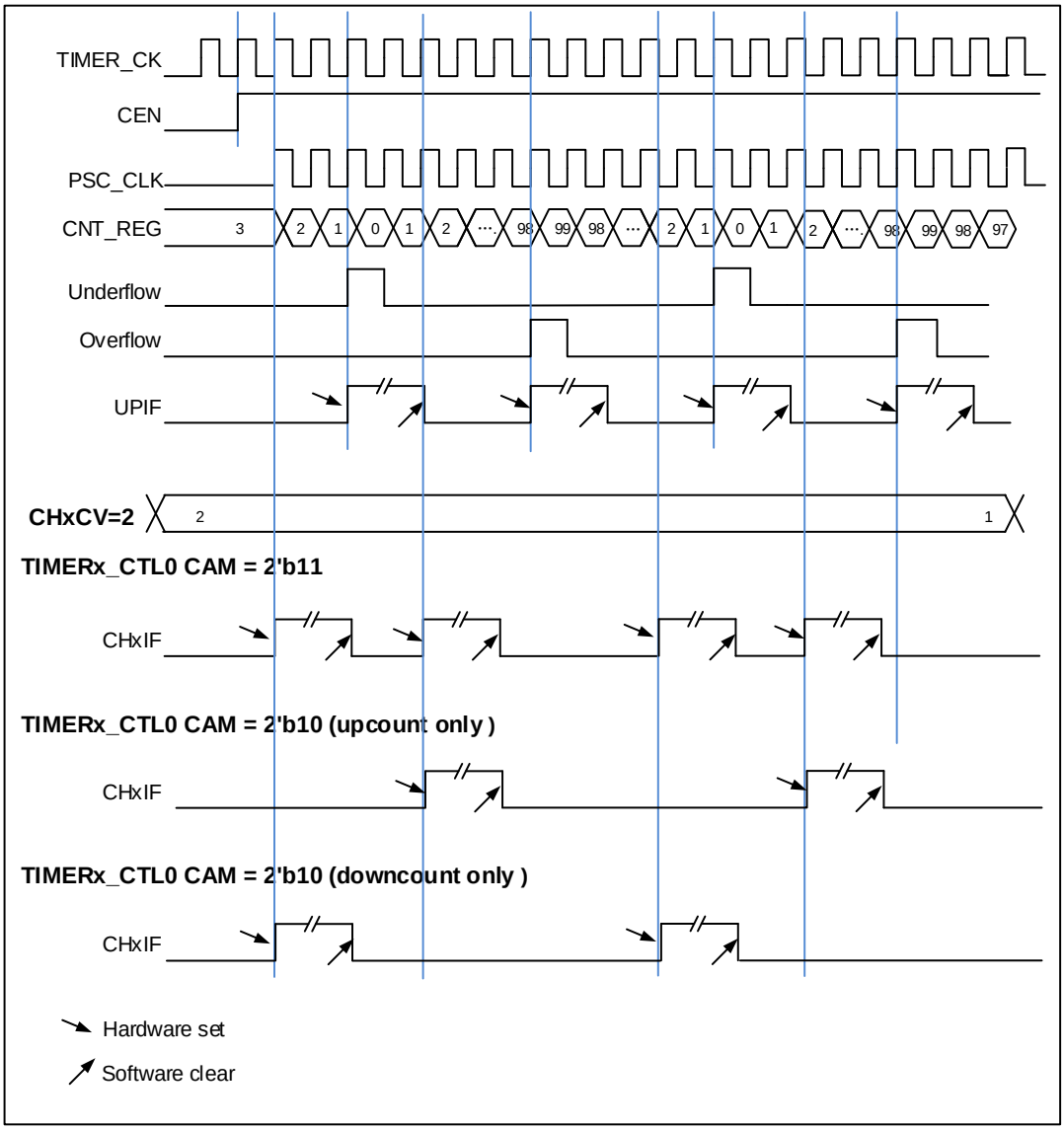
The UPIF bit in the TIMERx_INTF register will be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 14-38. Timing chart of center-aligned counting mode***.

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

***Figure 14-38. Timing chart of center-aligned counting mode*** shows the example of the counter behavior when TIMERx_CAR=0x99, TIMERx_PSC=0x0

**Figure 14-38. Timing chart of center-aligned counting mode**



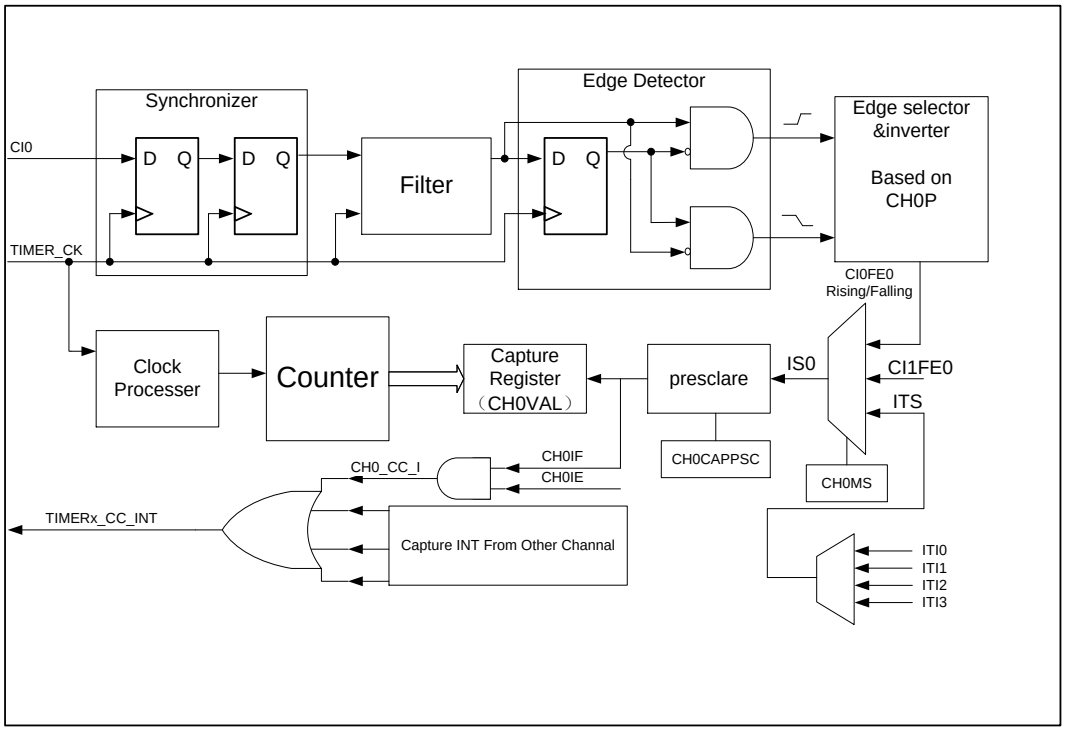

## Input capture and output compare channels

The general level0 Timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Channel input capture function**

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if it is enabled when CHxIE=1.

**Figure 14-39. Channel input capture principle**




The input signals of channelx (CIx) can be the TIMERx_CHx signal or the XOR signal of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the input signal of channel (CIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT in TIMERx_CHCTL0).

Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT.

**Step2**: Edge selection (CHxP/CHxNP in TIMERx_CHCTL2).

Rising edge or falling edge, choose one by configuring CHxP/CHxNP bits.

**Step3**: Capture source selection (CHxMS in TIMERx_CHCTL0)

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN in TIMERx_DMAINTEN)

Enable the related interrupt to get the interrupt and DMA request.

**Step5:** Capture enable (CHxEN in TIMERx_CHCTL2)

**Result**: When the wanted input signal is captured, TIMERx_CHxCV will be set by counter's

value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

The channel input capture function can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty cycle.

- **Channel output compare function**

**Figure 14-40. Channel output compare principle (x=0,1,2,3)**

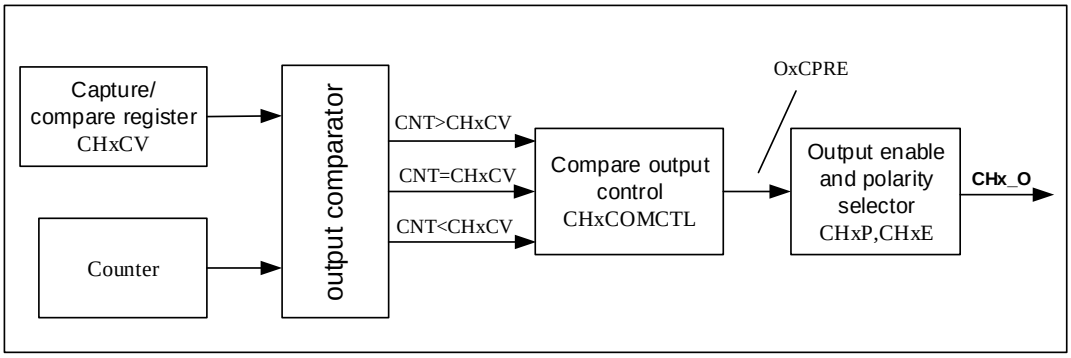



*Figure 14-40. Channel output compare principle (x=0,1,2,3)* shows the logic circuit of output compare mode. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CH0P bit (please refer to the TIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxE=1 (the output of CHx_O is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level.

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CxCDE=1.

So, the process can be divided into several steps as below:

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL.
- Select the active polarity by CHxP.
- Enable the output by CHxEN.

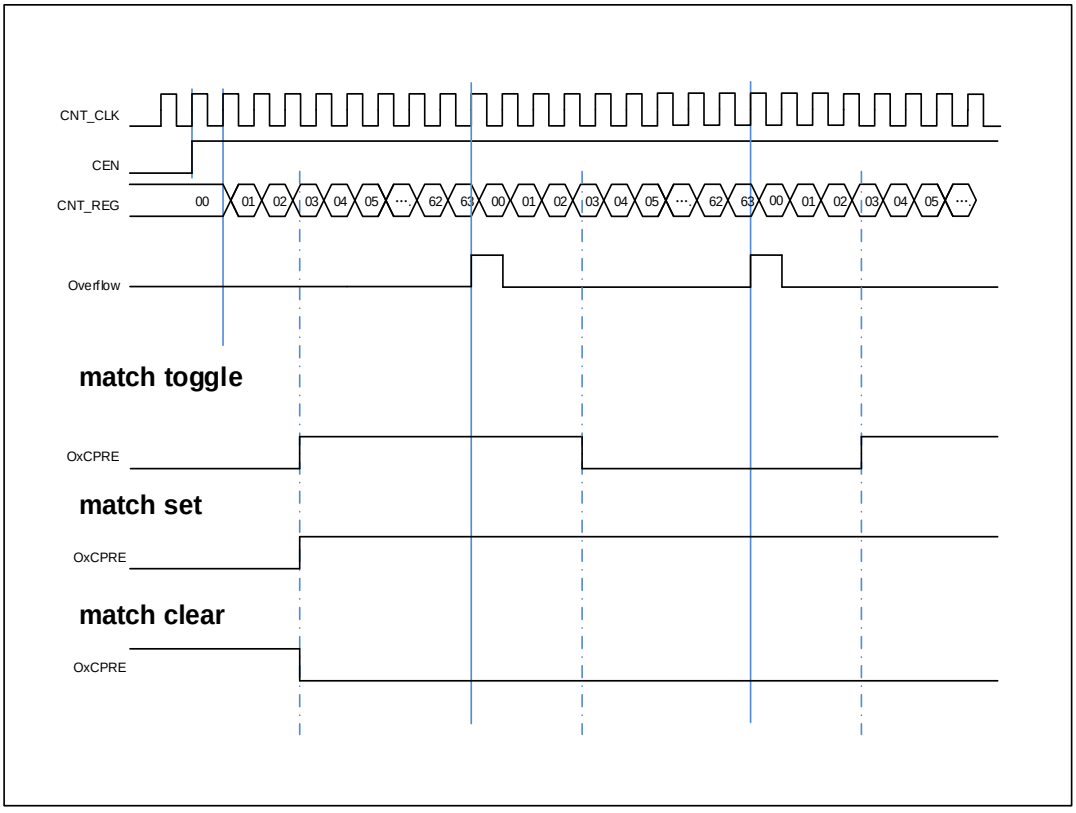
**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CxCDE.

**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.
The TIMERx_CHxCV can be changed onging to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

The timing chart below shows the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 14-41. Output-compare under three modes**




## Output PWM function

In the output PWM function (by setting the CHxCOMCTL bit to 3'b110 (PWM mode 0) or to 3'b 111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined

by TIMERx_CHxCV.*Figure 14-42. Timing chart of EAPWM* shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is determined by 2*TIMERx_CHxCV. *Figure 14-43. Timing chart of CAPWM* shows the CAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 0 (CHxCOMCTL=3'b110). And if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 1 (CHxCOMCTL=3'b111).

**Figure 14-42. Timing chart of EAPWM**

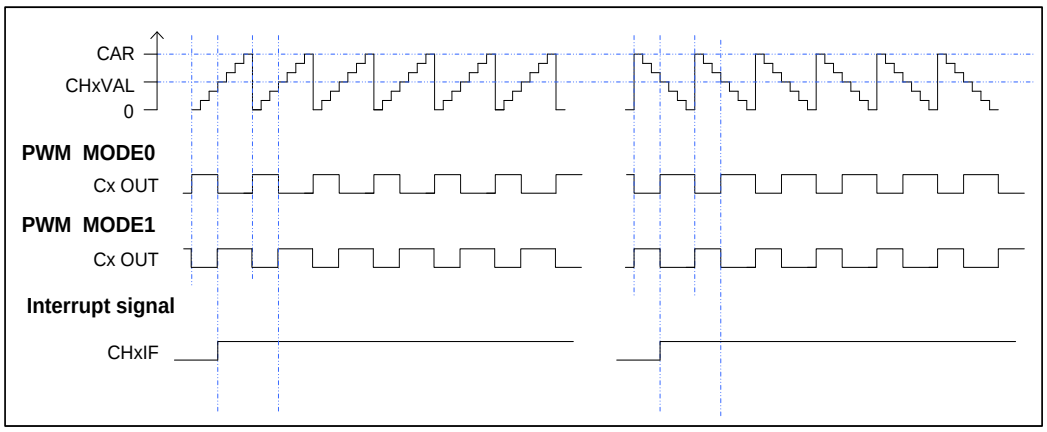
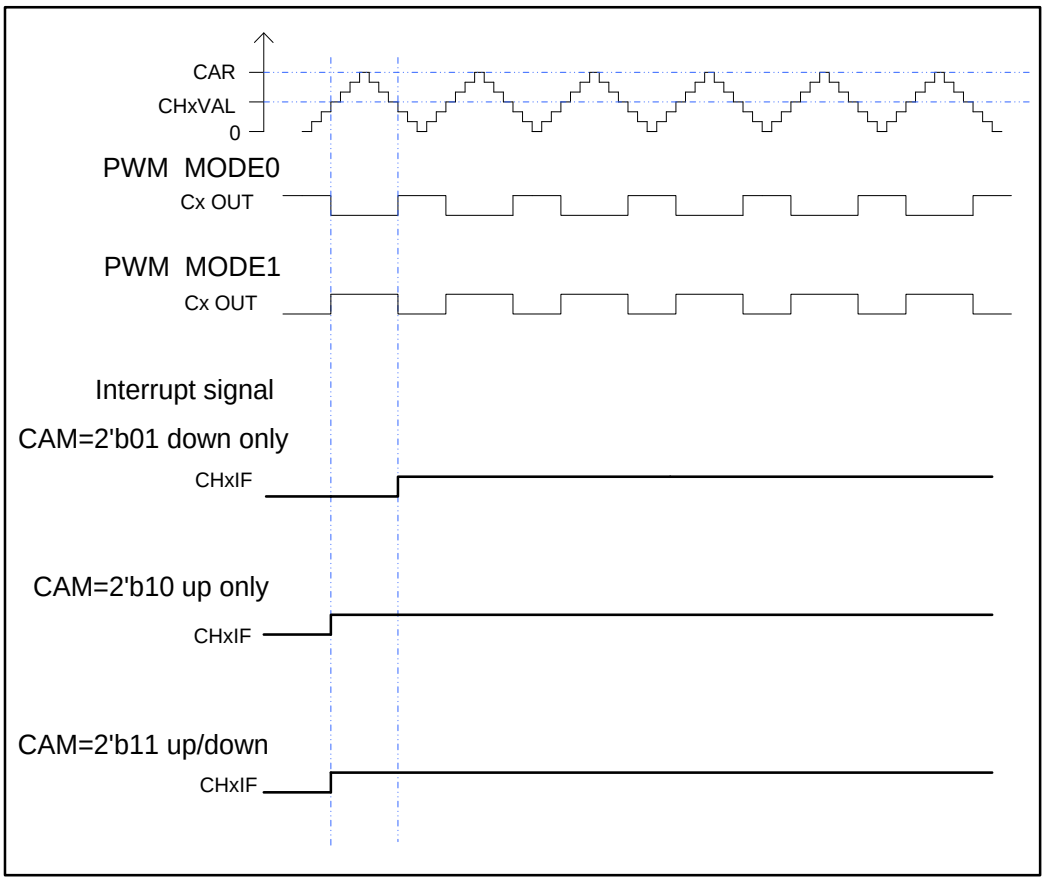


**Figure 14-43. Timing chart of CAPWM**




### Channel output prepare signal

As is shown in ***Figure 14-40. Channel output compare principle (x=0,1,2,3)*** when TIMERx is configured in compare match output mode,a middle signal which is OxCPRE signal (Channel x output prepare signal) will be generated before the channel outputs signal. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit. The OxCPRE signal has several types of output function. These include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is a forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Quadrature decoder

Refer to ***Quadrature decoder***.

### Hall sensor function

Refer to ***Hall sensor function.***

### Master-slave management

Refer to ***Master-slave management***.

### Single pulse mode

Refer to ***Single pulse mode***.

### Timers interconnection

Refer to ***Timers interconnection.***

### Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx_DMATB. In fact, TIMERx_DMATB register is only a buffer, timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0 (1 transfer), the timer sends only one DMA request. While if TIMERx_DMATC is not 0, such as 3 (4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8 and DMATA+0xC at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

### Timer debug mode

When the Cortex®-M23 is halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

### 14.2.5. TIMERx registers(x=2)

TIMER2 base address: 0x4000 0400

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CKDIV[1:0] | | ARSE | CAM[1:0] | | DIR | SPM | UPS | UPDIS | CEN |
| | | | | | | rw | | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS).<br>00: $f_{DTS}=f_{CK_TIMER}$<br>01: $f_{DTS}= f_{CK_TIMER}$ /2<br>10: $f_{DTS}= f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:5 | CAM[1:0] | Counter aligns mode selection<br>00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.<br>01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting down, CHxF bit can be set.<br>10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting up, CHxF bit can be set.<br>11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when counting up and counting down, CHxF bit can be set. |

| | | |
|---|---|---|
| | | After the counter is enabled, cannot be switched from 0x00 to non 0x00. |
| 4 | DIR | Direction<br>0: Count up<br>1: Count down<br>If the timer work in center-aligned mode or quadrature decoder mode, this bit is read only. |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode disable. The counter continues after update event.<br>1: Single pulse mode enable. The counter counts until the next update event occurs. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: This event generates update interrupts or DMA requests:<br>The counter generates an overflow or underflow event |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: Update event disable.<br>**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decoder mode. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | TI0S | MMC[2:0] | | | DMAS | Reserved | | |
| | | | | | | | | rw | rw | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | TI0S | Channel 0 trigger input selection<br>0: The TIMERx_CH0 pin input is selected as channel 0 trigger input.<br>1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input. |
| 6:4 | MMC[2:0] | Master mode control<br>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.<br>000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source:<br>Master timer generate a reset<br>the UPG bit in the TIMERx_SWEVG register is set<br>001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source :<br>CEN control bit is set<br>The trigger input in pause mode is high<br>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.<br>011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.<br>100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.<br>101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE.<br>110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE.<br>111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE. |
| 3 | DMAS | DMA request source selection<br>0: When capture or compare event occurs, the DMA request of channel x is sent<br>1: When update event occurs, the DMA request of channel x is sent. |
| 2:0 | Reserved | Must be kept at reset value. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETP | SMC1 | ETPSC[1:0] | | ETFC[3:0] | | | | MSM | TRGS[2:0] | | | OCRC | SMC[2:0] | | |
| rw | rw | rw | | rw | | | | rw | rw | | | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal<br>0: ETI is active at rising edge or high level .<br>1: ETI is active at falling edge or low level . |
| 14 | SMC1 | Part of SMC for enable External clock mode1<br>In external clock mode 1, the counter is clocked by any active edge on the ETIF signal.<br>0: External clock mode 1 disabled<br>1: External clock mode 1 enabled.<br>When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case.<br>The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time.<br>**Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed. |
| 13:12 | ETPSC[1:0] | The prescaler of external trigger<br>The frequency of external trigger signal ETIFP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETIFP frequency.<br>00: Prescaler disable.<br>01: The prescaler is 2.<br>10: The prescaler is 4.<br>11: The prescaler is 8. |
| 11:8 | ETFC[3:0] | External trigger filter control<br>The external trigger can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the external trigger signal |

according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.

The filtering capability configuration is as follows:

| EXTFC[3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS_CK}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS_CK}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS_CK}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS_CK}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS_CK}/32$ |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

7 MSM

Master-slave mode

This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.

0: Master-slave mode disable

1: Master-slave mode enable

6:4 TRGS[2:0]

Trigger selection

This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.

000: ITI0

001: ITI1

010: ITI2

011: ITI3

100: CI0F_ED

101: CI0FE0

110: CI1FE1

111: ETIFP

These bits must not be changed when slave mode is enabled.

3 OCRC

OCPRE clear source selection

0: OCPRE clear source is connected to the OCPRE_CLR input

| | | |
|---|---|---|
| | | 1: OCPRE clear source is connected to ETIFP |
| 2:0 | SMC[2:0] | Slave mode control<br>000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high.<br>001: Quadrature decoder mode 0.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level.<br>010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level.<br>011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other.<br>100: Restart Mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.<br>101: Pause Mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.<br>110: Event Mode. A rising edge of the trigger input enables the counter.<br>111: External Clock Mode 0. The counter counts on the rising edges of the selected trigger.<br>Because CI0F_ED outputs 1 pulse for each transition on CI0F, and the pause mode checks the level of the trigger signal, when CI0F_ED is selected as the trigger input, the pause mode must not be used. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | TRGDEN | Reserved | CH3DEN | CH2DEN | CH1DEN | CH0DEN | UPDEN | Reserved | TRGIE | Reserved | CH3IE | CH2IE | CH1IE | CH0IE | UPIE |
| | rw | | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: disabled<br>1: enabled |
| 13 | Reserved | Must be kept at reset value. |
| 12 | CH3DEN | Channel 3 capture/compare DMA request enable<br>0: disabled |

| | | |
|---|---|---|
| | | 1: enabled |
| 11 | CH2DEN | Channel 2 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 8 | UPDEN | Update DMA request enable<br>0: disabled<br>1: enabled |
| 7 | Reserved | Must be kept at reset value. |
| 6 | TRGIE | Trigger interrupt enable<br>0: disabled<br>1: enabled |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3IE | Channel 3 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 3 | CH2IE | Channel 2 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | CH3OF | CH2OF | CH1OF | CH0OF | Reserved | | TRGIF | Reserved | CH3IF | CH3IF | CH1IF | CH0IF | UPIF |
| | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12 | CH3OF | Channel 3 over capture flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 over capture flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8:7 | Reserved | Must be kept at reset value. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event.<br>0: No trigger event occurred.<br>1: Trigger interrupt occurred. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3IF | Channel 3 ‘s capture/compare interrupt enable<br>Refer to CH0IF description |
| 3 | CH2IF | Channel 2 ‘s capture/compare interrupt enable<br>Refer to CH0IF description |
| 2 | CH1IF | Channel 1 ‘s capture/compare interrupt flag<br>Refer to CH0IF description |

| | | |
|---|---|---|
| 1 | CH0IF | Channel 0 ‘s capture/compare interrupt flag<br>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No Channel 1 interrupt occurred<br>1: Channel 1 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | TRGG | Reserved | CH3G | CH2G | CH1G | CH0G | UPG |
| | | | | | | | | | w | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event<br>1: Generate a trigger event |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3G | Channel 3’s capture or compare event generation<br>Refer to CH0G description |
| 3 | CH2G | Channel 2’s capture or compare event generation<br>Refer to CH0G description |
| 2 | CH1G | Channel 1’s capture or compare event generation<br>Refer to CH0G description |
| 1 | CH0G | Channel 0’s capture or compare event generation |

| | | |
|---|---|---|
| | | This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 1 capture or compare event<br>1: Generate a channel 1 capture or compare event |
| 0 | UPG | This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMCEN | CH1COMCTL[2:0] | | | CH1COMSEN | CH1COMFEN | CH1MS[1:0] | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| CH1CAPFLT[3:0] | | | | CH1CAPPSC[1:0] | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| Rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | CH1COMFEN | Channel 1 output compare fast enable<br>Refer to CH0COMFEN description |

| | | |
|---|---|---|
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is programmed as output mode<br>01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1<br>10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1<br>11: Channel 1 is programmed as input mode, IS1 is connected to ITS.<br>**Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, if the OCPRE clear source signal is detected as high level, the O0CPRE signal will be cleared. The OCPRE clear source signal is selected by OCRC bit in TIMERx_SMCFG register.<br>0: Channel 0 output compare clear disable<br>1: Channel 0 output compare clear enable |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced to low level.<br>101: Force high. O0CPRE is forced to high level.<br>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.<br>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable |

| | | |
|---|---|---|
| | | When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.<br>0: Channel 0 output compare shadow disable<br>1: Channel 0 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH0COMFEN | Channel 0 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.<br>0: Channel 0 output quickly compare disable.<br>1: Channel 0 output quickly compare enable. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).<br>00: Channel 0 is programmed as output mode<br>01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0<br>10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0<br>11: Channel 0 is programmed as input mode, IS0 is connected to ITS<br>**Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>Same as Output compare mode |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI0 input signal according to |

$f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

| CH0CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}/32$ |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

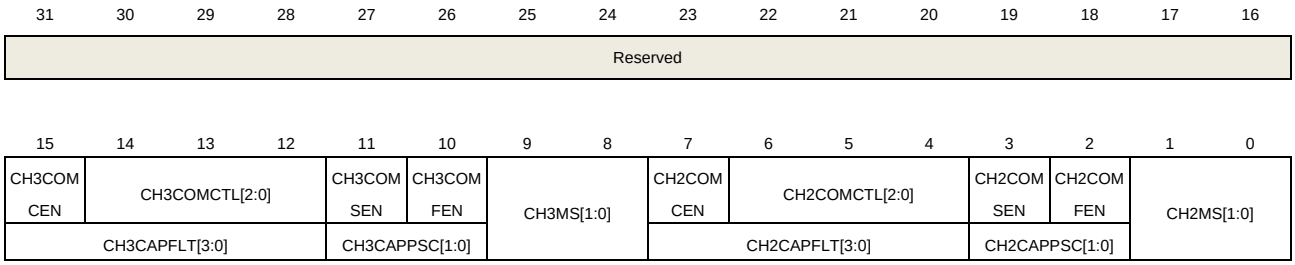
Same as Output compare mode

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



| Rw | rw | rw | rw | rw | rw |
|---|---|---|---|---|---|

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | CH3COMCEN | Channel 3 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH3COMCTL[2:0] | Channel 3 compare output control<br>Refer to CH0COMCTL description |
| 11 | CH3COMSEN | Channel 3 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | CH3COMFEN | Channel 3 output compare fast enable<br>Refer to CH0COMFEN description |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH3EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 3 is programmed as output mode<br>01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3<br>10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3<br>11: Channel 3 is programmed as input mode, IS3 is connected to ITS.<br>**Note:** When CH3MS[1:0]=11, it is necessary to ensure that an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable.<br>When this bit is set, if the OCPRE clear source signal is detected as high level, the O0CPRE signal will be cleared. The OCPRE clear source signal is selected by OCRC bit in TIMERx_SMCFG register.<br>0: Channel 2 output compare clear disable<br>1: Channel 2 output compare clear enable |
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT.<br>001: Set the channel output. O2CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH2CV.<br>010: Clear the channel output. O2CPRE signal is forced low when the counter is |

| | | |
|---|---|---|
| | | equals to the output compare register TIMERx_CH2CV.<br>011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV.<br>100: Force low. O2CPRE is forced to low level.<br>101: Force high. O2CPRE is forced to high level.<br>110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise.<br>111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise.<br>If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE). |
| 3 | CH2COMSEN | Channel 2 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled.<br>0: Channel 2 output compare shadow disable<br>1: Channel 2 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH2COMFEN | Channel 2 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH2_O is set to the compare level independently from the result of the comparison.<br>0: Channel 2 output quickly compare disable.<br>1: Channel 2 output quickly compare enable. |
| 1:0 | CH2MS[1:0] | Channel 2 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH2EN bit in TIMERx_CHCTL2 register is reset).).<br>00: Channel 2 is programmed as output mode<br>01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2<br>10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2<br>11: Channel 2 is programmed as input mode, IS2 is connected to ITS.<br>**Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input |

through TRGS bits in TIMERx_SMCFG register.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | CH3CAPFLT[3:0] | Channel 3 input capture filter control<br>Refer to CH0CAPFLT description |
| 11:10 | CH3CAPPSC[1:0] | Channel 3 input capture prescaler<br>Refer to CH0CAPPSC description |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>Same as Output compare mode |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI2 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| CH2CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}/32$ |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

| | | |
|---|---|---|
| 3:2 | CH2CAPPSC[1:0] | Channel 2 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 register is clear.<br>00: Prescaler disable, input capture occurs on every channel input edge<br>01: The input capture occurs on every 2 channel input edges |

10: The input capture occurs on every 4 channel input edges
11: The input capture occurs on every 8 channel input edges

1:0 CH2MS[1:0] Channel 2 mode selection
Same as output compare mode

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3NP | Reserved | CH3P | CH3EN | CH2NP | Reserved | CH2P | CH2EN | CH1NP | Reserved | CH1P | CH1EN | CH0NP | Reserved | CH0P | CH0EN |
| rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | CH3NP | Channel 3 complementary output polarity<br>Refer to CH0NP description |
| 14 | Reserved | Must be kept at reset value |
| 13 | CH3P | Channel 3 capture/compare function polarity<br>Refer to CH0P description |
| 12 | CH3EN | Channel 3 capture/compare function enable<br>Refer to CH0EN description |
| 11 | CH2NP | Channel 2 complementary output polarity<br>Refer to CH0NP description |
| 10 | Reserved | Must be kept at reset value |
| 9 | CH2P | Channel 2 capture/compare function polarity<br>Refer to CH0P description |
| 8 | CH2EN | Channel 2 capture/compare function enable<br>Refer to CH0EN description |
| 7 | CH1NP | Channel 1 complementary output polarity<br>Refer to CH0NP description |
| 6 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 5 | CH1P | Channel 1 capture/compare function polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 capture/compare function enable<br>Refer to CH0EN description |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 is configured in output mode, this bit should be keep reset value.<br>When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 2 | Reserved | Must be kept at reset value |
| 1 | CH0P | Channel 0 capture/compare function polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 high level is active level<br>1: Channel 0 low level is active level<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: Reserved.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |

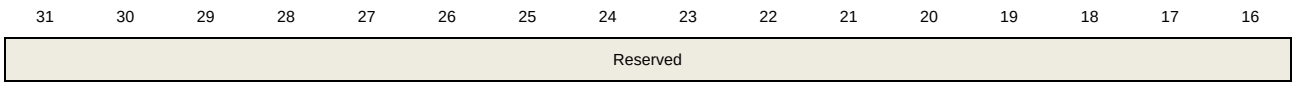
rw

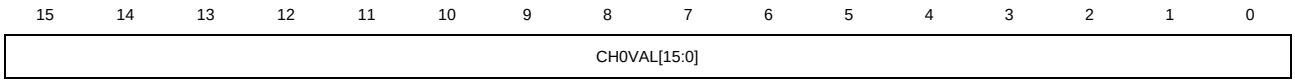
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter.<br>**Note:** When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



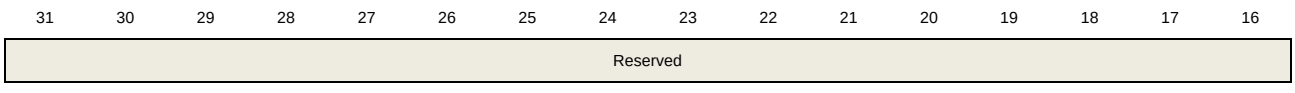

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel0<br>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



| CH1VAL[15:0] |
|---|

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH1VAL[15:0] | Capture or compare value of channel1<br>When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2VAL[15:0] | | | | | | | | | | | | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH2VAL[15:0] | Capture or compare value of channel 2<br>When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

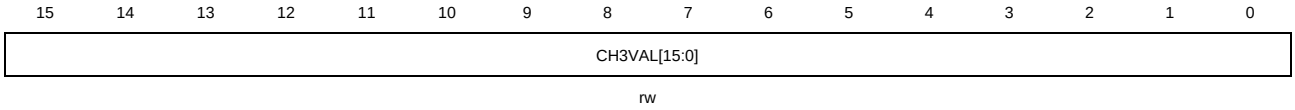
### Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

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



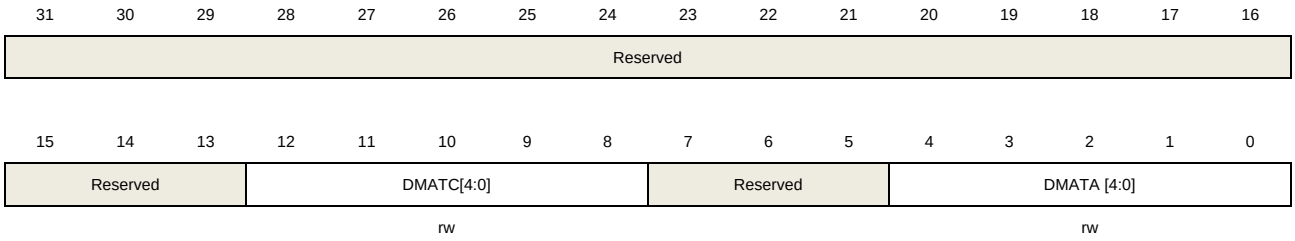

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH3VAL[15:0] | Capture or compare value of channel 3<br>When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



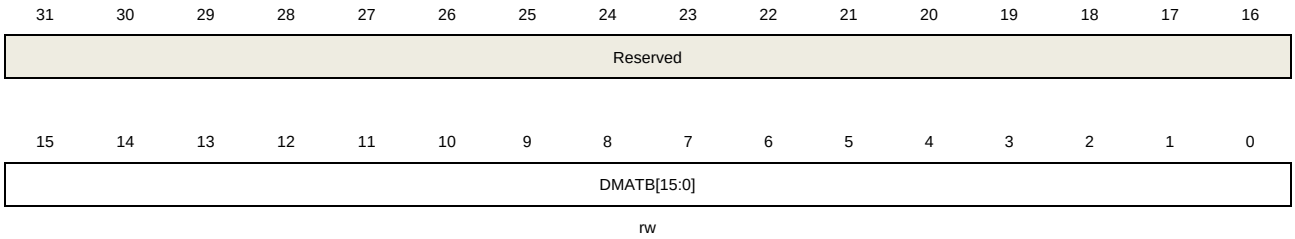

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:8 | DMATC [4:0] | DMA transfer count<br>This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | DMATA [4:0] | DMA transfer access start address<br>This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4. |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



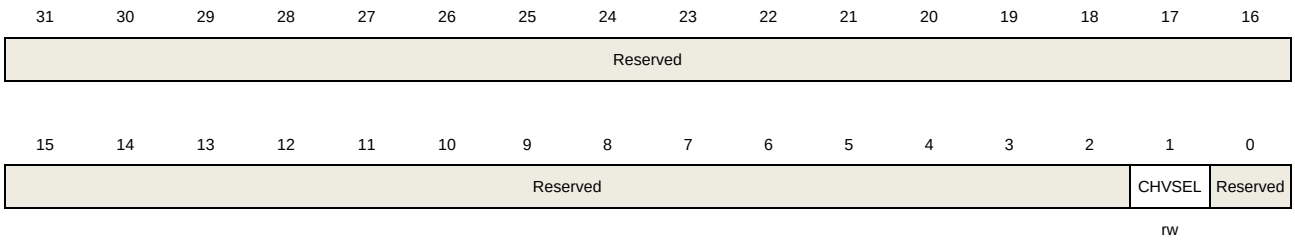

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | DMATB[15:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.<br>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG )

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CHVSEL | Write CHxVAL register selection<br>This bit-field set and reset by software.<br>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored<br>0: No effect |
| 0 | Reserved | Must be kept at reset value |

## 14.3. General level2 timer (TIMERx, x=13)

### 14.3.1. Overview

The general level2 timer module (TIMER13) is a one-channel timer that supports input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level2 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level2 timers can be programmed and be used for counting, their external events can be used to drive other timers.

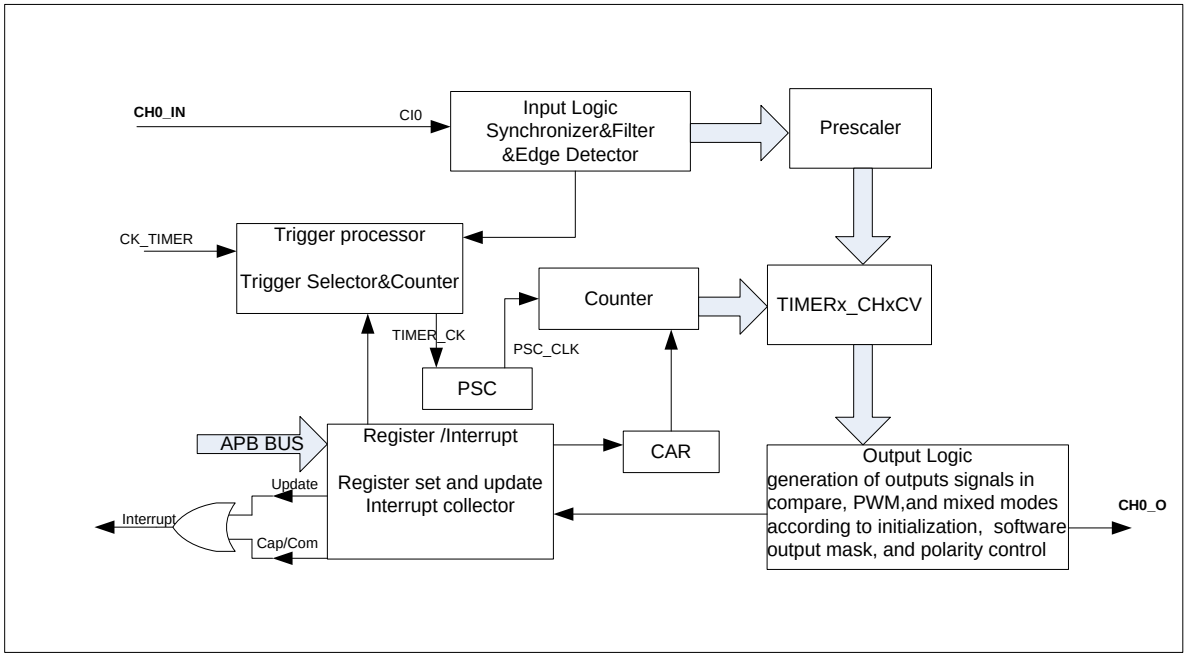
### 14.3.2. Characteristics

- Total channel num: 1.
- Counter width: 16 bits.
- Source of count clock: internal clock.
- Counter mode: count up only.
- Programmable prescaler: 16 bits. Factor can be changed ongoing.
- Each channel is user-configurable:
  Input capture mode, output compare mode, programmable PWM mode
- Auto-reload function.
- Interrupt output on: update, compare/capture event.

### 14.3.3. Block diagram

*Figure 14-44. General level2 timer block diagram* provides details on the internal configuration of the general level2 timer.

**Figure 14-44. General level2 timer block diagram**



## 14.3.4. Function overview

### Clock source configuration

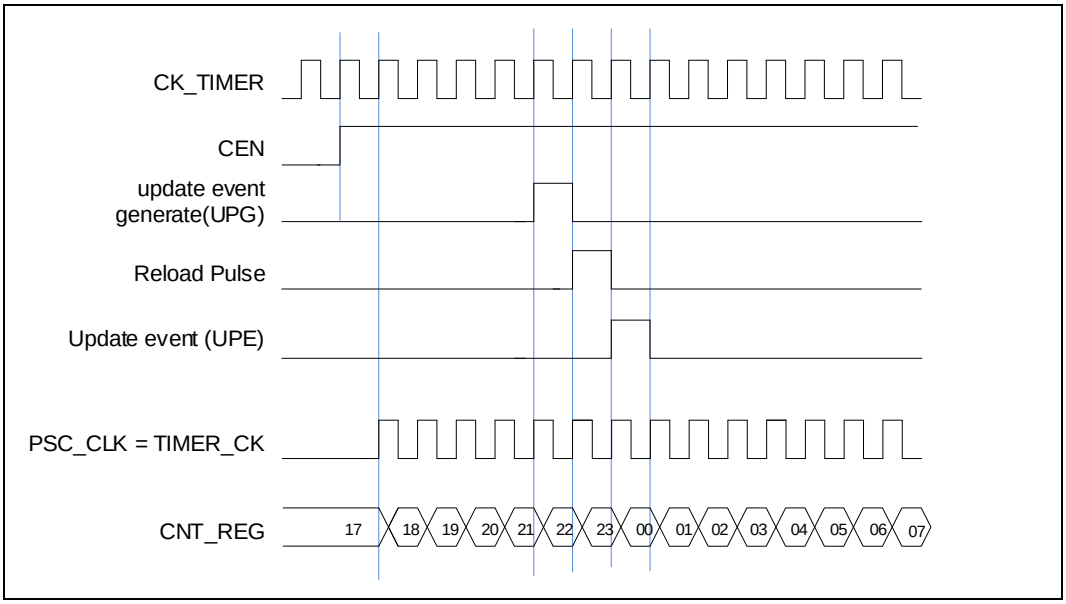
The general level2 TIMER can only being clocked by the CK_TIMER.

- Internal timer clock CK_TIMER which is from module RCU

The general level2 TIMER has only one clock source which is the internal CK_TIMER, used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU

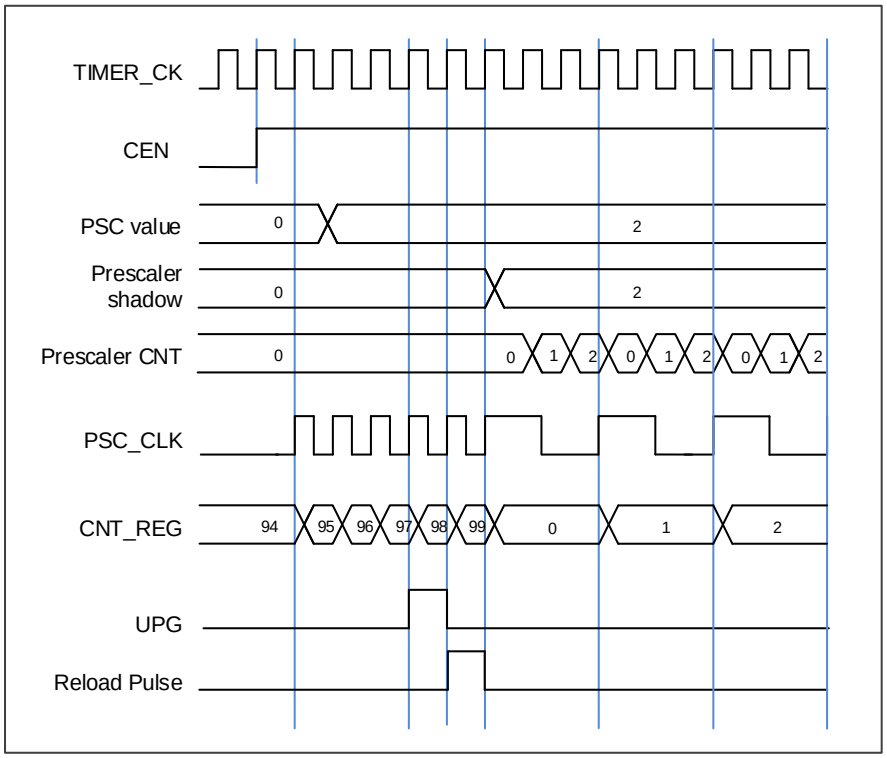
**Figure 14-45. Timing chart of internal clock divided by 1**




### Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-46. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-47. Timing chart of up counting mode, PSC=0/2**

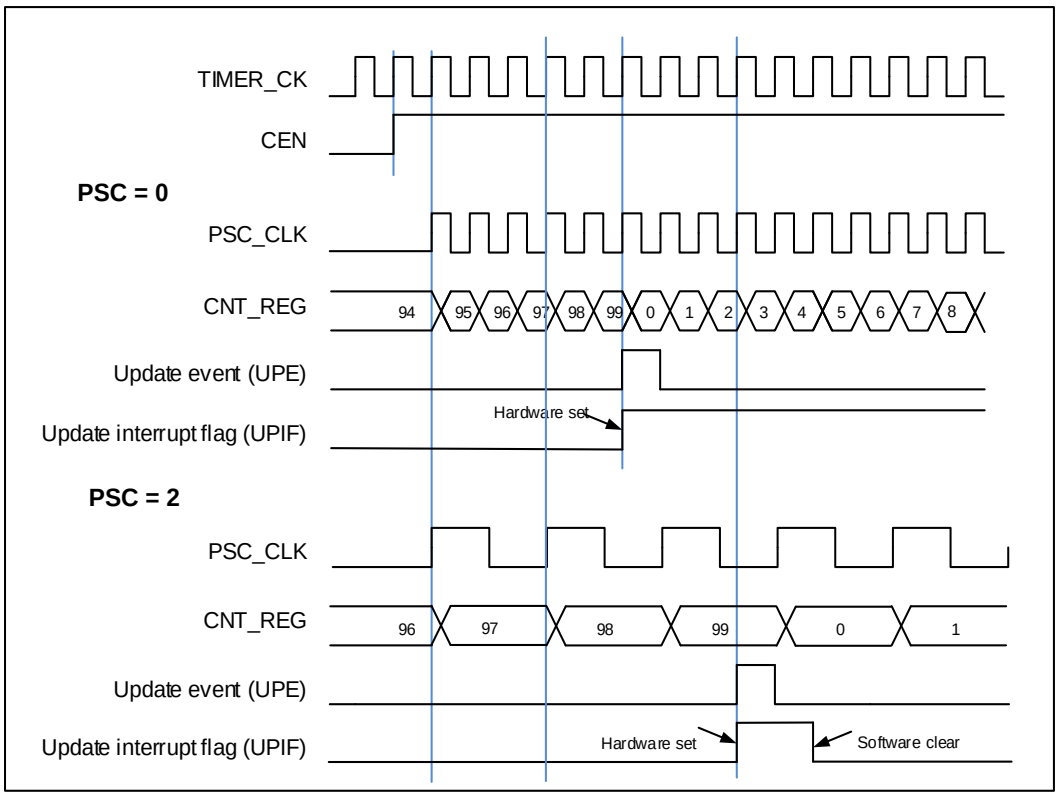
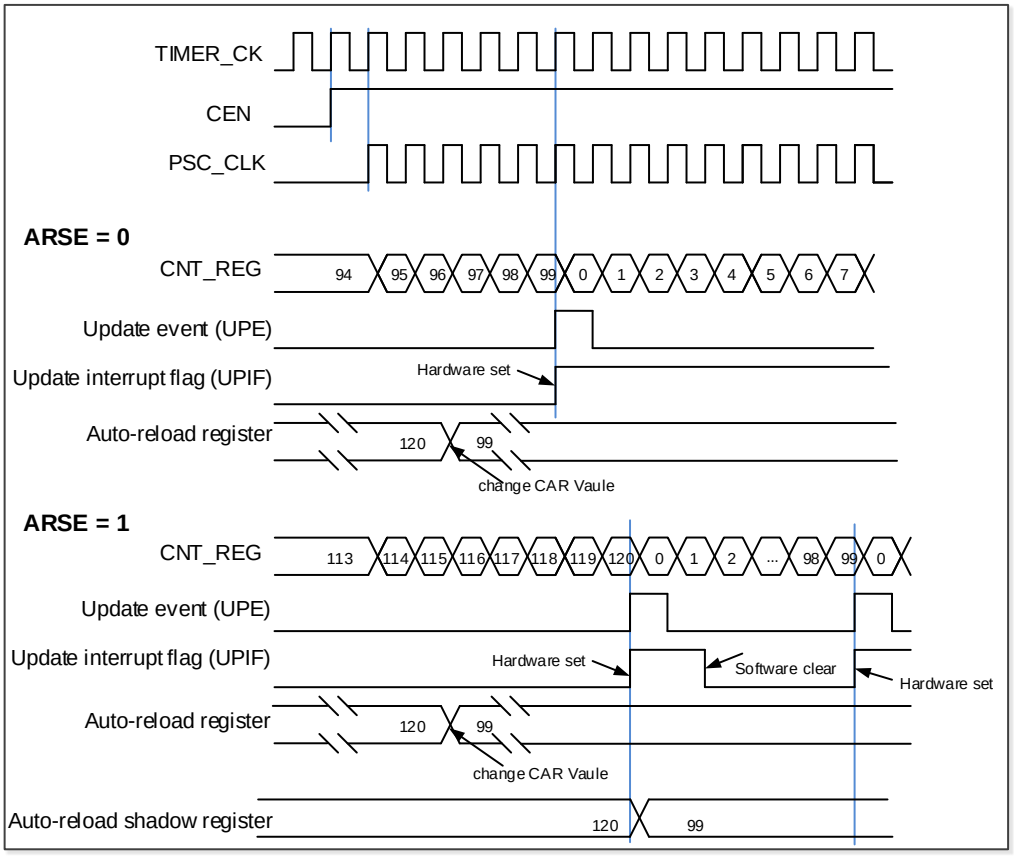



**Figure 14-48. Timing chart of up counting mode, change TIMERx_CAR on the go**



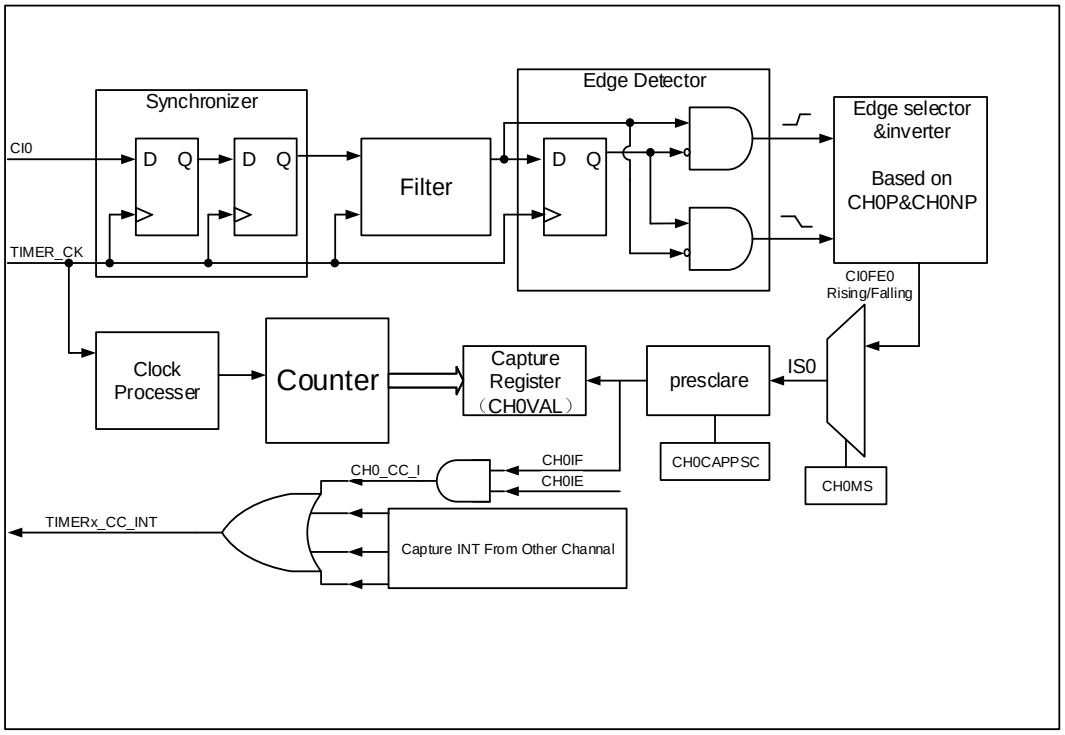
### Input capture and output compare channels

The general level2 timer has one independent channel which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Channel input capture function**

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if it is enabled when CHxIE=1.

**Figure 14-49. Channel input capture principle**




First, the input signal of channel (CIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT in TIMERx_CHCTL0).

Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT.

**Step2**: Edge selection.(CHxP/CHxNP in TIMERx_CHCTL2).

Rising edge, falling edge or both edges (rising and falling edge), choose one by configuring CHxP/CHxNP bits.

**Step3**: Capture source selection (CHxMS in TIMERx_CHCTL0).

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN in TIMERx_DMAINTEN)

Enable the related interrupt to get the interrupt and DMA request.

**Step5:** Capture enable (CHxEN in TIMERx_CHCTL2).

**Result**: When the wanted input signal is captured, TIMERx_CHxCV will be set by counter's value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE in TIMERx_DMAINTEN.

**Direct generation**: An interrupt is generated by setting CHxG directly.

- **Channel output compare function**

**Figure 14-50. Channel output compare principle**

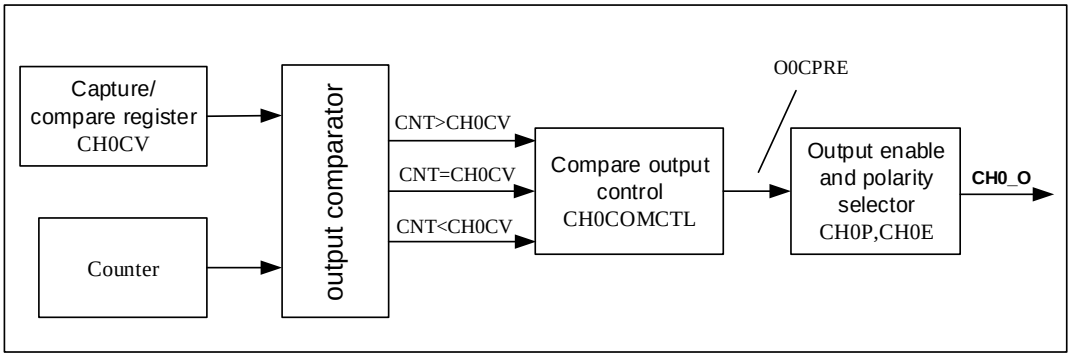



*Figure 14-50. Channel output compare principle* shows the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CH0P bit (please refer to the TIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxE=1 (the output of CHx_O is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated

if CHxIE = 1.

So, the process can be divided into several steps as below:

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL.
- Select the active polarity by CHxP.
- Enable the output by CHxEN.

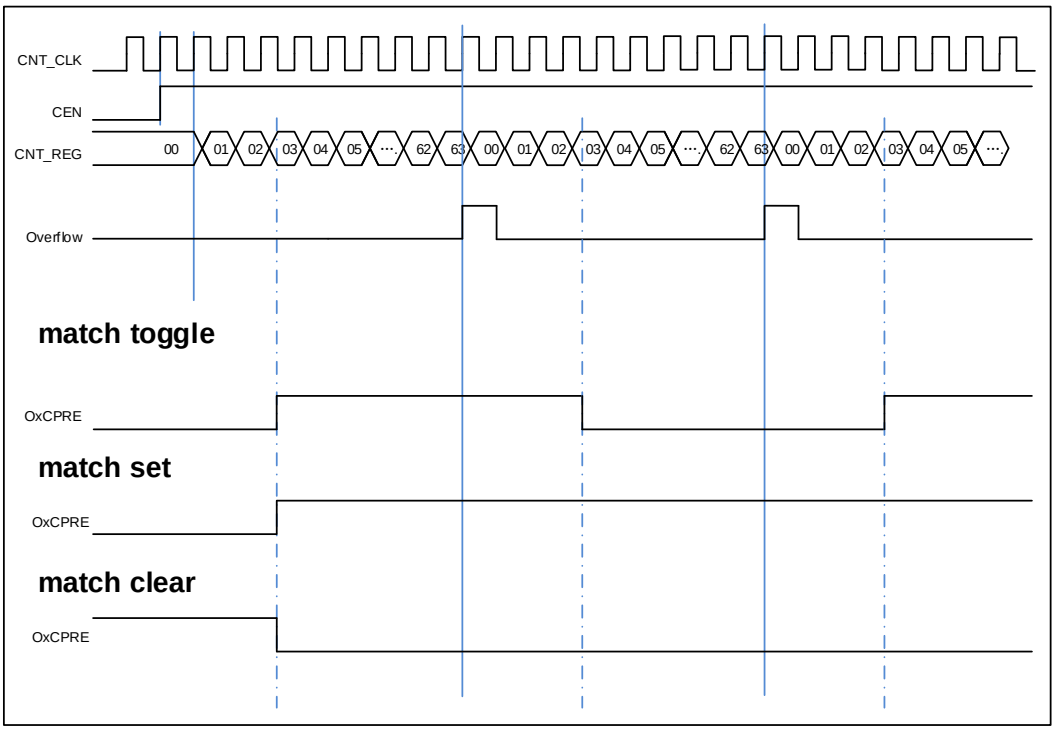
**Step3:** Interrupt/DMA-request enables configuration by CHxIE.

**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV. The TIMERx_CHxCV can be changed onging to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

The timing chart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 14-51. Output-compare in three modes**




## Output PWM function

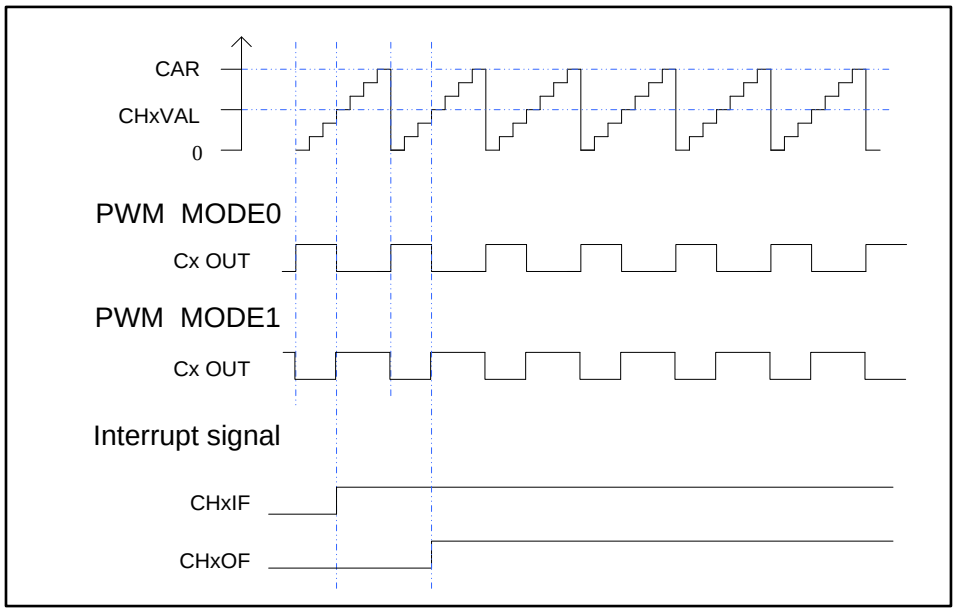
In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

The period is determined by TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV.

*Figure 14-52. PWM mode timechart* shows the PWM output mode and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 14-52. PWM mode timechart**




### Channel output prepare signal

As is shown in *Figure 14-50. Channel output compare principle*, when TIMERx is configured in compare match output mode,a middle signal which is OxCPRE signal (Channel x output prepare signal) will be generated before the channel outputs signal. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit. The OxCPRE signal has several types of output function. These include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is a forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

### Timer debug mode

When the Cortex®-M23 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

### 14.3.5. TIMERx registers(x=13)

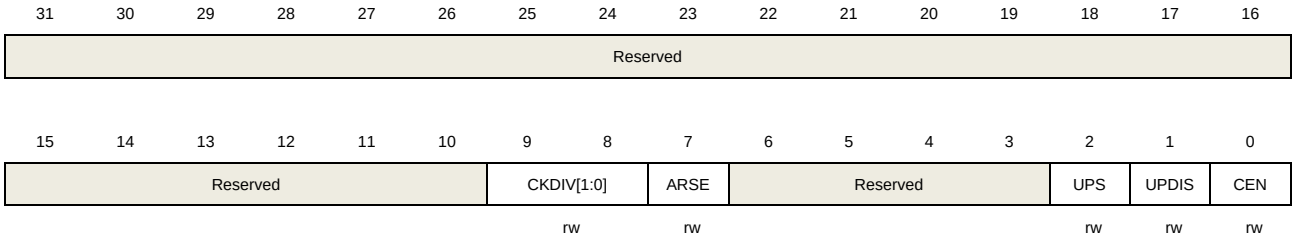
TIMER13 base address: 0x4000 2000

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS).<br>00: $f_{DTS}=f_{CK_TIMER}$<br>01: $f_{DTS}= f_{CK_TIMER}$ /2<br>10: $f_{DTS}= f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:3 | Reserved | Must be kept at reset value. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: This event generates update interrupts or DMA requests:<br>The counter generates an overflow or underflow event |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow |

| | | |
|---|---|---|
| | | registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: Update event disable.<br>**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decoder mode. |

### Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CH0IE | UPIE |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CH0OF | Reserved. | | | | | | | CH0IF | UPIF |
| | | | | | | rc_w0 | | | | | | | | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IF | Channel 0 ‘s capture/compare interrupt flag<br>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No Channel 1 interrupt occurred<br>1: Channel 1 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CH0G | UPG |
| | | | | | | | | | | | | | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |

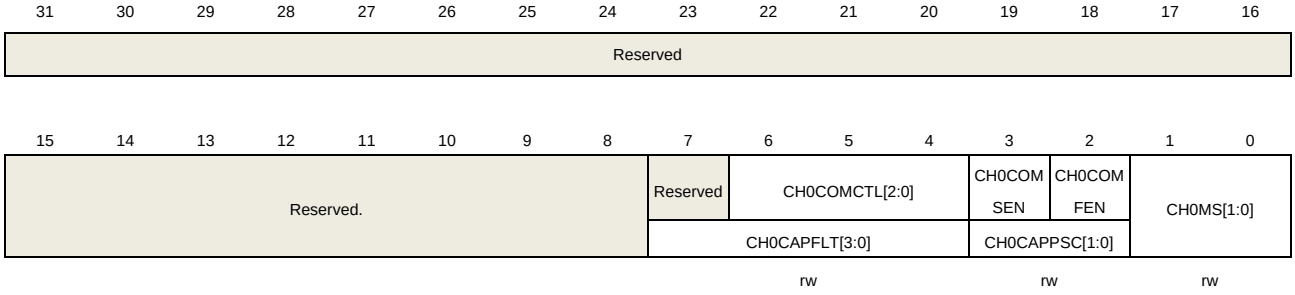
| | | |
|---|---|---|
| 1 | CH0G | Channel 0's capture or compare event generation<br>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 1 capture or compare event<br>1: Generate a channel 1 capture or compare event |
| 0 | UPG | This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter is equals to the output |

compare register TIMERx_CH0CV.
100: Force low. O0CPRE is forced to low level.
101: Force high. O0CPRE is forced to high level.
110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.
111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.
If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.
This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).

| | | |
|---|---|---|
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.<br>0: Channel 0 output compare shadow disable<br>1: Channel 0 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH0COMFEN | Channel 0 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.<br>0: Channel 0 output quickly compare disable.<br>1: Channel 0 output quickly compare enable. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).<br>00: Channel 0 is programmed as output mode<br>01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0<br>10: Reserved<br>11: Reserved |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| CH0CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}/32$ |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

| | | |
|---|---|---|
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.<br>00: Prescaler disable, input capture occurs on every channel input edge<br>01: The input capture occurs on every 2 channel input edges<br>10: The input capture occurs on every 4 channel input edges<br>11: The input capture occurs on every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 mode selection<br>Same as output compare mode |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | CH0NP | Reserved | CH0P | CH0EN |
| | | | | | | | | | | | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 2 | Reserved | Must be kept at reset value |
| 1 | CH0P | Channel 0 capture/compare polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 high level is active level<br>1: Channel 0 low level is active level<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: Reserved.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in input mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in output mode, setting this bit enables the capture event in channel0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter.<br>**Note:** When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel0<br>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel input remap register(TIMERx_IRMP)

Address offset: 0x50
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CI0_RMP[1:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1:0 | CI0_RMP[1:0] | Channel 0 input remap<br>00: Channel 0 input is connected to GPIO(TIMER13_CH0)<br>01: Channel 0 input is connected to the RTCCLK<br>10: Channel 0 input is connected to HXTAL/32 clock<br>11: Channel 0 input is connected to CKOUTSEL |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CHVSEL | Reserved |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CHVSEL | Write CHxVAL register selection<br>This bit-field set and reset by software.<br>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored<br>0: No effect |
| 0 | Reserved | Must be kept at reset value |

# 14.4. General level3 timer (TIMERx, x=14)

## 14.4.1. Overview

The general level3 timer module (TIMER14) is a two-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level3 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level3 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which is suitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

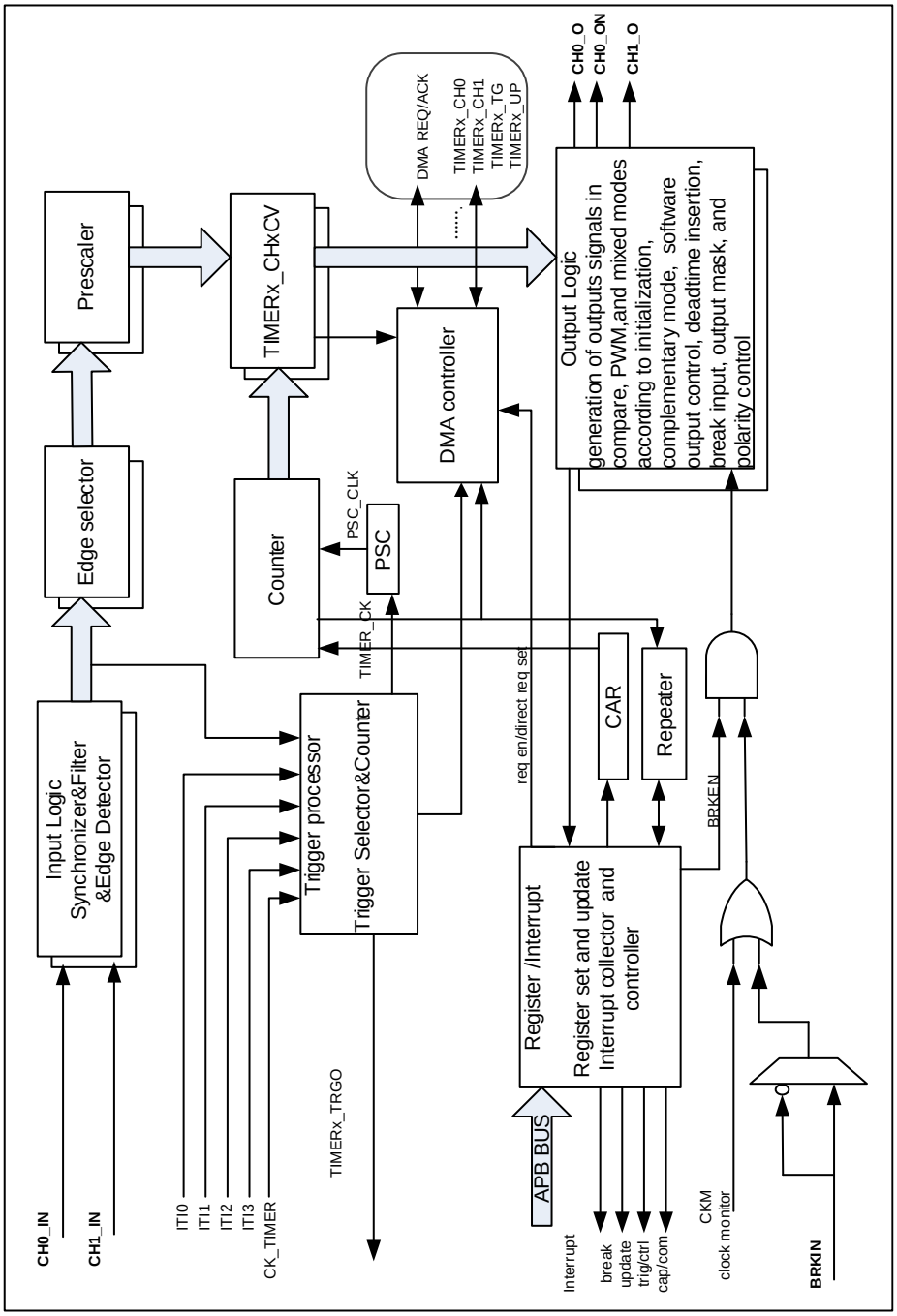
## 14.4.2. Characteristics

- Total channel num: 2.
- Counter width: 16 bits.
- Clock source of timer is selectable:
  internal clock, internal trigger, external input.
- Counter modes: count up only.
- Programmable prescaler: 16 bits.The factor can be changed ongoing.
- Each channel is user-configurable:
  input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update event, trigger event, compare/capture event, and break input.
- Daisy chaining of timer modules allows a single timer to start multiple timers.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master-slave management.

## 14.4.3. Block diagram

*Figure 14-53. General level3 timer block diagram* provides details of the internal configuration of the general level3 timer.

**Figure 14-53. General level3 timer block diagram**




## 14.4.4. Function overview

### Clock source configuration

The clock source of the advanced timer can be either the CK_TIMER or an alternate clock source controlled by SMC bits (TIMERx_SMCFG bit[2:0]).

- SMC [2:0] == 3'b000. Internal clock CK_TIMER is selected as timer clock source which

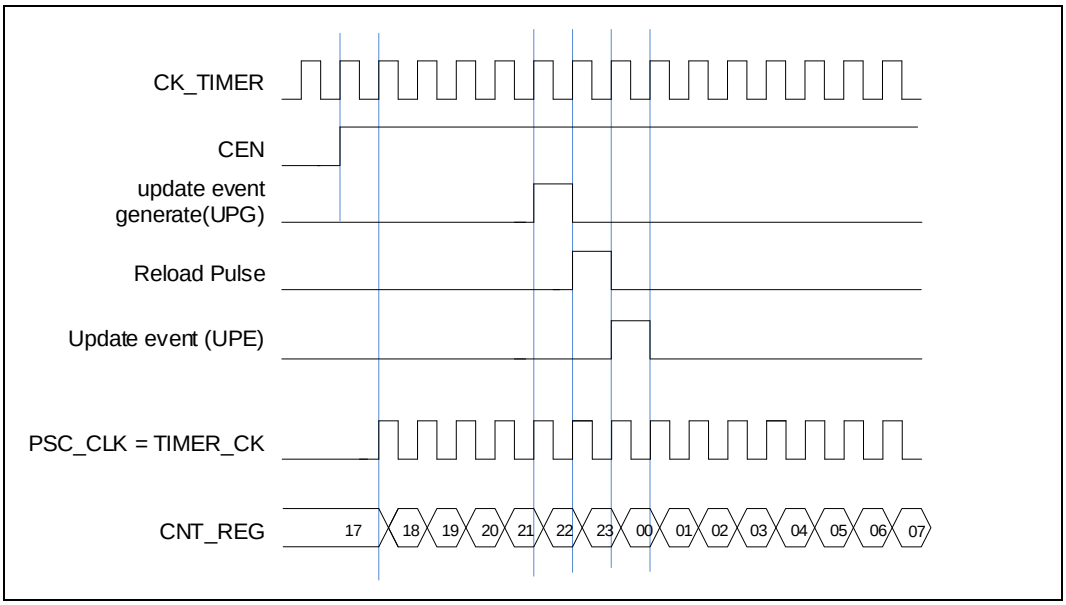
is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when the SMC [2:0] == 3'b000. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, which drives counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register, details as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

**Figure 14-54. Timing chart of internal clock divided by 1**




- SMC [2:0] == 3'b111 (external clock mode 0). External input pin is selected as timer clock source

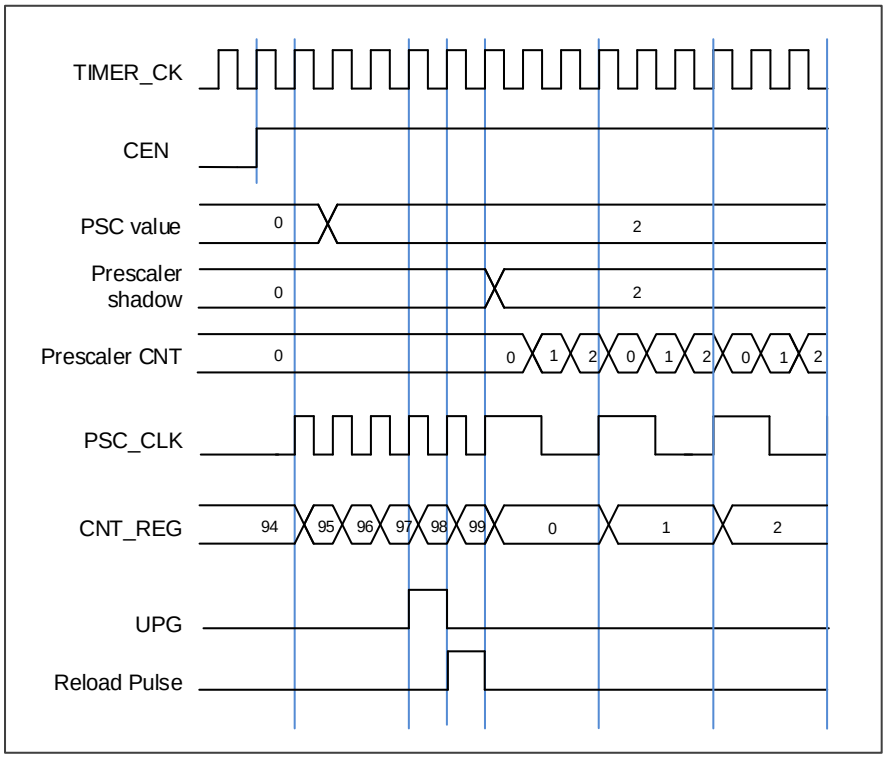
The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

### Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-55. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again and an overflow event will be generated. In addition, the update events will be generated after (TIMERx_CREP+1) times of overflow events. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

***Figure 14-56. Timing chart of up counting mode, PSC=0/2*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-56. Timing chart of up counting mode, PSC=0/2**

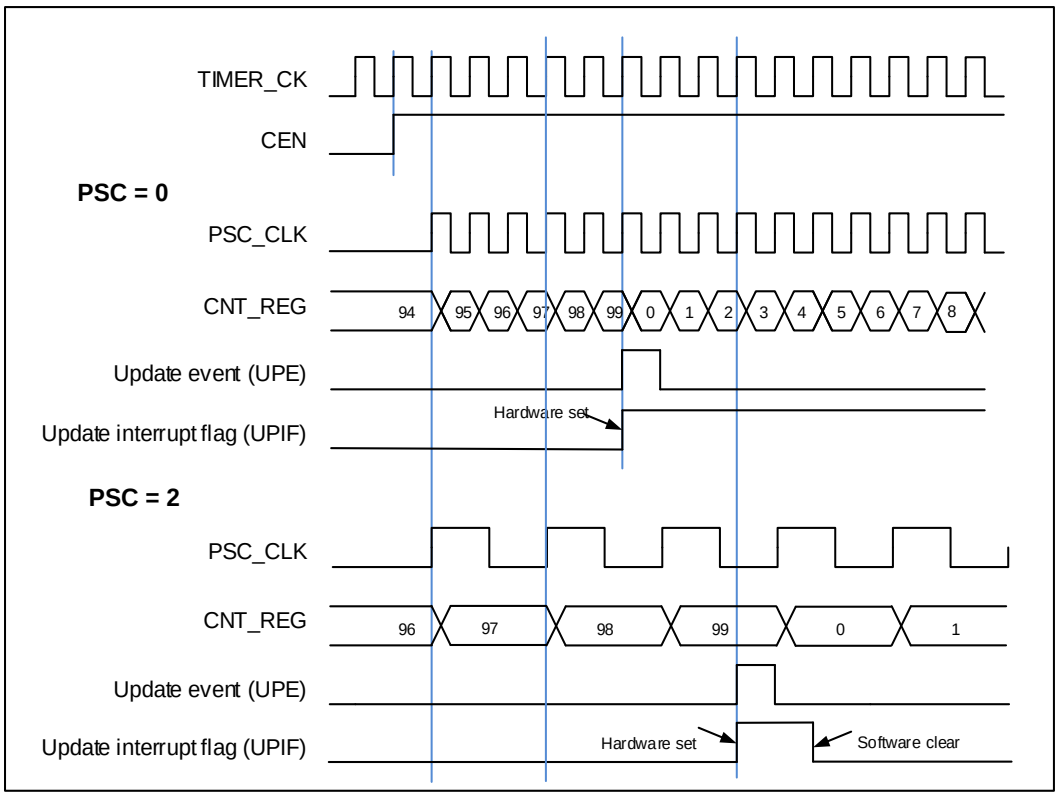
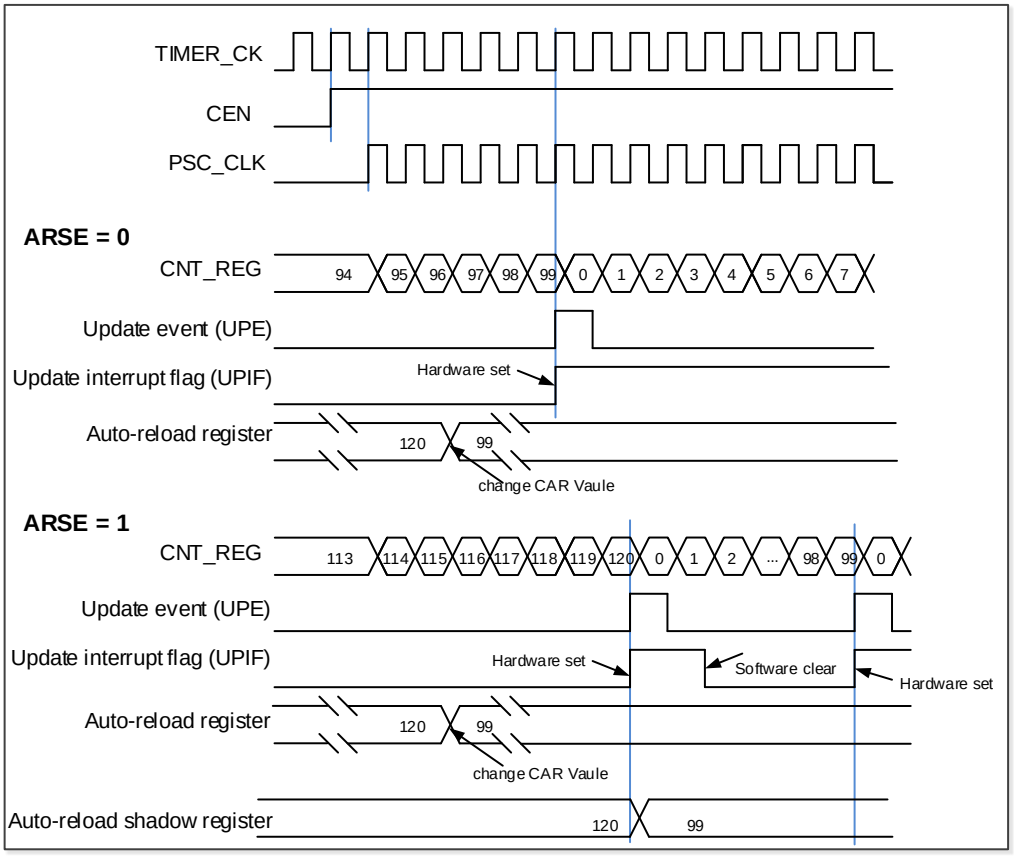



**Figure 14-57. Timing chart of up counting mode, change TIMERx_CAR on the go**

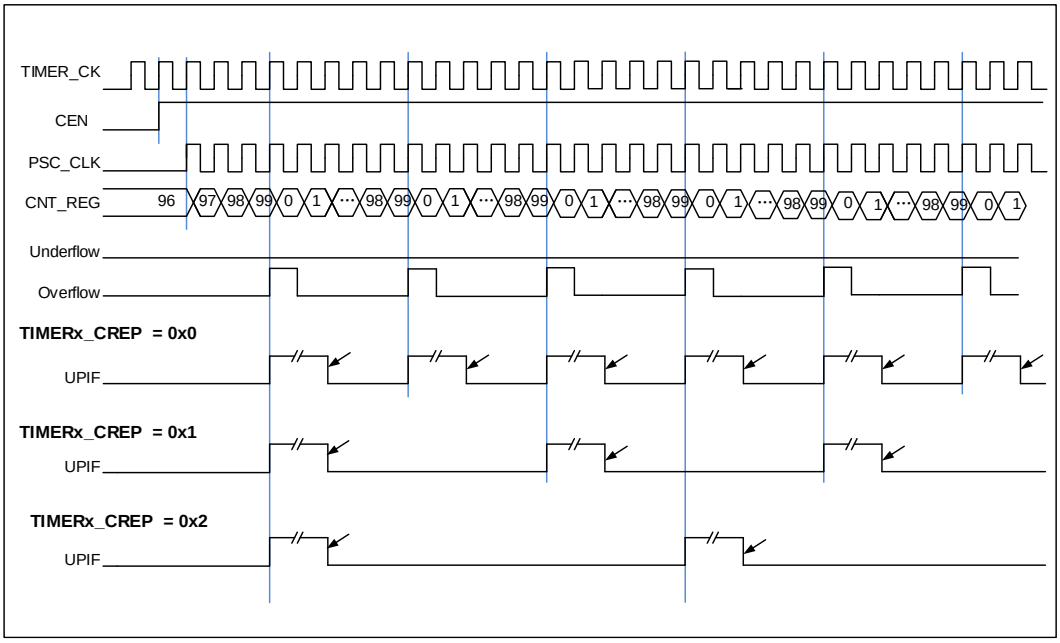


### Update event (from overflow/underflow) rate configuration

The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx_CREP register. Counter repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in TIMERx_CREP register. The repetition counter is decremented at each counter overflow in up-counting mode.

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP in TIMERx_CREP register and generator an update event.

**Figure 14-58. Repetition counter timing chart of up counting mode**



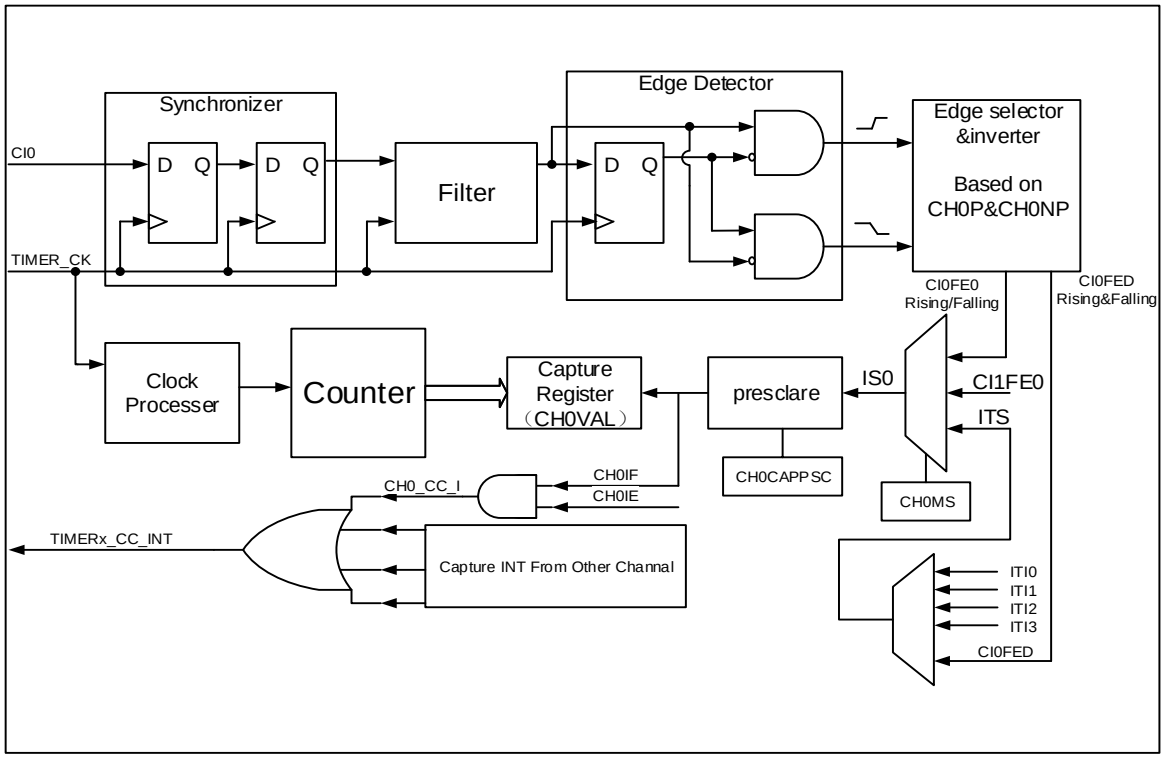

## Input capture and output compare channels

The general level3 timer has two independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Channel input capture function**

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if it is enabled when CHxIE=1.

**Figure 14-59. Channel input capture principle**




Channels' input signals (CIx) is the TIMERx_CHx signal. First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

**Step1**: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2**: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3**: Capture source selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode ( CHxMS!=0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable. (CHxIE and CHxDEN in TIMERx_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5**: Capture enables. (CHxEN in TIMERx_CHCTL2)

**Result**: when you wanted input signal is got, TIMERx_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and

CHxDEN in TIMERx_DMAINTEN

**Direct generation**: if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

The channel input capture function can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Channel output compare function**

**Figure 14-60. Channel output compare principle (with complementary output, x=0)**

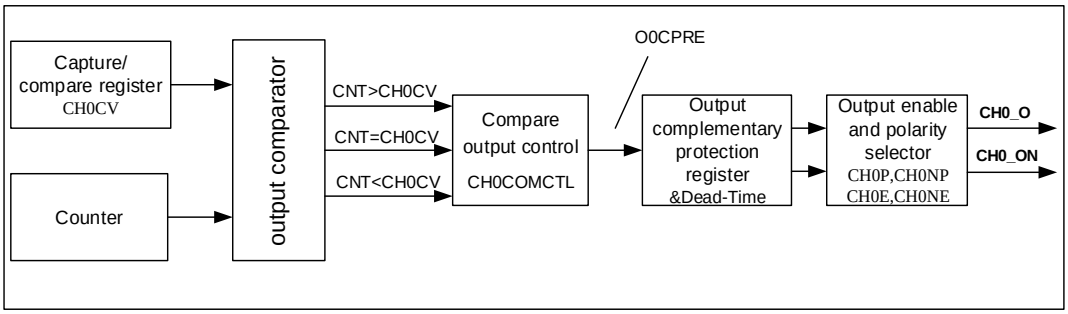



**Figure 14-61. Channel output compare principle (CH1_O)**

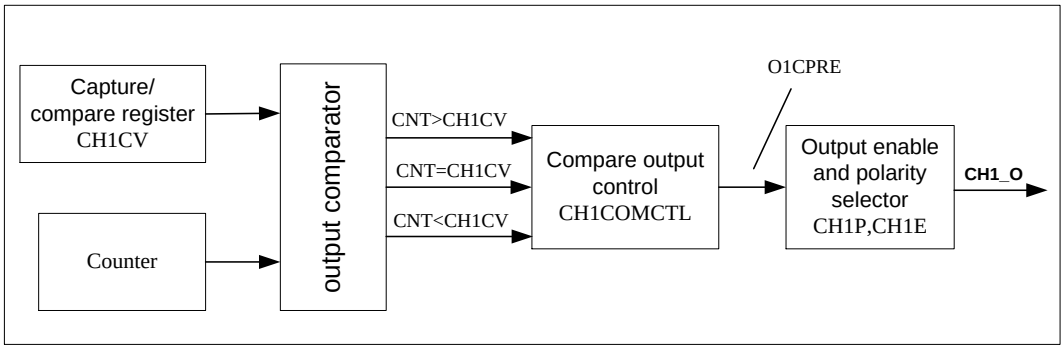



*Figure 14-60. Channel output compare principle (with complementary output, x=0)* and *Figure 14-61. Channel output compare principle (CH1_O)* show the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O/CHx_ON and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O/CH0_ON depends on OxCPRE signal, CHxP/CHxNP bit and CH0E/CH0NE bit (please refer to the TIMERx_CHCTL2 register for more details). For examples, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxE=1 (the output of CHx_O is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

Configure CHxNP=0 (the active level of CHx_ON is low, contrary to OxCPRE), CHxNE=1 (the output of CHx_ON is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(low) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(high) level.

When CH0_O and CH0_ON are output at the same time, the specific outputs of CH0_O and CH0_ON are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register.

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CHxDEN =1.

So the process can be divided to several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

* Set the shadow enable mode by CHxCOMSEN
* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
* Select the active high polarity by CHxP/CHxNP
* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/ CHxDEN

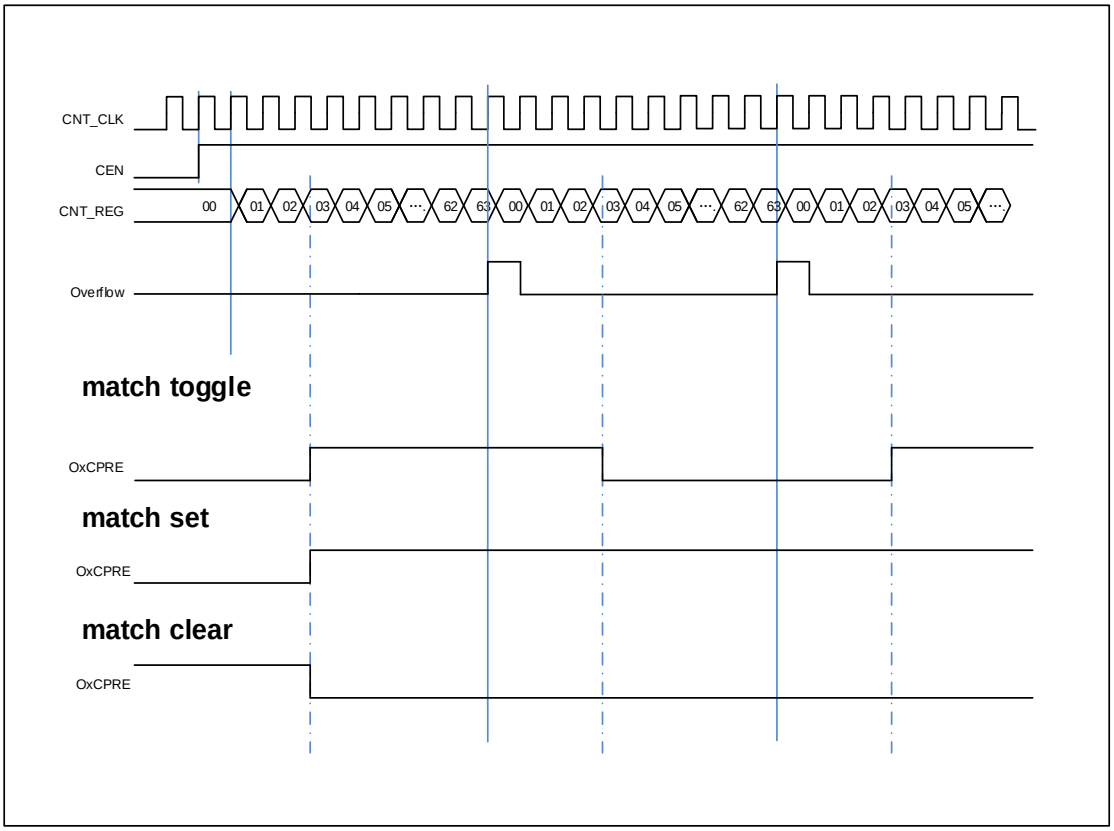
**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV

About the CHxVAL; you can change it ongoing to meet the waveform you expected.

**Step5:** Start the counter by CEN.

***Figure 14-62. Output-compare in three modes*** show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 14-62. Output-compare in three modes**




Output PWM function

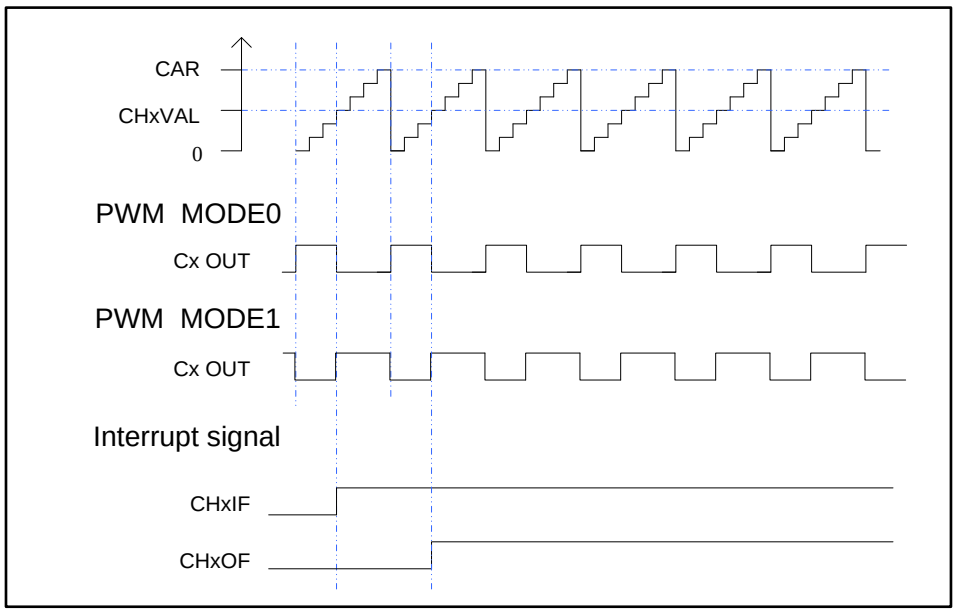
In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

The period is determined by TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV. ***Figure 14-63. PWM mode timechart*** shows the PWM output mode and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 14-63. PWM mode timechart**




### Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

### Channel output complementary PWM

Function of complementary is for a pair of channels, CHx_O and CHx_ON, the two output signals cannot be active at the same time. The TIMERx has 4 channels, but only the first three channels have this function. The complementary signals CHx_O and CHx_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and CHxNP bits in

the TIMERx_CHCTL2 register.

**Table 14-5. Complementary outputs controlled by parameters**

| Complementary Parameters | | | | | Output Status | |
|---|---|---|---|---|---|---|
| POEN | ROS | IOS | CHxEN | CHxNEN | CHx_O | CHx_ON |
| 0 | 0/1 | 0 | 0 | 0 | CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1). | |
| | | | | 1 | CHx_O/ CHx_ON output “off-state” (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3) | |
| | | | 1 | 0 | | |
| | | | | 1 | | |
| | | 1 | x | x | CHx_O/ CHx_ON output “off-state”:<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. | |
| 1 | 0 | 0/1 | 0 | 0 | CHx_O/CHx_ON = LOW<br>CHx_O/CHx_ON output disable. | |
| | | | | 1 | CHx_O = LOW<br>CHx_O output disable. | CHx_ON =OxCPRE⊕(4)CHxNP<br>CHx_ON output enable. |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | CHx_ON = LOW<br>CHx_ON output disable. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | CHx_ON =(!OxCPRE)(5)⊕ CHxNP.<br>CHx_ON output enable. |
| | 1 | | 0 | 0 | CHx_O = CHxP<br>CHx_O output “off-state”. | CHx_ON = CHxNP<br>CHx_ON output “off-state”. |
| | | | | 1 | CHx_O = CHxP<br>CHx_O output “off-state” | CHx_ON =OxCPRE⊕CHxNP<br>CHx_ON output enable |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON = CHxNP<br>CHx_ON output “off-state”. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON =(!OxCPRE)⊕ CHxNP<br>CHx_ON output enable. |

**Note:**

(1) Output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “Off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (!OxCPRE)：the complementary output of the OxCPRE signal.

### Insertion dead time for complementary PWM

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for channel 0. The detail about the delay time, refer to the register TIMERx_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

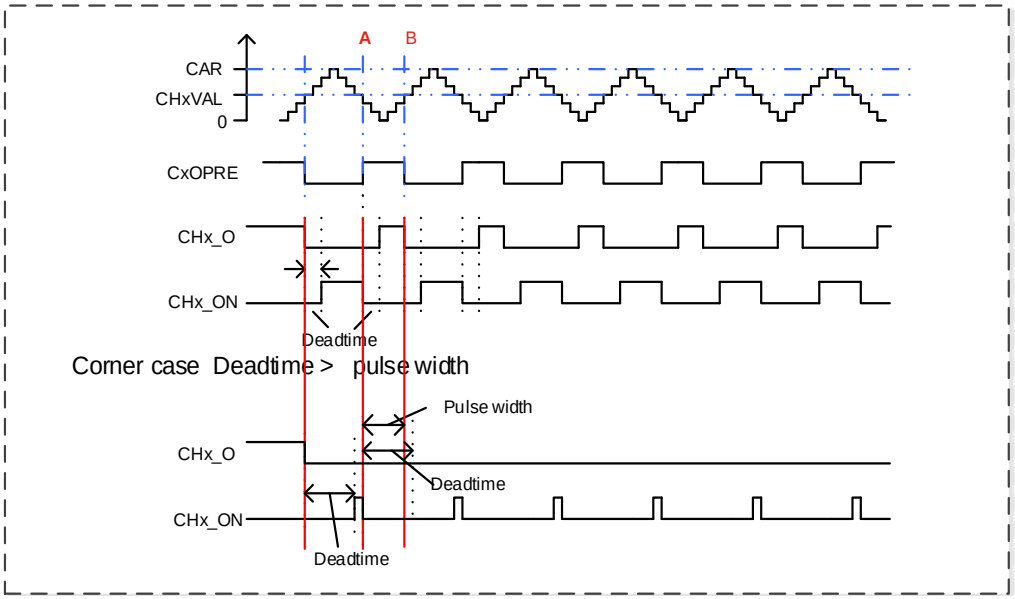
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the ***Figure 14-64. Complementary output with dead-time insertion.*** CHx_O signal remains at the low value until the end of the deadtime delay, while CHx_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx_O signal will be cleared at once, while CHx_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx_O duty cycle, then the CHx_O signal is always the inactive value. (as show in the ***Figure 14-64. Complementary output with dead-time insertion.***)

The dead time delay is greater than or equal to the CHx_ON duty cycle, then the CHx_ON signal is always the inactive value.

**Figure 14-64. Complementary output with dead-time insertion.**



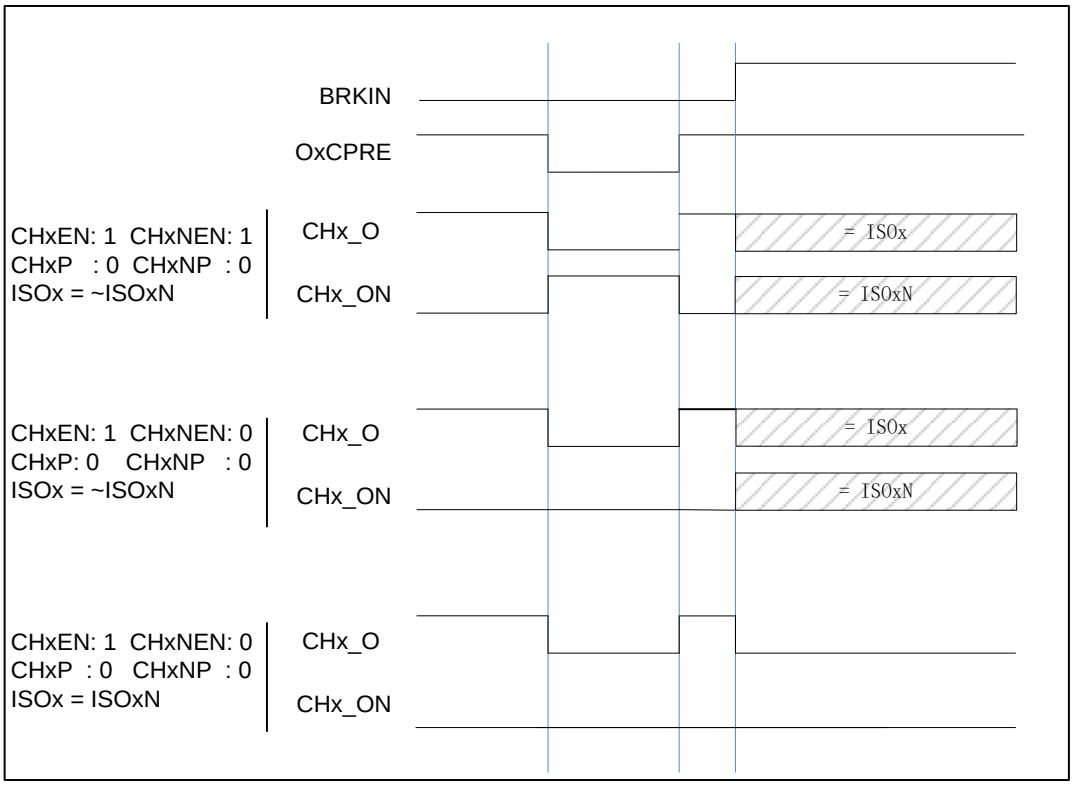
### Break mode

In this mode, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx_O and CHx_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 14-65. Output behavior in response to a break(The break high active)**

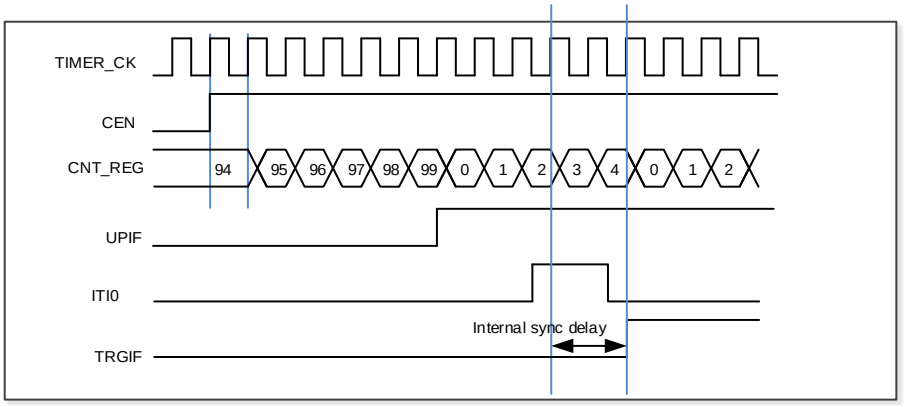


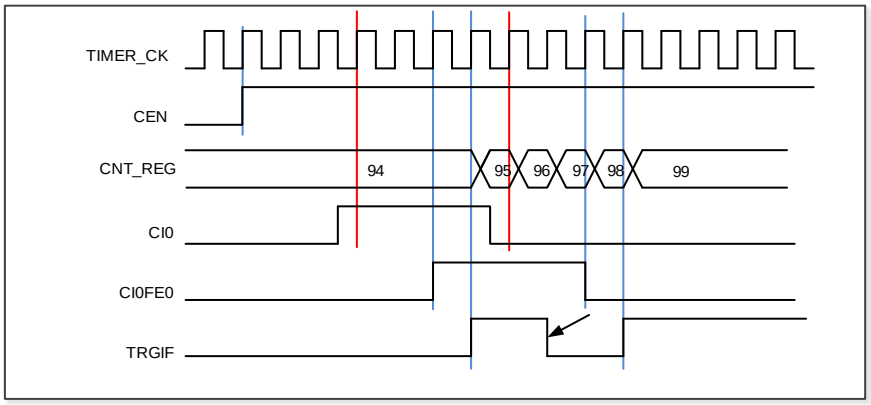
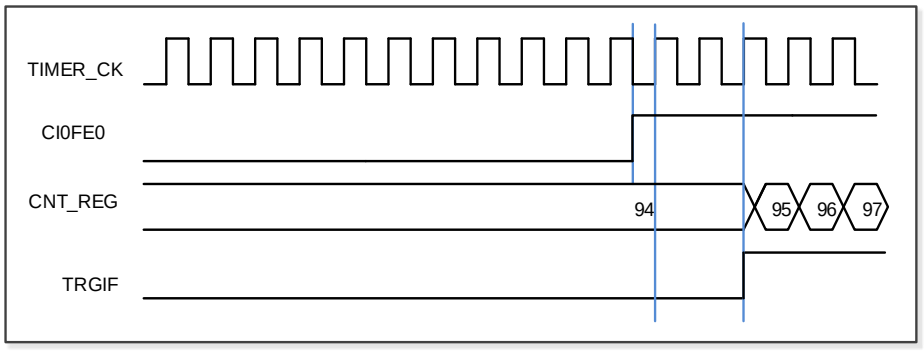

### Master-slave management

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC[2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the

TRGS[2:0] in the TIMERx_SMCFG register.

**Table 14-6. Slave mode example table**

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| LIST | SMC[2:0]<br>3'b100 (restart mode)<br>3'b101 (pause mode)<br>3'b110 (event mode) | TRGS[2:0]<br>000: ITI0<br>001: ITI1<br>010: ITI2<br>011: ITI3<br>100: CI0F_ED<br>101: CI0FE0<br>110: CI1FE1<br>111: Reserved | If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. | For the ITIx no filter and prescaler can be used.<br>For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. |
| Exam1 | Restart mode<br>The counter can be clear and restart when a rising trigger input. | TRGS[2:0]=3'b000<br>ITI0 is the selection. | -<br>For ITI0, no polarity selector can be used. | -<br>For the ITI0, no filter and prescaler can be used. |
| | **Figure 14-66. Restart mode**<br> | | | |
| Exam2 | Pause mode<br>The counter can be paused when the trigger input is low. | TRGS[2:0]=3'b101<br>CI0FE0 is the selection. | TI0S=0.(Non-xor) [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only. | Filter is bypass in this example. |

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| | **Figure 14-67. Pause mode**<br> | | | |
| Exam3 | Event mode<br>The counter will start to count when a rising trigger input. | TRGS[2:0]=3'b101<br>CI0FE0 is the selection. | TI0S=0.(Non-xor)<br>[CH0NP==0, CH0P==0]<br>no inverted. | Filter is bypass in this example. |
| | **Figure 14-68. Event mode**<br> | | | |

### Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

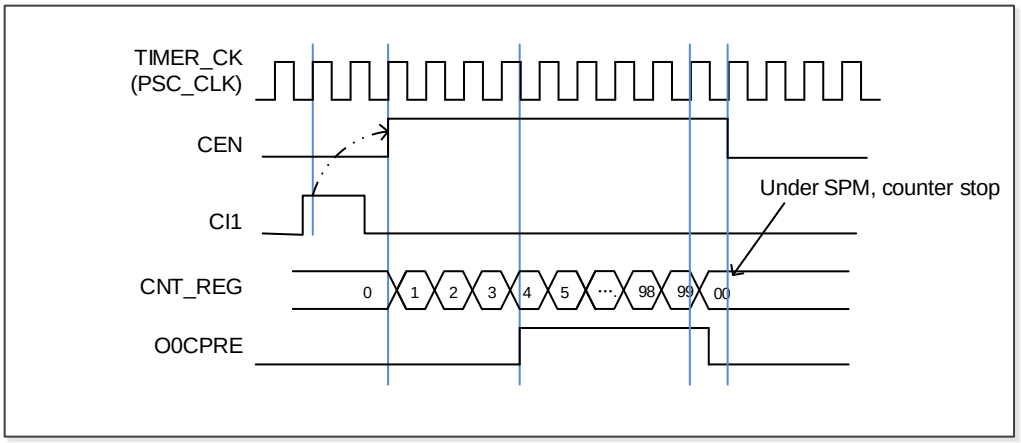
Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or

the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

***Figure 14-69. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99*** shows an example.

**Figure 14-69. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99**




## Timers interconnection

Refer to ***Timers interconnection***

## Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG . If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx_DMATC is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx_DMATB. In one word, one time DMA internal interrupt event

assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### Timer debug mode

When the Cortex®-M23 halted, and the TIMERx_HOLD configuration bit in DBG_CTL1 register set to 1, the TIMERx counter stops.

## 14.4.5. TIMERx registers(x=14)

TIMER14 base address: 0x4001 4000

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CKDIV[1:0] | | ARSE | Reserved | | | SPM | UPS | UPDIS | CEN |
| | | | | | | rw | | rw | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS).<br>00: $f_{DTS}=f_{CK_TIMER}$<br>01: $f_{DTS}= f_{CK_TIMER}$ /2<br>10: $f_{DTS}= f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value. |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode disable. The counter continues after update event.<br>1: Single pulse mode enable. The counter counts until the next update event occurs. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: This event generates update interrupts or DMA requests: |

| | | |
|---|---|---|
| | | The counter generates an overflow or underflow event |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: Update event disable.<br>**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decoder mode. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | ISO1 | ISO0N | ISO0 | Reserved | MMC[2:0] | | | DMAS | CCUC | Reserved | CCSE |
| | | | | | rw | rw | rw | | rw | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value |
| 10 | ISO1 | Idle state of channel 1 output<br>Refer to ISO0 bit |
| 9 | ISO0N | Idle state of channel 0 complementary output<br>0: When POEN bit is reset, CH0_ON is set low.<br>1: When POEN bit is reset, CH0_ON is set high<br>This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output |

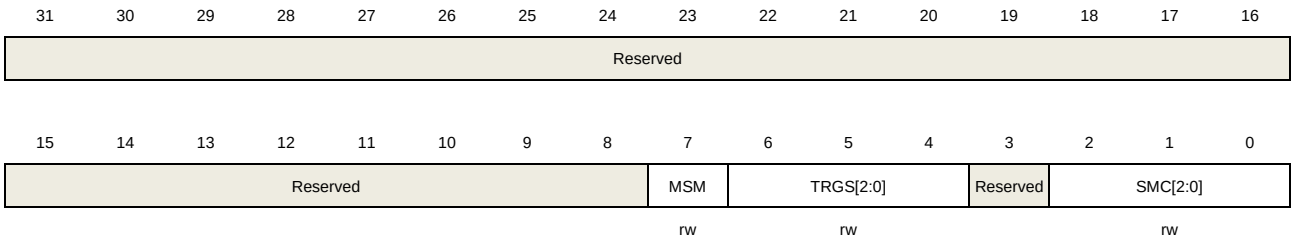
| | | |
|---|---|---|
| | | 0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high<br>The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 7 | Reserved | Must be kept at reset value |
| 6:4 | MMC[2:0] | Master mode control<br>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.<br>000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source:<br>Master timer generate a reset<br>the UPG bit in the TIMERx_SWEVG register is set<br>001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source :<br>CEN control bit is set<br>The trigger input in pause mode is high<br>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.<br>011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.<br>100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.<br>101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE.<br>110: Reserved<br>111: Reserved |
| 3 | DMAS | DMA request source selection<br>0: When capture or compare event occurs, the DMA request of channel x is sent<br>1: When update event occurs, the DMA request of channel x is sent. |
| 2 | CCUC | Commutation control shadow register update control<br>When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:<br>0: The shadow registers update by when CMTG bit is set.<br>1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.<br>When a channel does not have a complementary output, this bit has no effect. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled. |

1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled. After these bits have been written, they are updated based when commutation event coming.
When a channel does not have a complementary output, this bit has no effect.

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | MSM | Master-slave mode<br>This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.<br>0: Master-slave mode disable<br>1: Master-slave mode enable |
| 6:4 | TRGS[2:0] | Trigger selection<br>This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.<br>000: ITI0<br>001: ITI1<br>010: ITI2<br>011: ITI3<br>100: CI0 edge flag (CI0F_ED)<br>101: Channel 0 input filtered output (CI0FE0)<br>110: Channel 1 input filtered output (CI1FE1)<br>111: Reserved<br>These bits must not be changed when slave mode is enabled. |
| 3 | Reserved | Must be kept at reset value |
| 2:0 | SMC[2:0] | Slave mode control |

000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high.
001: Reserved
010: Reserved
011: Reserved
100: Restart Mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.
101: Pause Mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.
110: Event Mode. A rising edge of the trigger input enables the counter.
111: External Clock Mode 0. The counter counts on the rising edges of the selected trigger.
Because CI0F_ED outputs 1 pulse for each transition on CI0F, and the pause mode checks the level of the trigger signal, when CI0F_ED is selected as the trigger input, the pause mode must not be used.

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | TRGDEN | CMTDEN | Reserved | CH1DEN | | CH0DEN | UPDEN | BRKIE | TRGIE | CMTIE | Reserved | | CH1IE | CH0IE | UPIE |
| | rw | rw | | | rw | rw | rw | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value |
| 14 | TRGDEN | Trigger DMA request enable<br>0: disabled<br>1: enabled |
| 13 | CMTDEN | Commutation DMA request enable<br>0: disabled<br>1: enabled |
| 12:11 | Reserved | Must be kept at reset value. |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: disabled<br>1: enabled |

| | | |
|---|---|---|
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 8 | UPDEN | Update DMA request enable<br>0: disabled<br>1: enabled |
| 7 | BRKIE | Break interrupt enable<br>0: disabled<br>1: enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: disabled<br>1: enabled |
| 5 | CMTIE | commutation interrupt enable<br>0: disabled<br>1: enabled |
| 4:3 | Reserved | Must be kept at reset value |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | CH1OF | CH0OF | Reserved | BRKIF | TRGIF | CMTIF | Reserved. | | CH1IF | CH0IF | UPIF |
| | | | | | rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |
| 7 | BRKIF | Break interrupt flag<br>When the break input is inactive, the bit is set by hardware.<br>When the break input is inactive, the bit can be cleared by software.<br>0: No active level break has been detected.<br>1: An active level has been detected. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event.<br>0: No trigger event occurred.<br>1: Trigger interrupt occurred. |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when channel's commutation event occurs, and cleared by software<br>0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4:3 | Reserved | Must be kept at reset value |
| 2 | CH1IF | Channel 1 's capture/compare interrupt flag<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 's capture/compare interrupt flag<br>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No Channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software. |

0: No update interrupt occurred

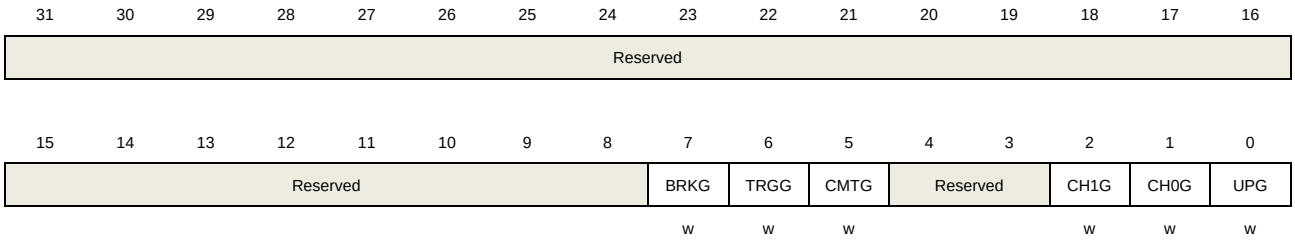
1: Update interrupt occurred

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | BRKG | Break event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a break event<br>1: Generate a break event |
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event<br>1: Generate a trigger event |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel's c/c control update event |
| 4:3 | Reserved | Must be kept at reset value |
| 2 | CH1G | Channel 1's capture or compare event generation<br>Refer to CH0G description |

| | | |
|---|---|---|
| 1 | CH0G | Channel 0's capture or compare event generation<br>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 1 capture or compare event<br>1: Generate a channel 1 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | CH1COMCTL[2:0] | | | CH1COMSEN | CH1COMFEN | CH1MS[1:0] | | Reserved | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| CH1CAPFLT[3:0] | | | | CH1CAPPSC[1:0] | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | CH1COMFEN | Channel 1 output compare fast enable<br>Refer to CH0COMFEN description |

| | | |
|---|---|---|
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is programmed as output mode<br>01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1<br>10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1<br>11: Channel 1 is programmed as input mode, IS1 is connected to ITS.<br>**Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced to low level.<br>101: Force high. O0CPRE is forced to high level.<br>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.<br>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.<br>0: Channel 0 output compare shadow disable<br>1: Channel 0 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single |

| | | |
|---|---|---|
| | | pulse mode (when SPM=1)<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00. |
| 2 | CH0COMFEN | Channel 0 output compare fast enable<br>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.<br>0: Channel 0 output quickly compare disable.<br>1: Channel 0 output quickly compare enable. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).<br>00: Channel 0 is programmed as output mode<br>01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0<br>10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0<br>11: Channel 0 is programmed as input mode, IS0 is connected to ITS<br>**Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>Same as Output compare mode |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.<br>Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.<br>The filtering capability configuration is as follows: |

| CH0CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4’b0000 | Filter disabled. | |

| | | | | |
|---|---|---|---|---|
| | | 4’b0001 | 2 | $f_{CK_TIMER}$ |
| | | 4’b0010 | 4 | |
| | | 4’b0011 | 8 | |
| | | 4’b0100 | 6 | $f_{DTS}/2$ |
| | | 4’b0101 | 8 | |
| | | 4’b0110 | 6 | $f_{DTS}/4$ |
| | | 4’b0111 | 8 | |
| | | 4’b1000 | 6 | $f_{DTS}/8$ |
| | | 4’b1001 | 8 | |
| | | 4’b1010 | 5 | $f_{DTS}/16$ |
| | | 4’b1011 | 6 | |
| | | 4’b1100 | 8 | |
| | | 4’b1101 | 5 | $f_{DTS}/32$ |
| | | 4’b1110 | 6 | |
| | | 4’b1111 | 8 | |

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

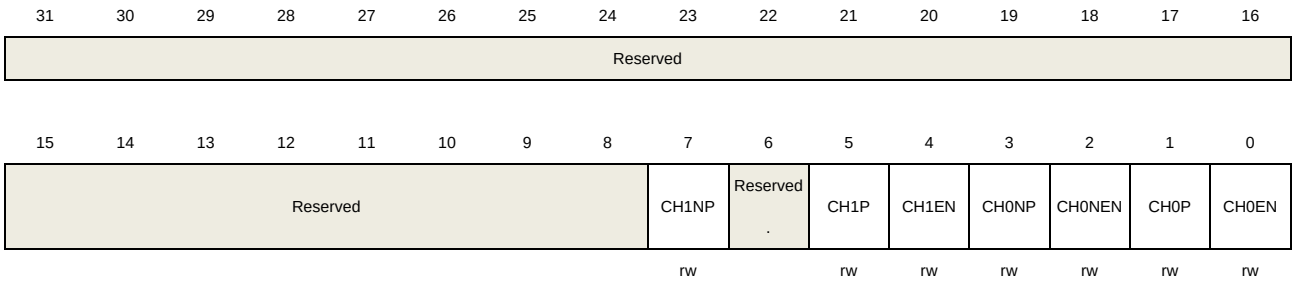
Same as Output compare mode

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | CH1NP | Channel 1 complementary output polarity |

| | | |
|---|---|---|
| | | Refer to CH0NP description |
| 6 | Reserved | Must be kept at reset value |
| 5 | CH1P | Channel 1 capture/compare function polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 capture/compare function enable<br>Refer to CH0EN description |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.<br>0: Channel 0 complementary output high level is active level<br>1: Channel 0 complementary output low level is active level<br>When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 2 | CH0NEN | Channel 0 complementary output enable<br>When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 capture/compare function polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 high level is active level<br>1: Channel 0 low level is active level<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: Reserved.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables |

the capture event in channel0.

0: Channel 0 disabled

1: Channel 0 enabled

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter. |

### Counter repetition register (TIMERx_CREP)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | CREP[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP[7:0] | Counter repetition value<br>This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel0<br>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH1VAL[15:0] | Capture or compare value of channel1<br>When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event. |

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POEN | OAEN | BRKP | BRKEN | ROS | IOS | PROT[1:0] | | DTCFG[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | POEN | Primary output enable<br>The bit can be set to 1 by:<br>- Write 1 to this bit<br>- If OAEN is set to 1, this bit is set to 1 at the next update event.<br>The bit can be cleared to 0 by:<br>- Write 0 to this bit<br>- Valid fault input (asynchronous).<br>When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set.<br>0: Disable channel outputs (CHxO or CHxON).<br>1: Enabled channel outputs (CHxO or CHxON).<br>**Note:** This bit is only valid when CHxMS=2'b00. |
| 14 | OAEN | Output automatic enable<br>0: The POEN bit can only be set by software.<br>1: POEN can be set at the next update event, if the break input is not active.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 13 | BRKP | Break polarity<br>This bit specifies the polarity of the BRKIN input signal.<br>0: BRKIN input active low<br>1: BRKIN input active high |
| 12 | BRKEN | Break enable<br>This bit can be set to enable the BRKIN and CCS clock failure event inputs.<br>0: Break inputs disabled<br>1: Break inputs enabled |

This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

11 ROS

Run mode “off-state” enable

When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 14-5. Complementary outputs controlled by parameters***.

0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.

1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.

This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.

10 IOS

Idle mode “off-state” enable

When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 14-5. Complementary outputs controlled by parameters***.

0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.

1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.

This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.

9:8 PROT[1:0]

Complementary register protect control

This bit-filed specifies the write protection property of registers.

00: protect disable. No write protection.

01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.

10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected.

11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN bits in TIMERx_CHCTL0 registers (if the related channel is configured in output) are writing protected.

This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.

7:0 DTCFG[7:0]

Dead time configure

The relationship between DTVAL value and the duration of dead-time is as follow:

| DTCFG[7:5] | The duration of dead-time |
|---|---|
| 3’b0xx | DTCFG[7:0] * $t_{DTS_CK}$ |
| 3’b10x | (64+ DTCFG[5:0]) * $t_{DTS_CK}$ *2 |
| 3’b110 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *8 |

| | |
|---|---|
| 3'b111 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *16 |

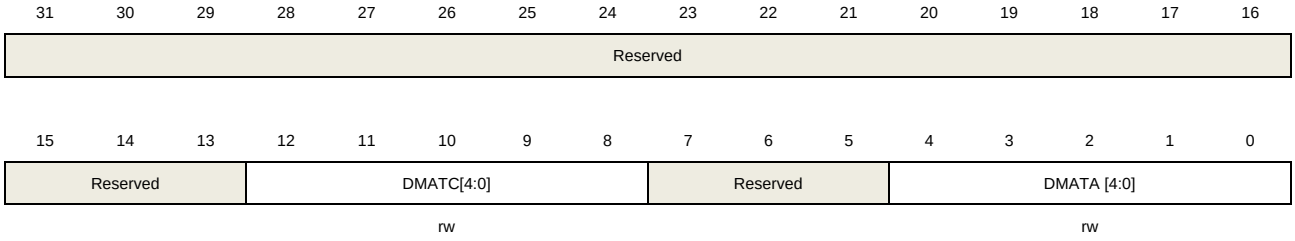
**Note:**

1. $t_{DTS_CK}$ is the period of DTS_CK which is configured by CKDIV[1:0] in TIMERx_CTL0.
2. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

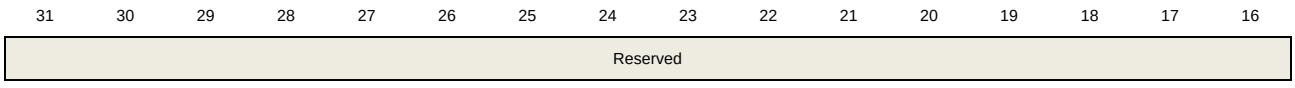



| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:8 | DMATC [4:0] | DMA transfer count<br>This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | DMATA [4:0] | DMA transfer access start address<br>This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4. |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | DMATB[15:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.<br>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CHVSEL | OUTSEL |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CHVSEL | Write CHxVAL register selection<br>This bit-field set and reset by software.<br>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored<br>0: No effect |
| 0 | OUTSEL | The output value selection<br>This bit-field set and reset by software<br>1: If POEN and IOS is 0, the output disabled<br>0: No effect |

# 14.5. General level4 timer (TIMERx, x=15, 16)

## 14.5.1. Overview

The general level4 timer module (TIMER15, TIMER16) is a one-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level4 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level4 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which is suitable for motor control applications.

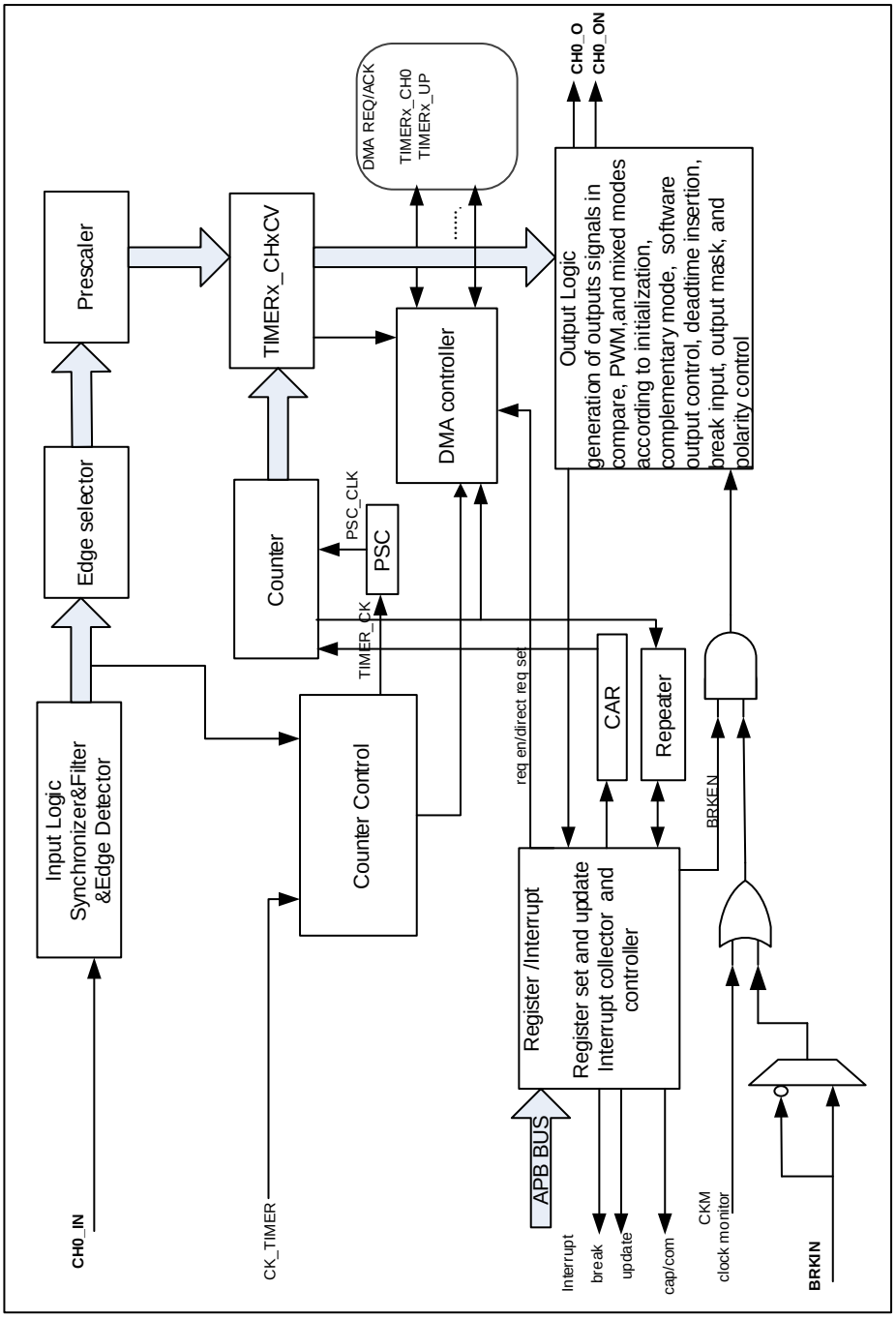
## 14.5.2. Characteristics

- Total channel num: 1.
- Counter width: 16 bits.
- Clock source of counter clock: internal clock.
- Counter modes: count up only.
- Programmable prescaler: 16 bits.The factor can be changed ongoing.
- Each channel is user-configurable:
  input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update event, compare/capture event, and break input.

## 14.5.3. Block diagram

***Figure 14-70. General level4 timer block diagram*** provides details of the internal configuration of the general level4 timer.

**Figure 14-70. General level4 timer block diagram**




## 14.5.4. Function overview

### Clock source configuration

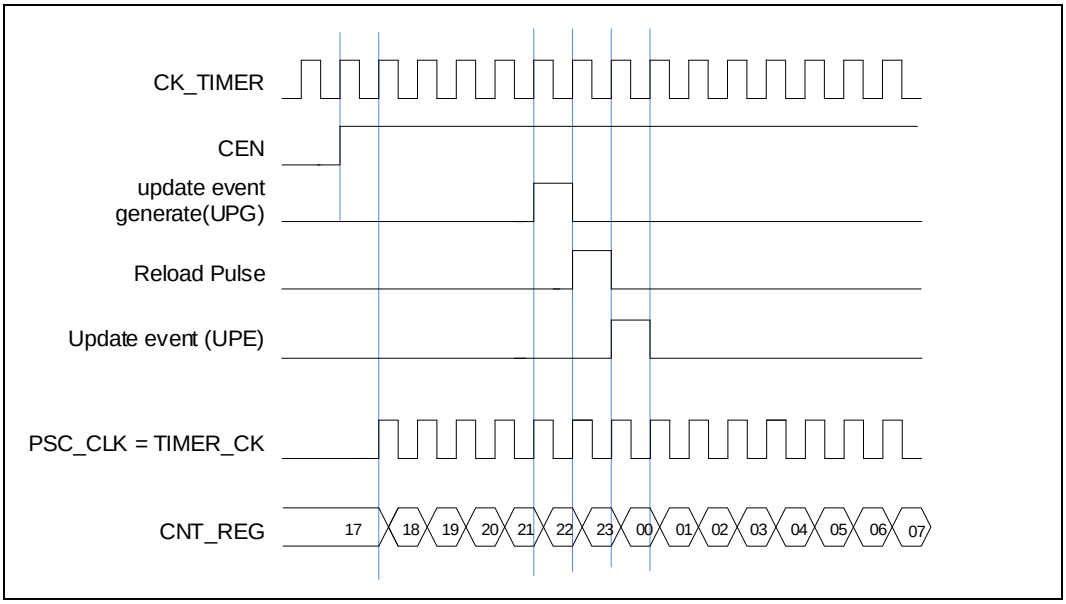
The general level4 TIMER can only being clocked by the CK_TIMER.

- Internal timer clock CK_TIMER which is from module RCU

The general level4 TIMER has only one clock source which is the internal CK_TIMER, used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU

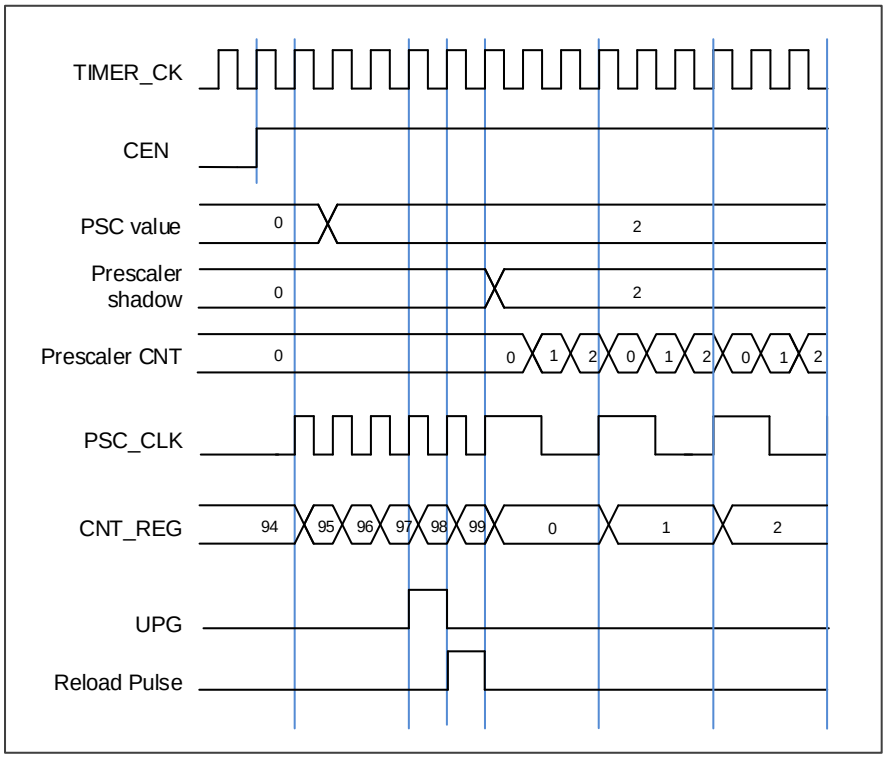
**Figure 14-71. Timing chart of internal clock divided by 1**




## Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-72. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again and an overflow event will be generated. In addition, the update events will be generated after (TIMERx_CREP+1) times of overflow events. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the shadow registers (repetition counter, auto reload register, prescaler register) are updated.

***Figure 14-73. Timing chart of up counting mode, PSC=0/2*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-73. Timing chart of up counting mode, PSC=0/2**

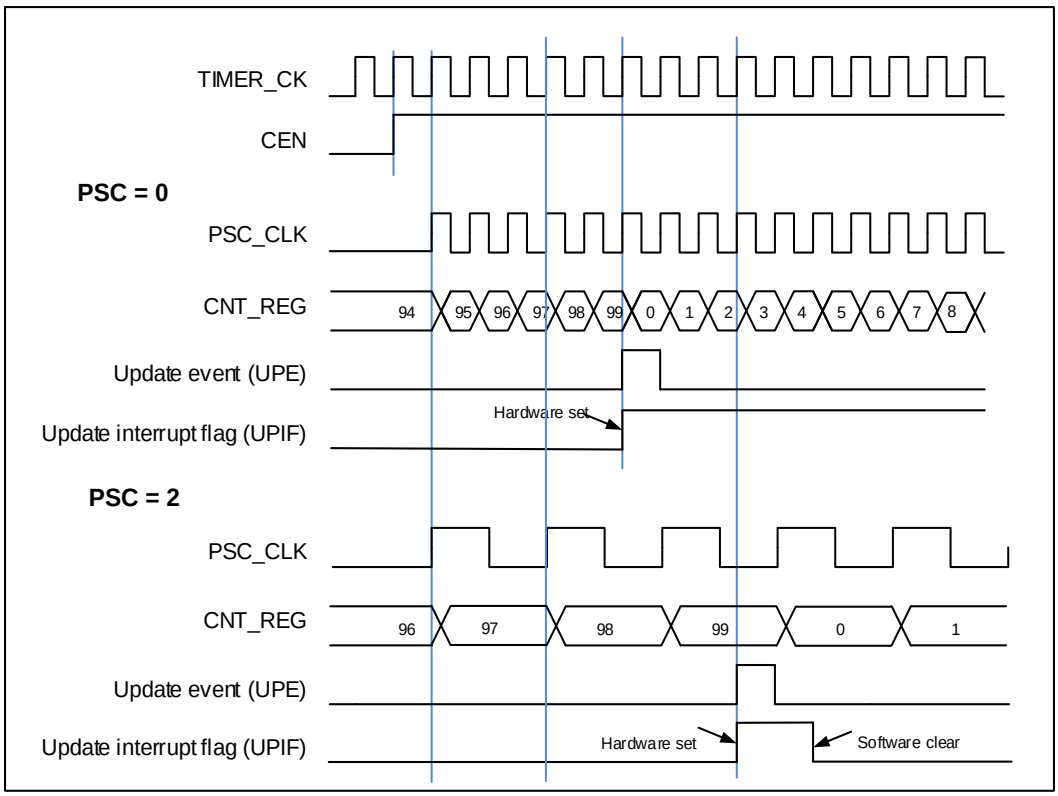
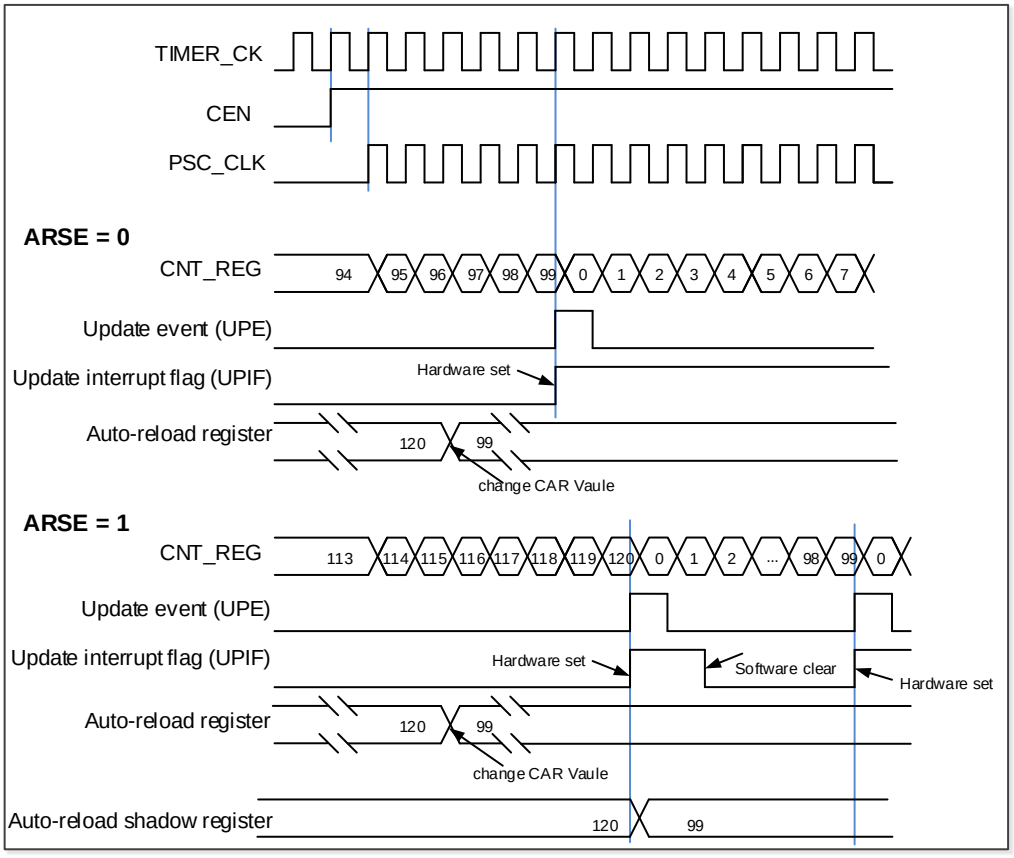



**Figure 14-74. Timing chart of up counting mode, change TIMERx_CAR on the go**

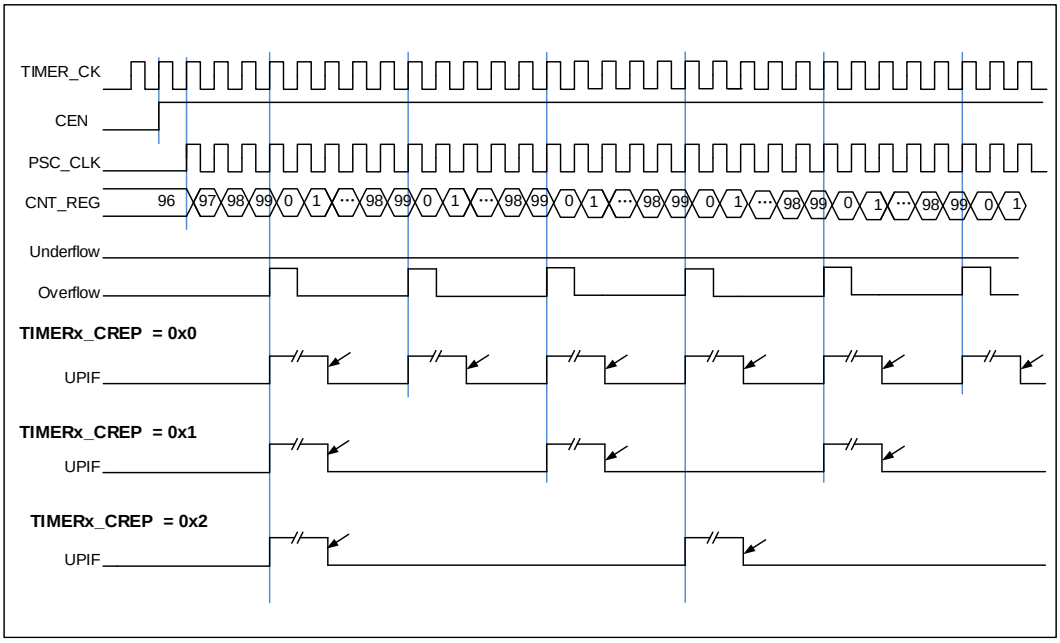


### Update event (from overflow/underflow) rate configuration

The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx_CREP register. Counter repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in TIMERx_CREP register. The repetition counter is decremented at each counter overflow in up counting mode.

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP in TIMERx_CREP register and generator an update event.

**Figure 14-75. Repetition counter timing chart of up counting mode**



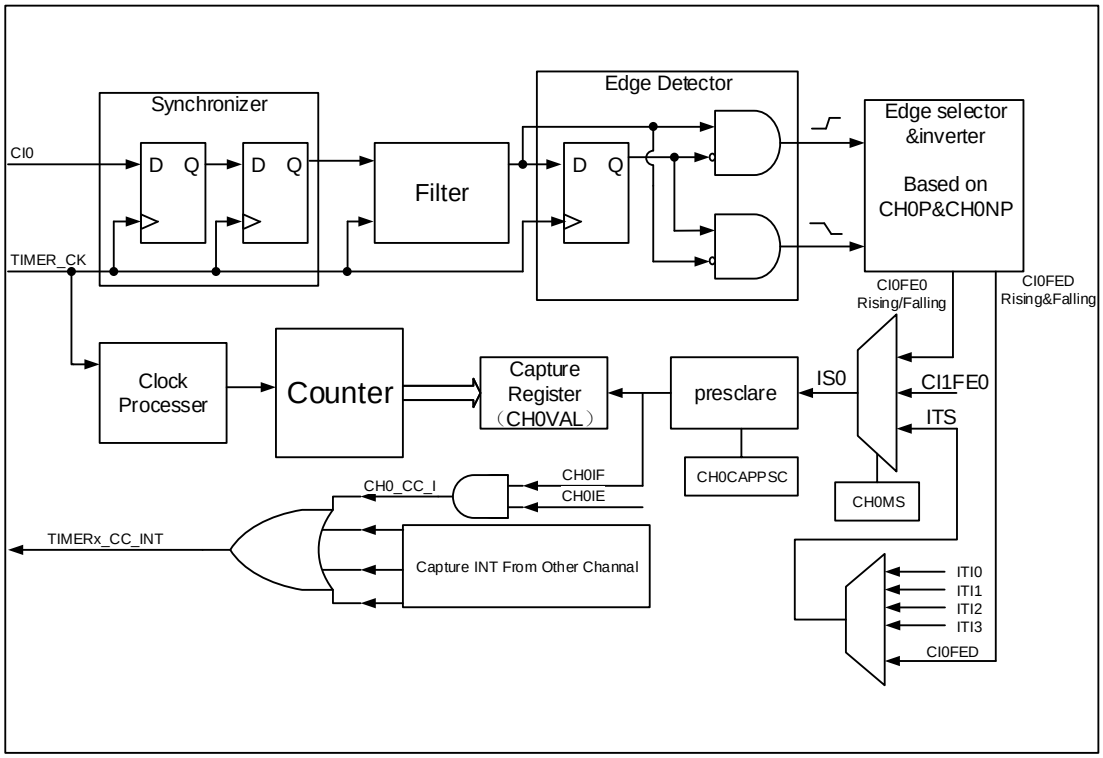

## Input capture and output compare channels

The general level4 timer has one independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Channel input capture function**

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

**Figure 14-76. Channel input capture principle**




Channels' input signals (CIx) is the TIMERx_CHx signal. First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

**Step1**: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2**: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3**: Capture source selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode ( CHxMS!=0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable. (CHxIE and CHxDEN in TIMERx_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5**: Capture enables. (CHxEN in TIMERx_CHCTL2)

**Result**: when you wanted input signal is got, TIMERx_CHxCV will be set by counter's

value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN

**Direct generation**: if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

The channel input capture function can be also used for pulse period measurement from signals on the TIMERx_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period.

- **Channel output compare function**

**Figure 14-77. Channel output compare principle (with complementary output, x=0)**

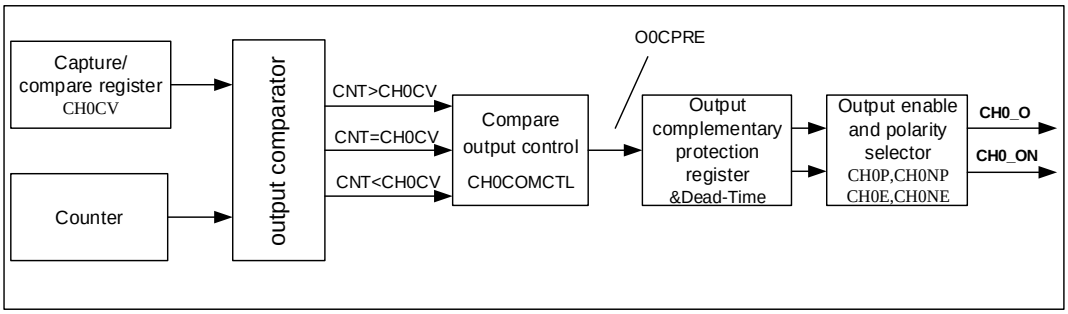



*Figure 14-77. Channel output compare principle (with complementary output, x=0)*show the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O/CHx_ON and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O/CH0_ON depends on OxCPRE signal, CHxP/CHxNP bit and CH0E/CH0NE bit (please refer to the TIMERx_CHCTL2 register for more details). For examples, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxE=1 (the output of CHx_O is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

Configure CHxNP=0 (the active level of CHx_ON is low, contrary to OxCPRE), CHxNE=1 (the output of CHx_ON is enabled):

If the output of OxCPRE is active(high) level, the output of CHx_O is active(low) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(high) level.

When CH0_O and CH0_ON are output at the same time, the specific outputs of CH0_O and CH0_ON are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared,

or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CHxDEN =1.

So the process can be divided to several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

* Set the shadow enable mode by CHxCOMSEN
* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
* Select the active high polarity by CHxP/CHxNP
* Enable the output by CHxEN

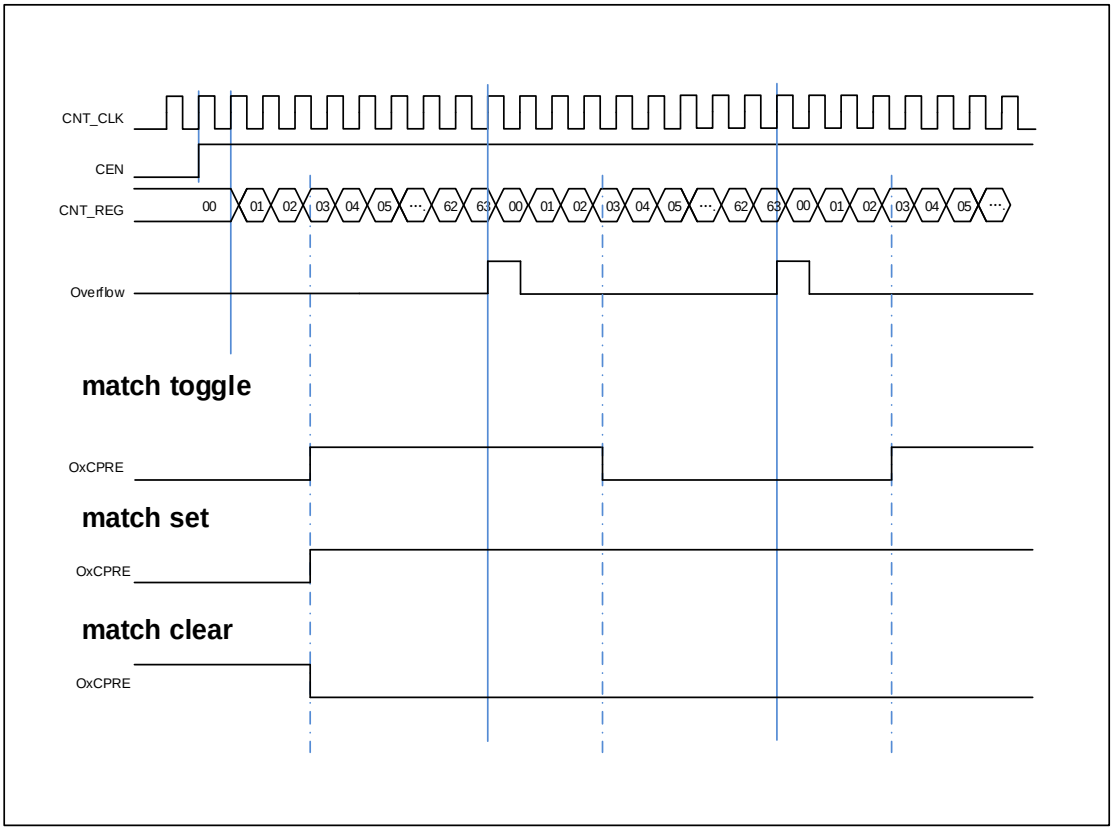
**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CHxDEN

**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV
About the CHxVAL; you can change it ongoing to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 14-78. Output-compare under three modes**



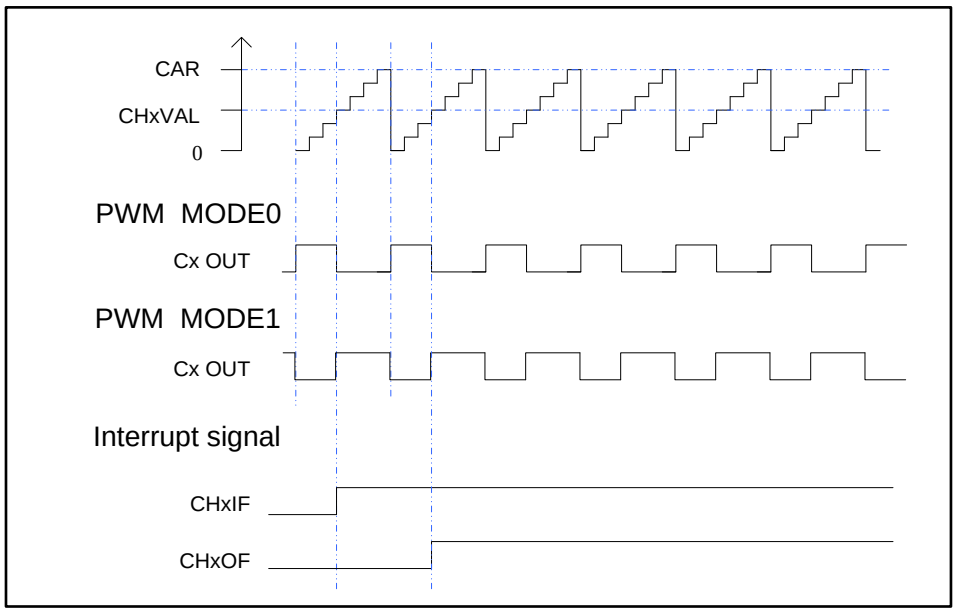
### Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

The period is determined by TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV. ***Figure 14-79. PWM mode timechart*** shows the PWM output mode and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 14-79. PWM mode timechart**




### Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. With regard to a more detail description refer to the

relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

### Channel output complementary PWM

Function of complementary is for a pair of channels, CHx_O and CHx_ON, the two output signals cannot be active at the same time. The TIMERx has 4 channels, but only the first three channels have this function. The complementary signals CHx_O and CHx_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register.

**Table 14-7. Complementary outputs controlled by parameters**

<table>
<tr><th colspan="5">Complementary Parameters</th><th colspan="2">Output Status</th></tr>
<tr><th>POEN</th><th>ROS</th><th>IOS</th><th>CHxEN</th><th>CHxNEN</th><th>CHx_O</th><th>CHx_ON</th></tr>
<tr><td rowspan="5">0</td><td rowspan="5">0/1</td><td rowspan="3">0</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1).</td></tr>
<tr><td>1</td><td colspan="2" rowspan="3">CHx_O/ CHx_ON output “off-state” (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3)</td></tr>
<tr><td rowspan="2">1</td><td>0</td></tr>
<tr><td>1</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td colspan="2">CHx_O/ CHx_ON output “off-state”:<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN.</td></tr>
<tr><td rowspan="5">1</td><td rowspan="3">0</td><td rowspan="5">0/1</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O/CHx_ON = LOW<br>CHx_O/CHx_ON output disable.</td></tr>
<tr><td>1</td><td>CHx_O = LOW<br>CHx_O output disable.</td><td>CHx_ON =OxCPRE⊕(4)CHxNP<br>CHx_ON output enable.</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable.</td><td>CHx_ON = LOW<br>CHx_ON output disable.</td></tr>
<tr><td>1</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable.</td><td>CHx_ON =(!OxCPRE)(5)⊕ CHxNP.<br>CHx_ON output enable.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>CHx_O = CHxP<br>CHx_O output “off-state”.</td><td>CHx_ON = CHxNP<br>CHx_ON output “off-state”.</td></tr>
</table>

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | 1 | CHx_O = CHxP<br>CHx_O output “off-state” | CHx_ON =OxCPRE⊕CHxNP<br>CHx_ON output enable |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON = CHxNP<br>CHx_ON output “off-state”. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | CHx_ON =(!OxCPRE)⊕<br>CHxNP<br>CHx_ON output enable. |

**Note:**

(1) Output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “Off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (!OxCPRE)：the complementary output of the OxCPRE signal.

### Insertion dead time for complementary PWM

The dead time insertion is enabled when both CHxEN and CHxNEN are 1’b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for channel 1. The detail about the delay time, refer to the register TIMERx_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

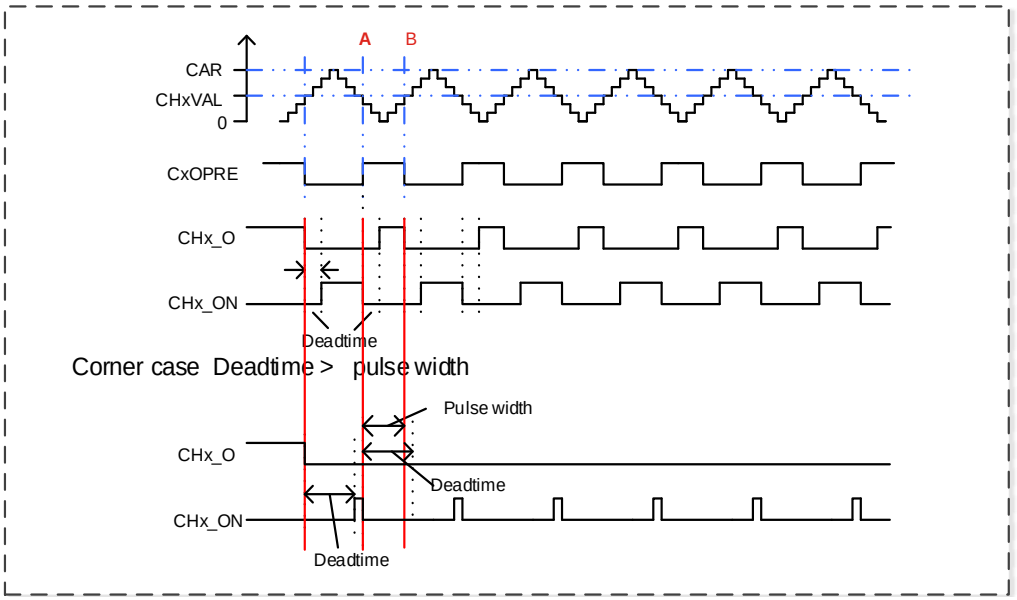
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the ***Figure 14-80. Complementary output with dead-time insertion.*** CHx_O signal remains at the low value until the end of the deadtime delay, while CHx_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx_O signal will be cleared at once, while CHx_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx_O duty cycle, then the CHx_O signal is always the inactive value. (as show in the ***Figure 14-80. Complementary output with dead-time insertion.***).

The dead time delay is greater than or equal to the CHx_ON duty cycle, then the CHx_ON signal is always the inactive value.

**Figure 14-80. Complementary output with dead-time insertion.**



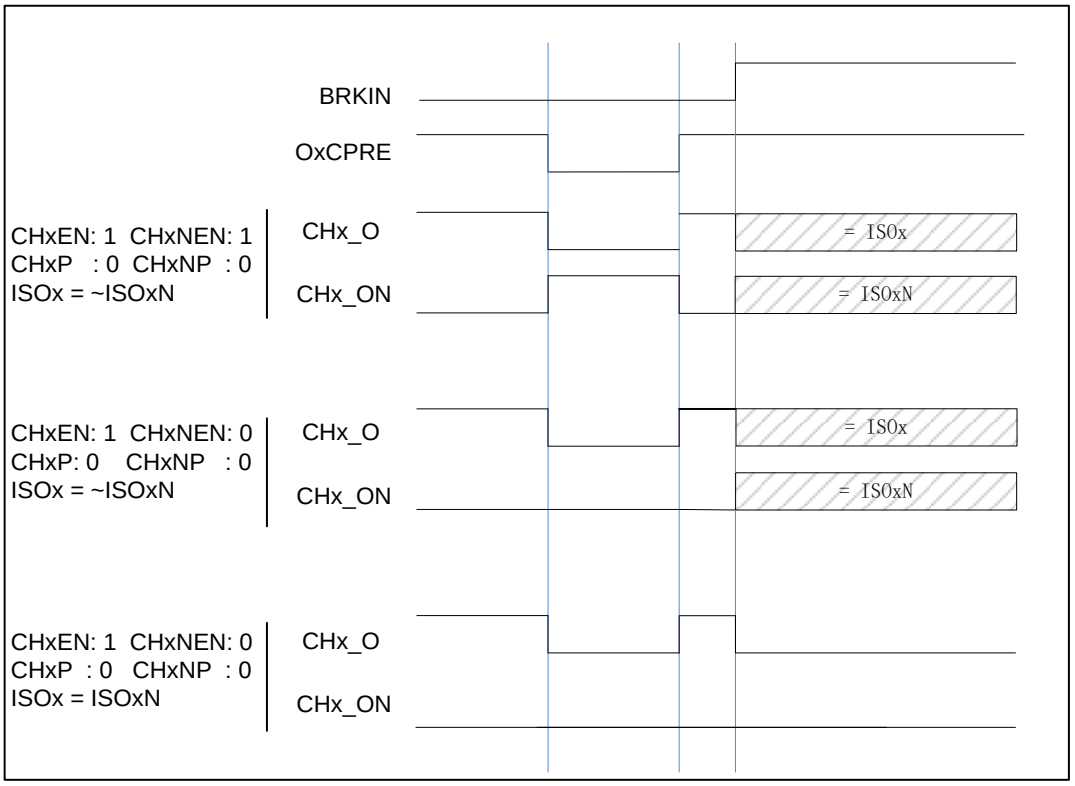

## Break mode

In this mode, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx_O and CHx_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 14-81. Output behavior in response to a break(The break high active)**




### Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

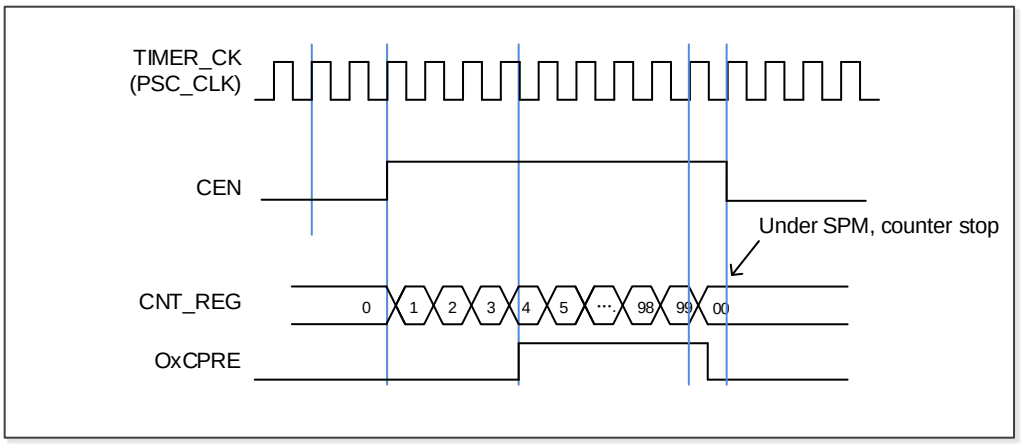
Once the timer is set to operate in the single pulse mode, it is necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

*Figure 14-82. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60* shows an

example.

**Figure 14-82. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60**




### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG . If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx_DMATC is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### Timer debug mode

When the Cortex®-M23 halted, and the TIMERx_HOLD configuration bit in DBG_CTL1 register set to 1, the TIMERx counter stops.

### 14.5.5. TIMERx registers(x=15, 16)

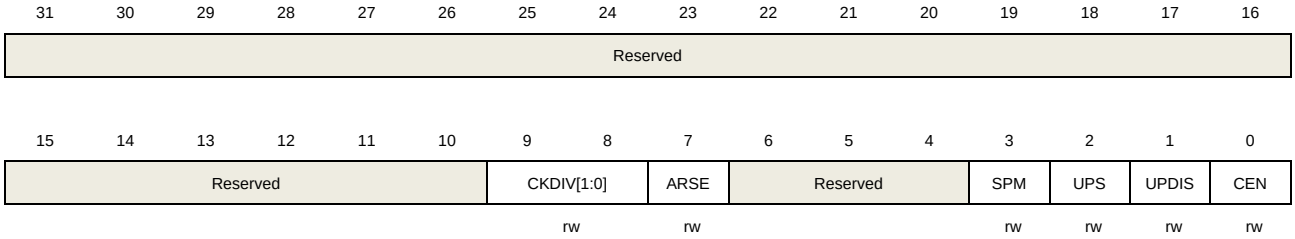
TIMER15 base address: 0x4001 4400

TIMER16 base address: 0x4001 4800

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS).<br>00: $f_{DTS}=f_{CK_TIMER}$<br>01: $f_{DTS}= f_{CK_TIMER}$ /2<br>10: $f_{DTS}= f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value. |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode disable. The counter continues after update event.<br>1: Single pulse mode enable. The counter counts until the next update event occurs. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event |

The restart mode generates an update event.

1: This event generates update interrupts or DMA requests:

The counter generates an overflow or underflow event

| | | |
|---|---|---|
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: Update event disable.<br>**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decoder mode. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9 | ISO0N | Idle state of channel 0 complementary output<br>0: When POEN bit is reset, CH0_ON is set low.<br>1: When POEN bit is reset, CH0_ON is set high<br>This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output |

| | | |
|---|---|---|
| | | 0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high<br>The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00. |
| 7:4 | Reserved | Must be kept at reset value |
| 3 | DMAS | DMA request source selection<br>0: When capture or compare event occurs, the DMA request of channel x is sent<br>1: When update event occurs, the DMA request of channel x is sent. |
| 2 | CCUC | Commutation control shadow register update control<br>When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:<br>0: The shadow registers update by when CMTG bit is set.<br>1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.<br>When a channel does not have a complementary output, this bit has no effect. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled.<br>1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled.<br>After these bits have been written, they are updated based when commutation event coming.<br>When a channel does not have a complementary output, this bit has no effect. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CH0DEN | UPDEN | BRKIE | Reserved | CMTIE | Reserved | | | CH0IE | UPIE |
| | | | | | | rw | rw | rw | | rw | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: disabled<br>1: enabled |
| 8 | UPDEN | Update DMA request enable<br>0: disabled<br>1: enabled |
| 7 | BRKIE | Break interrupt enable<br>0: disabled<br>1: enabled |
| 6 | Reserved | Must be kept at reset value |
| 5 | CMTIE | Commutation interrupt enable<br>0: disabled<br>1: enabled |
| 4:2 | Reserved | Must be kept at reset value |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: disabled<br>1: enabled |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | CH0OF | Reserved | BRKIF | Reserved | CMTIF | Reserved. | | | CH0IF | UPIF |
| | | | | | | rc_w0 | | rc_w0 | | rc_w0 | | | | rc_w0 | rc_w0 |

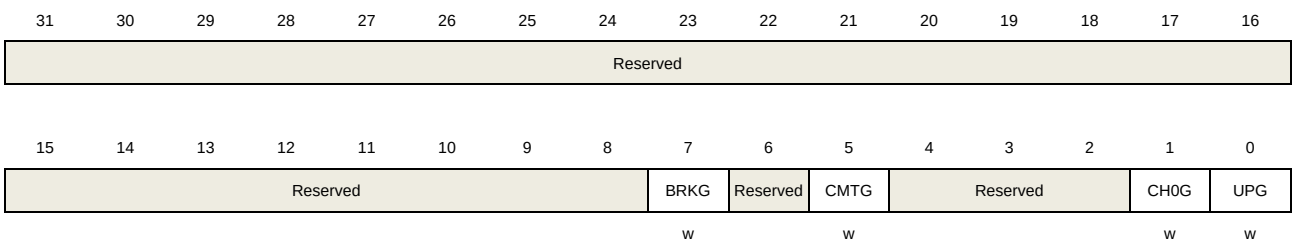
| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by |

| | | |
|---|---|---|
| | | software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |
| 7 | BRKIF | Break interrupt flag<br>When the break input is inactive, the bit is set by hardware.<br>When the break input is inactive, the bit can be cleared by software.<br>0: No active level break has been detected.<br>1: An active level has been detected. |
| 6 | Reserved | Must be kept at reset value |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when channel's commutation event occurs, and cleared by software<br>0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4:2 | Reserved | Must be kept at reset value |
| 1 | CH0IF | Channel 0 's capture/compare interrupt flag<br>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No Channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | BRKG | Break event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a break event<br>1: Generate a break event |
| 6 | Reserved | Must be kept at reset value |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel's c/c control update event |
| 4:2 | Reserved | Must be kept at reset value |
| 1 | CH0G | Channel 0's capture or compare event generation<br>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 1 capture or compare event<br>1: Generate a channel 1 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared in up counting mode is selected. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | Reserved | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| | | | | | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| | | | | | | | | | rw | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits.<br>000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced to low level.<br>101: Force high. O0CPRE is forced to high level.<br>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.<br>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.<br>0: Channel 0 output compare shadow disable<br>1: Channel 0 output compare shadow enable<br>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1) |

This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.

2 CH0COMFEN Channel 0 output compare fast enable

When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.

0: Channel 0 output quickly compare disable.

1: Channel 0 output quickly compare enable.

1:0 CH0MS[1:0] Channel 0 I/O mode selection

This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).

00: Channel 0 is programmed as output

01: Channel 0 is programmed as input, IS0 is connected to CI0FE0

10: Reserved.

11: Reserved.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |

7:4 CH0CAPFLT[3:0] Channel 0 input capture filter control

The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.

Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.

The filtering capability configuration is as follows:

| CH0CAPFLT [3:0] | Times | $f_{SAMP}$ |
|---|---|---|
| 4'b0000 | Filter disabled. | |
| 4'b0001 | 2 | $f_{CK_TIMER}$ |
| 4'b0010 | 4 | |
| 4'b0011 | 8 | |
| 4'b0100 | 6 | $f_{DTS}/2$ |
| 4'b0101 | 8 | |
| 4'b0110 | 6 | $f_{DTS}/4$ |
| 4'b0111 | 8 | |
| 4'b1000 | 6 | $f_{DTS}/8$ |
| 4'b1001 | 8 | |
| 4'b1010 | 5 | $f_{DTS}/16$ |

| | | |
|---|---|---|
| 4'b1011 | 6 | |
| 4'b1100 | 8 | |
| 4'b1101 | 5 | $f_{DTS}$/32 |
| 4'b1110 | 6 | |
| 4'b1111 | 8 | |

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

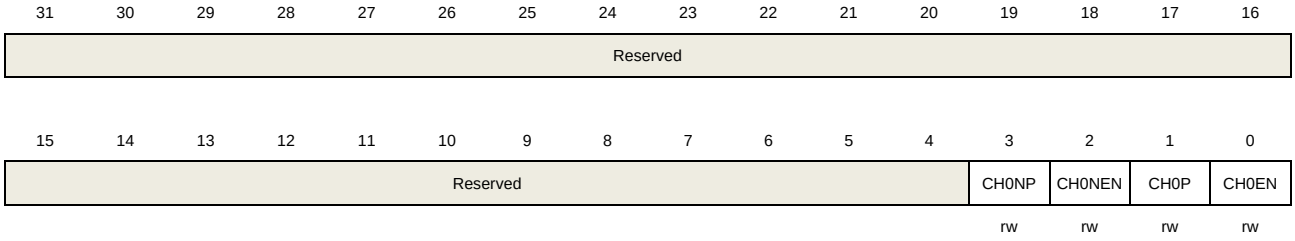
Same as Output compare mode

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.<br>0: Channel 0 complementary output high level is active level<br>1: Channel 0 complementary output low level is active level<br>When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 2 | CH0NEN | Channel 0 complementary output enable<br>When channel 0 is configured in output mode, setting this bit enables the |

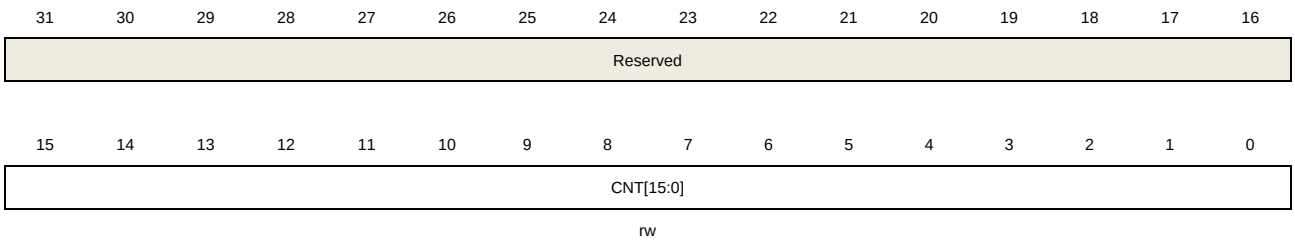
| | | |
|---|---|---|
| | | complementary output in channel0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 capture/compare function polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 high level is active level<br>1: Channel 0 low level is active level<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: Reserved.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



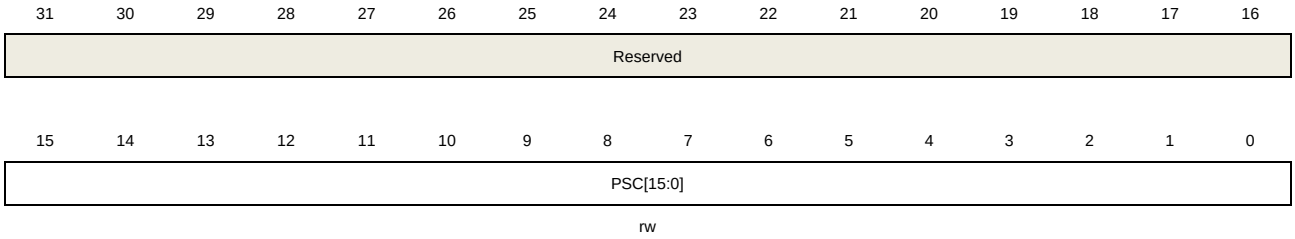

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

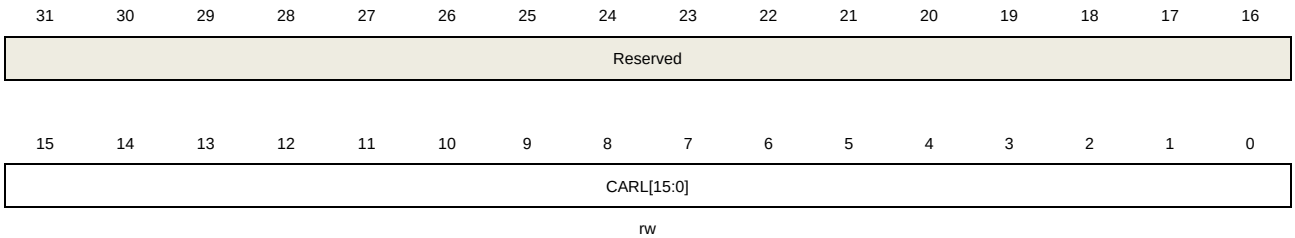



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



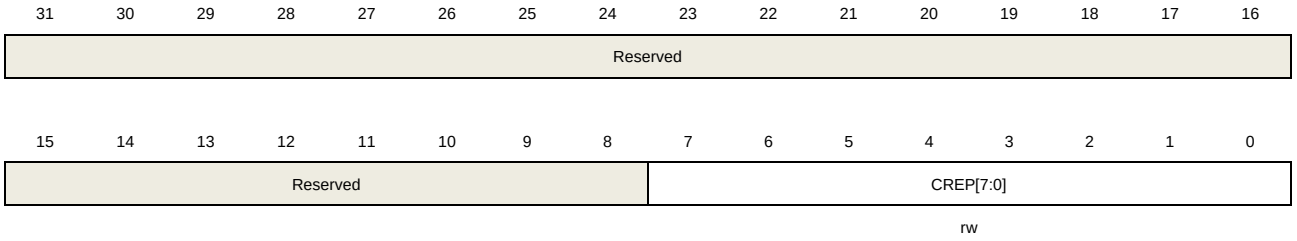

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter.<br>**Note:** When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected |

value

### Counter repetition register (TIMERx_CREP)

Address offset: 0x30
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

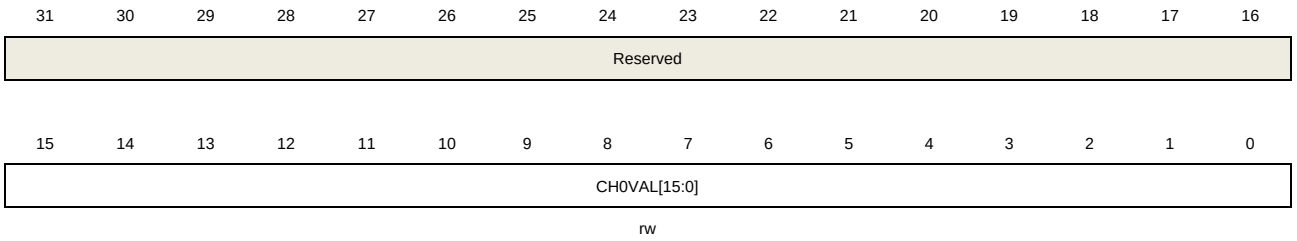



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP[7:0] | Counter repetition value<br>This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel0<br>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.<br>When channel 0 is configured in output mode, this bit-filed contains value to be |

compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POEN | OAEN | BRKP | BRKEN | ROS | IOS | PROT[1:0] | | DTCFG[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | POEN | Primary output enable<br>The bit can be set to 1 by:<br>- Write 1 to this bit<br>- If OAEN is set to 1, this bit is set to 1 at the next update event.<br>The bit can be cleared to 0 by:<br>- Write 0 to this bit<br>- Valid fault input (asynchronous).<br>When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set.<br>0: Disable channel outputs (CHxO or CHxON).<br>1: Enabled channel outputs (CHxO or CHxON).<br>**Note:** This bit is only valid when CHxMS=2'b00. |
| 14 | OAEN | Output automatic enable<br>0: The POEN bit can only be set by software.<br>1: POEN can be set at the next update event, if the break input is not active.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 13 | BRKP | Break polarity<br>This bit specifies the polarity of the BRKIN input signal.<br>0: BRKIN input active low<br>1: BRKIN input active high |
| 12 | BRKEN | Break enable |

This bit can be set to enable the BRKIN and CCS clock failure event inputs.
0: Break inputs disabled
1: Break inputs enabled
This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

11 ROS

Run mode “off-state” enable
When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 14-7. Complementary outputs controlled by parameters***.
0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.
1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.
This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.

10 IOS

Idle mode “off-state” enable
When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 14-7. Complementary outputs controlled by parameters***.
0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.
1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.
This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.

9:8 PROT[1:0]

Complementary register protect control
This bit-filed specifies the write protection property of registers.
00: protect disable. No write protection.
01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.
10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected.
11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN bits in TIMERx_CHCTL0 registers (if the related channel is configured in output) are writing protected.
This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.

7:0 DTCFG[7:0]

Dead time configure
The relationship between DTVAL value and the duration of dead-time is as follow:

| DTCFG[7:5] | The duration of dead-time |
|---|---|

| | |
|---|---|
| 3'b0xx | DTCFG[7:0] * $t_{DTS_CK}$ |
| 3'b10x | (64+ DTCFG[5:0]) * $t_{DTS_CK}$ *2 |
| 3'b110 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *8 |
| 3'b111 | (32+ DTCFG[4:0]) * $t_{DTS_CK}$ *16 |

**Note:**

1. $t_{DTS_CK}$ is the period of DTS_CK which is configured by CKDIV[1:0] in TIMERx_CTL0.
2. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | DMATC[4:0] | | | | | Reserved | | | DMATA [4:0] | | | | |
| | | | rw | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 12:8 | DMATC [4:0] | DMA transfer count<br>This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | DMATA [4:0] | DMA transfer access start address<br>This filed define the first address for the DMA access the TIMERx_DMATB.<br>When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4. |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | DMATB[15:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.<br>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | CHVSEL | OUTSEL |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CHVSEL | Write CHxVAL register selection<br>This bit-field set and reset by software.<br>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored<br>0: No effect |
| 0 | OUTSEL | The output value selection<br>This bit-field set and reset by software<br>1: If POEN and IOS is 0, the output disabled<br>0: No effect |

# 14.6. Basic timer (TIMERx, x=5)

## 14.6.1. Overview

The basic timer module (TIMER5) reference is a 16-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate DMA request.

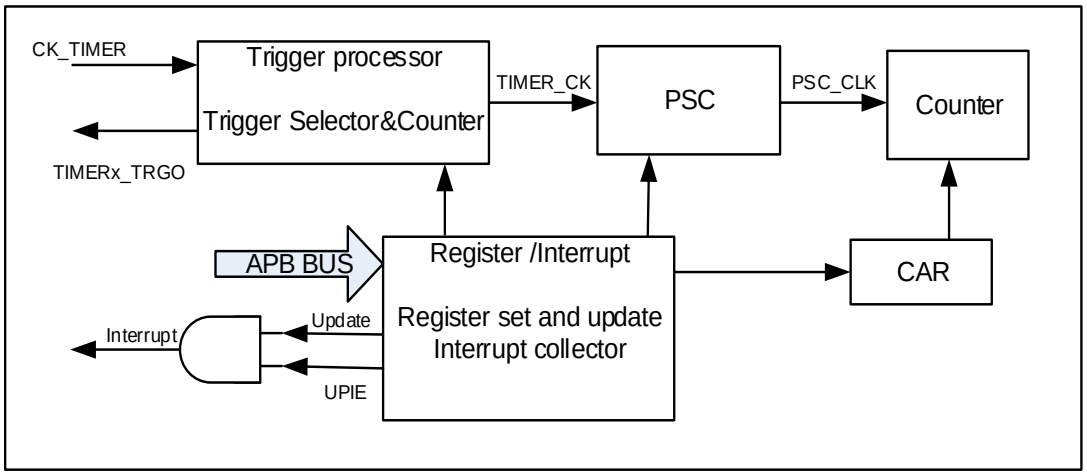
## 14.6.2. Characteristics

- Counter width: 16 bits.
- Source of count clock is internal clock only.
- Counter modes: only count up.
- Programmable prescaler: 16 bits. Factor can be changed ongoing.
- Auto-reload function.
- Interrupt output or DMA request on update event.

## 14.6.3. Block diagram

*Figure 14-83. Basic timer block diagram* provides details on the internal configuration of the basic timer.

**Figure 14-83. Basic timer block diagram**




## 14.6.4. Function overview

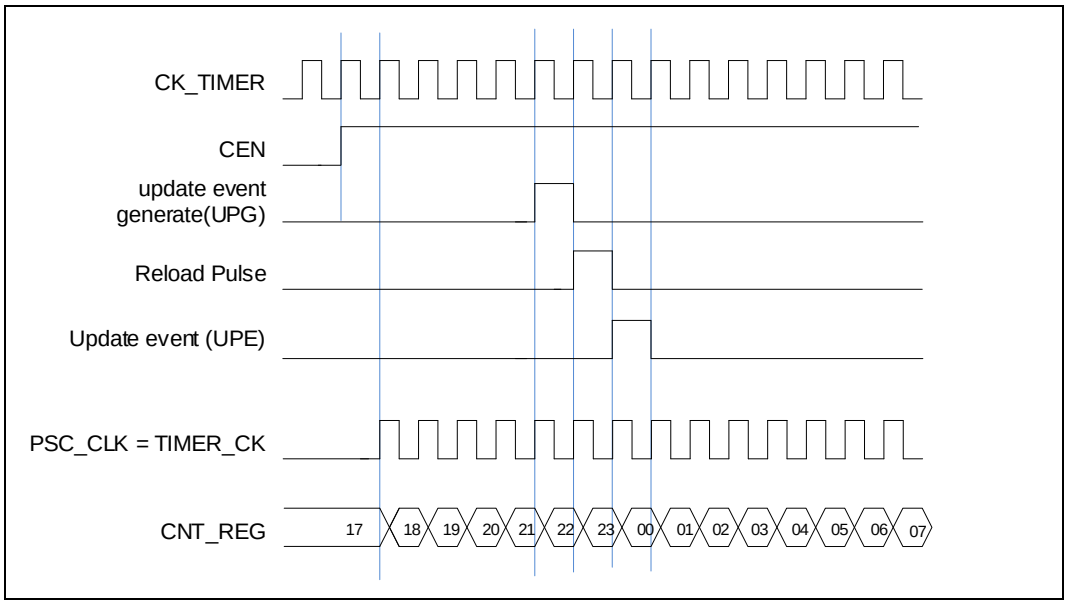
### Clock source configuration

The basic TIMER can only be clocked by the internal timer clock CK_TIMER, which is from the source named CK_TIMER in RCU

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER used to drive the

counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

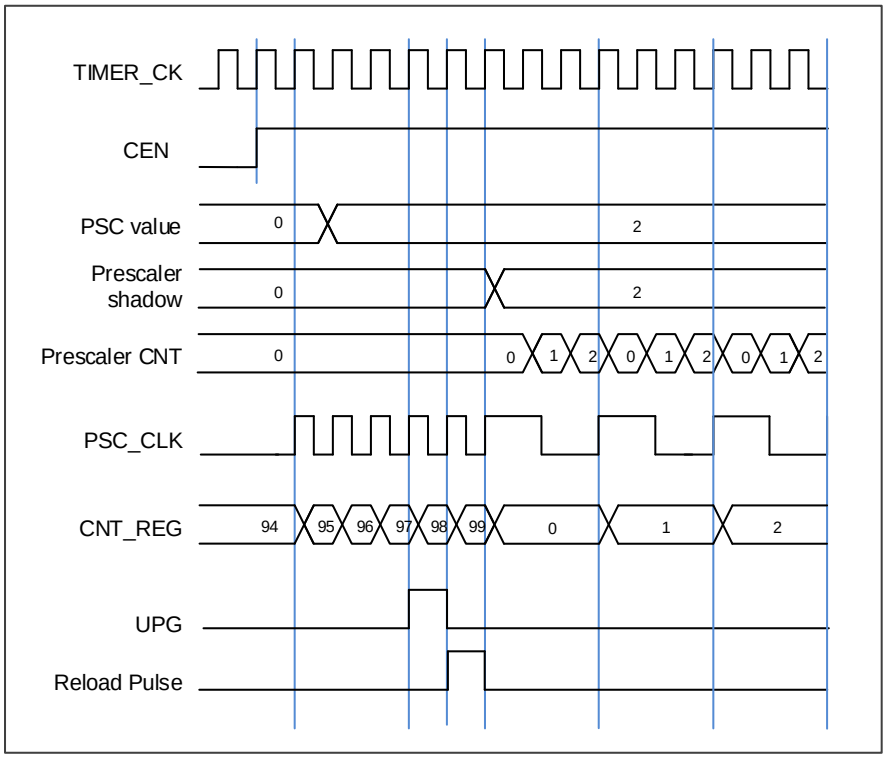
**Figure 14-84. Timing chart of internal clock divided by 1**




## Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 14-85. Timing chart of PSC value change from 0 to 2**




### Counter up counting

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 14-86. Timing chart of up counting mode, PSC=0/2**

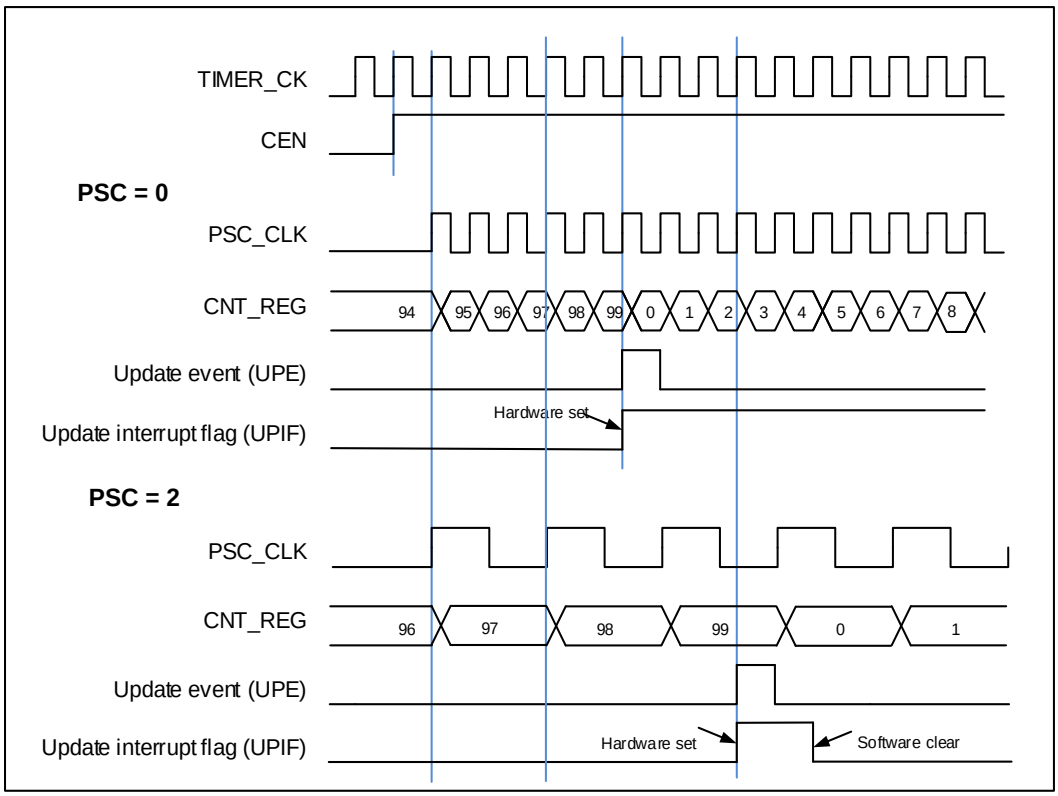
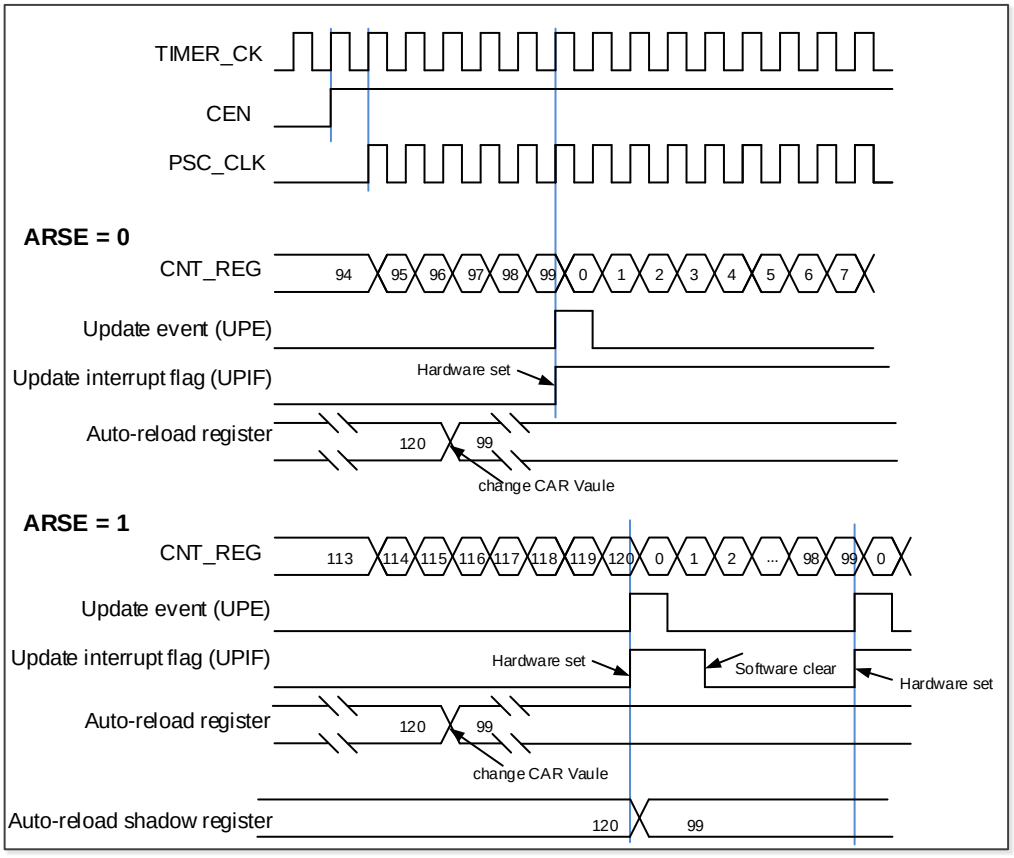



**Figure 14-87. Timing chart of up counting mode, change TIMERx_CAR on the go**



### Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event.

Once the timer is set to operate in the single pulse mode, it is necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter, then the CEN bit keeps at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held.

### Timer debug mode

When the Cortex®-M23 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

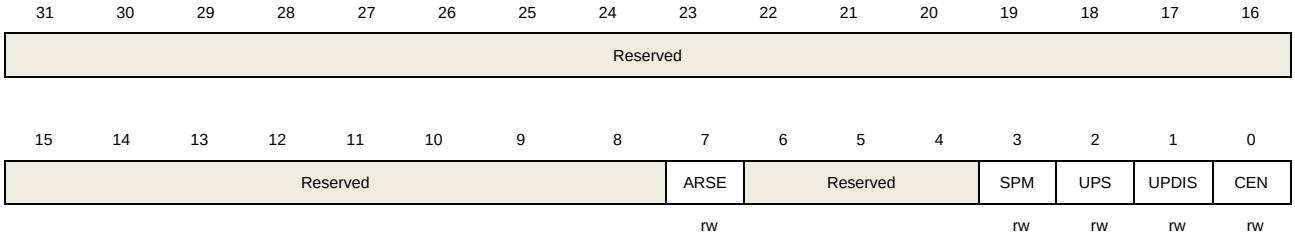
### 14.6.5. TIMERx registers(x=5)

TIMER5 base address: 0x4000 1000

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode disable. The counter continues after update event.<br>1: Single pulse mode enable. The counter counts until the next update event occurs. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: These events generate update interrupts or DMA requests:<br>The UPG bit is set<br>The counter generates an overflow or underflow event<br>The restart mode generates an update event.<br>1: This event generates update interrupts or DMA requests:<br>The counter generates an overflow or underflow event |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:<br>The UPG bit is set |

The counter generates an overflow or underflow event

The restart mode generates an update event.

1: Update event disable.

**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.

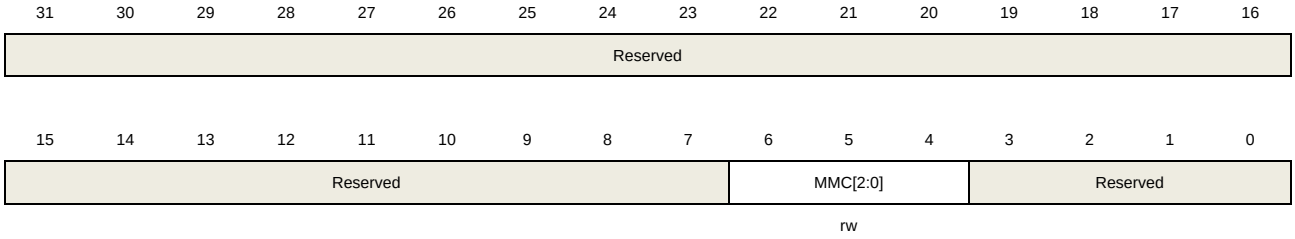
| | | |
|---|---|---|
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value |
| 6:4 | MMC[2:0] | Master mode control<br>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.<br>000: When a counter reset event occurs, a TGRO trigger signal is output. The counter resert source:<br>Master timer generate a reset<br>The UPG bit in the TIMERx_SWEVG register is set<br>001: Enable. When a conter start event occurs, a TGRO trigger signal is output. The counter start source :<br>CEN control bit is set<br>The trigger input in pause mode is high<br>010: When an update event occurs, a TGRO trigger signal is output. The update source depends on UPDIS bit and UPS bit. |
| 3:0 | Reserved | Must be kept at reset value. |

### Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | UPDEN | Reserved | | | | | | | UPIE |
| | | | | | | | rw | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | UPDEN | Update DMA request enable<br>0: disabled<br>1: enabled |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | UPIE | Update interrupt enable<br>0: disabled<br>1: enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | | UPIF |
| | | | | | | | | | | | | | | | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware on an update event and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 | | | | | | | | | | | | | | | UPG |
| | | | | | | | | | | | | | | | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | UPG | This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter. |

# 15. Infrared ray port (IFRP)

## 15.1. Overview

Infrared ray port (IFRP) is used to control infrared light LED, and send out infrared data to implement infrared ray remote control.

There is no register in this module, which is controlled by TIMER15 and TIMER16. You can improve the module's output to high current capacity by set the GPIO pin to Fast Mode.

## 15.2. Characteristics

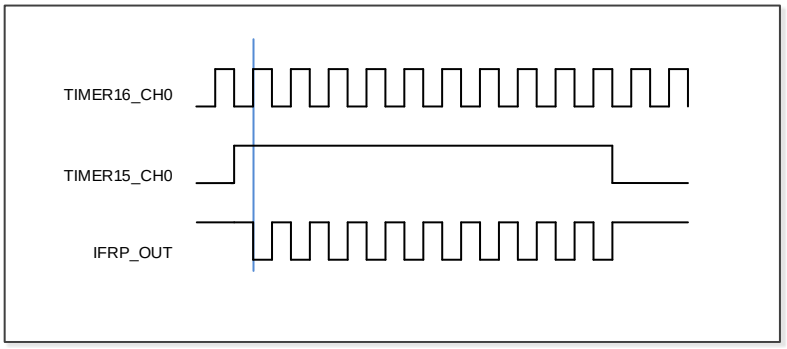
- The IFRP output signal is decided by TIMER15_CH0 and TIMER16_CH0
- To get correct infrared ray signal, TIMER15 should generate low frequence modulation envelope signal, and TIMER16 should generate high frequence carrier signal
- The IFRP output (PB9) can provide high current to control LED interface by setting PB9_HCCE in SYSCFG_CFG0

## 15.3. Function overview

IFRP is a module which is able to integrate the output of TIMER15 and TIMER16 to generate an infrared ray signal.

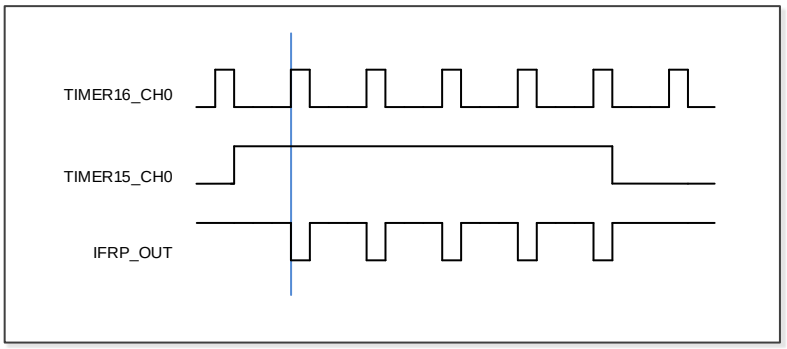
1. The TIMER15's CH0 is programed to generate the low frequence PWM signal which is the modulation evalope signal. The TIMER16's CH0 is programed to generate the high frquence PWM signal which is the carrier signal. And the channel need to be enabled before generating these signals.
2. Program the GPIO remap regisger and enable the pin.
3. If you want to get the high current capacity of output, remapping IFRP_OUT to PB9 and setting the PB9 to Fast Mode by the register in SYS_CFG module are required.

**Figure 15-1. IFRP output timechart 1**



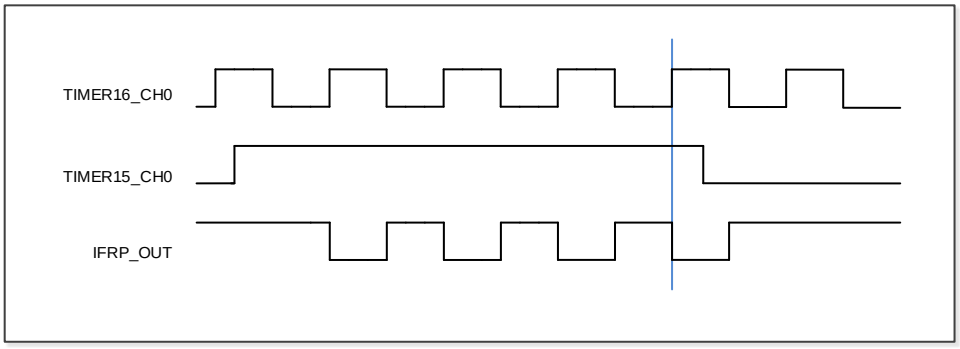

**Note:** IFRP_OUT has one APB clock delay from TIMER16_CH0.

**Figure 15-2. IFRP output timechart 2**




**Note:** Carrier (TIMER15_CH0)'s duty cycle can be changed, and IFRP_OUT has inverted relationship with TIMER16_CH0 when TIMER15_CH0 is high.

**Figure 15-3. IFRP output timechart 3**




**Note:** IFRP_OUT will keep the integrity of TIMER16_CH0, even if evelope signal (TIMER15_CH0) is no active.

# 16. Universal synchronous/asynchronous receiver /transmitter (USART)

## 16.1. Overview

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. Data frames can be transferred in full duplex or half duplex mode, synchronously or asynchronously through this interface. A programmable baud rate generator divides the the UCLK (PCLK1, PCLK2 and CK_USART0 available only for USART0) to produces a dedicated wide range baudrate clock for the USART transmitter and receiver.

Besides the standard asynchronous receiver and transmitter mode, the USART implements several other types of serial data exchange modes, such as IrDA (infrared data association) SIR mode, smartcard mode, LIN (local interconnection network) mode, half-duplex mode and synchronous mode. It also supports multiprocessor communication mode, and hardware flow control protocol (CTS/RTS). The data frame can be transferred from LSB or MSB bit. The polarity of the TX/RX pins can be configured independently and flexibly.

All USARTs support DMA function for high-speed data communication.

## 16.2. Characteristics

- NRZ standard format
- Asynchronous, full duplex communication
- Half duplex single wire communication
- Receive FIFO function
- Dual clock domain:
  - Asynchronous pclk and USART clock
  - Baud rate programming independent from the PCLK reprogramming
- Programmable baud-rate generator allowing speed up to 9 MBits/s when the clock frequency is 72 MHz and oversampling is by 8.
- Fully programmable serial interface characteristics:
  - A data word (8 or 9 bits) LSB or MSB first
  - Even, odd or no-parity bit generation/detection
  - 0.5, 1, 1.5 or 2 stop bit generation
- Swappable Tx/Rx pin
- Configurable data polarity
- Hardware modem operations (CTS/RTS) and RS485 drive enable
- Configurable multibuffer communication using centralized DMA
- Separate enable bits for transmitter and receiver

- Parity control
  - Transmits parity bit
  - Checks parity of received data byte
- LIN break generation and detection
- Support IrDA
- Synchronous mode and transmitter clock output for synchronous transmission
- ISO 7816-3 compliant smartcard interface
  - Character mode (T=0)
  - Block mode (T=1)
  - Direct and inverse convention
- Multiprocessor communication
  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle line or address mask detection
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition
- Wake up from Deep-sleep mode
  - By standard RBNE interrupt
  - By WUF interrupt
- Various status flags
  - Flags for transfer detection: receive buffer not empty (RBNE), receive FIFO full (RFF), transmit buffer empty (TBE), transfer complete (TC)
  - Flags for error detection: overrun error (ORERR), noise error (NERR), frame error (FERR) and parity error (PERR)
  - Flag for hardware flow control: CTS changes (CTSF)
  - Flag for LIN mode: LIN break detected (LBDF)
  - Flag for multiprocessor communication: IDLE frame detected (IDLEF)
  - Flag for ModBus communication: Address/character match (AMF) and receiver timeout (RTF)
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF)
  - Wakeup from Deep-sleep mode flag
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set

While USART0 is fully implemented, USART1 is only partially implemented with the following features not supported.

- Smartcard mode
- IrDA SIR ENDEC block
- LIN mode
- Dual clock domain and wakeup from Deep-sleep mode
- Receiver timeout interrupt
- ModBus communication

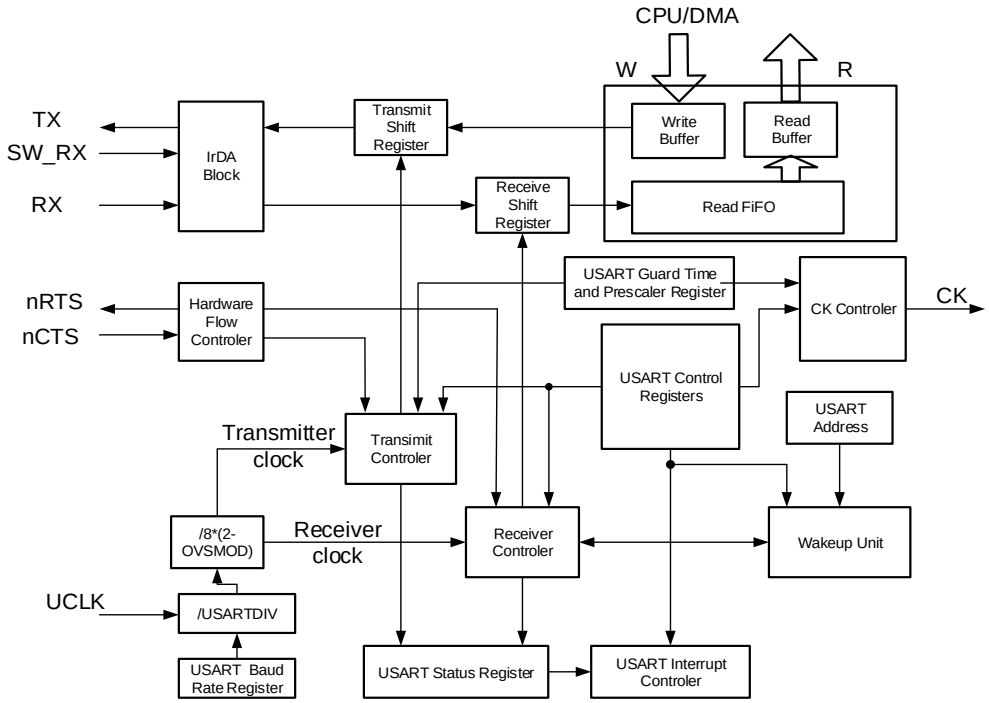
## 16.3. Function overview

The interface is externally connected to another device by the main pins listed in ***Table 16-1. Description of USART important pins***.

**Table 16-1. Description of USART important pins**

| Pin | Type | Description |
|---|---|---|
| RX | Input | Receive data |
| TX | Output I/O (single-wire/smartcard mode) | Transmit data. High level when enabled but nothing to be transmitted |
| CK | Output | Serial clock for synchronous communication |
| nCTS | Input | Clear to send in hardware flow control mode |
| nRTS | Output | Request to send in hardware flow control mode |

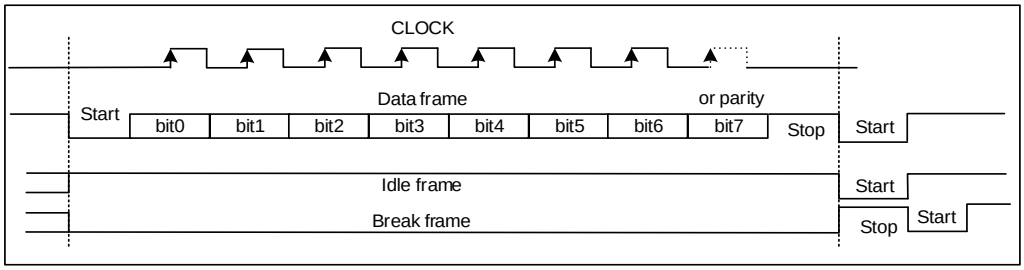
**Figure 16-1. USART module block diagram**




### 16.3.1. USART frame format

The USART frame starts with a start bit and ends up with a number of stop bits. The length of the data frame is configured by the WL bit in the USART_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit of in USART_CTL0 register. When the WL bit is reset, the parity bit is the 7th bit. When the WL bit is set, the parity bit is the 8th bit. The method of calculating the parity bit is selected by the PM bit in USART_CTL0 register.

**Figure 16-2. USART character frame (8 bits data and 1 stop bit)**




In transmission and reception, the number of stop bits can be configured by the STB[1:0] bits in the USART_CTL1 register.

**Table 16-2. Configuration of stop bits**

| STB[1:0] | stop bit length (bit) | usage description |
|---|---|---|
| 00 | 1 | Default value |
| 01 | 0.5 | Smartcard mode for receiving |
| 10 | 2 | Normal USART and single-wire modes |
| 11 | 1.5 | Smartcard mode for transmitting and receiving |

In an idle frame, all the frame bits are logic 1. The frame length is equal to the normal USART frame.

The break frame structure is a number of low bits followed by the configured number of stop bits. The transfer speed of a USART frame depends on the frequency of the UCLK, the configuration of the baud rate generator and the oversampling mode.

### 16.3.2. Baud rate generation

The baud-rate divider is a 16-bit number which consists of a 12-bit integer and a 4-bit fractional part. The number formed by these two values is used by the baud rate generator to determine the bit period. Having a fractional baud-rate divider allows the USART to generate all the standard baud rates.

The baud-rate divider (USARTDIV) has the following relationship with the UCLK:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (16\text{-}1)$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}} \quad (16\text{-}2)$$

For example, when oversampled by 16:

1. Get USARTDIV by caculating the value of USART_BUAD:

   If USART_BUAD=0x21D, then INTDIV=33 (0x21), FRADIV=13 (0xD).

USARTDIV=33+13/16=33.81.

2. Get the value of USART_BUAD by calculating the value of USARTDIV:

   If USARTDIV=30.37, then INTDIV=30 (0x1E).

   16*0.37=5.92, the nearest integer is 6, so FRADIV=6 (0x6).

   USART_BUAD=0x1E6.

   **Note:** If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

### 16.3.3. USART transmitter

If the transmit enable bit (TEN) in USART_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shifts out the transmit data frame through the TX pin. The polarity of the TX pin can be configured by the TINV bit in the USART_CTL1 register. Clock pulses can output through the CK pin.

After the TEN bit is set, an idle frame will be sent. The TEN bit should not be cleared while the transmission is ongoing.

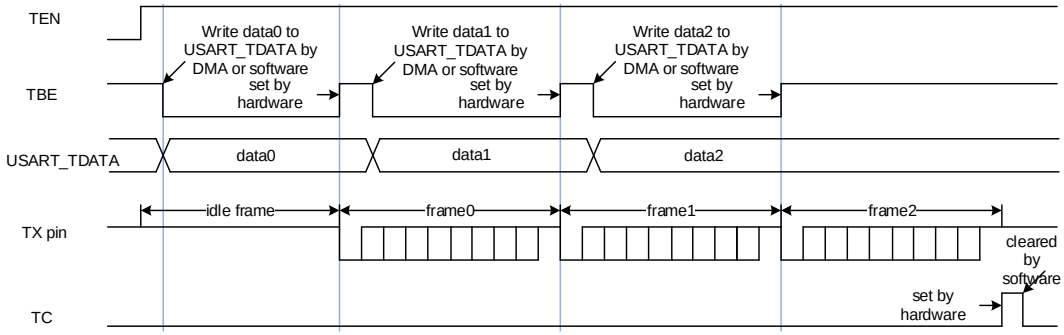
After power on, the TBE bit is high by default. Data can be written to the USART_TDATA when the TBE bit in the USART_STAT register is asserted. The TBE bit is cleared by writing USART_TDATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART_TDATA register while a transmission is ongoing, it will be firstly stored in the transmit buffer, and transferred to the transmit shift register after the current transmission is done. If a data is written to the USART_TDATA register while no transmission is ongoing, the TBE bit will be cleared and set soon, because the data will be transferred to the transmit shift register immediately.

If a frame is transmitted and the TBE bit is asserted, the TC bit of the USART_STAT register will be set. An interrupt will be generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

The USART transmit procedure is shown in ***Figure 16-3.USART transmit procedure***. The software operating process is as follows:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1 to configure the number of stop bits.
3. Enable DMA (DENT bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
6. Set the TEN bit in USART_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to the USART_TDATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

**Figure 16-3.USART transmit procedure**




It It is necessary to wait for the TC bit to be asserted before disabling the USART or entering the power saving mode. The TC bit can be cleared by writing 1 to TCC bit in USART_INTC register.

The break frame is sent when the SBKCMD bit is set, and SBKCMD bit is reset after the transmission.

### 16.3.4. USART receiver

After power on, the USART receiver can be enabled by the following procedure:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1.
3. Enable DMA (DENR bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
6. Set the REN bit in USART_CTL0.

After being enabled, the receiver receives a bit stream after a valid start pulse has been detected. Detection on noisy error, parity error, frame error and overrun error is performed during the reception of a frame.

When a frame is received, the RBNE bit in USART_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register. The status of the reception are stored in the USART_STAT register.

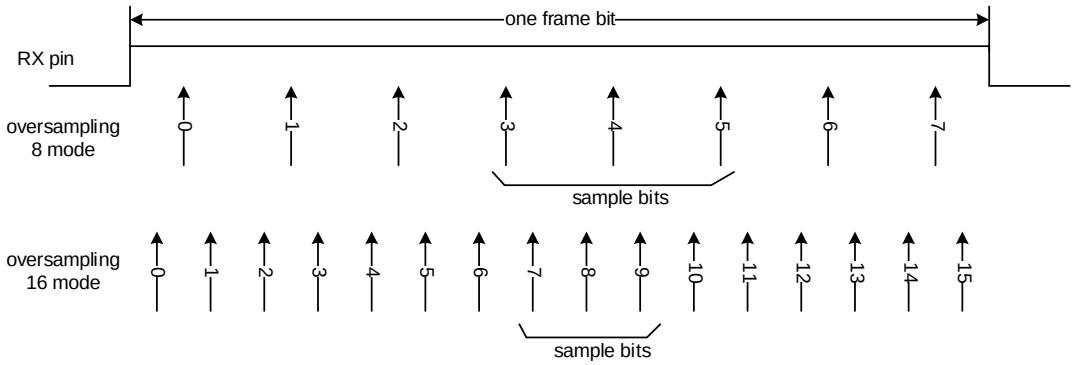
The software can get the received data by reading the USART_RDATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART_RDATA register, whatever it is performed by software directly, or through DMA.

The REN bit should not be disabled when reception is ongoing, or the current frame will be lost.

By default, the receiver gets three samples to evaluate the value of a frame bit. If the oversampling 8 mode is enabled, the 3rd, 4th and 5th samples are used, while in the oversampling 16 mode, the 7th, 8th, and 9th samples are used. If two or more samples of a frame bit is 0, the frame bit is confirmed as a 0, else 1. If the value of the three samples of any bit are not the same, whatever it is a data bit, parity bit or stop bit, a noisy error (NERR)

status will be generated for the frame. An interrupt will be generated, If the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set. If the OSB bit in USART_CTL2 register is set, the receiver gets only one sample to evaluate a bit value. In this situation, no noisy error will be detected.

**Figure 16-4.Oversampling method of a receive frame bit (OSB=0)**




If the parity check function is enabled by setting the PCEN bit in the USART_CTL0 register, the receiver calculates the expected parity value while receiving a frame. The received parity bit will be compared with this expected value. If they are not the same, the parity error (PERR) bit in USART_STAT register will be set. An interrupt is generated, if the PERRIE bit in USART_CTL0 register is set.

If the RX pin is evaluated as 0 during a stop bit, the frame error (FERR) bit in USART_STAT register will be set. An interrupt is generated, If the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set.

When a frame is received, if the RBNE bit is not cleared yet, the last frame will not be stored in the receive data buffer. The overrun error (ORERR) bit in USART_STAT register will be set. An interrupt is generated, if the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set, or if the RBNEIE is set.

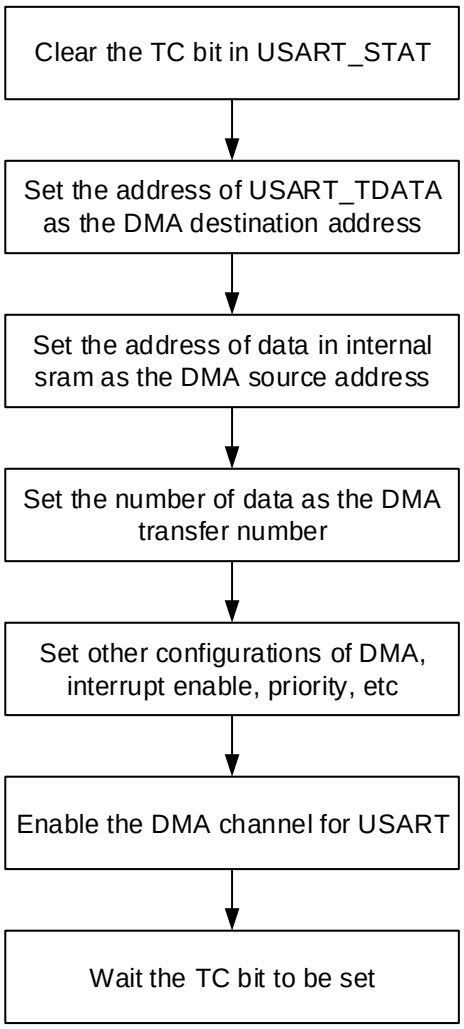
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) occurs during reception, NERR, PERR, FERR or ORERR will be set at the same time with RBNE. If the receive DMA is not enabled, when the RBNE interrupt occurs, software need to check whether there is a noise error, parity error, frame error or overrun error.

### 16.3.5. Use DMA for data buffer access

To reduce the burden of the processor, DMA can be used to access the transmitting and receiving data buffer. The DENT bit in USART_CTL2 is used to enable the DMA transmission, and the DENR bit in USART_CTL2 is used to enable the DMA reception.

When DMA is used for USART transmission, DMA transfers data from internal SRAM to the transmit data buffer of the USART. The configuration step are shown in ***Figure 16-5. Configuration step when using DMA for USART transmission***.

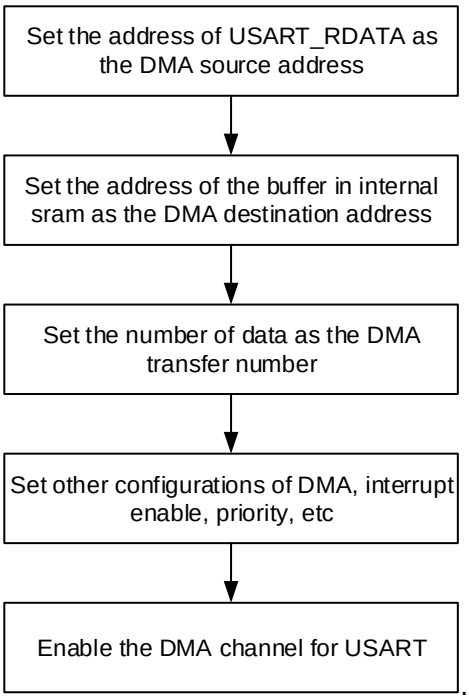
**Figure 16-5. Configuration step when using DMA for USART transmission**




After all of the data frames are transmitted, the TC bit in USART_STAT is set. An interrupt occurs if the TCIE bit in USART_CTL0 is set.

When DMA is used for USART reception, DMA transfers data from the receive data buffer of the USART to the internal SRAM. The configuration steps are shown in ***Figure 16-6. Configuration step when using DMA for USART reception***. If the ERRIE bit in USART_CTL2 is set, interrupts can be generated by the error status bits (FERR, ORERR and NERR) in USART_STAT.

**Figure 16-6. Configuration step when using DMA for USART reception**

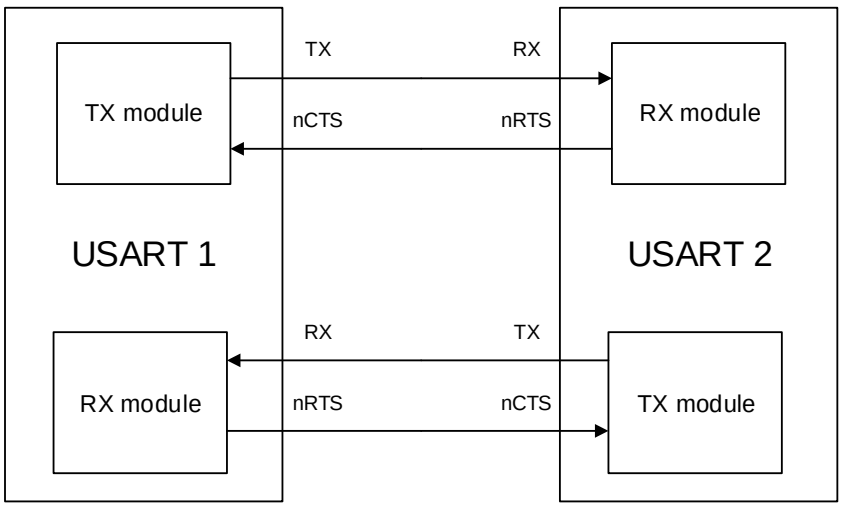



When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt can be generated in the DMA module.

### 16.3.6. Hardware flow control

The hardware flow control function is realized by the nCTS and nRTS pins. The RTS flow control is enabled by writing ‘1’ to the RTSEN bit in USART_CTL2 and the CTS flow control is enabled by writing ‘1’ to the CTSEN bit in USART_CTL2.

**Figure 16-7. Hardware flow control between two USARTs**



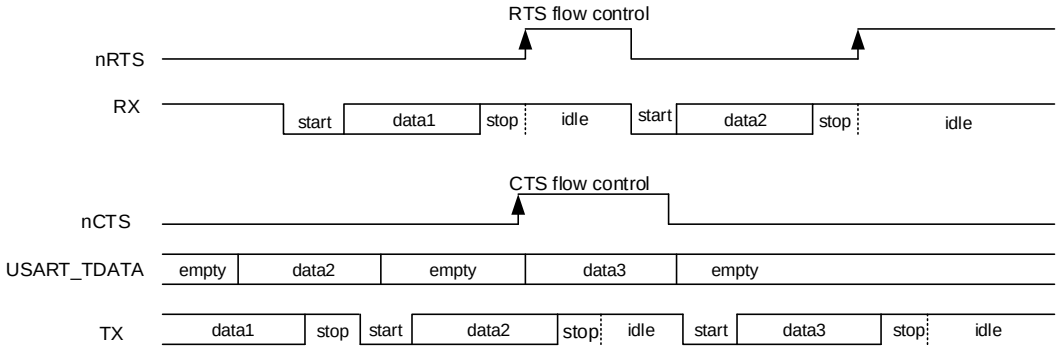
### RTS flow control

The USART receiver outputs the nRTS, which reflects the status of the receive buffer. When data frame is received, the nRTS signal goes high to prevent the transmitter from sending next frame. The nRTS signal keeps high when the receive buffer is full.

### CTS flow control

The USART transmitter monitors the nCTS input pin to decide whether a data frame can be transmitted. If the TBE bit in USART_STAT is '0' and the nCTS signal is low, the transmitter transmits the data frame. When the nCTS signal goes high during a transmission, the transmitter stops after the current transmission is accomplished.

**Figure16-8. Hardware flow control**




### RS485 driver enable

The driver enable feature, which is enabled by setting bit DEM in the USART_CTL2 control register, allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time, which is programmed using the DEA [4:0] bits field in the USART_CTL0 control register, is the time between the activation of the DE signal and the beginning of the START bit. The de-assertion time, which is programmed using the DED [4:0] bits field in the USART_CTL0 control register, is the time between the end of the last stop bit and the de-activation of the DE signal. The polarity of the DE signal can be configured using the DEP bit in the USART_CTL2 control register.

## 16.3.7. Multi-processor communication

In multiprocessor communication, several USARTs are connected as a network. It will be a big burden for a device to monitor all of the messages on the RX pin. To reduce the burden of a device, software can put an USART module into a mute mode by writing 1 to the MMCMD bit in USART_CMD register.

If a USART is in mute mode, all of the receive status bits cannot be set. The USART can also be wake up by hardware by one of the two methods: idle frame method and address match method.

The idle frame wake up method is selected by default. When an idle frame is detected on the RX pin, the hardware clears the RWU bit and exits the mute mode. When it is woken up by an idle frame, the IDLEF bit in USART_STAT will not be set.

When the WM bit of in USART_CTL0 register is set, the MSB bit of a frame is detected as the address flag. If the address flag is high, the frame is treated as an address frame. If the address flag is low, the frame is treated as a data frame. If the LSB 4 or 7 bits, which are configured by the ADDM bit of the USART_CTL1 register, of an address frame is the same as the ADDR bits in the USART_CTL1 register, the hardware will clear the RWU bit and exits the mute mode. The RBNE bit will be set when the frame that wakes up the USART. The status bits are available in the USART_STAT register. If the LSB 4/7 bits of an address frame defers from the ADDR bits in the USART_CTL1 register, the hardware sets the RWU bit and enters mute mode automatically. In this situation, the RBNE bit is not set.

If the PCEN bit in USART_CTL0 is set, the MSB bit will be checked as the parity bit, and the bit preceding the MSB bit is detected as the address bit. If the ADDM bit is set and the receive frame is a 7bit data, the LSB 6 bits will be compared with ADDR[5:0]. If the ADDM bit is set and the receive frame is a 9bit data, the LSB 8 bits will be compared with ADDR[7:0].

### 16.3.8. LIN mode

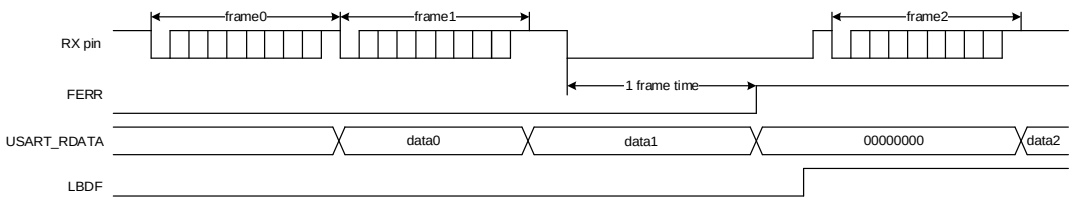
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, STB[1:0] bit in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. The data bits length can only be 8. And the break frame is 13-bit ’0’, followed by 1 stop bit.

The break detection function is totally independent of the normal USART receiver. So a break frame can be detected during the idle state or during a frame. The expected length of a break frame can be selected by configuring LBLEN in USART_CTL1. When the RX pin is detected at low state for a time that is equal to or longer than the expected break frame length (10 bits when LBLEN=0, or 11 bits when LBLEN=1), the LBDF bit in USART_STAT is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

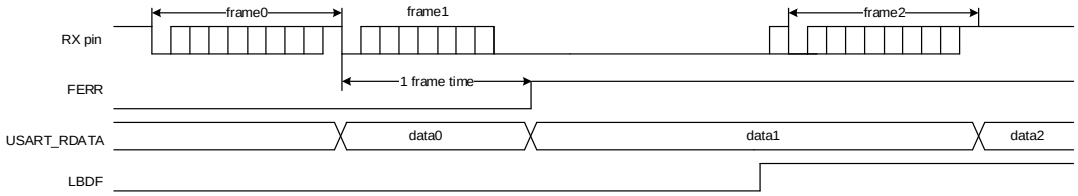
As shown in ***Figure 16-9. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all ‘0’ frame, with an asserted FERR status.

**Figure 16-9. Break frame occurs during idle state**



As shown in ***Figure 16-10. Break frame occurs during a frame***, if a break frame occurs during a frame on the RX pin, the FERR status will be asserted for the current frame.

**Figure 16-10. Break frame occurs during a frame**




## 16.3.9. Synchronous mode

The USART can be used for full-duplex synchronous serial communications only in master mode, by setting the CKEN bit in USART_CTL1. The LMEN bit in USART_CTL1 and SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in synchronous mode. The CK pin is the clock output of the synchronous USART transmitter, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the transmission of the start bit and stop bit. The CLEN bit in USART_CTL1 can be used to determine whether the clock is output or not during the last (address flag) bit transmission. The clock output is also not activated during idle and break frame sending. The CPH bit in USART_CTL1 can be used to determine whether data is captured on the first or the second clock edge. The CPL bit in USART_CTL1 can be used to configure the clock polarity in the USART synchronous idle state.

The CPL, CPH and CLEN bits in USART_CTL1 determine the waveform on the CK pin. Software can only change them when the USART is disabled (UEN=0).

The clock is synchronized with the data transmitted. The receiver in synchronous mode samples the data on the transmitter clock without any oversampling.

**Figure 16-11. Example of USART in synchronous mode**

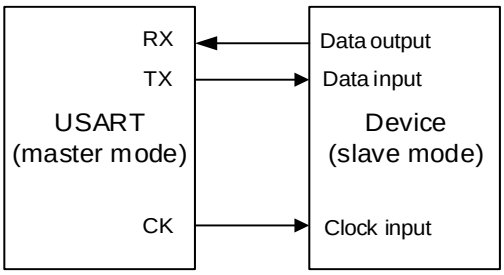
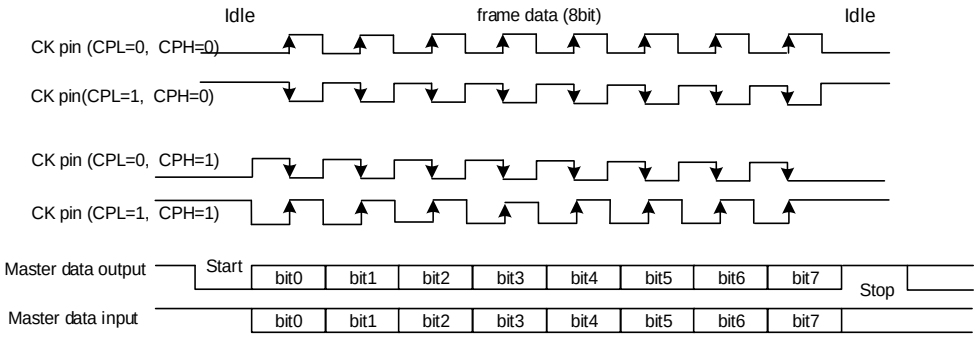


**Figure 16-12. 8-bit format USART synchronous waveform (CLEN=1)**

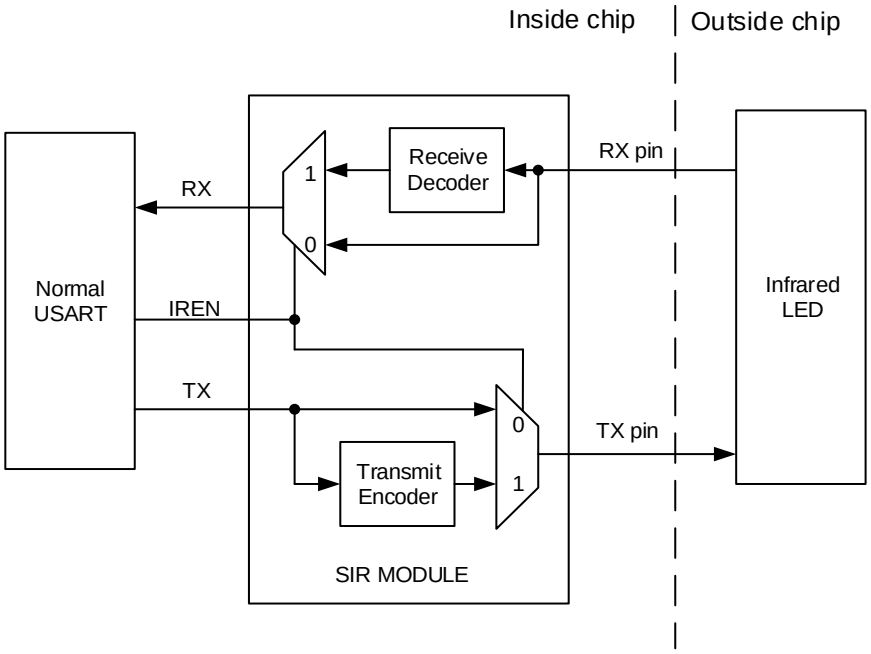



## 16.3.10. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART_CTL2. The LMEN, STB[1:0], CKEN bits in USART_CTL1 and HDEN, SCEN bits in USART_CTL2 should be cleared in IrDA mode.

In IrDA mode, the USART transmission data frame is modulated in the SIR transmit encoder and transmitted to the infrared LED through the TX pin. The SIR receive decoder receives the modulated signal from the infrared LED through the RX pin, and puts the demodulated data frame to the USART receiver. The baud rate should not be larger than 115200 for the encoder.

**Figure 16-13. IrDA SIR ENDEC module**

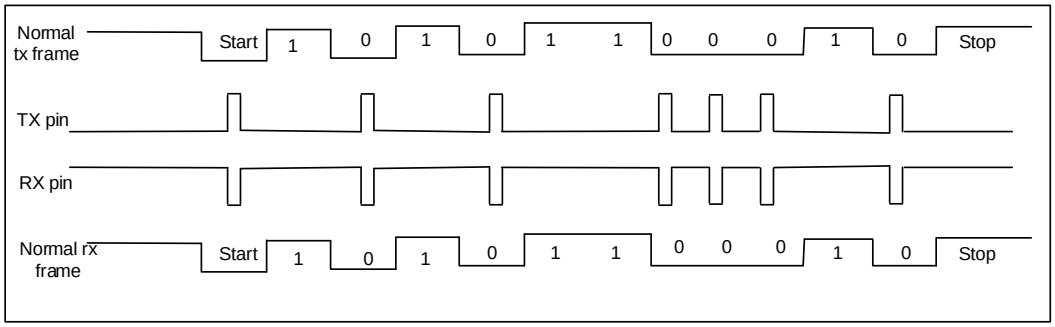



In IrDA mode, the polarity of the TX and RX pins is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic ‘1’, while an infrared light pulse on the IrDA pins (a Return to Zero signal) represents the

logic '0'. The pulse width should be 3/16 of a bit period. The IrDA could not detect any pulse if the pulse width is less than 1 PSC clock. While it can detect a pulse by chance if the pulse width is greater than 1 but smaller than 2 times of PSC clock.

Because the IrDA is a half-duplex protocol, the transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

**Figure 16-14. IrDA data modulation**




The SIR sub module can work in low power mode by setting the IRLP bit in USART_CTL2. The transmit encoder is driven by a low speed clock, which is divided from the PCLK. The division ratio is configured by the PSC[7:0] bits in USART_GP register, USART_BAUD register needs to be configured as 16*PSC[7:0]. The pulse width on the TX pin is 3 cycles of this low speed period. The receiver decoder works in the same manner as the normal IrDA mode.

### 16.3.11. Half-duplex communication mode

The half-duplex communication mode is enabled by setting the HDEN bit in USART_CTL2. The LMEN, CKEN bits in USART_CTL1 and SCEN, IREN bits in USART_CTL2 should be cleared in half-duplex communication mode.

Only one wire is used in half-duplex mode. The TX and RX pins are connected together internally, and the RX pin is no longer used. The TX pin should be configured as open drain mode and communication conflicts are handled by software.

### 16.3.12. Smartcard (ISO7816-3) mode

The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character (T=0) mode and the block (T=1) mode are supported. The smartcard mode is enabled by setting the SCEN bit in USART_CTL2. The LMEN bit in USART_CTL1 and HDEN, IREN bits in USART_CTL2 should be reset in smartcard mode.

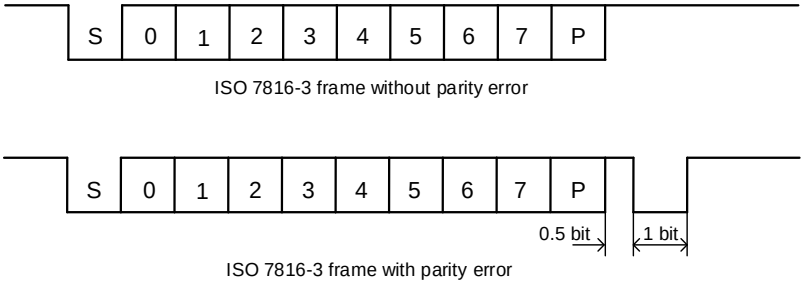
A clock is provided to the smartcard if the CKEN bit is set. The clock can be divided for other use.

The frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits.

The smartcard mode is a half-duplex communication protocol. When connected to a

smartcard, the TX pin must be configured as open drain mode, and drives a bidirectional line that is also driven by the smartcard.

**Figure 16-15. ISO7816-3 frame format**




### Character (T=0) mode

Compared to the timing in normal operation, the transmission time from transmit shift register to the TX pin is delayed by half baud clock, and the TC flag assertion time is delayed by a guard time that is configured by the GUAT[7:0] bits in USART_GP. In Smartcard mode, the internal guard time counter starts counting up after the stop bits of the last data frame, and the GUAT[7:0] bits should be configured as the character guard time (CGT) in ISO7816-3 protocol minus 12. The TC status is forced reset while the guard time counter is counting up. When the counter reaches the programmed value TC is asserted high.

During USART transmission, if a parity error event is detected, the smartcard may NACK the current frame by pulling down the TX pin during the last 1 bit time of the stop bits. The USART can automatically resend data according to the protocol for SCRTNUM times. An interframe gap of 2 bits time will be inserted before the start of a resented frame. At the end of the last repeated character the TC bit is set immediately without guard time. The USART will stop transmitting and assert the frame error status if it still receives the NACK signal after the programmed number of retries. The USART will not take the NACK signal as the start bit.

During USART reception, if the parity error is detected in the current frame, the TX pin is pulled low during the last 1 bit time of the stop bits. This signal is the NACK signal to smartcard. Then a frame error occurs in smartcard side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and the error is regarded as a parity error if the received character is still erroneous after the maximum number of retries which is specified in the SCRTNUM bit field. The NACK signal is enabled by setting the NKEN bit in USART_CTL2.

The idle frame and break frame are not supported in the Smartcard mode.

### Block (T=1) mode

In block (T=1) mode, the NKEN bit in the USART_CTL2 register should be cleared to deactivate the NACK transmission.

When requesting a read from the smartcard, the RT[23:0] bits in USART_RT register should

be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. A timeout interrupt will be generated, if no answer is received from the card before the expiration of this period. If the first character is received before the expiration of the period, it is signaled by the RBNE interrupt. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

In order to allow the automatic check of the maximum wait time between two consecutive characters, the USART_RT register must be programmed to the CWT (character wait time) - 11 value, which is expressed in baudtime units, after the reception of the first character (RBNE interrupt). The USART signals to the software through the RT flag and interrupt (when RTIE bit is set), if the smartcard doesn't send a new character in less than the CWT period after the end of the previous character.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE=0), to count the number of received characters. The length of the block, which must be programmed in the BL[7:0] bits in the USART_RT register, is received from the smartcard in the third byte of the block (prologue field). This register field must be programmed to the minimum value (0x0), before the start of the block, when using DMA mode. With this value, an interrupt is generated after the 4th received character. The software must read the third byte as block length from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BL value. However, before the start of the block, the maximum value of BL (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

The total block length (including prologue, epilogue and information fields) equals BL+4. The end of the block is signaled to the software through the EBF flag and interrupt (when EBIE bit is set). The RT interrupt may occur in case of an error in the block length.

#### Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to high state of the line and parity is even. In this case, the following control bits must be programmed: MSBF=0, DINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to a low state on the signal line and parity is even. In this case, the following control bits must be programmed: MSBF=1, DINV=1.

### 16.3.13. ModBus communication

The USART offers basic support for the implementation of ModBus/RTU and ModBus/ASCII protocols by implementing an end of block detection.

In the ModBus/RTU mode, the end of one block is recognized by an idle line for more than 2

characters time. This function is implemented through the programmable timeout function.

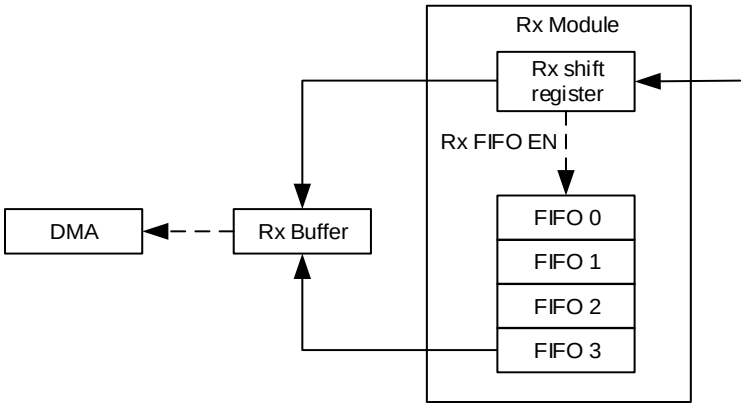
To detect the idle line, the RTEN bit in the USART_CTL1 register and the RTIE in the USART_CTL0 register must be set. The USART_RT register must be set to the value corresponding to a timeout of 2 characters time. After the last stop bit is received, when the receive line is idle for this duration, an interrupt will be generated, informing the software that the current block reception is completed.

In the ModBus/ASCII mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function by programming the LF ASCII code in the ADDR field and activating the address match interrupt (AMIE=1). When a LF has been received or can check the CR/LF in the DMA buffer, the software will be informed.

### 16.3.14. Receive FIFO

The receive FIFO can be enabled by setting the RFEN bit of the USART_RFCS register to avoid the overrun error when the CPU can't serve the RBNE interrupt immediately. Up to 5 frames receive data can be stored in the receive FIFO and receive buffer. The RFFINT flag will be set when the receive FIFO is full. An interrupt is generated if the RFFIE bit is set.

**Figure 16-16. USART receive FIFO structure**




If the software read receive data buffer in the routing of the RBNE interrupt, the RBNEIE bit should be reset at the beginning of the routing and set after all of the receive data is read out. The PERR/NERR/FERR/EBF flags should be cleared before reading a receive data out.

### 16.3.15. Wakeup from Deep-sleep mode

The USART is able to wake up the MCU from Deep-sleep mode by the standard RBNE interrupt or the WUM interrupt.

The UESM bit must be set and the USART clock must be set to IRC8M or LXTAL (refer to the reset and clock unit RCU section).

When using the standard RBNE interrupt, the RBNEIE bit must be set before entering Deep-sleep mode.

When using the WUIE interrupt, the source of WUIE interrupt may be selected through the WUM bit fields.

DMA must be disabled before entering Deep-sleep mode. Before entering Deep-sleep mode, software must check that the USART is not performing a transfer, by checking the BSY flag in the USART_STAT register. The REA bit must be checked to ensure the USART is actually enabled.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUIE bit is set, independently of whether the MCU is in stop or active mode.

### 16.3.16. USART interrupts

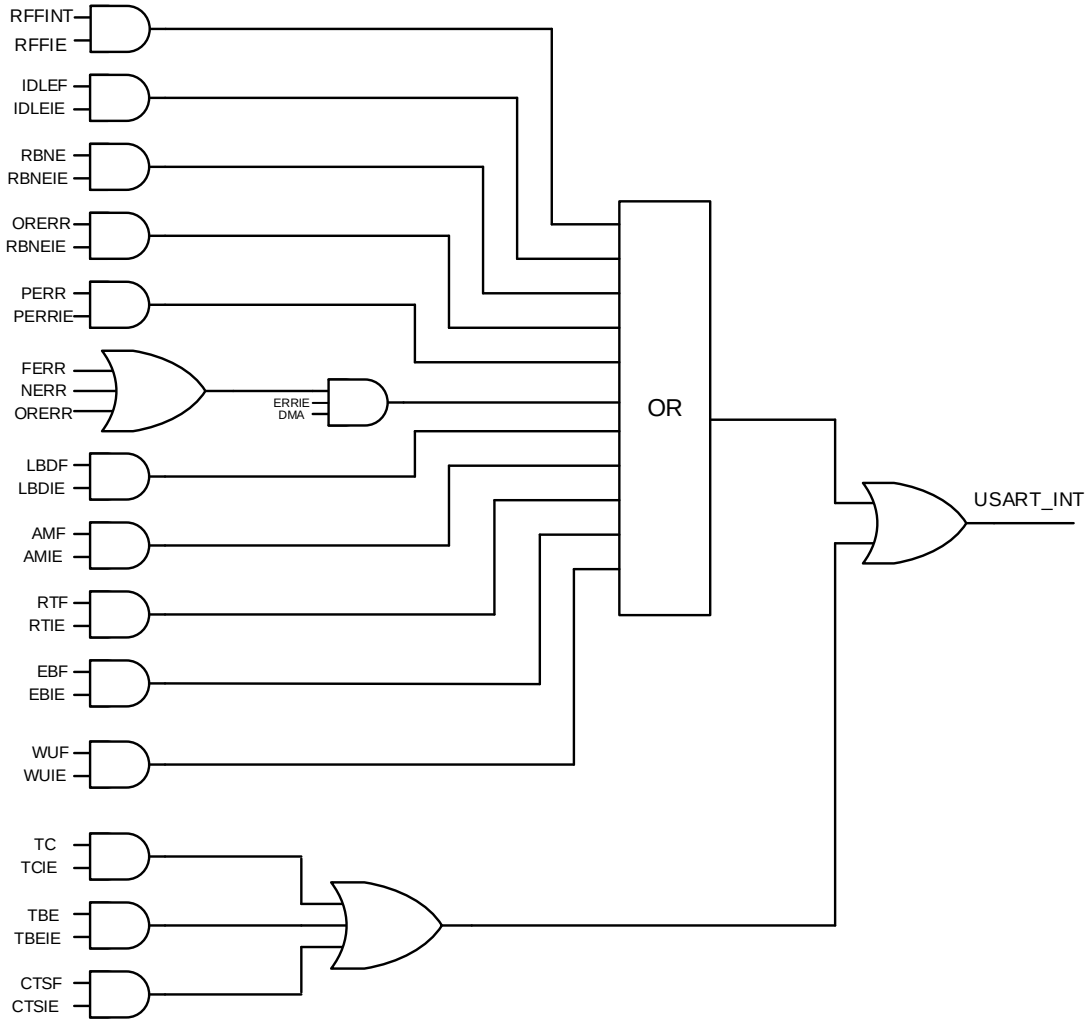
The USART interrupt events and flags are listed in ***Table 16-3. USART interrupt requests***.

**Table 16-3. USART interrupt requests**

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| Transmit data register empty | TBE | TBEIE |
| CTS flag | CTSF | CTSIE |
| Transmission complete | TC | TCIE |
| Received data ready to be read | RBNE | RBNEIE |
| Overrun error detected | ORERR | |
| Receive FIFO full | RFFINT | RFFIE |
| Idle line detected | IDLEF | IDLEIE |
| Parity error flag | PERR | PERRIE |
| Break detected flag in LIN mode | LBDF | LBDIE |
| Reception errors (Noise flag, overrun error, framing error) in DMA reception | NERR or ORERR or FERR | ERRIE |
| Character match | AMF | AMIE |
| Receiver timeout error | RTF | RTIE |
| End of Block | EBF | EBIE |
| Wakeup from Deep-sleep mode | WUF | WUIE |

All of the interrupt events are ORed together before being sent to the interrupt controller, so the USART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine

**Figure 16-17. USART interrupt mapping diagram**



## 16.4. Register definition

USART0 base address: 0x4001 3800
USART1 base address: 0x4000 4400

### 16.4.1. Control register 0 (USART_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | EBIE | RTIE | DEA[4:0] | | | | | DED[4:0] | | | | |
| | | | | rw | rw | | | rw | | | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OVSMOD | AMIE | MEN | WL | WM | PCEN | PM | PERRIE | TBEIE | TCIE | RBNEIE | IDLEIE | TEN | REN | UESM | UEN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | EBIE | End of block interrupt enable<br>0: Disable end of block interrupt<br>1: Enable end of block interrupt<br>This bit is reserved in USART1. |
| 26 | RTIE | Receiver timeout interrupt enable<br>0: Disable receiver timeout interrupt<br>1: Enable receiver timeout interrupt<br>This bit is reserved in USART1. |
| 25:21 | DEA[4:0] | Driver enable assertion time<br>These bits are used to define the time between the activation of the DE (driver enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 20:16 | DED[4:0] | Driver enable de-assertion time<br>These bits are used to define the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (driver enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 15 | OVSMOD | Oversample mode<br>0: Oversampling by 16 |

| | | |
|---|---|---|
| | | 1: Oversampling by 8<br>This bit must be kept cleared in LIN, IrDA and smartcard modes.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 14 | AMIE | ADDR match interrupt enable<br>0: Disable ADDR match interrupt<br>1: Enable ADDR match interrupt |
| 13 | MEN | Mute mode enable<br>0: Disable mute mode<br>1: Enable mute mode |
| 12 | WL | Word length<br>0: 8 Data bits<br>1: 9 Data bits<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 11 | WM | Wakeup method in mute mode<br>0: Idle Line<br>1: Address mask<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 10 | PCEN | Parity control enable<br>0: Disable parity control<br>1: Enable parity control<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 9 | PM | Parity mode<br>0: Even parity<br>1: Odd parity<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 8 | PERRIE | Parity error interrupt enable<br>0: Disable parity error interrupt<br>1: Enable parity error interrupt. An interrupt will occur when the PERR bit is set in USART_STAT. |
| 7 | TBEIE | Transmitter register empty interrupt enable<br>0: Disable transmitter register empty interrupt<br>1: Enable transmitter register empty interrupt. An interrupt will occur when the TBE bit is set in USART_STAT. |
| 6 | TCIE | Transmission complete interrupt enable<br>0: Disable transmission complete interrup<br>1: Enable transmission complete interrupt. An interrupt occurs when the TC bit in USART_STAT is set. |
| 5 | RBNEIE | Read data buffer not empty interrupt and overrun error interrupt enable |

| | | |
|---|---|---|
| | | 0: Disable read data register not empty interrupt and overrun error interrupt<br>1: Enable read data register not empty interrupt and overrun error interrupt. An interrupt will occur whenever the ORERR bit is set or the RBNE bit is set in USART_STAT. |
| 4 | IDLEIE | IDLE line detected interrupt enable<br>0: Disable IDLE line detected interrupt<br>1: Enable IDLE line detected interrupt. An interrupt will occur when the IDLEF bit is set in USART_STAT. |
| 3 | TEN | Transmitter enable<br>0: Disable transmitter<br>1: Enable transmitter |
| 2 | REN | Receiver enable<br>0: Disable receiver<br>1: Enable receiver and begins searching for a start bit |
| 1 | UESM | USART enable in Deep-sleep mode<br>0: USART not able to wake up the MCU from Deep-sleep mode.<br>1: USART able to wake up the MCU from Deep-sleep mode. Providing that the clock source for the USART must be IRC8M or LXTAL.<br>This bit is reserved in USART1. |
| 0 | UEN | USART enable<br>0: Disable USART prescaler and outputs<br>1: Enable USART prescaler and outputs |

## 16.4.2. Control register 1 (USART_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR[7:0] | | | | | | | | RTEN | Reserved | | | MSBF | DINV | TINV | RINV |
| rw | | | | | | | | rw | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STRP | LMEN | STB[1:0] | | CKEN | CPL | CPH | CLEN | Reserved | LBDIE | LBLEN | ADDM | Reserved | | | |
| rw | rw | rw | | rw | rw | rw | rw | | rw | rw | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | ADDR[7:0] | Address of the USART terminal<br>These bits give the address of the USART terminal.<br>In multiprocessor communication during mute mode or Deep-sleep mode, this is used for wakeup with address mask detection. The received frame, the MSB of |

| | | |
|---|---|---|
| | | which is equal to 1, will be compared to these bits. When the ADDM bit is reset, only the ADDR[3:0] bits are used to compare.<br>In normal reception, these bits are also used for character detection. The whole received character (8-bit) is compared to the ADDR[7:0] value and AMF flag is set on matching.<br>This bit field cannot be written when both reception (REN=1) and USART (UEN=1) are enabled. |
| 23 | RTEN | Receiver timeout enable<br>0: Disable receiver timeout function<br>1: Enable receiver timeout function<br>This bit is reserved in USART1. |
| 22:20 | Reserved | Must be kept at reset value. |
| 19 | MSBF | Most significant bit first<br>0: Data is transmitted/received with the LSB first<br>1: Data is transmitted/received with the MSB first<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 18 | DINV | Data bit level inversion<br>0: Data bit signal values are not inverted<br>1: Data bit signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 17 | TINV | TX pin level inversion<br>0: TX pin signal values are not inverted<br>1: TX pin signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 16 | RINV | RX pin level inversion<br>0: RX pin signal values are not inverted<br>1: RX pin signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 15 | STRP | Swap TX/RX pins<br>0: The TX and RX pins functions are not swapped<br>1: The TX and RX pins functions are swapped<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 14 | LMEN | LIN mode enable<br>0: LIN mode disabled<br>1: LIN mode enabled<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 13:12 | STB[1:0] | STOP bits length<br>00: 1 Stop bit |

| | | |
|---|---|---|
| | | 01: 0.5 Stop bit<br>10: 2 Stop bits<br>11: 1.5 Stop bit<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 11 | CKEN | CK pin enable<br>0: Disable CK pin<br>1: Enable CK pin<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 10 | CPL | Clock polarity<br>0: Steady low value on CK pin outside transmission window in synchronous mode<br>1: Steady high value on CK pin outside transmission window in synchronous mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 9 | CPH | Clock phase<br>0: The first clock transition is the first data capture edge in synchronous mode<br>1: The second clock transition is the first data capture edge in synchronous mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 8 | CLEN | CK length<br>0: The clock pulse of the last data bit (MSB) is not output to the CK pin in synchronous mode.<br>1: The clock pulse of the last data bit (MSB) is output to the CK pin in synchronous mode.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 7 | Reserved | Must be kept at reset value. |
| 6 | LBDIE | LIN break detection interrupt enable<br>0: Disable LIN break detection interrupt<br>1: Enable LIN break detection interrupt. An interrupt will occur when the LBDF bit is set in USART_STAT.<br>This bit is reserved in USART1. |
| 5 | LBLEN | LIN break frame length<br>0: 10 bit break detection<br>1: 11 bit break detection<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 4 | ADDM | Address detection mode<br>This bit is used to select between 4-bit address detection and full-bit address detection.<br>0: 4-bit address detection<br>1: Full-bit address detection. In 7-bit, 8-bit and 9-bit data modes, the address detection is done on 6-bit, 7-bit and 8-bit address (ADDR[5:0], ADDR[6:0] and |

ADDR[7:0]) respectively.
This bit field cannot be written when the USART is enabled (UEN=1).

| | | |
|---|---|---|
| 3:0 | Reserved | Must be kept at reset value. |

### 16.4.3. Control register 2 (USART_CTL2)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | WUIE | WUM[1:0] | | SCRTNUM[2:0] | | | Reserved |
| | | | | | | | | | rw | rw | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DEP | DEM | DDRE | OVRD | OSB | CTSIE | CTSEN | RTSEN | DENT | DENR | SCEN | NKEN | HDEN | IRLP | IREN | ERRIE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | WUIE | Wakeup from Deep-sleep mode interrupt enable<br>0: Disable wakeup from Deep-sleep mode interrupt<br>1: Enable wakeup from Deep-sleep mode interrupt<br>This bit is reserved in USART1. |
| 21:20 | WUM[1:0] | Wakeup mode from Deep-sleep mode<br>These bits are used to specify the event which activates the WUF (wakeup from Deep-sleep mode flag) in the USART_STAT register.<br>00: WUF active on address match, which is defined by ADDR and ADDM<br>01: Reserved<br>10: WUF active on Start bit<br>11: WUF active on RBNE<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 19:17 | SCRTNUM[2:0] | Smartcard auto-retry number<br>In smartcard mode, these bits specify the number of retries in transmission and reception.<br>In transmission mode, a transmission error (FERR bit set) will occur after this number of automatic retransmission retries.<br>In reception mode, reception error (RBNE and PERR bits set) will occur after this number or erroneous reception trials.<br>When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode. |

| | | |
|---|---|---|
| | | This bit field is only can be cleared to 0 when the USART is enabled (UEN=1), to stop retransmission.<br>This bit is reserved in USART1. |
| 16 | Reserved | Must be kept at reset value. |
| 15 | DEP | Driver enable polarity mode<br>0: DE signal is active high<br>1: DE signal is active low<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 14 | DEM | Driver enable mode<br>This bit is used to activate the external transceiver control, through the DE signal, which is output on the RTS pin.<br>0: Disable DE function<br>1: Enable DE function<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 13 | DDRE | Disable DMA on reception error<br>0: DMA is not disabled in case of reception error. The DMA request is not asserted to make sure the erroneous data is not transferred, but the next correct received data will be transferred. The RBNE is kept 0 to prevent overrun, but the corresponding error flag is set. This mode can be used in smartcard mode.<br>1: The DMA request is not asserted in case of reception error until the error flag is cleared. The RBNE flag and corresponding error flag will be set. The software must first disable the DMA request (DMAR = 0) or clear RBNE before clearing the error flag.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 12 | OVRD | Overrun disable<br>0: Enable overrun functionality. The ORERR error flag will be set when received data is not read before receiving new data, and the new data will be lost.<br>1: Disable overrun functionality. The ORERR error flag will not be set when received data is not read before receiving new data, and the new received data overwrites the previous content of the USART_RDATA register<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 11 | OSB | One sample bit method<br>0: Three sample bit method<br>1: One sample bit method<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 10 | CTSIE | CTS interrupt enable<br>0: Disable CTS interrupt<br>1: Enable CTS interrupt. An interrupt will occur when the CTS bit is set in USART_STAT. |

| | | |
|---|---|---|
| 9 | CTSEN | CTS enable<br>0: Disable CTS hardware flow control<br>1: Enable CTS hardware flow control<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 8 | RTSEN | RTS enable<br>0: Disable RTS hardware flow control<br>1: Enable RTS hardware flow control. Data can be requested only when there is space in the receive buffer.<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 7 | DENT | DMA enable for transmission<br>0: Disable<br>1: Enable |
| 6 | DENR | DMA enable for reception<br>0: Disable<br>1: Enable |
| 5 | SCEN | Smartcard mode enable<br>0: Disable Smartcard mode<br>1: Enable Smartcard mode<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 4 | NKEN | NACK enable in Smartcard mode<br>0: Disable NACK transmission when parity error<br>1: Enable NACK transmission when parity error<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 3 | HDEN | Half-duplex enable<br>0: Disable Half duplex mode<br>1: Enable Half duplex mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 2 | IRLP | IrDA low-power<br>0: Normal mode<br>1: Low-power mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 1 | IREN | IrDA mode enable<br>0: Disable IrDA<br>1: Enable IrDA<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 0 | ERRIE | Error interrupt enable |

0: Disable error interrupt

1: Enable error interrupt. An interrupt will occur whenever the FERR bit or the ORERR bit or the NERR bit is set in USART_STAT in multibuffer communication.

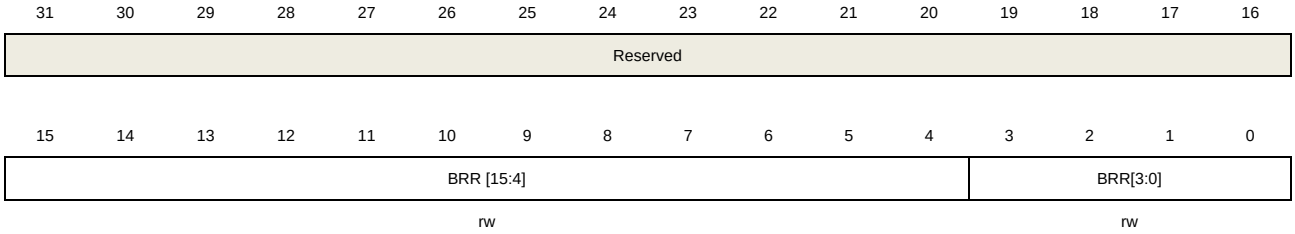
### 16.4.4. Baud rate generator register (USART_BAUD)

Address offset: 0x0C

Reset value: 0x0000 0000

This register cannot be written when the USART is enabled (UEN=1).

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:4 | BRR[15:4] | Integer of baud-rate divider<br>INTDIV[11:0] = BRR[15:4] |
| 3:0 | BRR [3:0] | Fraction of baud-rate divider<br>If OVSMOD = 0, FRADIV [3:0] = BRR [3:0];<br>If OVSMOD = 1, FRADIV [3:1] = BRR [2:0], BRR [3] must be reset. |

### 16.4.5. Prescaler and guard time configuration register (USART_GP)

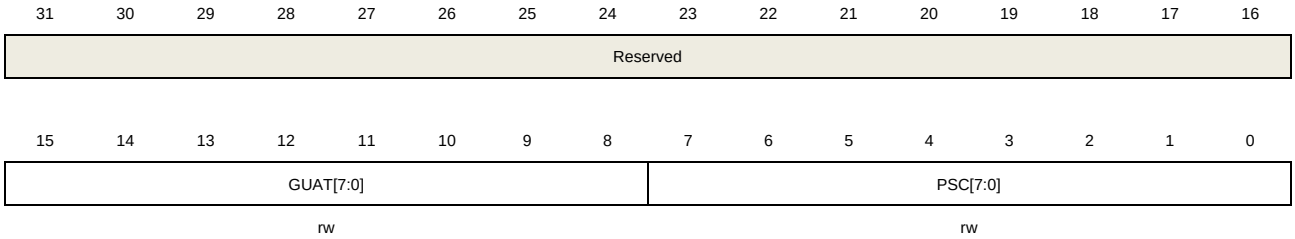
Address offset: 0x10

Reset value: 0x0000 0000

This register cannot be written when the USART is enabled (UEN=1).

This register is reserved in USART1.

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | GUAT[7:0] | Guard time value in smartcard mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 7:0 | PSC[7:0] | Prescaler value for dividing the system clock<br>In IrDA low-power mode, the division factor is the prescaler value.<br>00000000: Reserved - do not program this value<br>00000001: divides the source clock by 1<br>00000010: divides the source clock by 2<br>...<br>In IrDA normal mode,<br>00000001: can be set this value only<br>In smartcard mode, the prescaler value for dividing the system clock is stored in PSC[4:0] bits. And the bits of PSC[7:5] must be kept at reset value. The division factor is twice as the prescaler value.<br>00000: Reserved - do not program this value<br>00001: divides the source clock by 2<br>00010: divides the source clock by 4<br>00011: divides the source clock by 6<br>...<br>This bit field cannot be written when the USART is enabled (UEN=1). |

### 16.4.6. Receiver timeout register (USART_RT)

Address offset: 0x14
Reset value: 0x0000 0000

This bit is reserved in USART1.

This register has to be accessed by word (32-bit).

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | BL[7:0] | Block length<br>These bits specify the block length in smartcard (T=1) reception. Its value equals the number of information characters + the length of the epilogue field (1-LEC/2-CRC) - 1.<br>This value, which must be programmed only once per received block, can be |

| | | |
|---|---|---|
| | | programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE=0 in smartcard mode.<br>In other modes, when REN=0 (receiver disabled) and/or when the EBC bit is written to 1, the block length counter is reset. |
| 23:0 | RT[23:0] | Receiver timeout threshold<br>These bits are used to specify receiver timeout value in terms of number of baud clocks.<br>In standard mode, the RTF flag is set if no new start bit is detected for more than the RT value after the last received character.<br>In smartcard mode, the CWT and BWT are implemented by this value. In this case, the timeout measurement is started from the start bit of the last received character.<br>These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the counter. These bits must only be programmed once per received character.<br>**Note**: When oversampling by 16, RT[23:0] cannot be configured as 0xFFFFFF. |

## 16.4.7. Command register (USART_CMD)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 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 | | | | | | | | | | | TXFCMD | RXFCMD | MMCMD | SBKCMD | Reserved |
| | | | | | | | | | | | w | w | w | w | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | TXFCMD | Transmit data flush request<br>Writing 1 to this bit sets the TBE flag, to discard the transmit data.<br>This bit is reserved in USART1. |
| 3 | RXFCMD | Receive data flush command<br>Writing 1 to this bit clears the RBNE flag to discard the received data without reading it. |
| 2 | MMCMD | Mute mode command<br>Writing 1 to this bit makes the USART into mute mode and sets the RWU flag. |
| 1 | SBKCMD | Send break command<br>Writing 1 to this bit sets the SBKF flag and makes the USART send a BREAK |

| | | |
|---|---|---|
| | | frame, as soon as the transmit machine is idle. |
| 0 | Reserved | Must be kept at reset value. |

## 16.4.8. Status register (USART_STAT)

Address offset: 0x1C

Reset value: 0x0000 00C0

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | REA | TEA | WUF | RWU | SBF | AMF | BSY |
| | | | | | | | | | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | EBF | RTF | CTS | CTSF | LBDF | TBE | TC | RBNE | IDLEF | ORERR | NERR | FERR | PERR |
| | | | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | REA | Receive enable acknowledge flag<br>This bit, which is set/reset by hardware, reflects the receive enable state of the USART core logic.<br>0: The USART core receiving logic has not been enabled<br>1: The USART core receiving logic has been enabled |
| 21 | TEA | Transmit enable acknowledge flag<br>This bit, which is set/reset by hardware, reflects the transmit enable state of the USART core logic.<br>0: The USART core transmitting logic has not been enabled<br>1: The USART core transmitting logic has been enabled |
| 20 | WUF | Wakeup from Deep-sleep mode flag<br>0: No wakeup from Deep-sleep mode<br>1: Wakeup from Deep-sleep mode. An interrupt is generated if WUFIE=1 in the USART_CTL2 register and the MCU is in Deep-sleep mode.<br>This bit is set by hardware when a wakeup event, which is defined by the WUM bit field, is detected.<br>Cleared by writing a 1 to the WUC in the USART_INTC register.<br>This bit can also be cleared when UESM is cleared.<br>This bit is reserved in USART1. |
| 19 | RWU | Receiver wakeup from mute mode<br>This bit is used to indicate if the USART is in mute mode.<br>0: Receiver in active mode<br>1: Receiver in mute mode |

| | | |
|---|---|---|
| | | It is cleared/set by hardware when a wakeup/mute sequence (address or IDLEIE) is recognized, which is selected by the WM bit in the USART_CTL0 register.<br>This bit can only be set by writing 1 to the MMCMD bit in the USART_CMD register when wakeup on IDLEIE mode is selected. |
| 18 | SBF | Send break flag<br>0: No break character is transmitted<br>1: Break character will be transmitted<br>This bit indicates that a send break character was requested.<br>Set by software, by writing 1 to the SBKCMD bit in the USART_CMD register.<br>Cleared by hardware during the stop bit of break transmission. |
| 17 | AMF | ADDR match flag<br>0: ADDR does not match the received character<br>1: ADDR matches the received character, An interrupt is generated if AMIE=1 in the USART_CTL0 register.<br>Set by hardware, when the character defined by ADDR [7:0] is received.<br>Cleared by writing 1 to the AMC in the USART_INTC register. |
| 16 | BSY | Busy flag<br>0: USART reception path is idle<br>1: USART reception path is working |
| 15:13 | Reserved | Must be kept at reset value. |
| 12 | EBF | End of block flag<br>0: End of Block not reached<br>1: End of Block (number of characters) reached. An interrupt is generated if the EBIE=1 in the USART_CTL1 register<br>Set by hardware when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.<br>Cleared by writing 1 to EBC bit in USART_INTC register.<br>This bit is reserved in USART1. |
| 11 | RTF | Receiver timeout flag<br>0: Timeout value not reached<br>1: Timeout value reached without any data reception. An interrupt is generated if RTIE bit in the USART_CTL1 register is set.<br>Set by hardware when the RT value, programmed in the USART_RT register has lapsed without any communication.<br>Cleared by writing 1 to RTC bit in USART_INTC register.<br>The timeout corresponds to the CWT or BWT timings in smartcard mode.<br>This bit is reserved in USART1. |
| 10 | CTS | CTS level<br>This bit equals to the inverted level of the nCTS input pin.<br>0: nCTS input pin is in high level |

| | | |
|---|---|---|
| | | 1: nCTS input pin is in low level |
| 9 | CTSF | CTS change flag<br>0: No change occurred on the nCTS status line<br>1: A change occurred on the nCTS status line. An interrupt will occur if the CTSIE bit is set in USART_CTL2.<br>Set by hardware when the nCTS input toggles.<br>Cleared by writing 1 to CTSC bit in USART_INTC register. |
| 8 | LBDF | LIN break detected flag<br>0: LIN Break is not detected<br>1: LIN Break is detected. An interrupt will occur if the LBDIE bit is set in USART_CTL1<br>Set by hardware when the LIN break is detected.<br>Cleared by writing 1 to LBDC bit in USART_INTC register.<br>This bit is reserved in USART1. |
| 7 | TBE | Transmit data register empty<br>0: Data is not transferred to the shift register<br>1: Data is transferred to the shift register. An interrupt will occur if the TBEIE bit is set in USART_CTL0.<br>Set by hardware when the content of the USART_TDATA register has been transferred into the transmit shift register or writing 1 to TXFCMD bit of the USART_CMD register.<br>Cleared by a write to the USART_TDATA. |
| 6 | TC | Transmission completed<br>0: Transmission is not completed<br>1: Transmission is complete. An interrupt will occur if the TCIE bit is set in USART_CTL0.<br>Set by hardware if the transmission of a frame containing data is completed and if the TBE bit is set.<br>Cleared by writing 1 to TCC bit in USART_INTC register. |
| 5 | RBNE | Read data buffer not empty<br>0: Data is not received<br>1: Data is received and ready to be read. An interrupt will occur if the RBNEIE bit is set in USART_CTL0.<br>Set by hardware when the content of the receive shift register has been transferred to the USART_RDATA.<br>Cleared by reading the USART_RDATA or writing 1 to RXFCMD bit of the USART_CMD register. |
| 4 | IDLEF | IDLE line detected flag<br>0: No Idle line is detected<br>1: Idle line is detected. An interrupt will occur if the IDLEIE bit is set in |

| | | |
|---|---|---|
| | | USART_CTL0.<br>Set by hardware when an Idle Line is detected. It will not be set again until the RBNE bit has been set itself.<br>Cleared by writing 1 to IDLEC bit in USART_INTC register. |
| 3 | ORERR | Overrun error<br>0: No overrun error is detected<br>1: Overrun error is detected. An interrupt will occur if the RBNEIE bit is set in USART_CTL0. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when the word in the receive shift register is ready to be transferred into the USART_RDATA register while the RBNE bit is set.<br>Cleared by writing 1 to OREC bit in USART_INTC register. |
| 2 | NERR | Noise error flag<br>0: No noise error is detected<br>1: Noise error is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when noise error is detected on a received frame.<br>Cleared by writing 1 to NEC bit in USART_INTC register. |
| 1 | FERR | Frame error flag<br>0: No framing error is detected<br>1: Frame error flag or break character is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when a de-synchronization, excessive noise or a break character is detected. This bit will be set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame), when USART transmits in smartcard mode.<br>Cleared by writing 1 to FEC bit in USART_INTC register. |
| 0 | PERR | Parity error flag<br>0: No parity error is detected<br>1: Parity error flag is detected. An interrupt will occur if the PERRIE bit is set in USART_CTL0.<br>Set by hardware when a parity error occurs in receiver mode.<br>Cleared by writing 1 to PEC bit in USART_INTC register. |

### 16.4.9. Interrupt status clear register (USART_INTC)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

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

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | w | | | w | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | EBC | RTC | Reserved | CTSC | LBDC | Reserved | TCC | Reserved | IDLEC | OREC | NEC | FEC | PEC |
| | | | w | w | | w | w | | w | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | WUC | Wakeup from Deep-sleep mode clear<br>Writing 1 to this bit clears the WUF bit in the USART_STAT register.<br>This bit is reserved in USART1. |
| 19:18 | Reserved | Must be kept at reset value. |
| 17 | AMC | ADDR match clear<br>Writing 1 to this bit clears the AMF bit in the USART_STAT register. |
| 16:13 | Reserved | Must be kept at reset value. |
| 12 | EBC | End of block clear<br>Writing 1 to this bit clears the EBF bit in the USART_STAT register.<br>This bit is reserved in USART1. |
| 11 | RTC | Receiver timeout clear<br>Writing 1 to this bit clears the RTF flag in the USART_STAT register.<br>This bit is reserved in USART1. |
| 10 | Reserved | Must be kept at reset value. |
| 9 | CTSC | CTS change clear<br>Writing 1 to this bit clears the CTSF bit in the USART_STAT register. |
| 8 | LBDC | LIN break detected clear<br>Writing 1 to this bit clears the LBDF flag in the USART_STAT register.<br>This bit is reserved in USART1. |
| 7 | Reserved | Must be kept at reset value. |
| 6 | TCC | Transmission complete clear<br>Writing 1 to this bit clears the TC bit in the USART_STAT register. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | IDLEC | Idle line detected clear<br>Writing 1 to this bit clears the IDLEF bit in the USART_STAT register. |
| 3 | OREC | Overrun error clear<br>Writing 1 to this bit clears the ORERR bit in the USART_STAT register. |
| 2 | NEC | Noise detected clear |

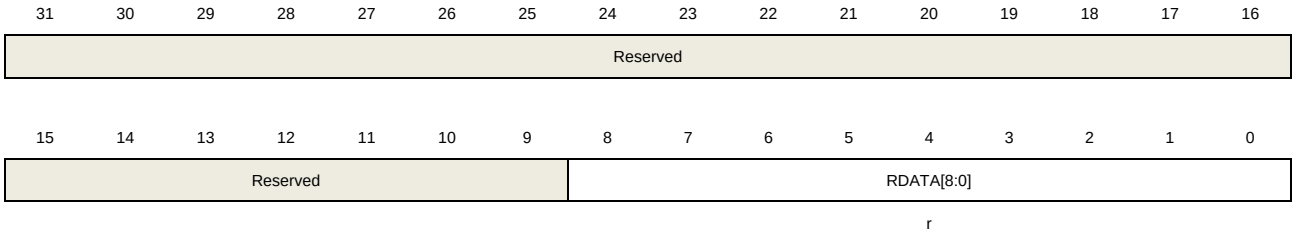
| | | |
|---|---|---|
| | | Writing 1 to this bit clears the NERR bit in the USART_STAT register. |
| 1 | FEC | Frame error flag clear<br>Writing 1 to this bit clears the FERR bit in the USART_STAT register. |
| 0 | PEC | Parity error clear<br>Writing 1 to this bit clears the PERR bit in the USART_STAT register. |

## 16.4.10. Receive data register (USART_RDATA)

Address offset: 0x24

Reset value: 0xXXXX XXXX

This register has to be accessed by word (32-bit).



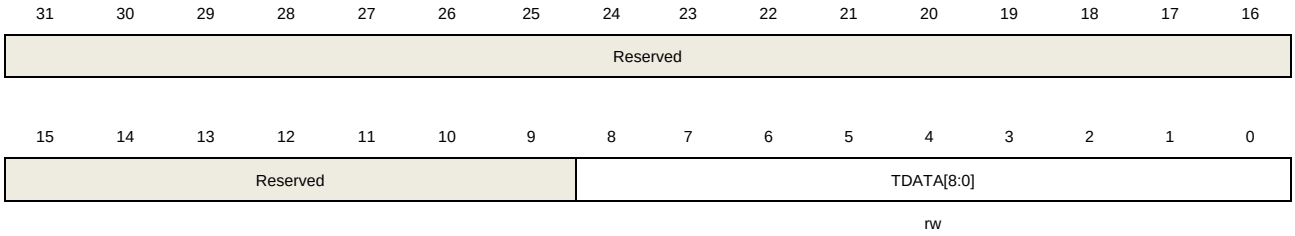

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8:0 | RDATA[8:0] | Receive data value<br>The received data character is contained in these bits.<br>The value read in the MSB (bit 7 or bit 8 depending on the data length) will be the received parity bit, if receiving with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register). |

## 16.4.11. Transmit data register (USART_TDATA)

Address offset: 0x28

Reset value: 0xXXXX XXXX

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 8:0 | TDATA[8:0] | Transmit data value<br>The transmit data character is contained in these bits.<br>The value written in the MSB (bit 7 or bit 8 depending on the data length) will be replaced by the parity, when transmitting with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register).<br>This register must be written only when TBE bit in USART_STAT register is set. |

## 16.4.12. USART coherence control register (USART_CHC)

Address offset: 0xC0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 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 | | | | | | | EPERR | Reserved | | | | | | | HCM |
| | | | | | | | w0c | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | EPERR | Early parity error flag. This flag will be set as soon as the parity bit has been detected, which is before RBNE flag. This flag is cleared by writing 0.<br>0: No parity error is detected<br>1: Parity error is detected. |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | HCM | Hardware flow control coherence mode<br>0: nRTS signal equals to the RBNE in status register<br>1: nRTS signal is set when the last data bit (parity bit when pce is set) has been sampled. |

## 16.4.13. USART receive FIFOcontrol and status register (USART_RFCS)

Address offset: 0xD0

Reset value: 0x0000 0400

This register has to be accessed by word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFFINT | RFCNT[2:0] | | | RFF | RFE | RFFIE | RFEN | Reserved | | | | | | | ELNACK |

r_w0 r r r rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | RFFINT | Receive FIFO full interrupt flag |
| 14:12 | RFCNT[2:0] | Receive FIFO counter number |
| 11 | RFF | Receive FIFO full flag<br>0: Receive FIFO not full<br>1: Receive FIFO full |
| 10 | RFE | Receive FIFO empty flag<br>0: Receive FIFO not empty<br>1: Receive FIFO empty |
| 9 | RFFIE | Receive FIFO full interrupt enable<br>0: Receive FIFO full interrupt disable<br>1: Receive FIFO full interrupt enable |
| 8 | RFEN | Receive FIFO enable<br>This bit can be set when UESM = 1.<br>0: Receive FIFO disable<br>1: Receive FIFO enable |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | ELNACK | Early NACK when smartcard mode is selected.<br>The NACK pulse occurs 1/16 bit time earlier when the parity error is detected.<br>0:Early NACKdisable when smartcard mode is selected<br>1:Early NACKenable when smartcard mode is selected<br>This bit is reserved in USART1. |

# 17. Inter-integrated circuit interface (I2C)

## 17.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface.I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard mode, fast mode and fast mode plus as well as CRC calculation and checking, SMBus (system management bus), PMBus (power management bus) and SAM_V (secure access and control module for validation) mode. It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

## 17.2. Characteristics

- Parallel-bus to I2C-bus protocol converter and interface.
- Both master and slave functions with the same interface.
- Bi-directional data transfer between master and slave.
- Supports 7-bit and 10-bit addressing and General Call Addressing.
- Multi-master capability.
- Supports standard mode (up to 100 kHz), fast mode (up to 400 kHz) and fast mode plus (up to 1MHz).
- Configurable SCL stretching in slave mode.
- Supports DMA mode.
- SMBus 2.0 and PMBus compatible.
- 2 Interrupts: one for successful byte transmission and the other for error event.
- Optional PEC (Packet Error Checking) generation and check.
- Supports SAM_V mode.

## 17.3. Function overview

***Figure 17-1. I2C module block diagram*** below provides details of the internal configuration of the I2C interface.

**Figure 17-1. I2C module block diagram**

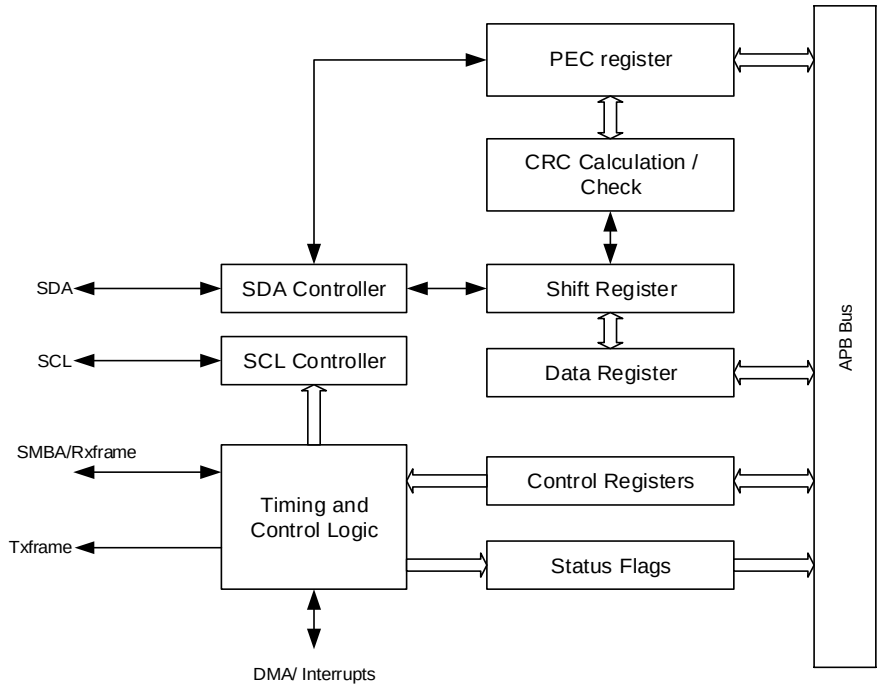



**Table 17-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)**

| Term | Description |
|---|---|
| Transmitter | The device which sends data to the bus |
| Receiver | The device which receives data from the bus |
| Master | The device which initiates a transfer, generates clock signals and terminates a transfer |
| Slave | The device addressed by a master |
| Multi-master | More than one master can attempt to control the bus at the same time without corrupting the message |
| Synchronization | Procedure to synchronize the clock signals of two or more devices |
| Arbitration | Procedure to ensure that, if more than one master tries to control the bus simultaneously, only one is allowed to do so and the winning master's message is not corrupted |

### 17.3.1. SDA and SCL lines

The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus.

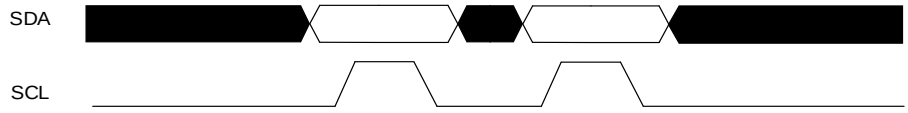
Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collect to perform the wired-AND function. Data on the I2C-bus can be transferred at rates of up to 100 Kbit/s in the standard mode, up to 400 Kbit/s in the fast mode and up to 1Mbit/s in the fast mode plus

if the FMPEN bit in I2C_FMPCFG is set. Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the voltage levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of $V_{DD}$.

### 17.3.2. Data validation

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the SDA line can only change when the clock signal on the SCL line is LOW (see ***Figure 17-2. Data validation***). One clock pulse is generated for each data bit to be transferred.

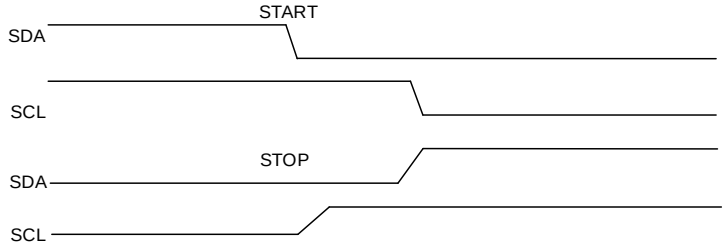
**Figure 17-2. Data validation**




### 17.3.3. START and STOP signal

All transmissions begin with a START and are terminated by a STOP (see ***Figure 17-3. START and STOP signal***). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START signal. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP signal.

**Figure 17-3. START and STOP signal**



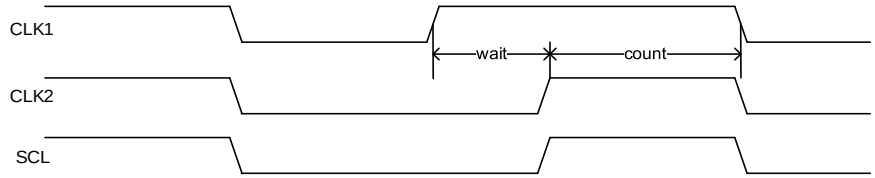

### 17.3.4. Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which master takes control of the bus and completes its transmission. This is done by clock synchronization and bus arbitration. In a single master system, clock synchronization and bus arbitration are unnecessary.

Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the masters concerned to start counting their LOW period, and once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see ***Figure 17-4. Clock synchronization***). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore

held LOW by the master with the longest LOW period. Masters with shorter LOW period enter a HIGH wait-state during this time.

**Figure 17-4. Clock synchronization**



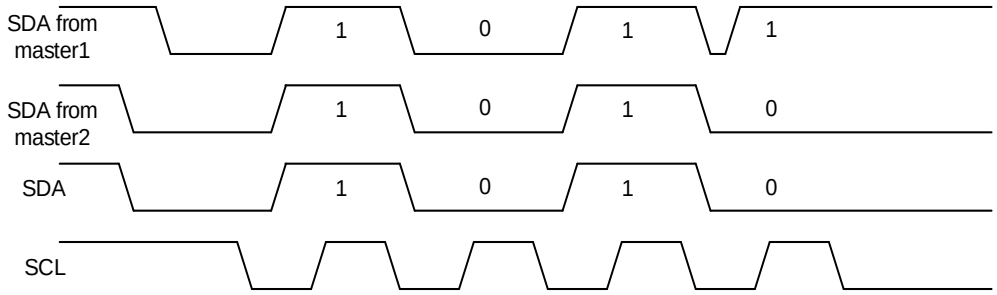

### 17.3.5. Arbitration

Arbitration, like synchronization, is part of the protocol where more than one master is used in the system. Slaves are not involved in the arbitration procedure.

A master may start a transfer only if the bus is free. Two masters may generate a START signal within the minimum hold time of the START signal which results in a valid START signal on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks whether the SDA level matches what it has been sent. This process may take many bits. Two masters can even complete an entire transmission without error, as long as the transmissions are identical. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, then the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transmission.

**Figure 17-5. SDA line arbitration**




### 17.3.6. I2C communication flow

Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can be operated as either a transmitter or receiver, depending on the function of the device.

An I2C slave will continue to detect addresses after a START signal on I2C bus and compare the detected address with its slave address which is programmed by software. Once the two addresses match with each other, the I2C slave will send an ACK to the I2C bus and respond to the following command on I2C bus: transmitting or receiving the desired data. Additionally,

if General Call is enabled by software, the I2C slave always responds to a General Call Address (0x00). The I2C block supports both 7-bit and 10-bit address modes.

An I2C master always initiates or ends a transfer using START or STOP signal and it's also responsible for SCL clock generation.

**Figure 17-6. I2C communication flow with 7-bit address**

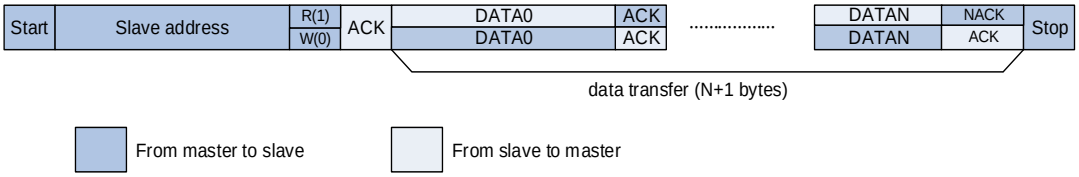



**Figure 17-7. I2C communication flow with 10-bit address (Master Transmit)**

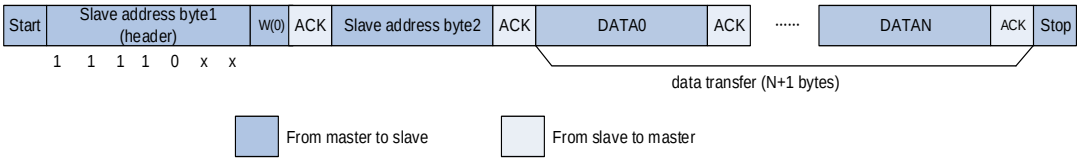
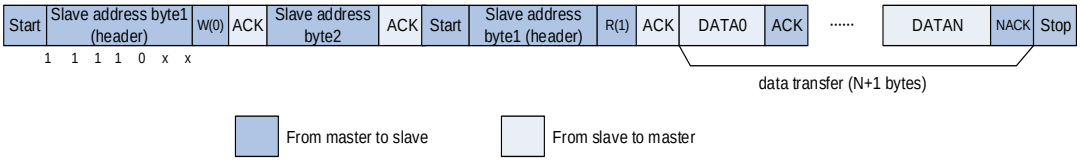



**Figure 17-8. I2C communication flow with 10-bit address (Master Receive)**




## 17.3.7. Programming model

An I2C device such as LCD driver may only be a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered as a slave.

An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- Master Transmitter
- Master Receiver
- Slave Transmitter
- Slave Receiver

I2C block supports all of the four I2C modes. After system reset, it works in slave mode. After sending a START signal on I2C bus, it changes into master mode. The I2C changes back to slave mode after sending a STOP signal on I2C bus.

### Programming model in slave transmitting mode

As is shown in ***Figure 17-9. Programming model for slave transmitting (10-bit address***

***mode)***, the following software procedure should be followed if users wish to transmit data in slave transmitter mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.

2. After receiving a START signal followed by a matched address, either in 7-bit format or in 10-bit format, the I2C hardware sets the ADDSEND bit in I2C_STAT0 register, which should be monitored by software either by polling or interrupt. After that, software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. If 10-bit addressing format is selected, the I2C master should then send a repeated START signal followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START signal and the following header. The ADDSEND bit must be cleared by software again by reading I2C_STAT0 and then I2C_STAT1.

3. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Once TBE is set, software should write the first byte of data to I2C_DATA register, TBE is not cleared in this case because the byte written in I2C_DATA is moved to the internal shift register immediately. I2C begins to transmit data to I2C bus as soon as the shift register is not empty.

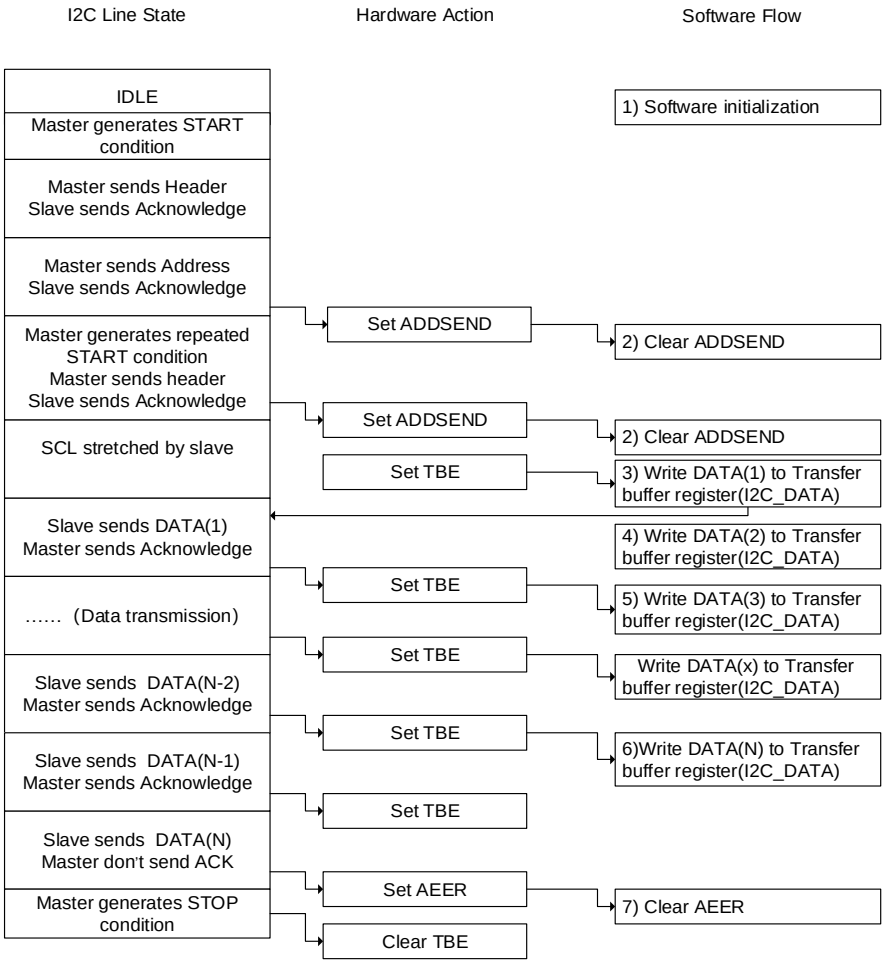
4. During the transmission of the first byte, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

5. After the transmission of the first byte, the TBE bit will be set, the software can write the third byte to the I2C_DATA register and TBE is cleared. After this, any time TBE is set, software can write a byte to I2C_DATA as long as there is still data to be transmitted.

6. During the transmission of the second last byte, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be set after the byte's transmission and not cleared until a STOP signal.

7. I2C master doesn't acknowledge to the last byte according to the I2C protocol, so after sending the last byte, I2C slave will wait for the STOP signal on I2C bus and sets AERR (Acknowledge Error) bit to notify software that the transmission completes. Software clears AERR bit by writing 0 to it.

**Figure 17-9. Programming model for slave transmitting (10-bit address mode)**




## Programming model in slave receiving mode

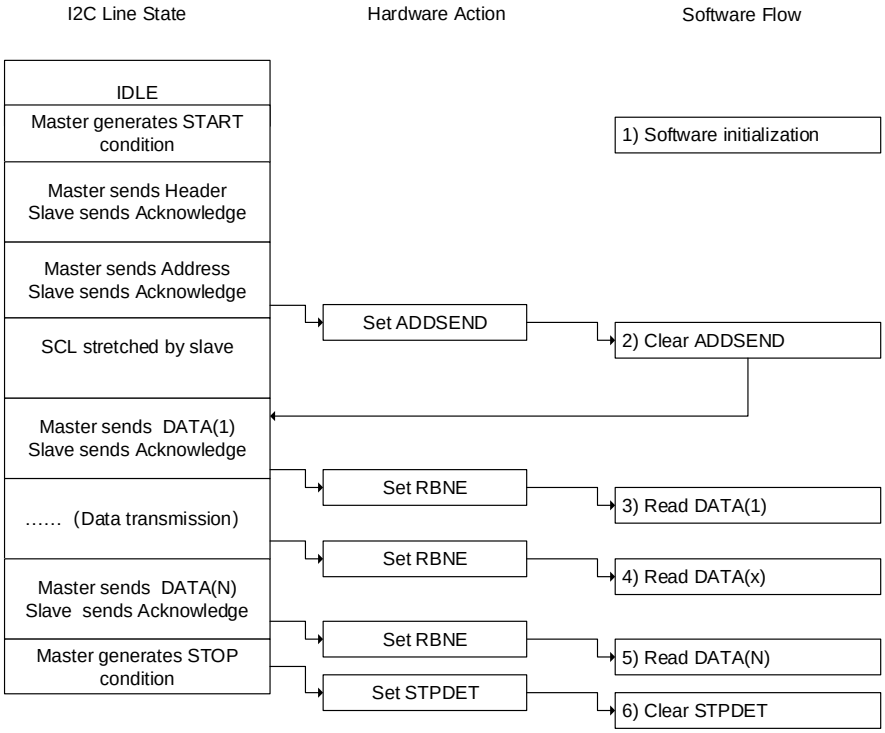
As is shown in ***Figure 17-10. Programming model for slave receiving (10-bit address mode)***, the following software procedure should be followed if users wish to receive data in slave receiver mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. After receiving a START signal followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register 0, which should be monitored by software either by polling or interrupt. After that software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. The I2C begins to receive data on I2C bus as soon as ADDSEND bit is cleared.
3. As soon as the first byte is received, RBNE is set by hardware. Software can now read the first byte from I2C_DATA and RBNE is cleared as well.
4. Any time RBNE is set, software can read a byte from I2C_DATA.

5. After the last byte is received, RBNE is set. Software reads the last byte.

6. STPDET bit is set when I2C detects a STOP signal on I2C bus and software reads I2C_STAT0 and then writes I2C_CTL0 to clear the STPDET bit.

**Figure 17-10. Programming model for slave receiving (10-bit address mode)**




## Programming model in master transmitting mode

As is shown in ***Figure 17-11. Programming model for master transmitting (10-bit address mode)***, the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.

2. Software sets START bit requesting I2C to generate a START signal on I2C bus.

3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending the header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.

4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then

I2C_STAT1.

5. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Software now writes the first byte data to I2C_DATA register, but the TBE will not be cleared because the byte written in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as the shift register is not empty.

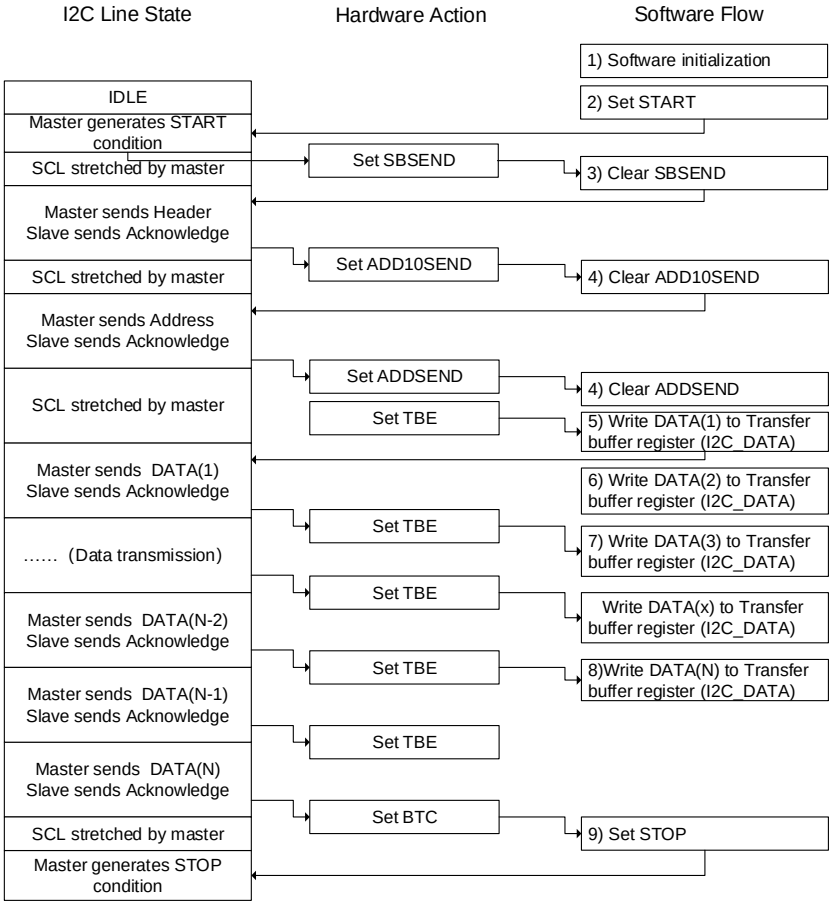
6. During the transmission of the first byte, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

7. Any time TBE is set, software can write a byte to I2C_DATA as long as there is still data to be transmitted.

8. During the transmission of the second last byte, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the transmission of the byte and not be cleared until a STOP signal.

9. After sending the last byte, I2C master sets BTC bit because both the shift register and I2C_DATA are empty. Software should set the STOP bit to generate a STOP signal, then the I2C clears both TBE and BTC flags.

**Figure 17-11. Programming model for master transmitting (10-bit address mode)**



### Programming model in master receiving mode

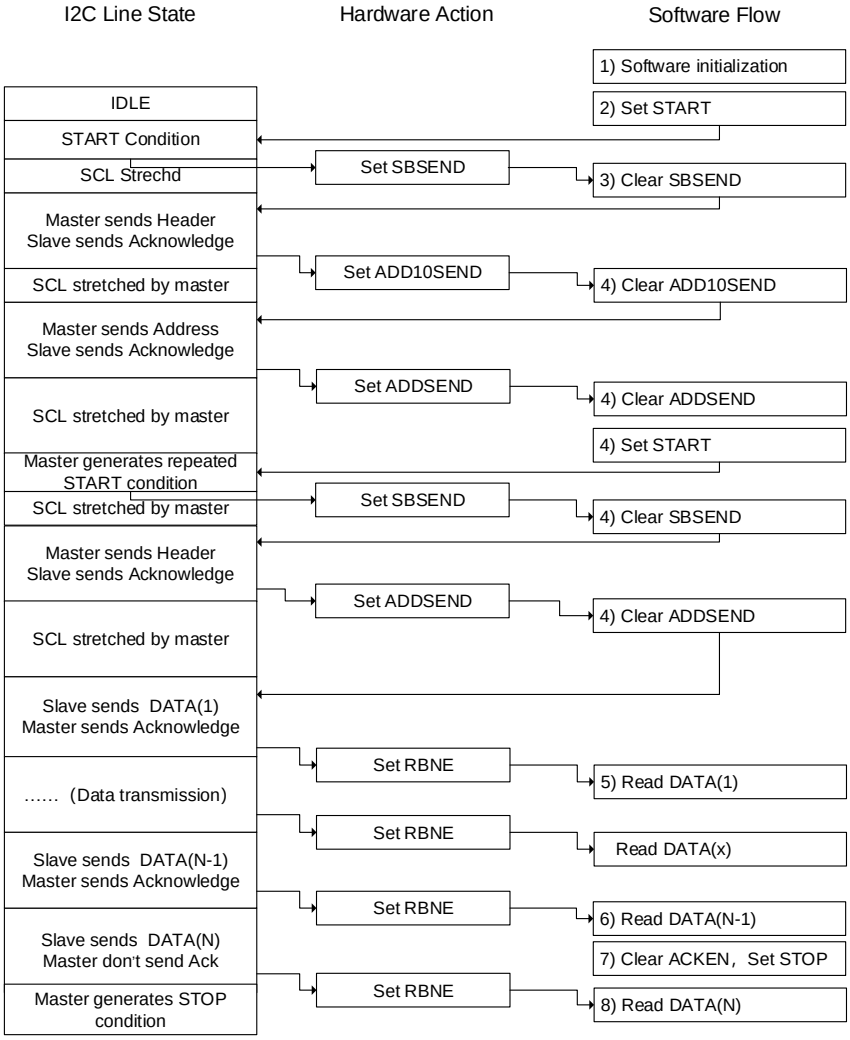
In master receiving mode, a master is responsible for generating NACK for the last byte reception and then sending a STOP signal on I2C bus. So, special attention should be paid to ensure the correct ending of data reception. Two solutions for master receiving are provided here for applications: Solution A and B. Solution A requires the software's quick response to I2C events, while Solution B doesn't.

**Solution A**

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.
4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START signal on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header to I2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C_DATA.
7. After the second last byte (N-1) is received, the software should clear ACKEN bit and set STOP bit. These actions should complete before the end of the last byte's receiving to ensure that NACK will be sent for the last byte.
8. After the last byte is received, RBNE is set. Software reads the last byte. Since ACKEN has been cleared in the previous step, I2C doesn't send ACK for the last byte and it generates a STOP signal after the transmission of the last byte.

The above steps require byte number N>1. If N=1, Step 7 should be performed after Step 4 and completed before the end of the single byte's receiving.

9. **Figure 17-12. Programming model for master receiving using Solution A (10-bit address mode)**




**Solution B**

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.

2. Software sets START bit requesting I2C to generate a START signal on I2C bus.

3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.

4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit

and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START signal on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header to I2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.

5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C_DATA until the master receives N-3 bytes.

As shown in ***Figure 17-13. Programming model for master receiving mode using solution B (10-bit address mode)***, the N-2 byte is not read out by software, so after the N-1 byte is received, both BTC and RBNE are asserted. The bus is stretched by master to prevent the reception of the last byte. Then software should clear ACKEN bit.

7. Software reads out N-2 byte, clearing BTC. After this, the N-1 byte is moved from shift register to I2C_DATA and bus is released and begins to receive the last byte. Master doesn't send an ACK for the last byte because ACKEN is already cleared.
8. After the last byte is received, both BTC and RBNE are set again, and SCL is stretched low. Software sets STOP bit and master sends out a STOP signal on bus.
9. Software reads the N-1 byte, clearing BTC. After this the last byte is moved from shift register to I2C_DATA.
10. Software reads the last byte, clearing RBNE.

The above steps require that byte number N>2. N=1 and N=2 are similar:

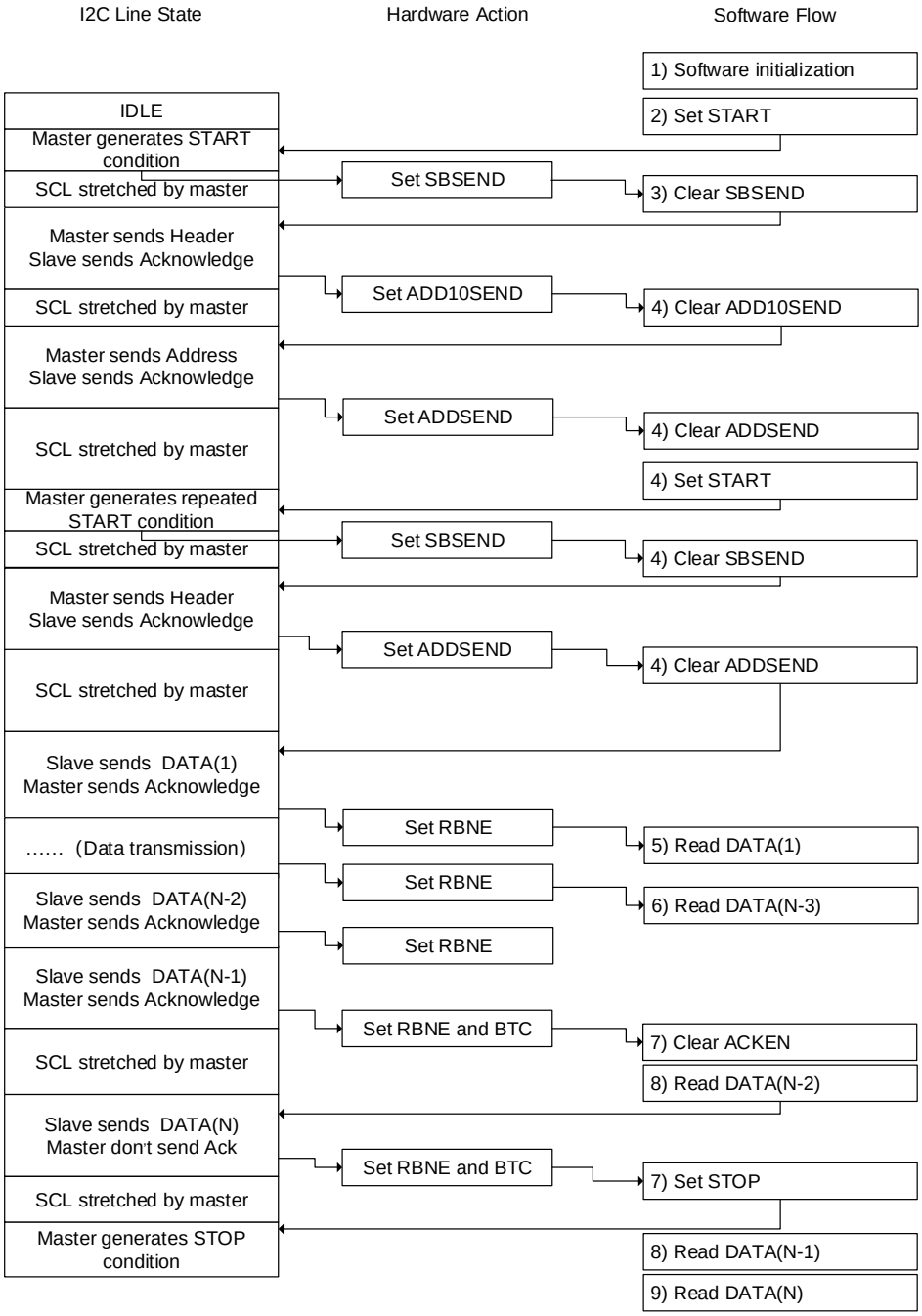
**N=1**

In Step4, software should reset ACKEN bit before clearing ADDSEND bit and set STOP bit after clearing ADDSEND bit. Step 5 is the last step when N=1.

**N=2**

In Step 2, software should set POAP bit before setting START bit. In Step 4, software should reset ACKEN bit before clearing ADDSEND bit. In Step 5, software should wait until BTC is set and then set STOP bit and read I2C_DATA twice.

**Figure 17-13. Programming model for master receiving mode using solution B (10-bit address mode)**




## 17.3.8. SCL line stretching

The SCL line stretching function is designed to avoid overflow error in reception and underflow error in transmission. As is shown in Programming Model, when the TBE and BTC bits are set in transmitting mode, the transmitter stretches the SCL line low until the transfer buffer register is filled with the next data to be transmitted. When the RBNE and BTC bits are set in receiving mode, the receiver stretches the SCL line low until the data in the transfer buffer is read out.

When works in slave mode, the SCL line stretching function can be disabled by setting the SS bit in the I2C_CTL0 register. If this bit is set, the software is required to be quick enough to serve the TBE, RBNE and BTC status, otherwise, overflow or underflow situation might occur.

### 17.3.9. Use DMA for data transfer

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU to be high overloaded. The DMA controller can be used to process TBE and RBNE flags: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically. It reduces the load on the CPU. See the DMA section for details on how to configure DMA.

The DMA request is enabled by the DMAON bit in the I2C_CTL1 register. This bit should be set after clearing the ADDSEND status. If the SCL line stretching function is disabled for a slave device, the DMAON bit should be set before the ADDSEND event.

Refer to the specification of the DMA controller for the configuration method of a DMA stream. The DMA controller must be configured and enabled before the I2C transfer. When the configured number of bytes have been transferred, the DMA controller generates End of Transfer (EOT) interrupt. DMA will send an End of Transmission (EOT) signal to the I2C interface and generates a DMA full transfer finish interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C_CTL1 register should be set. The I2C master will send NACK after the last byte. The STOP bit can be set by software to generate a STOP signal in the ISR of the DMA full transfer finish interrupt.

When a master receives only one byte, the ACKEN bit must be cleared before clearing the ADDSEND status. Software can set the STOP bit to generate a STOP signal after clearing the ADDSEND status, or in the ISR of the DMA full transfer finish interrupt.

### 17.3.10. Packet error checking

There is a CRC-8 calculator in I2C block to perform PEC (Packet Error Checking) for I2C data. The polynomial of the CRC is x8 + x2 + x + 1 which is compatible with the SMBus protocol. If enabled by setting PECEN bit, the PEC will calculate all the data transmitted through I2C including address. I2C is able to send out the PEC value after the last data byte or check the received PEC value with its calculated PEC using the PECTRANS bit. In DMA mode, the I2C will send or check PEC value automatically if PECEN bit and PECTRANS bit are set.

### 17.3.11. SMBus support

The System Management Bus (abbreviated to SMBus or SMB) is a single-ended simple two-wire bus for the purpose of lightweight communication. Most commonly it is found in computer motherboards for communication with power source for ON / OFF instructions. It is

derived from I2C for communication with low-bandwidth devices on a motherboard, especially power related chips such as a laptop's rechargeable battery subsystem (see Smart Battery Data).

### SMBus protocol

Each message transmission on SMBus follows the format of one of the defined SMBus protocols. The SMBus protocols are a subset of the data transfer formats defined in the I2C specifications. I2C devices that can be accessed through one of the SMBus protocols are compatible with the SMBus specifications. I2C devices that do not adhere to these protocols cannot be accessed by standard methods as defined in the SMBus and Advanced Configuration and Power Management Interface (abbreviated to ACPI) specifications.

### Address resolution protocol

The SMBus is realized based on I2C hardware and it uses I2C hardware addressing, but it adds the second-level software for building special systems. Additionally, its specifications include an Address Resolution Protocol that can make dynamic address allocations. Dynamic reconfiguration of the hardware and software allows bus devices to be ‘hot-plugged’ and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In this protocol there is a very useful distinction between a system host and all the other devices in the system, that is the host provides address assignment function.

### Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency is 10 kHz to prevent locking up the bus. I2C can be a ‘DC’ bus, which means that a slave device stretches the master clock when performing some routines while the master is accessing it. This will notify the master that the slave is busy but does not want to lose the communication. The slave device will continue the communication after its task is completed. There is no limit in the I2C bus protocol of how long this delay can be, whereas for a SMBus system, it would be limited to 25ms. SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to solve the problem. Slave devices are not allowed to hold the clock low too long.

### Packet error checking

SMBus 2.0 and 1.1 allow Packet Error Checking (PEC). In that mode, a PEC byte is appended at the end of each transaction. The byte is a CRC-8 checksum of the entire message including the address and read/write bit. The polynomial used is x8+x2+x+1 (the CRC-8-ATM HEC algorithm, initialized to zero).

### SMBus alert

The SMBus has an extra optional shared interrupt signal called SMBALERT# which can be used by slaves to tell the host to ask its slaves about events of interest. SMBus also defines a less common "Host Notify Protocol", providing similar notifications which is based on the I2C multi-master mode but it can pass more data.

### SMBus programming flow

The programming flow for SMBus is similar to normal I2C. In order to use SMBus mode, the application should configure several SMBus specific registers, respond to some SMBus specific flags and implement the upper protocols described in SMBus specification.

1. Before communication, SMBEN bit in I2C_CTL0 should be set and SMBSEL and ARPEN bits should be configured to desired values.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should respond to HSTSMB flag in SMBus Host Mode (SMBSEL =1) or DEFSMB flag in SMBus Device Mode, and implement the function of ARP protocol.
3. In order to support SMBus Alert Mode, the software should respond to SMBALT flag and implement the related function.

## 17.3.12. SAM_V support

To support the SAM_V standard, two additional pins are added to the I2C module: txframe and rxframe. Txframe is an output pin, in master mode, it indicates the I2C is busy when it is asserted. Rxframe is an input pin that is supposed to be multiplexed together with the SMBALERT signal.

The SAM_V mode is enabled by setting the SAMEN bit of the I2C_SAMCS register. The status of the txframe and rxframe pin can be reflected by the RFR, RFF, TFR, TFF, RXF, and TXF flags of the I2C_SAMCS register. I2C interrupts will be generated if the corresponding interrupt enable bits are set.

## 17.3.13. Status, errors and interrupts

There are several status and error flags in I2C, and interrupts may be asserted from these flags by setting some register bits (refer to ***Register definition*** for detail).

**Table 17-2. Event status flags**

| Event Flag Name | Description |
|---|---|
| SBSEND | START signal sent (master) |
| ADDSEND | Address sent or received |
| ADD10SEND | Header of 10-bit address sent |
| STPDET | STOP signal detected |

| Event Flag Name | Description |
|---|---|
| BTC | Byte transmission completed |
| TBE | I2C_DATA is empty when transmitting |
| RBNE | I2C_DATA is not empty when receiving |
| RFR | SAM_V mode rxframe pin rising edge is detected |
| RFF | SAM_V mode rxframe pin falling edge is detected |
| TFR | SAM_V mode txframe pin rising edge is detected |
| TFF | SAM_V mode txframe pin falling edge is detected |

**Table 17-3. Error flags**

| Error Name | Description |
|---|---|
| BERR | Bus error |
| LOSTARB | Arbitration lost |
| OUERR | Over-run or under-run when SCL stretch is disabled. |
| AERR | No acknowledge received |
| PECERR | CRC value doesn’t match |
| SMBTO | Bus timeout in SMBus mode |
| SMBALT | SMBus Alert |

## 17.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

### 17.4.1. Control register 0 (I2C_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRESET | Reserved | SALT | PECTRANS | POAP | ACKEN | STOP | START | SS | GCEN | PECEN | ARPEN | SMBSEL | Reserved | SMBEN | I2CEN |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | SRESET | Software resets I2C, software should wait until the I2C lines are released to reset the I2C.<br>0: I2C is not under reset<br>1: I2C is under reset |
| 14 | Reserved | Must be kept at reset value. |
| 13 | SALT | SMBus Alert.<br>Issue alert through SMBA pin.<br>Software can set and clear this bit and hardware can clear this bit.<br>0: Don’t issue alert through SMBA pin<br>1: Issue alert through SMBA pin |
| 12 | PECTRANS | PEC transfer<br>Software sets and clears this bit while hardware clears this bit when PEC is transferred or START / STOP signal is detected or I2CEN=0.<br>0: Don’t transfer PEC value<br>1: Transfer PEC value |
| 11 | POAP | Position of ACK and PEC when receiving<br>This bit is set and cleared by software and cleared by hardware when I2CEN=0.<br>0: ACKEN bit specifies whether to send ACK or NACK for the current byte that is being received. PECTRANS bit indicates that the current receiving byte is a PEC |

| | | |
|---|---|---|
| | | byte.<br>1: ACKEN bit specifies whether to send ACK or NACK for the next byte that is to be received, PECTRANS bit indicates the next byte that is to be received is a PEC byte. |
| 10 | ACKEN | Whether or not to send an ACK<br>This bit is set and cleared by software and cleared by hardware when I2CEN=0.<br>0: ACK will not be sent<br>1: ACK will be sent |
| 9 | STOP | Generate a STOP signal on I2C bus<br>This bit is set and cleared by software and set by hardware when SMBus timeout and cleared by hardware when STOP signal is detected.<br>0: STOP will not be sent<br>1: STOP will be sent |
| 8 | START | Generate a START signal on I2C bus<br>This bit is set and cleared by software and cleared by hardware when a START signal is detected or I2CEN=0.<br>0: START will not be sent<br>1: START will be sent |
| 7 | SS | Whether to stretch SCL low when data is not ready in slave mode.<br>This bit is set and cleared by software.<br>0: SCL stretching is enabled<br>1: SCL stretching is disabled |
| 6 | GCEN | Whether or not to response to a General Call (0x00)<br>0: Slave won't respond to a General Call<br>1: Slave will respond to a General Call |
| 5 | PECEN | PEC calculation enable<br>0: PEC calculation disable<br>1: PEC calculation enable |
| 4 | ARPEN | ARP protocol enable in SMBus mode<br>0: ARP is disabled<br>1: ARP is enabled |
| 3 | SMBSEL | SMBus type selection<br>0: Device<br>1: Host |
| 2 | Reserved | Must be kept at reset value. |
| 1 | SMBEN | SMBus / I2C mode switch<br>0: I2C mode<br>1: SMBus mode |

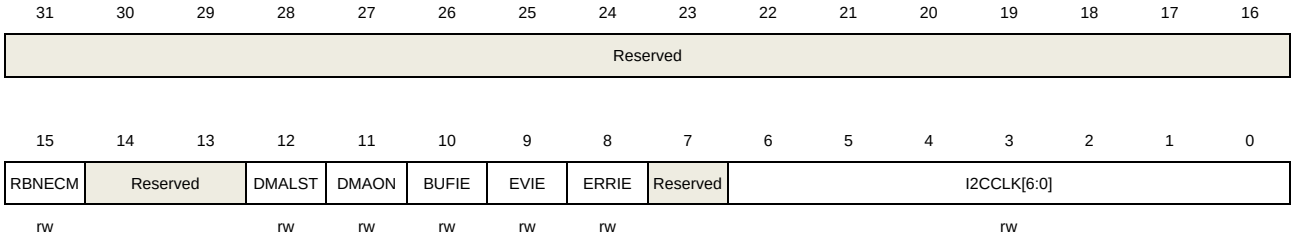
| | | |
|---|---|---|
| 0 | I2CEN | I2C peripheral enable<br>0: I2C is disabled<br>1: I2C is enabled |

### 17.4.2. Control register 1 (I2C_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).



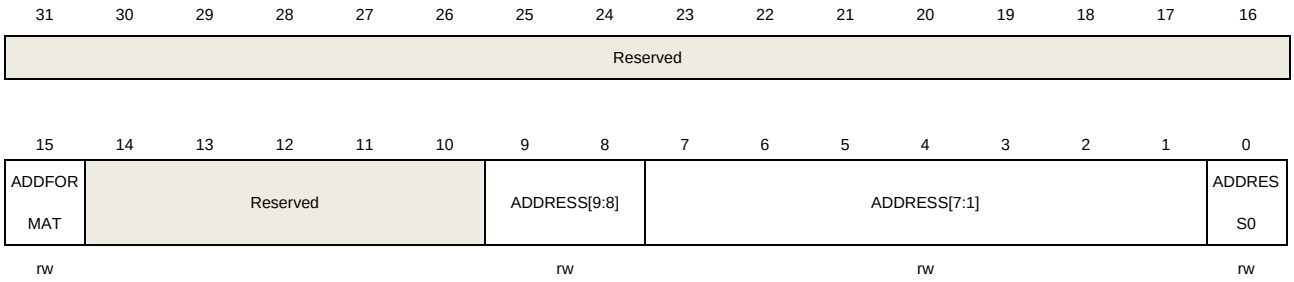

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | RBNECM | RBNE clear mode configure<br>0: RBNE can be cleared when I2C_DATA is read and BTC is 0<br>1: RBNE can be cleared when I2C_DATA is read |
| 14:13 | Reserved | Must be kept at reset value. |
| 12 | DMALST | DMA last transfer configure<br>0: Next DMA EOT is not the last transfer<br>1: Next DMA EOT is the last transfer |
| 11 | DMAON | DMA mode switch<br>0: DMA mode switched off<br>1: DMA mode switched on |
| 10 | BUFIE | Buffer interrupt enable<br>0: Buffer interrupt is disabled, TBE = 1 or RBNE = 1 when EVIE=1 will not generate an interrupt.<br>1: Buffer interrupt is enabled, which means that interrupt will be generated when TBE = 1 or RBNE = 1 if EVIE=1. |
| 9 | EVIE | Event interrupt enable<br>0: Event interrupt is disabled<br>1: Event interrupt is enabled, which means that interrupt will be generated when SBSEND, ADDSEND, ADD10SEND, STPDET or BTC flag asserted or TBE=1 or RBNE=1 if BUFIE=1. |
| 8 | ERRIE | Error interrupt enable |

| | | |
|---|---|---|
| | | 0: Error interrupt is disabled<br>1: Error interrupt is enabled, which means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag is asserted. |
| 7 | Reserved | Must be kept at reset value. |
| 6:0 | I2CCLK[6:0] | I2C peripheral clock frequency<br>I2CCLK[6:0]should be the frequency of input APB1 clock in MHz which is at least 2.<br>0000000 - 0000001: Not allowed<br>0000010 - 1001000: 2 MHz~72MHz<br>1001001 - 1111111: Not allowed due to the limitation of APB1 clock<br>**Note:**<br>In I2C standard mode, the frequencies of APB1 must be equal or greater than 2MHz. In I2C fast mode, the frequencies of APB1 must be equal or greater than 8MHz. In I2C fast mode plus, the frequencies of APB1 must be equal or greater than 24MHz. |

### 17.4.3. Slave address register 0 (I2C_SADDR0)

Address offset: 0x08
Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).



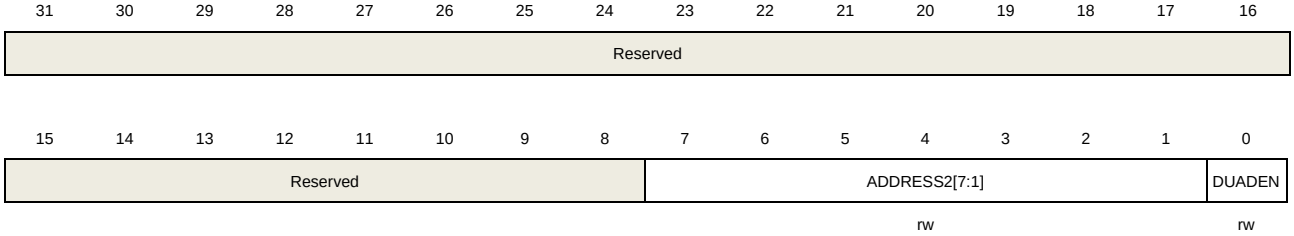

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | ADDFORMAT | Address format for the I2C slave<br>0: 7-bit address<br>1: 10-bit address |
| 14:10 | Reserved | Must be kept at reset value. |
| 9:8 | ADDRESS[9:8] | Highest two bits of a 10-bit address |
| 7:1 | ADDRESS[7:1] | 7-bit address or bits 7:1 of a 10-bit address |
| 0 | ADDRESS0 | Bit 0 of a 10-bit address |

### 17.4.4. Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).



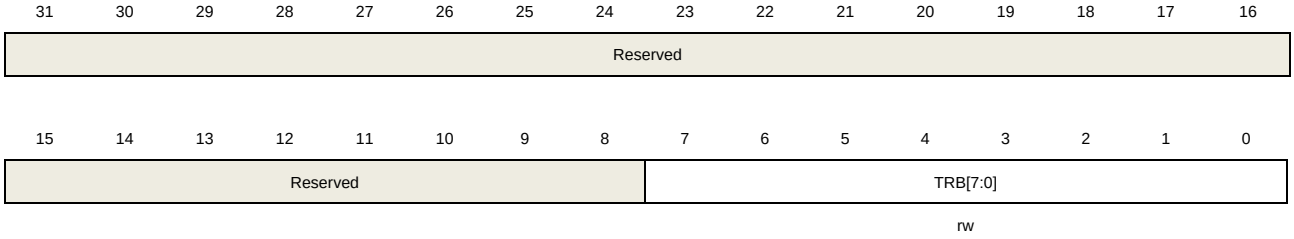

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:1 | ADDRESS2[7:1] | The second I2C address for the slave in Dual-Address mode |
| 0 | DUADEN | Dual-Address mode enable<br>0: Dual-Address mode is disabled<br>1: Dual-Address mode is enabled |

### 17.4.5. Transfer buffer register (I2C_DATA)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | TRB[7:0] | Transmission or reception data buffer |

### 17.4.6. Transfer status register 0 (I2C_STAT0)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMBALT | SMBTO | Reserved | PECERR | OUERR | AERR | LOSTARB | BERR | TBE | RBNE | Reserved | STPDET | ADD10SEND | BTC | ADDSEND | SBSEND |
| rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | r | r | | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | SMBALT | SMBus Alert status<br>This bit is set by hardware and cleared by writing 0.<br>0: SMBA pin not pulled down (device mode) or no Alert detected (host mode)<br>1: SMBA pin pulled down and Alert address received (device mode) or Alert detected (host mode) |
| 14 | SMBTO | Timeout signal in SMBus mode<br>This bit is set by hardware and cleared by writing 0.<br>0: No timeout error<br>1: Timeout event occurs (SCL is low for 25 ms) |
| 13 | Reserved | Must be kept at reset value. |
| 12 | PECERR | PEC error when receiving data<br>This bit is set by hardware and cleared by writing 0.<br>0: Received PEC matches the calculated PEC<br>1: Received PEC doesn’t match the calculated PEC, I2C will send NACK careless of ACKEN bit. |
| 11 | OUERR | Over-run or under-run situation occurs in slave mode, when SCL stretching is disabled. In slave receiving mode, if the last byte in I2C_DATA is not read out while the following byte is already received, over-run occurs. In slave transmitting mode, if the current byte is already sent out, while the I2C_DATA is still empty, under-run occurs.<br>This bit is set by hardware and cleared by writing 0.<br>0: No over-run or under-run occurs.<br>1: Over-run or under-run occurs. |
| 10 | AERR | Acknowledge error<br>This bit is set by hardware and cleared by writing 0.<br>0: No acknowledge error<br>1: Acknowledge error |
| 9 | LOSTARB | Arbitration lost in master mode<br>This bit is set by hardware and cleared by writing 0.<br>0: No arbitration lost<br>1: Arbitration lost occurs and the I2C block changes back to slave mode. |

| | | |
|---|---|---|
| 8 | BERR | Bus error<br>A bus error occurs when an unexpected START or STOP signal on I2C bus.<br>This bit is set by hardware and cleared by writing 0.<br>0: No bus error<br>1: A bus error detected |
| 7 | TBE | I2C_DATA is empty during transmitting<br>This bit is set by hardware after it moves a byte from I2C_DATA to shift register and cleared by writing a byte to I2C_DATA. If both the shift register and I2C_DATA are empty, writing I2C_DATA won't clear TBE (refer to Programming Model for detail).<br>0: I2C_DATA is not empty<br>1: I2C_DATA is empty, software can write |
| 6 | RBNE | I2C_DATA is not empty during receiving<br>This bit is set by hardware after it moves a byte from shift register to I2C_DATA and cleared by reading I2C_DATA. If both BTC and RBNE are asserted, reading I2C_DATA won't clear RBNE because the byte in shift register will be moved to I2C_DATA immediately.<br>0: I2C_DATA is empty<br>1: I2C_DATA is not empty, software can read |
| 5 | Reserved | Must be kept at reset value. |
| 4 | STPDET | STOP signal is detected in slave mode<br>This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0.<br>0: STOP signal not detected in slave mode<br>1: STOP signal detected in slave mode |
| 3 | ADD10SEND | Header of 10-bit address is sent in master mode<br>This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA.<br>0: No header of 10-bit address is sent in master mode<br>1: Header of 10-bit address is sent in master mode |
| 2 | BTC | Byte transmission is completed.<br>If a byte is already received in shift register but I2C_DATA is still full in receiving mode or a byte is already sent out from shift register but I2C_DATA is still empty in transmitting mode, the BTC flag is asserted if SCL stretching enabled.<br>This bit is set by hardware and cleared by 3 ways as follow:<br>1. Software clearing: reading I2C_STAT0 followed by reading or writing I2C_DATA<br>2. Hardware clearing: sending the STOP signal or START signal<br>3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset.<br>0: BTC not asserted<br>1: BTC asserted |

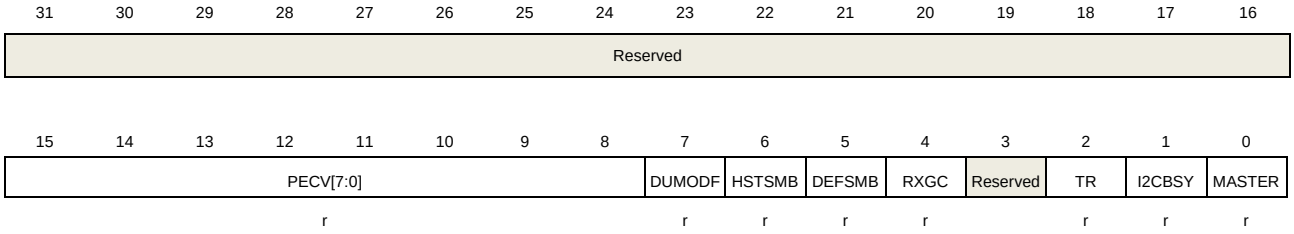
| | | |
|---|---|---|
| 1 | ADDSEND | Address is sent and ACK is received in master mode or address is received and matches with its own address in slave mode.<br>This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1.<br>0: In slave mode, no address is received or the received address does not match witih its own address. In master mode, no address is sent or address has been sent but not received the ACK from slave.<br>1: In slave mode, address is received and matches witih its own address. In master mode, address has been sent and receives the ACK from slave. |
| 0 | SBSEND | START signal is sent out in master mode<br>This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA.<br>0: No START signal sent<br>1: START signal sent |

### 17.4.7. Transfer status register 1 (I2C_STAT1)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | PECV[7:0] | Packet Error Checking value that calculated by hardware when PEC is enabled. |
| 7 | DUMODF | Dual flag in slave mode indicates which address matches with the address in Dual-Address mode<br>This bit is cleared by hardware after a STOP or a START signal or I2CEN=0<br>0: The address matches with SADDR0 address<br>1: The address matches with SADDR1 address |
| 6 | HSTSMB | SMBus host header detected in slave mode<br>This bit is cleared by hardware after a STOP or a START signal or I2CEN=0<br>0: No SMBus host header is detected<br>1: SMBus host header is detected |
| 5 | DEFSMB | Default address of SMBus device |

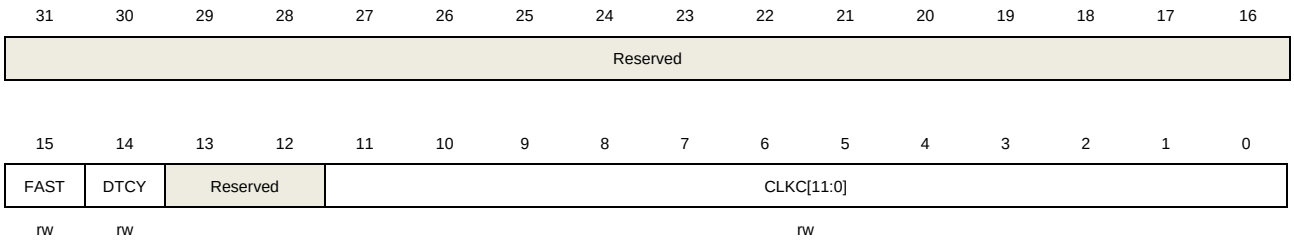
| | | |
|---|---|---|
| | | This bit is cleared by hardware after a STOP or a START signal or I2CEN=0.<br>0: The default address has not been received for SMBus device<br>1: The default address has been received for SMBus device |
| 4 | RXGC | General call address (0x00) received.<br>This bit is cleared by hardware after a STOP or a START signal or I2CEN=0.<br>0: No general call address received<br>1: General call address received |
| 3 | Reserved | Must be kept at reset value. |
| 2 | TR | Transmitter or receiver<br>This bit indicates whether the I2C is a transmitter or a receiver. It is cleared by hardware after a STOP or a START signal or I2CEN=0 or LOSTARB=1.<br>0: Receiver<br>1: Transmitter |
| 1 | I2CBSY | Busy flag<br>This bit is cleared by hardware after a STOP signal<br>0: No I2C communication.<br>1: I2C communication active. |
| 0 | MASTER | A flag indicating whether I2C block is in master or slave mode.<br>This bit is set by hardware when a START signal generates.<br>This bit is cleared by hardware after a STOP signal or I2CEN=0 or LOSTARB=1.<br>0: Slave mode<br>1: Master mode |

### 17.4.8. Clock configure register (I2C_CKCFG)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | FAST | I2C speed selection in master mode<br>0: Standard speed |

1: Fast speed

| | | |
|---|---|---|
| 14 | DTCY | Duty cycle in fast mode or fast mode plus<br>0: $T_{low}/T_{high}=2$<br>1: $T_{low}/T_{high}=16/9$ |
| 13:12 | Reserved | Must be kept at reset value. |
| 11:0 | CLKC[11:0] | I2C clock control in master mode<br>In standard speed mode: $T_{high}=T_{low}=CLKC*T_{PCLK1}$<br>In fast speed mode or fast mode plus, if DTCY=0:<br>$T_{high}=CLKC*T_{PCLK1}$，$T_{low}=2*CLKC*T_{PCLK1}$<br>In fast speed mode or fast mode plus, if DTCY=1:<br>$T_{high}=9*CLKC*T_{PCLK1}$，$T_{low}=16*CLKC*T_{PCLK1}$<br>**Note:** If DTCY is 0, when PCLK1 is an integral multiple of 3, the baud rate will be more accurate. If DTCY is 1, when PCLK1 is an integral multiple of 25, the baud rate will be more accurate. |

## 17.4.9. Rise time register (I2C_RT)

Address offset: 0x20

Reset value: 0x0000 0002

This register can be accessed by half-word (16-bit) or word (32-bit).

| 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 | | | | | | | | | RISETIME[6:0] | | | | | | |
| | | | | | | | | | rw | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6:0 | RISETIME[6:0] | Maximum rise time in master mode<br>The RISETIME value should be the maximum SCL rise time incremented by 1. |

## 17.4.10. SAM control and status register (I2C_SAMCS)

Address offset: 0x80

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFR | RFF | TFR | TFF | Reserved | | RXF | TXF | RFRIE | RFFIE | TFRIE | TFFIE | Reserved | | STOEN | SAMEN |
| rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | r | r | rw | rw | rw | rw | | | rw | rw |

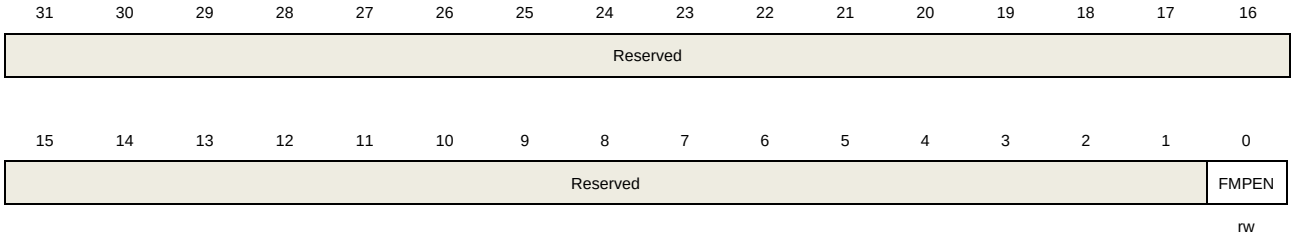
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | RFR | Rxframe rise flag, cleared by software by writing 0 |
| 14 | RFF | Rxframe fall flag, cleared by software by writing 0 |
| 13 | TFR | Txframe rise flag, cleared by software by writing 0 |
| 12 | TFF | Txframe fall flag, cleared by software by writing 0 |
| 11:10 | Reserved | Must be kept at reset value. |
| 9 | RXF | Level of rxframe signal |
| 8 | TXF | Level of txframe signal |
| 7 | RFRIE | Rxframe rise interrupt enable<br>0: Rxframe rise interrupt disabled<br>1: Rxframe rise interrupt enabled |
| 6 | RFFIE | Rxframe fall interrupt enable<br>0: Rxframe fall interrupt disabled<br>1: Rxframe fall interrupt enabled |
| 5 | TFRIE | Txframe rise interrupt enable<br>0: Txframe rise interrupt disabled<br>1: Txframe rise interrupt enabled |
| 4 | TFFIE | Txframe fall interrupt enable<br>0: Txframe fall interrupt disabled<br>1: Txframe fall interrupt enabled |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | STOEN | SAM_V interface timeout detect enable<br>0: SAM_V interface timeout detect disabled<br>1: SAM_V interface timeout detect enabled |
| 0 | SAMEN | SAM_V interface enable<br>0: SAM_V interface disabled<br>1: SAM_V interface enabled |

### 17.4.11. Fast mode plus configure register (I2C_FMPCFG)

Address offset: 0x90

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | FMPEN | Fast mode plus enable.<br>The I2C device supports up to 1MHz when this bit is set. |

# 18. Serial peripheral interface/Inter-IC sound (SPI/I2S)

## 18.1. Overview

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol.

The Serial Peripheral Interface (SPI) provides a SPI protocol of data transmission and reception function in master or slave mode. Both full-duplex and simplex communication modes are supported, with hardware CRC calculation and checking. Quad-SPI master mode is also supported in SPI1.

The Inter-IC sound (I2S) supports four audio standards: I2S Philips standard, MSB justified standard, LSB justified standard, and PCM standard. I2S works at either master or slave mode for transmission and reception.

## 18.2. Characteristics

### 18.2.1. SPI characteristics

- Master or slave operation with full-duplex or half-duplex or simplex mode.
- Separate transmission and reception buffer, 16 bits wide (only in SPI0).
- Separate transmission and reception 32-bit FIFO (only in SPI1).
- Data frame size can be 8 or 16 bits (only in SPI0).
- Data frame size can be 4 to 16 bits (only in SPI1).
- Bit order can be LSB or MSB.
- Software and hardware NSS management.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.
- SPI TI mode supported.
- SPI NSS pulse mode supported.
- Quad-SPI configuration available in master mode (only in SPI1).

### 18.2.2. I2S characteristics

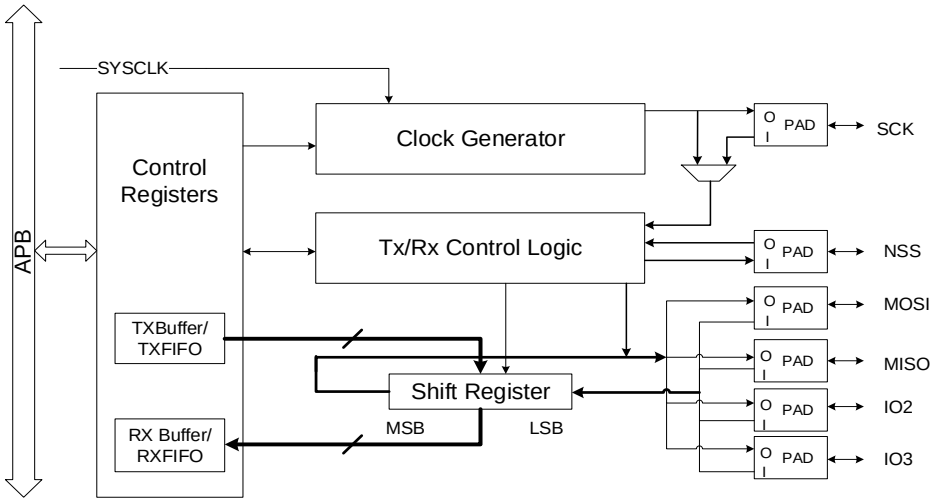
- Master or slave operation for transmission/reception.
- Four I2S standards supported: Philips, MSB justified, LSB justified and PCM standard.
- Data length can be 16 bits, 24 bits or 32 bits.
- Channel length can be 16 bits or 32 bits.
- Transmission and reception using a 16 bits wide buffer.
- Audio sample frequency can be 8 kHz to 192 kHz using I2S clock divider.
- Programmable idle state clock polarity.

- Master clock (MCK) can be output.
- Transmission and reception using DMA.

# 18.3. SPI function overview

## 18.3.1. SPI block diagram

**Figure 18-1. Block diagram of SPI**




## 18.3.2. SPI signal description

### Normal configuration (not Quad-SPI mode)

**Table 18-1. SPI signal description**

| Pin name | Direction | Description |
|---|---|---|
| SCK | I/O | Master: SPI clock output<br>Slave: SPI clock input |
| MISO | I/O | Master: data reception line<br>Slave: data transmission line<br>Master with bidirectional mode: not used<br>Slave with bidirectional mode: data transmission and reception line. |
| MOSI | I/O | Master: data transmission line<br>Slave: data reception line<br>Master with bidirectional mode: data transmission and reception line.<br>Slave with bidirectional mode: not used |
| NSS | I/O | Software NSS mode: not used<br>Master in hardware NSS mode: when NSSDRV=1, it is NSS |

| Pin name | Direction | Description |
|---|---|---|
| | | output, suitable for single master application; when NSSDRV=0, it is NSS input, suitable for multi-master application.<br>Slave in hardware NSS mode: NSS input, as a chip select signal for slave. |

### Quad-SPI configuration

SPI is in single wire mode by default and enters into Quad-SPI mode after QMOD bit in SPI_QCTL register is set (only available in SPI1). Quad-SPI mode can only work in master mode.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

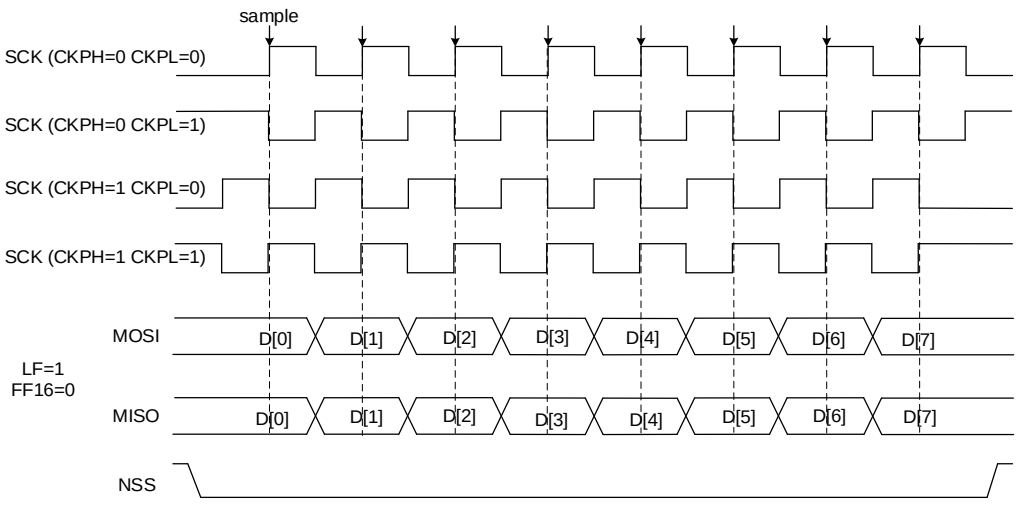
**Table 18-2. Quad-SPI signal description**

| Pin name | Direction | Description |
|---|---|---|
| SCK | O | SPI clock output |
| MOSI | I/O | Transmission/Reception data 0 |
| MISO | I/O | Transmission/Reception data 1 |
| IO2 | I/O | Transmission/Reception data 2 |
| IO3 | I/O | Transmission/Reception data 3 |
| NSS | O | NSS output |

## 18.3.3. SPI clock timing and data format

CKPL and CKPH bits in SPI_CTL0 register decide the timing of SPI clock and data signal. The CKPL bit decides the SCK level when SPI is in idle state and CKPH bit decides either first or second clock edge is a valid sampling edge. These bits take no effect in TI mode.

**Figure 18-2. SPI0 timing diagram in normal mode**




In SPI0 normal mode, the length of data is configured by the FF16 bit in the SPI_CTL0

register. Data length is 16 bits if FF16=1, otherwise is 8 bits.

Data order is configured by LF bit in SPI_CTL0 register, and SPI will send the LSB first if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

**Figure 18-3. SPI1 timing diagram in normal mode**

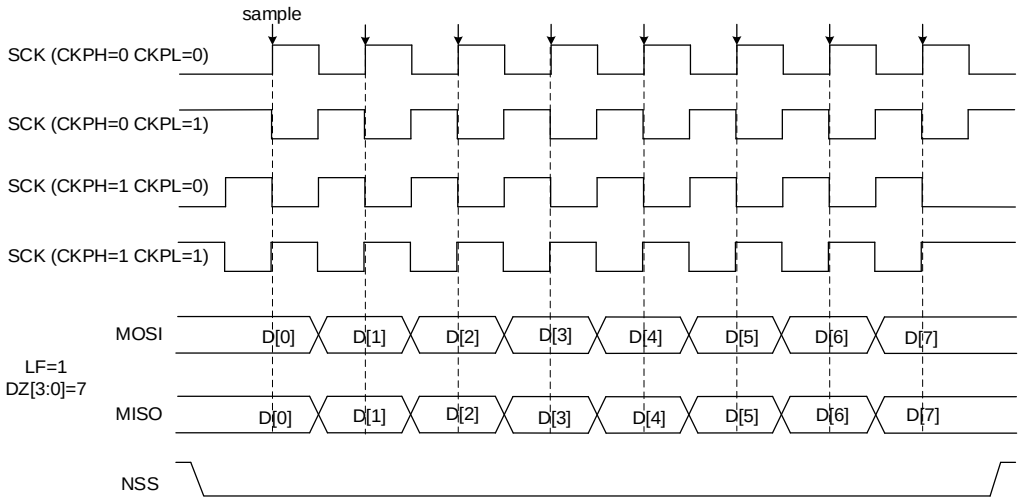
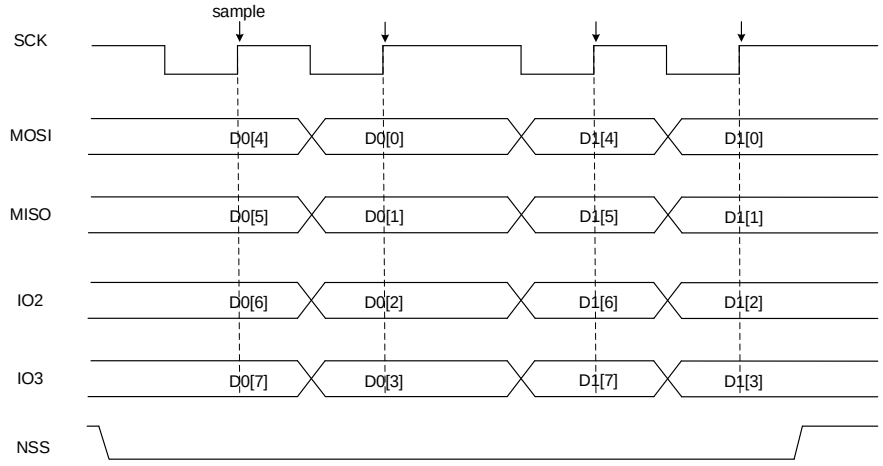



**Figure 18-4. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0)**

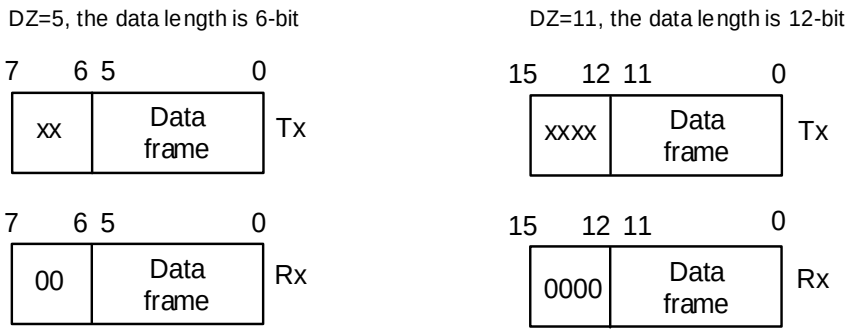



In SPI1 normal mode, the length of data is configured by the DZ[3:0] bits in the SPI_CTL1 register. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception, and the read access to the FIFO must be aligned with the BYTEN bit in the SPI_CTL1 register. The data frame length is fixed to 8 bits in Quad-SPI mode.

Data order is configured by LF bit in SPI_CTL0 register, and SPI will send the LSB first if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

When the SPI_DATA register is accessed, data frames are always right-aligned into either a byte (If the data length is less than or equal to one byte) or a half-word. During communication, only bits within the data frame will be output with the clock.

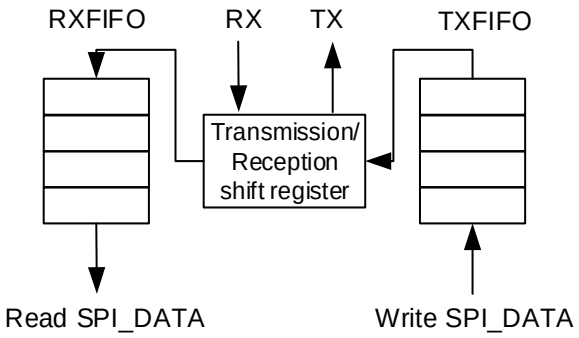
**Figure 18-5. SPI1 data frame right-aligned diagram**




## 18.3.4. Separate transmission and reception FIFO

The separate 32-bit reception FIFO (RXFIFO) and transmission FIFO (TXFIFO) are used in different directions for SPI data transactions, and they can enable the SPI to work in a continuous flow (only available in SPI1).

**Figure 18-6. Transmission and reception FIFO**




When the current TXFIFO level is less than or equal to half of its capacity, the TXFIFO is considered empty(1) and TBE is set to 1 by hardware at this time. When the TBE bit is set, writing data to the SPI_DATA register will store the data at the end of the TXFIFO. Hardware sets the RBNE bit when the RXFIFO is considered non-empty(2). When the RBNE bit is set, reading data from the SPI_DATA register will get the oldest data from the RXFIFO.

**Note:** (1) For SPI1, the TXFIFO empty means that the TXFIFO level is less than or equal to half of its capacity. The meaning of TXFIFO full is the opposite. Therefore, when the data frame format is not greater than 8 bits, the TXFIFO can store up to three data frames. If the TXFIFO empty or full appears below and there is no special explanation, the meaning is the same as that described here.

(2) For SPI1, the meaning of RXFIFO empty is divided into the following two conditions: If BYTEN bit in SPI_CTL1 is set, the RXFIFO empty means the RXFIFO level is less than quarter of its capacity. At this time, when the data frame format is not more than 8 bits, the RXFIFO can store up to 4 data frames. If BYTEN is cleared, the RXFIFO empty means the

RXFIFO level is less than half of its capacity. The meaning of RXFIFO full is the opposite. If the RXFIFO empty or full appears below and there is no special explanation, the meaning is the same as that described here.

#### Data merging (Only for SPI1)

When DZ[3:0] in the SPI_CTL1 register configures the transmission data bit width to be 8 bits or less than 8 bits, by configuring the BYTEN bit in the SPI_CTL1 register to 0, the data merge transmission mode function is enabled. When DZ[3:0] in the configuration SPI_CTL1 register configures the transmission data bit width to be less than or equal to 8 bits, this function can realize that when 16-bit write access is performed to the SPI_DATA register, two data frames are sent in parallel instead of serial line method.

Similarly, at the receiving end, the receiver obtains these two data frames through a 16-bit read access to SPI_DATA, and only one RBNE event will be generated when the two frames of data are received.

**Note:** when an odd number of data bytes will be transferred, on the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPI_DATA is enough. The receiver has to change BYTEN for the last data frame received in the odd sequence of frames in order to generate the RBNE event.

### 18.3.5. NSS function

#### Slave mode

When slave mode is configured (MSTMOD=0), SPI gets NSS level from NSS pin in hardware NSS mode (SWNSSEN = 0) or from SWNSS bit in software NSS mode (SWNSSEN = 1) and transmits/receives data only when NSS level is low. In software NSS mode, NSS pin is not used.

**Table 18-3. NSS function in slave mode**

| Mode | Register configuration | Description |
|---|---|---|
| Slave hardware NSS mode | MSTMOD = 0<br>SWNSSEN = 0 | SPI slave gets NSS level from NSS pin. |
| Slave software NSS mode | MSTMOD = 0<br>SWNSSEN = 1 | SPI slave NSS level is determined by the SWNSS bit.<br>SWNSS = 0: NSS level is low<br>SWNSS = 1: NSS level is high |

#### Master mode

In master mode (MSTMOD=1) if the application uses multi-master connection, NSS can be configured to hardware input mode (SWNSSEN=0, NSSDRV=0) or software mode

(SWNSSEN=1). Then, once the NSS pin (in hardware NSS mode) or the SWNSS bit (in software NSS mode) goes low, the SPI automatically enters slave mode and triggers a master fault flag CONFERR.

If the application wants to use NSS line to control the SPI slave, NSS should be configured to hardware output mode (SWNSSEN=0, NSSDRV=1). NSS goes low after SPI is enabled.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

**Table 18-4. NSS function in master mode**

| Mode | Register configuration | Description |
|---|---|---|
| Master hardware NSS output mode | MSTMOD = 1<br>SWNSSEN = 0<br>NSSDRV=1 | Applicable to single-master mode. The master uses the NSS pin to control the SPI slave device. At this time, the NSS is configured as the hardware output mode. NSS goes low after enabling SPI. |
| Master hardware NSS input mode | MSTMOD = 1<br>SWNSSEN = 0<br>NSSDRV=0 | Applicable to multi-master mode. At this time, NSS is configured as hardware input mode. Once the NSS pin is pulled low, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be set to 1. |
| Master software NSS mode | MSTMOD = 1<br>SWNSSEN = 1<br>SWNSS = 0<br>NSSDRV: Don’t care | Applicable to multi-master mode. Once SWNSS = 0, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be 1. |
| | MSTMOD = 1<br>SWNSSEN = 1<br>SWNSS = 1<br>NSSDRV: Don’t care | The slave can use hardware or software NSS mode. |

## 18.3.6. SPI operation modes

**Table 18-5. SPI operation modes**

| Mode | Description | Register configuration | Data pin usage |
|---|---|---|---|
| MFD | Master full-duplex | MSTMOD = 1<br>RO = 0<br>BDEN = 0<br>BDOEN: Don’t care | MOSI: transmission<br>MISO: reception |
| MTU | Master transmission with unidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 0 | MOSI: transmission<br>MISO: not used |

| Mode | Description | Register configuration | Data pin usage |
|---|---|---|---|
| | | BDOEN: Don't care | |
| MRU | Master reception with unidirectional connection | MSTMOD = 1<br>RO = 1<br>BDEN = 0<br>BDOEN: Don't care | MOSI: not used<br>MISO: reception |
| MTB | Master transmission with bidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 1<br>BDOEN = 1 | MOSI: transmission<br>MISO: not used |
| MRB | Master reception with bidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 1<br>BDOEN = 0 | MOSI: reception<br>MISO: not used |
| SFD | Slave full-duplex | MSTMOD = 0<br>RO = 0<br>BDEN = 0<br>BDOEN: Don't care | MOSI: reception<br>MISO: transmission |
| STU | Slave transmission with unidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 0<br>BDOEN: Don't care | MOSI: not used<br>MISO: transmission |
| SRU | Slave reception with unidirectional connection | MSTMOD = 0<br>RO = 1<br>BDEN = 0<br>BDOEN: Don't care | MOSI: reception<br>MISO: not used |
| STB | Slave transmission with bidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 1<br>BDOEN = 1 | MOSI: not used<br>MISO: transmission |
| SRB | Slave reception with bidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 1<br>BDOEN = 0 | MOSI: not used<br>MISO: reception |

**Figure 18-7. A typical full-duplex connection**

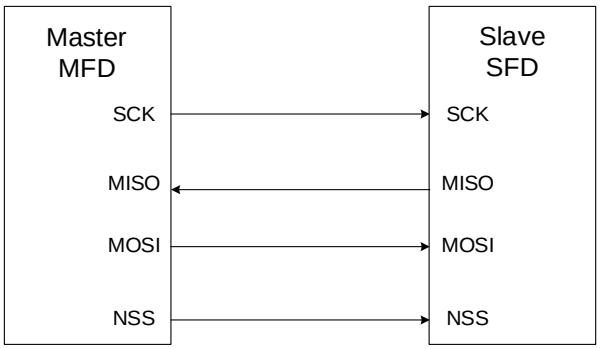



**Figure 18-8. A typical simplex connection (Master: receive, Slave: transmit)**

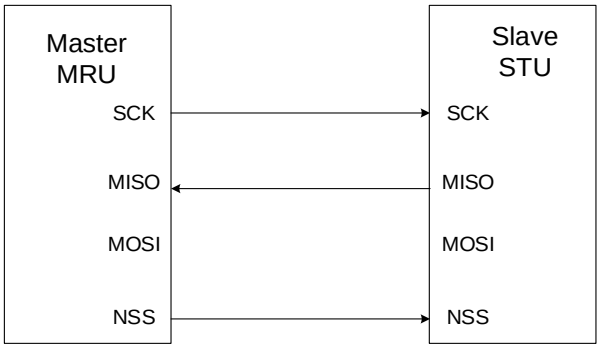



**Figure 18-9. A typical simplex connection (Master: transmit only, Slave: receive)**

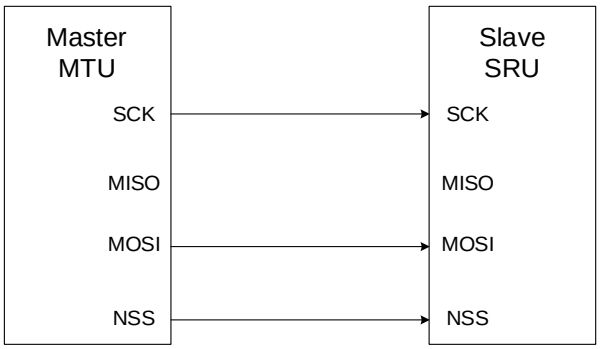
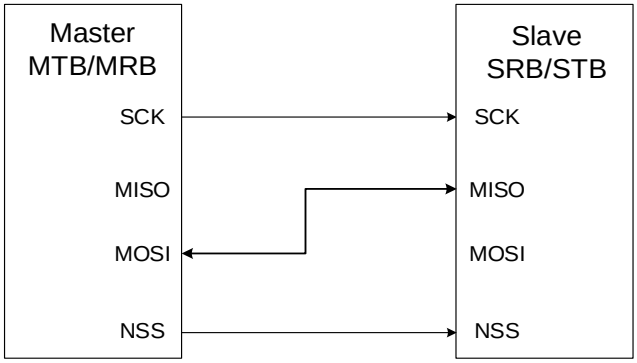



**Figure 18-10. A typical bidirectional connection**



### Initialization sequence

**SPI0:**

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or slave TI mode is used, program the PSC [2:0] bits in SPI_CTL0 register to generate SCK with desired baud rate or configure the Td time in TI mode, otherwise, ignore this step.
2. Configure data format (FF16 bit in the SPI_CTL0 register).
3. Configure the clock timing register (CKPL and CKPH bits in the SPI_CTL0 register).
4. Configure the frame format (LF bit in the SPI_CTL0 register).
5. Configure the NSS mode (SWNSSEN and NSSDRV bits in the SPI_CTL0 register) according to the application's demand as described above in ***NSS function*** section.
6. If TI mode is used, set TMOD bit in SPI_CTL1 register, otherwise, ignore this step.
7. If NSSP mode is used, set NSSP bit in SPI_CTL1 register, otherwise, ignore this step.
8. Configure MSTMOD, RO, BDEN and BDOEN depending on the operating modes described in ***SPI operation modes*** section.
9. Enable the SPI (set the SPIEN bit).

**Note:** During communication, CKPH, CKPL, MSTMOD, PSC[2:0] and LF bits should not be changed.

**SPI1:**

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or slave TI mode is used, program the PSC [2:0] bits in SPI_CTL0 register to generate SCK with desired baud rate or configure the Td time in TI mode, otherwise, ignore this step.
2. Configure the clock timing register (CKPL and CKPH bits in the SPI_CTL0 register).
3. Configure the frame format (LF bit in the SPI_CTL0 register).
4. Configure data format (DZ[3:0] bits in the SPI_CTL1 register) and the access size for the SPI_DATA register (BYTEN bit in the SPI_CTL1 register).
5. Configure the NSS mode (SWNSSEN and NSSDRV bits in the SPI_CTL0 register) according to the application's demand as described above in ***NSS function*** section.
6. If TI mode is used, set TMOD bit in SPI_CTL1 register, otherwise, ignore this step.
7. If NSSP mode is used, set NSSP bit in SPI_CTL1 register, otherwise, ignore this step.
8. Configure MSTMOD, RO, BDEN and BDOEN depending on the operation modes described in ***SPI operation modes*** section.
9. Configure the TXDMA_ODD and RXDMA_ODD bits, according to the needs of the application.
10. If Quad-SPI mode is used, set the QMOD bit in SPI_QCTL register. Ignore this step if Quad-SPI mode is not used.
11. Enable the SPI (set the SPIEN bit).

**Note:** During communication, CKPH, CKPL, MSTMOD, PSC[2:0], LF and DZ[3:0] bits should not be changed.

### Basic transmission and reception sequence

#### Transmission sequence

After the initialization sequence, the SPI is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the transmission buffer/TXFIFO. In slave mode the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is low, so application should ensure that data is already written into transmission buffer/TXFIFO before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the transmission buffer/TXFIFO to the shift register and then begins to transmit the loaded data frame. After TBE flag is set, which means the transmission buffer/TXFIFO is empty, the application should write SPI_DATA register again if it has more data to transmit.

In master mode, software should write the next data into SPI_DATA register before the transmission of current data frame is completed if it desires to generate continuous transmission.

#### Reception sequence

After the last valid sample clock, the incoming data will be moved from shift register to the reception buffer/RXFIFO and RBNE will be set. The application should read SPI_DATA register to get the received data and this will clear the RBNE flag automatically when reception buffer/RXFIFO is empty. In MRU and MRB modes, hardware continuously sends clock signal to receive the next data frame, while in full-duplex master mode (MFD), hardware only receives the next data frame when the transmission buffer/TXFIFO is not empty.

### SPI operation sequence in different modes (not Quad-SPI, TI mode or NSSP mode)

In full-duplex mode, either MFD or SFD, the RBNE and TBE flags should be monitored and then follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to the transmission sequence of full-duplex mode except that the RBNE bit and RXORERR bit need to be ignored.

The master reception mode (MRU or MRB) is different from the reception sequence of full-duplex mode. In MRU or MRB mode, after SPI is enabled, the SPI continuously generates SCK until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer/RXFIFO in time after the RBNE flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to the reception sequence of full-duplex

mode except that the TBE bit need to be ignored.

### SPI TI mode

SPI TI mode takes NSS as a special frame header flag signal and its operation sequence is similar to normal mode described above. The modes described above (MFD, MTU, MRU, MTB, MRB, SFD, STU, SRU, STB and SRB) are still supported in TI mode. While, in TI mode the CKPL and CKPH bits in SPI_CTL0 registers take no effect and the SCK sample edge is falling edge.

**Figure 18-11. Timing diagram of TI master mode with discontinuous transfer**

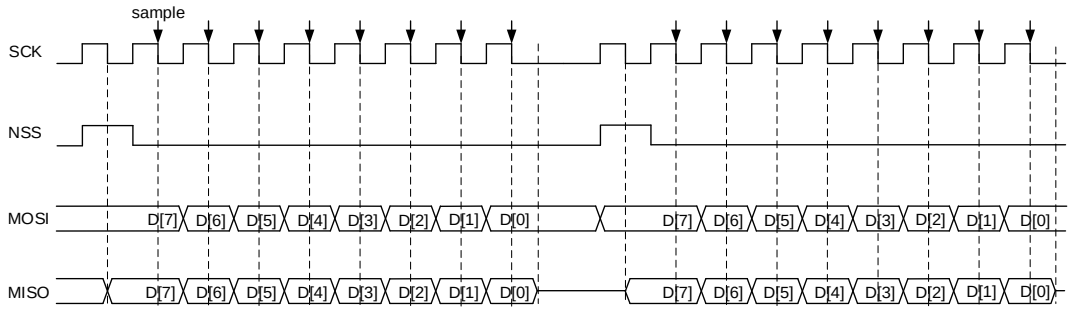
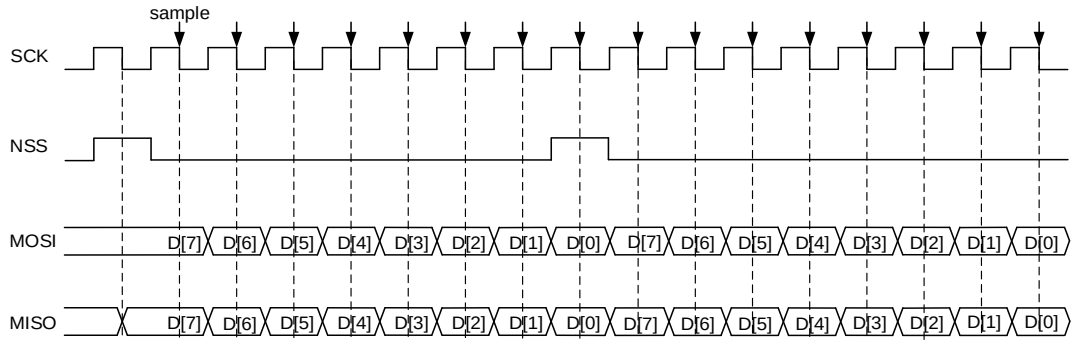
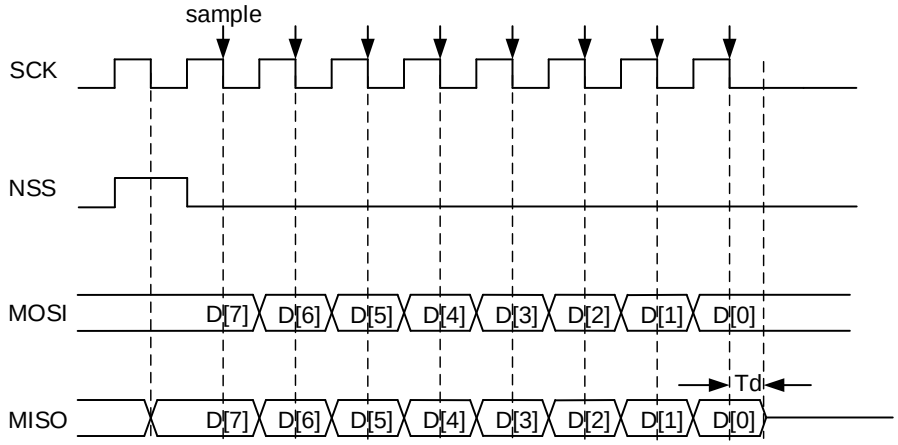



**Figure 18-12. Timing diagram of TI master mode with continuous transfer**




In master TI mode, SPI can perform continuous or non-continuous transfer. If the master writes SPI_DATA register fast enough, the transfer is continuous, otherwise non-continuous. In non-continuous transfer there is an extra header clock cycle before each byte. While in continuous transfer, the extra header clock cycle only exists before the first byte and the following bytes' header clock is overlaid at the last bit of pervious bytes.

**Figure 18-13. Timing diagram of TI slave mode**




In slave TI mode, after the last rising edge of SCK in transfer, the slave begins to transmit the LSB bit of the last data byte, and after a half-bit time, the master begins to sample the line. To make sure that the master samples the right value, the slave should continue to drive this bit after the falling sample edge of SCK for a period of time before releasing the pin. This time is called$T_d$, $T_d$ is decided by PSC [2:0] bits in SPI_CTL0 register.

$$T_d = \frac{T_{bit}}{2} + 5 * T_{pclk} \quad (18\text{-}1)$$

For example, if PSC [2:0] = 010, $T_d$ is 9*$T_{pclk}$.

In slave mode, the slave also monitors the NSS signal and sets an error flag FERR if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

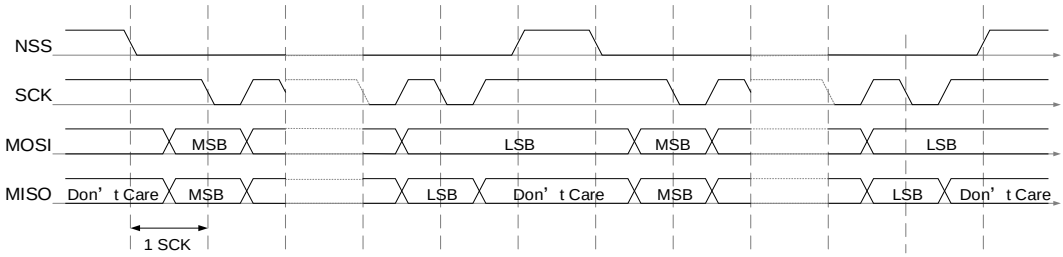
### NSS pulse mode operation sequence

This function is controlled by NSSP bit in SPI_CTL1 register. In order to implement this function, several additional conditions must be met: configure the device to master mode, frame format should follow the normal SPI protocol, select the first clock transition as the data capture edge.

In summary, MSTMOD = 1, NSSP = 1, CKPH = 0.

When NSS pulse mode is enabled, a pulse duration of at least 1 SCK clock period is inserted between two successive data frames depending on the status of internal data transmission buffer/TXFIFO. Multiple SCK clock cycle intervals are possible if the transfer buffer/TXFIFO stays empty. This function is designed for single master-slave configuration for the slave to latch data. The following diagram depicts its timing diagram.

**Figure 18-14. Timing diagram of NSS pulse with continuous transmit**




## Quad-SPI mode operation sequence

The Quad-SPI mode is designed to control Quad-SPI Flash.

In order to enter Quad-SPI mode, the software should first verify that the TBE bit is set and TRANS bit is cleared, then set QMOD bit in SPI_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCNT, CRCL, RO and LF in SPI_CTL0 register should be kept cleared and DZ[3:0] should be set to ensure that SPI data size is 8-bit, MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

There are two operation modes in Quad-SPI mode: quad write and quad read, decided by QRD bit in SPI_QCTL register.

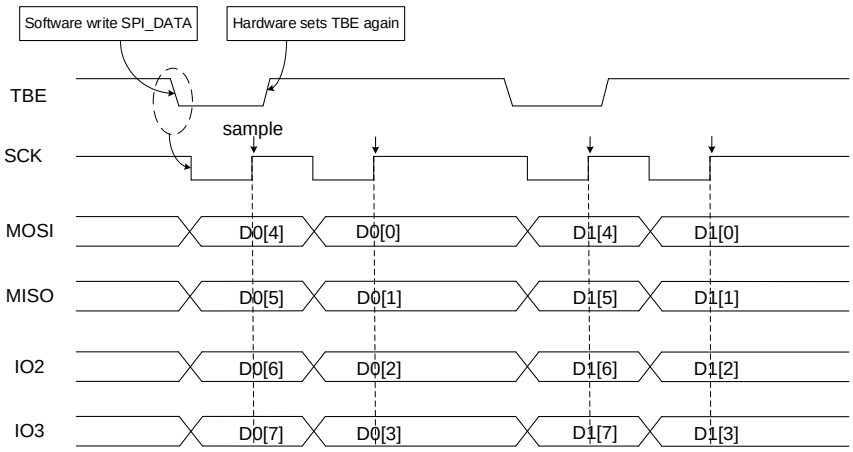
### Quad write operation

SPI works in quad write mode when QMOD is set and QRD is cleared in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as output pins. SPI begins to generate clock on SCK line and transmit data on MOSI, MISO, IO2 and IO3 as soon as data is written into SPI_DATA (TBE is cleared) and SPIEN is set. Once SPI starts transmission, it always checks TBE status at the end of a frame and stops when condition is not met.

The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 based on application requirements.
2. Set QMOD bit in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0.
3. Write a byte to SPI_DATA register and the TBE will be cleared.
4. Wait until TBE is set by hardware again before writing the next byte.

**Figure 18-15. Timing diagram of write operation in Quad-SPI mode**



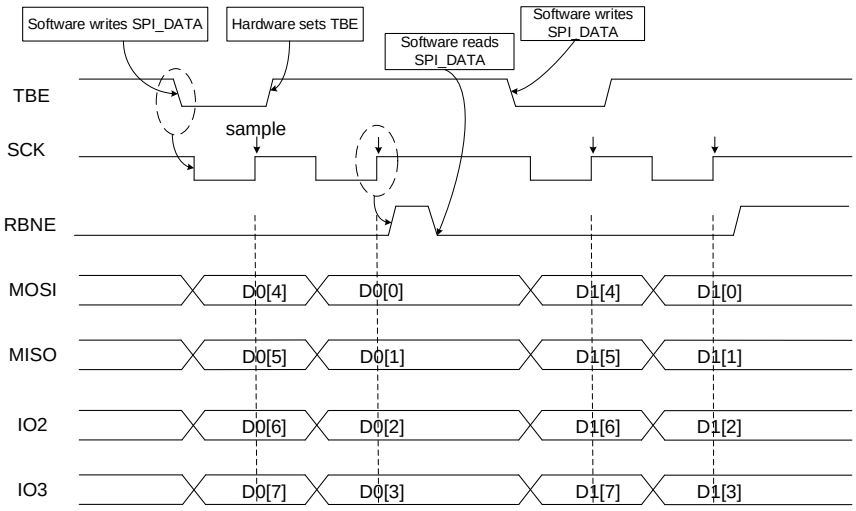

### Quad read operation

SPI works in quad read mode when QMOD and QRD are both set in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as input pins. SPI begins to generate clock on SCK line as soon as a data is written into SPI_DATA (TBE is cleared) and SPIEN is set. Writing data into SPI_DATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition is not met. So, dummy data should always be written into SPI_DATA to generate SCK.

The operation flow for receiving in quad mode is shown below:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 register based on application requirements.
2. Set QMOD and QRD bits in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0 register.
3. Write an arbitrary byte (for example, 0xFF) to SPI_DATA register.
4. Wait until the RBNE flag is set and read SPI_DATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI_DATA to receive the next byte.

**Figure 18-16. Timing diagram of read operation in Quad-SPI mode**




## SPI disabling sequence

Different sequences are used to disable the SPI in different operation modes.

### MFD SFD

For SPI0, wait for the last RBNE flag and then receive the last data. Confirm that TBE = 1 and TRANS = 0. At last, disable the SPI by clearing SPIEN bit.

For SPI1, wait until TXLVL[1:0] = 00 and confirm TRANS = 0. Then disable the SPI by clearing SPIEN bit. At last, read data until RXTVL[1:0] = 00.

### MTU MTB STU STB

For SPI0, write the last data into SPI_DATA and wait until the TBE flag is set and then wait until the TRANS flag is cleared. Disable the SPI by clearing SPIEN bit.

For SPI1, wait until TXLVL[1:0] = 00 and confirm TRANS = 0. Then disable the SPI by clearing SPIEN bit.

### MRU MRB

For SPI0, after getting the second last RBNE flag, read out this data and delay for a SCK clock time and then, disable the SPI by clearing SPIEN bit. Wait until the last RBNE flag is set and read out the last data.

For SPI1, after getting the second last RBNE flag, read out this data and delay for a SCK clock time and disable the SPI by clearing SPIEN bit. At last, wait until the last RBNE flag is set and read data until RXTVL[1:0] = 00.

### SRU SRB

For SPI0, application can disable the SPI when it doesn't want to receive data, and then wait

until the TRANS = 0 to ensure the ongoing transfer completed.

For SPI1, application can disable the SPI when it doesn't want to receive data, and then confirm the TRANS=0 and read data until RXLVL[1:0] = 00.

**TI mode**

The disabling sequence of TI mode is the same as the sequences described above.

**NSS pulse mode**

The disabling sequence of NSSP mode is the same as the sequences described above.

**Quad-SPI mode**

Before leaving quad wire mode or disabling SPI, software should first check that TBE bit is set and TRANS bit is cleared, then the QMOD bit in SPI_QCTL register and SPIEN bit in SPI_CTL0 register are cleared.

### 18.3.7. DMA function

The DMA frees the application from data writing and reading process during transfer, to improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI_CTL1 register. To use DMA function, application should first configure DMA modules correctly, then configure SPI module according to the initialization sequence, at last enable SPI.

After being enabled, If DMATEN is set, SPI will generate a DMA request each time when TBE = 1, then DMA will acknowledge to this request and write data into the SPI_DATA register automatically. If DMAREN is set, SPI will generate a DMA request each time when RBNE = 1, then DMA will acknowledge to this request and read data from the SPI_DATA register automatically.

#### Data merging with DMA (Only for SPI1)

In the case of using DMA for data transmission, when BYTEN is set to 0 and the data length configured by DZ[3:0] is less than or equal to 8 bits and the data merging mode is enabled, the DMA will access the SPI_DATA register in 16-bit mode, automatically Complete the data transmission.

In the case that the data packetization mode is enabled and the frame number of the data frame is not an even multiple, in order to avoid the problem of one more frame of data in the last DMA transmission, the TXDMA_ODD/RXDMA_ODD bit in the SPI_CTL1 register needs to be set to 1.

### 18.3.8. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial defined in SPI_CRCPOLY register.

Application can enable the CRC function by setting CRCEN bit in SPI_CTL0 register. The CRC calculators calculate CRC for each bit transmitted and received on lines continuously, and the calculated CRC values can be read from SPI_TCRC and SPI_RCRC registers.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI_CTL0 register after the last data is written to the transmission buffer/TXFIFO. In full-duplex mode (MFD or SFD), when the SPI transmits a CRC and prepares to check the received CRC value, the SPI treats the incoming data as a CRC value. In reception mode (MRB, MRU, SRU and SRB), the application should set the CRCNT bit after the second last data frame is received. When CRC checking fails, the CRCERR flag will be set.

For SPI0, if the data length is 8-bit, the CRC calculation is based on the CRC8 standard. If i the data length is 16-bit, the CRC calculation is based on the CRC16 standard. If the DMA function is enabled, the software does not need to set the CRCNT bit, and the hardware will handle the CRC transmission and verification automatically.

For SPI1, only when the data length is 8-bit or 16-bit, SPI provides CRC calculation function, and independent of the data length, it can be fixedly set to 8-bit or 16-bit CRC calculation. For all other data lengths, CRC function is invalid. CRC data exchange usually requires one or more data communication time after the end of the data sequence. For example, when setting the data length to 8 bits and doing a 16-bit CRC check, it takes two frames to send the complete CRC data. If the DMA function is enabled, the hardware will handle the CRC transmission and verification automatically, but the SPI needs to set the counter value of the DMA transmitting channel and the receiving channel. The transmitting DMA counter value is the number of data frames that do not include the CRC frame. The configuration of receiving the DMA counter value is as follows:

1. Full duplex mode: Suppose the amount of data received by SPI is L, when CRCL = 0 and DZ = 8, then the count of the DMA receive channel is L + 1, otherwise the count of the DMA receive channel is L + 2.

2. Receive only mode: DMA receive channel count is only equal to the amount of data received .After receiving data, the CRC value is obtained by reading SPI_RCRC register by software.

**Note:** When SPI is in slave mode and CRC function is enable, the CRC calculator is sensitive to input SCK clock whether SPI is enable or not. The software must enable CRC only when the clock is stable to avoid wrong CRC calculation. And when SPI works as a slave, the NSS internal signal needs to be kept low between the data phase and CRC phase.

When SPI is configured as slave mode and the CRC function is used, the CRC calculation will still be performed even when the NSS pin is high (when the NSS signal is high, as long

as there is a clock pulse on the SCK pin, the CRC calculation will continue). This situation occurs when the master device alternately communicates with multiple slave devices. At this time, it is recommended to restart the CRC function when the NSS signal is low. When the slave device is not selected (NSS signal is high) and switches to being selected as a new slave device (NSS signal is low), in order to maintain the synchronization of the next CRC calculation results between the master and slave devices, the CRC values at both ends should be cleared. It is recommended to clear the CRC value according to the following steps:

1. Disable the SPI module (SPIEN=0);
2. Clear the CRCEN bit (CRCEN=0);
3. Set the CRCEN bit (CRCEN=1);
4. Enable the SPI module (SPIEN=1).

### 18.3.9. SPI interrupts

#### Status flags

- **Transmission buffer/TXFIFO empty flag (TBE)**

This bit is set when the transmission buffer is empty or the TXFIFO level is lower or equal to 1/2 of FIFO depth, the software can write the next data to the transmission buffer/TXFIFO by writing the SPI_DATA register.

- **Reception buffer/RXFIFO not empty flag (RBNE)**

For SPI0, this bit is set when reception buffer is not empty, which means that one data is received and stored in the reception buffer, and software can read the data by reading the SPI_DATA register.

For SPI1, this bit is set depending on the BYTEN bit in the SPI_CTL1: If BYTEN = 0, the RBNE is set when the RXFIFO level is greater or equal to 1/2(16-bit). If BYTEN = 1, the RBNE is set when the RXFIFO level is greater or equal to 1/4(8-bit).

- **SPI transmitting ongoing flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag doesn't generate any interrupt.

#### Error conditions

- **Configuration fault error (CONFERR)**

CONFERR is an error flag in master mode. In NSS hardware mode and the NSSDRV is not enabled, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the SWNSS bit is 0. When the CONFERR is set, the SPIEN bit and the MSTMOD bit are cleared by hardware, the SPI is disabled and the device is forced into slave mode.

The SPIEN and MSTMOD bit are write protection until the CONFERR is cleared. The CONFERR bit of the slave cannot be set. In a multi-master configuration, the device can be in slave mode with CONFERR bit set, which means there might have been a multi-master conflict for system control.

- **Rx overrun error (RXORERR)**

The RXORERR bit is set if a data is received when the RBNE is set. For SPI0, that means the last data has not been read out and the newly incoming data is received. For SPI1, that means the RXFIFO has not enough space to store this received data. The reception buffer/RXFIFO contents won't be covered with the newly incoming data, so the newly incoming data is lost.

- **Format error (FERR)**

In slave TI mode, the slave also monitors the NSS signal and set an error flag if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

- **CRC error (CRCERR)**

When the CRCEN bit is set, the CRC calculation result of the received data in the SPI_RCRC register is compared with the received CRC value after the last data, the CRCERR is set when they are different.

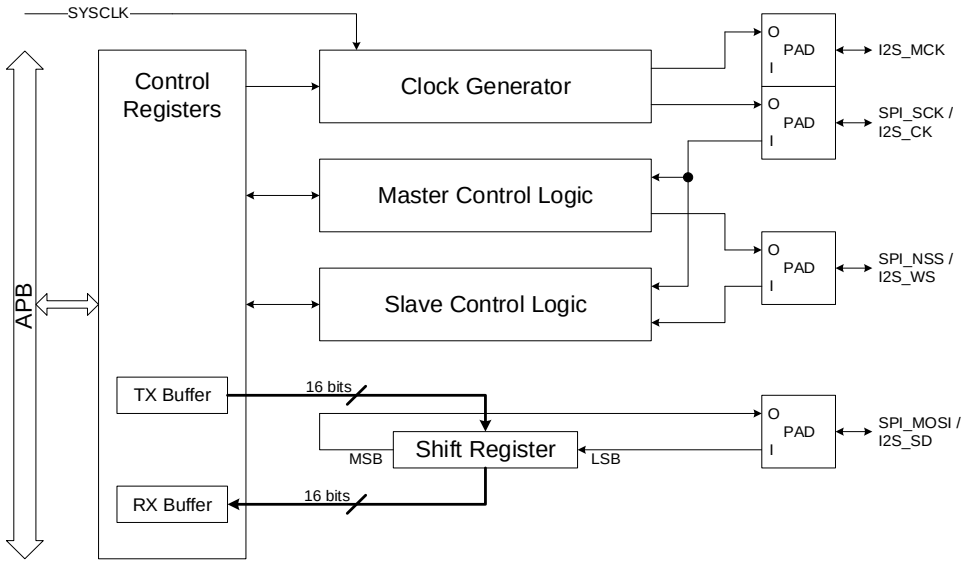
**Table 18-6. SPI interrupt requests**

| Flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| TBE | Transmission buffer/TXFIFO empty | Write SPI_DATA register. | TBEIE |
| RBNE | Reception buffer/RXFIFO not empty | Read SPI_DATA register. | RBNEIE |
| CONFERR | Configuration fault error | Read or write SPI_STAT register, then write SPI_CTL0 register. | ERRIE |
| RXORERR | Rx overrun error | Read SPI_DATA register, then read SPI_STAT register. | |
| CRCERR | CRC error | Write 0 to CRCERR bit | |
| FERR | TI mode format error | Write 0 to FERR bit | |

# 18.4. I2S function overview

## 18.4.1. I2S block diagram

**Figure 18-17. Block diagram of I2S**




There are five sub modules to support I2S function, including control registers, clock generator, master control logic, slave control logic and shift register. All the user configuration registers are implemented in the control registers module, including the TX buffer and RX buffer. The clock generator is used to produce I2S communication clock in master mode. The master control logic is implemented to generate the I2S_WS signal and control the communication in master mode. The slave control logic is implemented to control the communication in slave mode according to the received I2SCK and I2S_WS. The shift register handles the serial data transmission and reception on I2S_SD.

## 18.4.2. I2S signal description

There are four pins on the I2S interface, including I2S_CK, I2S_WS, I2S_SD and I2S_MCK. I2S_CK is the serial clock signal, which shares the same pin with SPI_SCK. I2S_WS is the frame control signal, which shares the same pin with SPI_NSS. I2S_SD is the serial data signal, which shares the same pin with SPI_MOSI. I2S_MCK is the master clock signal. It produces a frequency rate equal to 256 x Fs, and Fs is the audio sampling frequency.

## 18.4.3. I2S audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI_I2SCTL register. Four audio standards are supported, including I2S Philips standard, MSB justified standard, LSB justified standard, and PCM standard. All standards except PCM handle audio data time-multiplexed on two channels (the left channel and the right channel). For these

standards, the I2S_WS signal indicates the channel side. For PCM standard, the I2S_WS signal indicates frame synchronization information.

The data length and the channel length are configured by the DTLEN bit and CHLEN bit in the SPI_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 24-bit data packed in 32-bit frame, and 32-bit data packed in 32-bit frame. The data buffer for transmission and reception is 16-bit wide. In the case that the data length is 24 bits or 32 bits, two write or read operations to or from the SPI_DATA register are needed to complete the transmission of a frame. In the case that the data length is 16-bit, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame. When using 16-bit data packed in 32-bit frame, 16-bit 0 is inserted by hardware automatically to extend the data to 32-bit format.

For all standards and packet types, the most significant bit (MSB) is always sent first. For all standards based on two channels time-multiplexed, the channel left is always sent first followed by the channel right.

### I2S Philips standard

For I2S Philips standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK, and I2S_WS becomes valid one clock before the data. The timing diagrams for each configuration are shown below.

**Figure 18-18. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

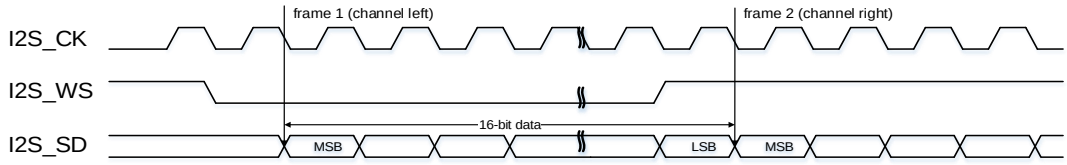
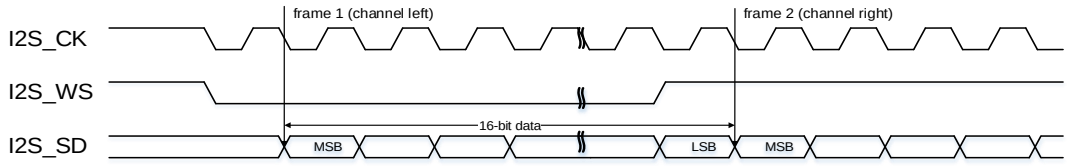



**Figure 18-19. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**




When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame.

**Figure 18-20. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

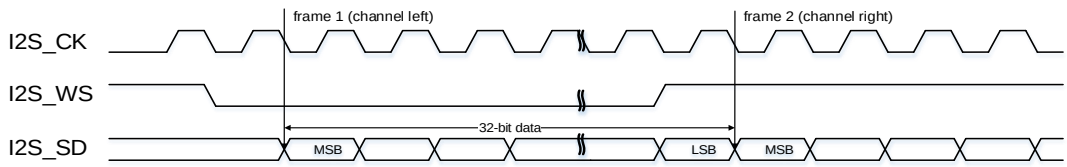
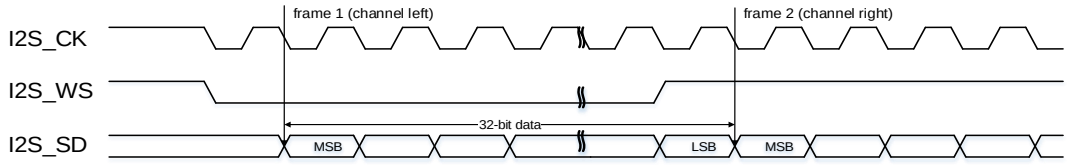


**Figure 18-21. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**




When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits, and the second one should be the lower 16 bits. In reception mode, if a 32-bit data is received, the first data read from the SPI_DATA register should be higher 16 bits, and the second one should be the lower 16 bits.

**Figure 18-22. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

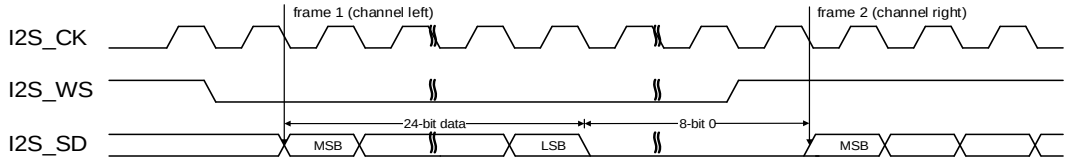
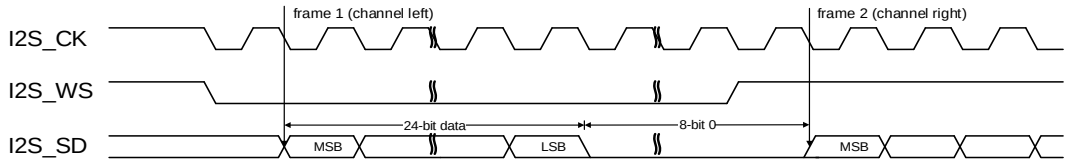



**Figure 18-23. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**




When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits: D[23:8], and the second one should be a 16-bit data. The higher 8 bits of this 16-bit data should be D[7:0] and the lower 8 bits can be any value. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI_DATA register is D[23:8], and the second one is a 16-bit data. The higher 8 bits of this 16-bit data are D[7:0] and the lower 8 bits are zeros.

**Figure 18-24. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

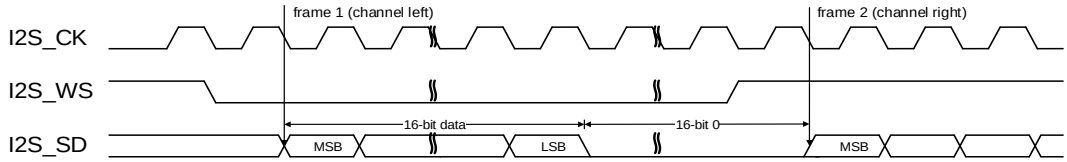
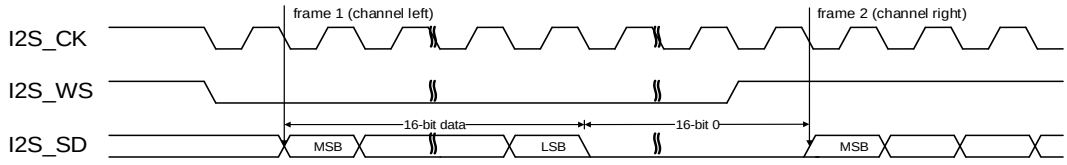


**Figure 18-25. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**




When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### MSB justified standard

For MSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The SPI_DATA register is handled in the exactly same way as that for I2S Philips standard. The timing diagrams for each configuration are shown below.

**Figure 18-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

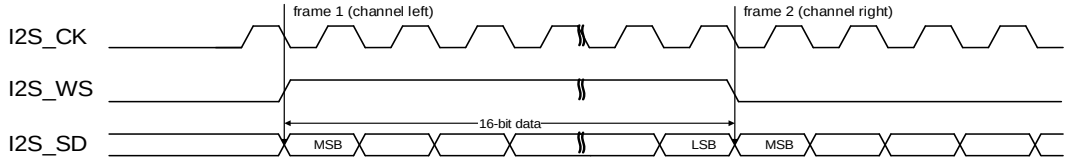



**Figure 18-27. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

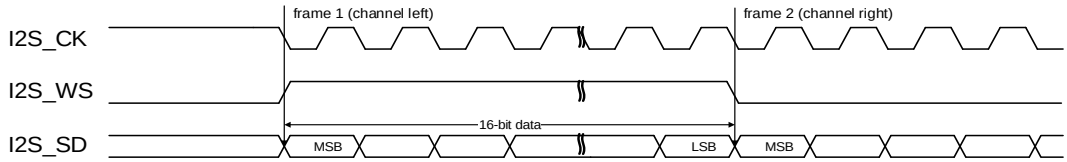



**Figure 18-28. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

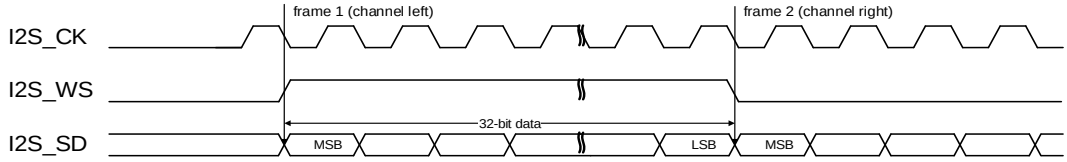



**Figure 18-29. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

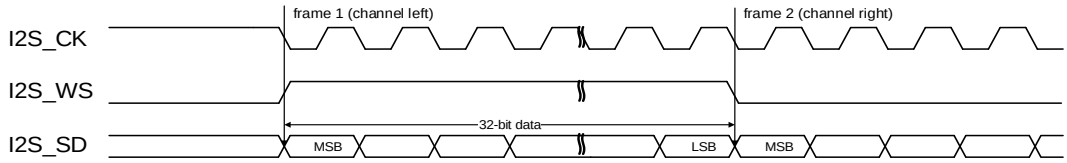


**Figure 18-30. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

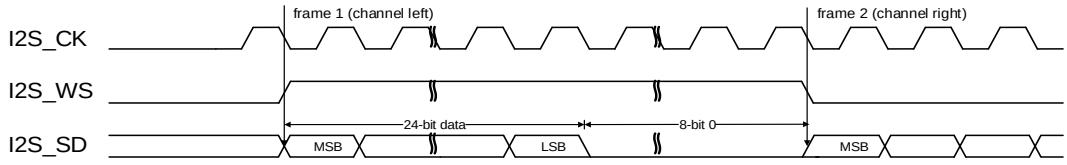



**Figure 18-31. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

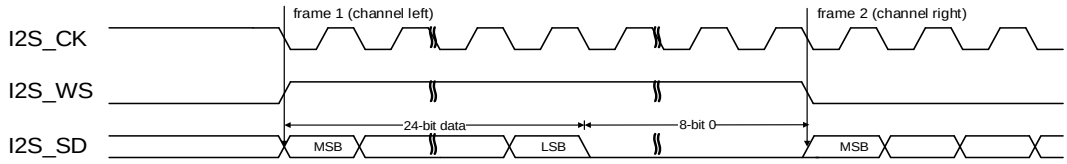



**Figure 18-32. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

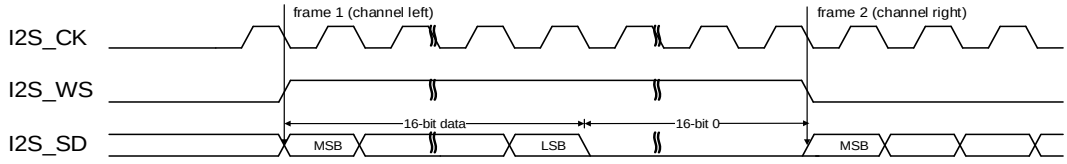
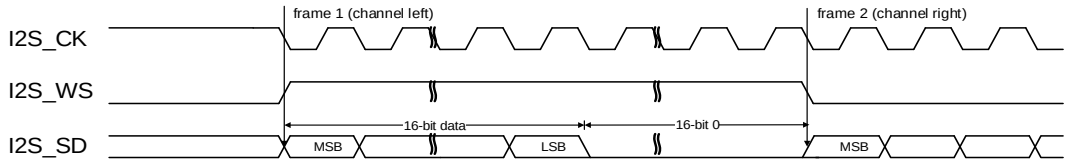



**Figure 18-33. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**




## LSB justified standard

For LSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. In the case that the channel length is equal to the data length, LSB justified standard and MSB justified standard are exactly the same. In the case that the channel length is greater than the data length, the valid data is aligned to LSB for LSB justified standard while the valid data is aligned to MSB for MSB justified standard. The timing diagrams for the cases that the channel length is greater than the data length are shown below.

**Figure 18-34. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

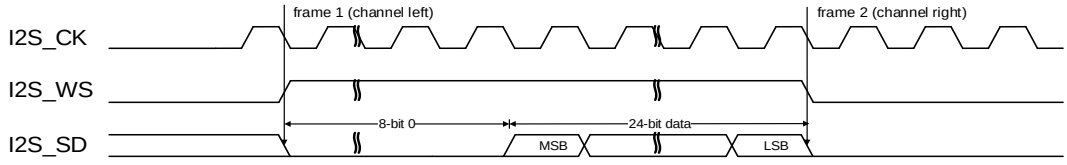
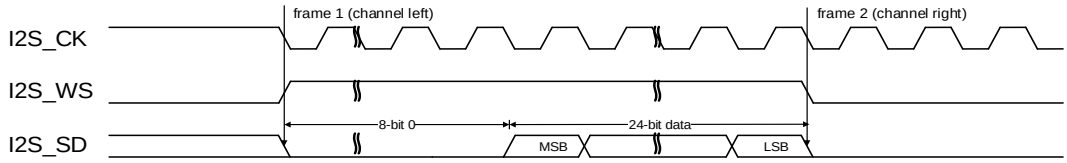


**Figure 18-35. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**




When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI_DATA register should be a 16-bit data. The higher 8 bits of the 16-bit data can be any value and the lower 8 bits should be D[23:16]. The second data written to the SPI_DATA register should be D[15:0]. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI_DATA register is a 16-bit data. The high 8 bits of this 16-bit data are zeros and the lower 8 bits are D[23:16]. The second data read from the SPI_DATA register is D[15:0].

**Figure 18-36. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

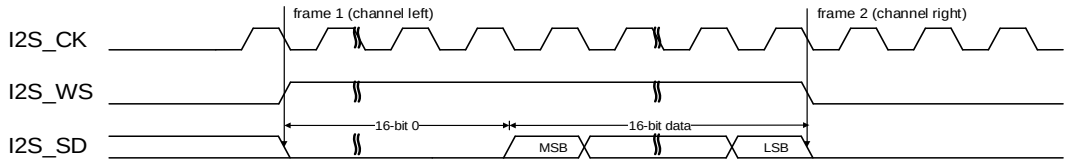
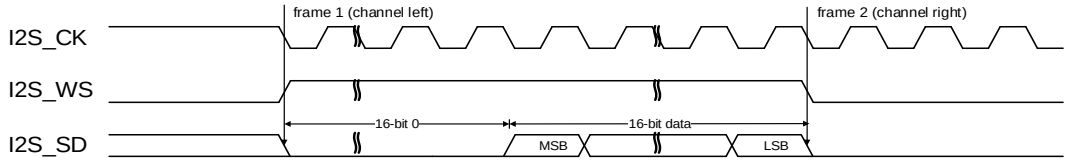



**Figure 18-37. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**




When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### PCM standard

For PCM standard, I2S_WS and I2S_SD are updated on the rising edge of I2S_CK, and the I2S_WS signal indicates frame synchronization information. Both the short frame synchronization mode and the long frame synchronization mode are available and configurable using the PCMSMOD bit in the SPI_I2SCTL register. The SPI_DATA register is handled in the exactly same way as that for I2S Philips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

**Figure 18-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

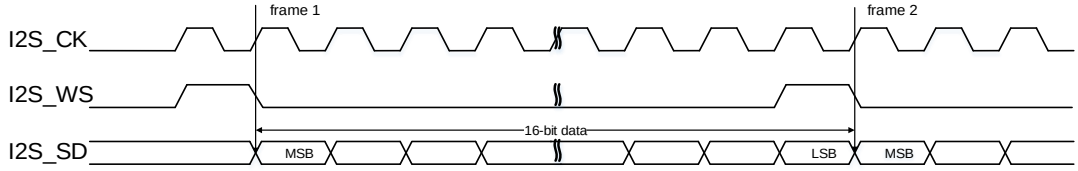



**Figure 18-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

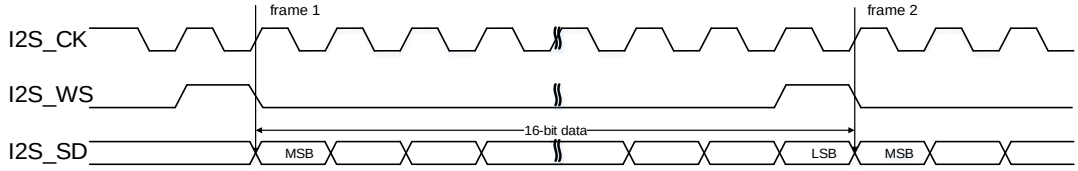



**Figure 18-40. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

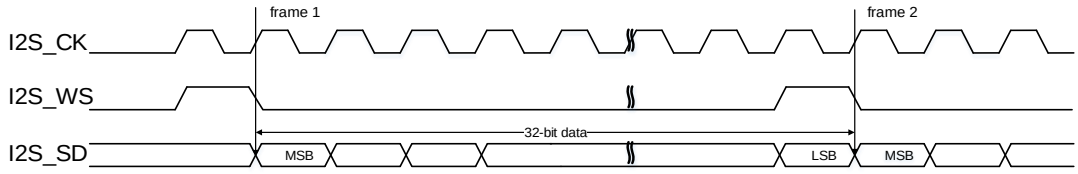



**Figure 18-41. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

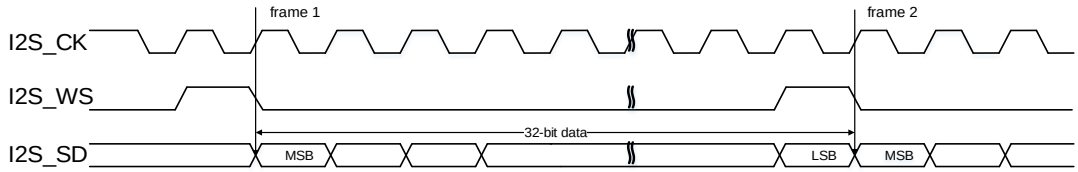



**Figure 18-42. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

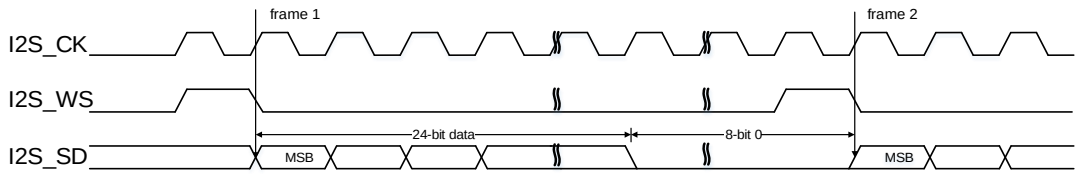


**Figure18-43. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

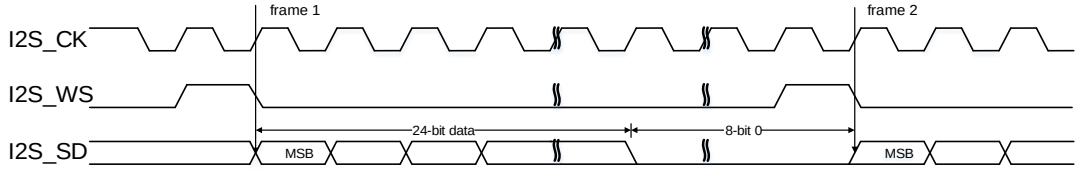



**Figure 18-44. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

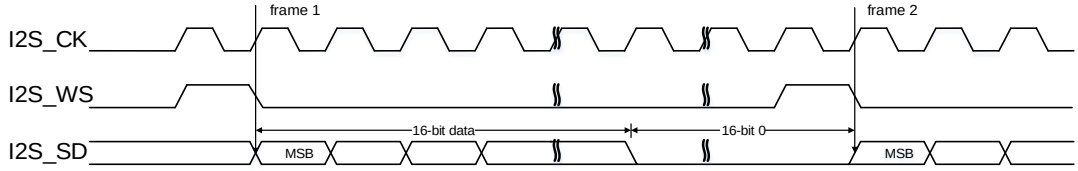
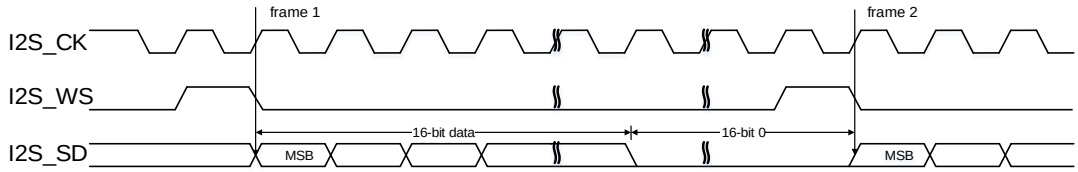



**Figure 18-45. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**




The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 18-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

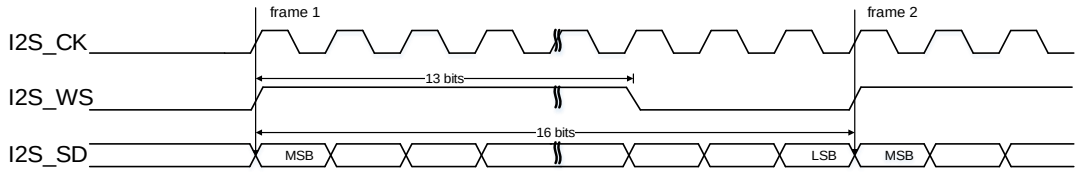



**Figure18-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

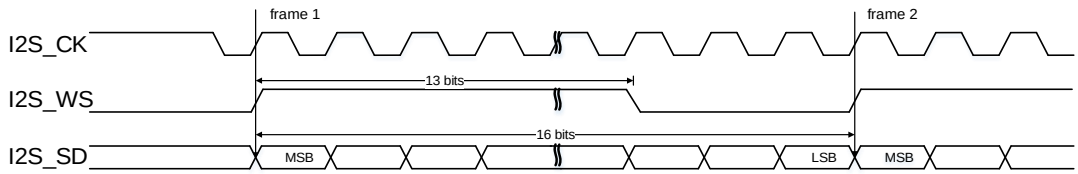


**Figure 18-48. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

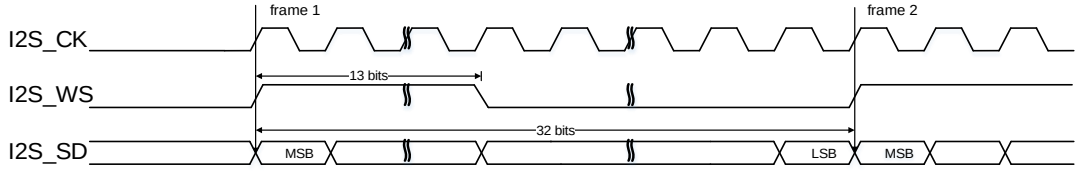



**Figure 18-49. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

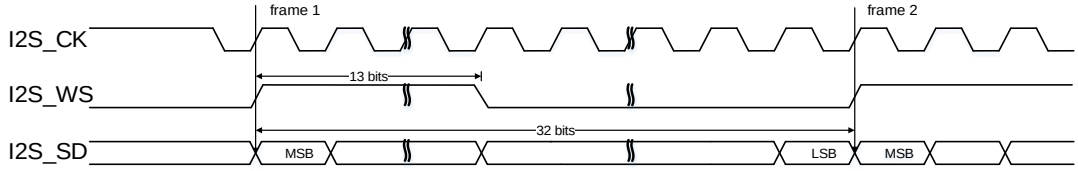



**Figure 18-50. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

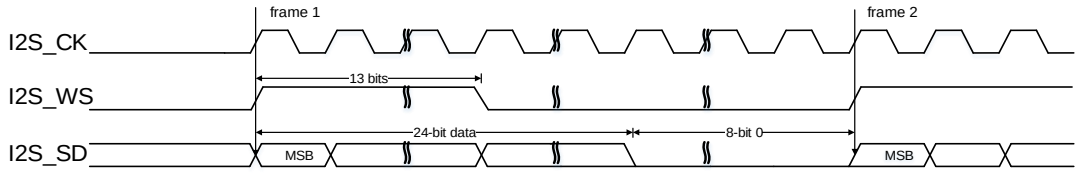



**Figure 18-51. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

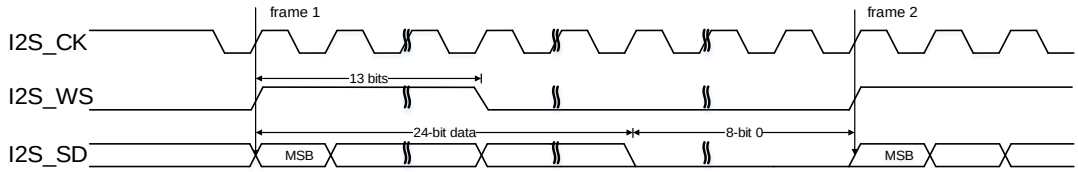



**Figure 18-52. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

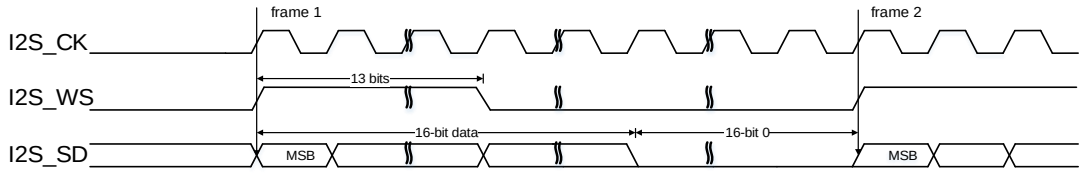
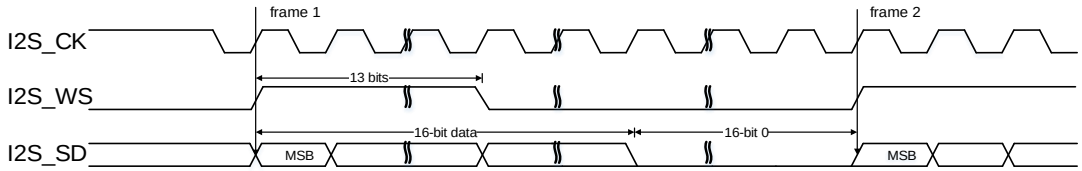
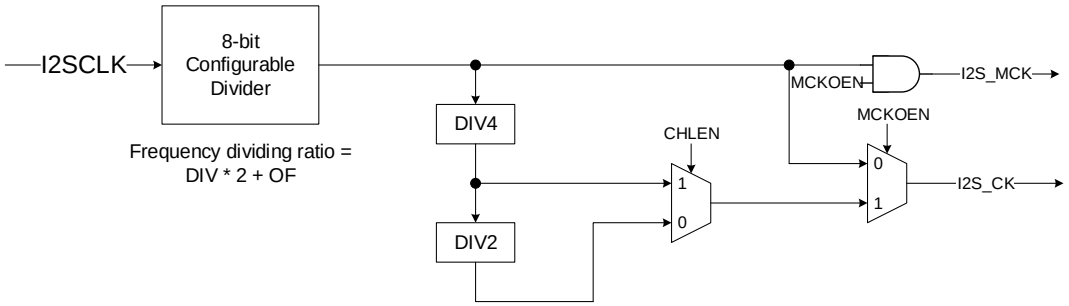


**Figure 18-53. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**




## 18.4.4. I2S clock

**Figure 18-54. Block diagram of I2S clock generator**




The block diagram of I2S clock generator is shown as ***Figure 18-54. Block diagram of I2S clock generator***.The I2S interface clocks are configured by the DIV bits, the OF bit, the MCKOEN bit in the SPI_I2SPSC register and the CHLEN bit in the SPI_I2SCTL register. The source clock is the system clock(CK_SYS). The I2S bitrate can be calculated by the formulas shown in ***Table 18-7. I2S bitrate calculation formulas***.

**Table 18-7. I2S bitrate calculation formulas**

| MCKOEN | CHLEN | Formula |
|---|---|---|
| 0 | 0 | I2SCLK / (DIV * 2 + OF) |
| 0 | 1 | I2SCLK / (DIV * 2 + OF) |
| 1 | 0 | I2SCLK / (8 * (DIV * 2 + OF)) |
| 1 | 1 | I2SCLK / (4 * (DIV * 2 + OF)) |

The relationship between audio sampling frequency (Fs) and I2S bitrate is defined by the following formula:

Fs = I2S bitrate / (number of bits per channel * number of channels)

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in ***Table 18-8. Audio sampling frequency calculation formulas***.

**Table 18-8. Audio sampling frequency calculation formulas**

| MCKOEN | CHLEN | Formula |
|---|---|---|
| 0 | 0 | I2SCLK / (32 * (DIV * 2 + OF)) |
| 0 | 1 | I2SCLK / (64 * (DIV * 2 + OF)) |

| | | |
|---|---|---|
| 1 | 0 | I2SCLK / (256 * (DIV * 2 + OF)) |
| 1 | 1 | I2SCLK / (256 * (DIV * 2 + OF)) |

**Note**: The configuration value of the I2S serial clock must be set to less than 1/6 of the PCLK clock (excluding 1/6).

## 18.4.5. Operation

### Operation modes

The operation mode is selected by the I2SOPMOD[1:0] bits in the SPI_I2SCTL register. There are four available operation modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode. The direction of I2S interface signals for each operation mode is shown in the ***Table 18-9. Direction of I2S interface signals for each operation mode***.

**Table 18-9. Direction of I2S interface signals for each operation mode**

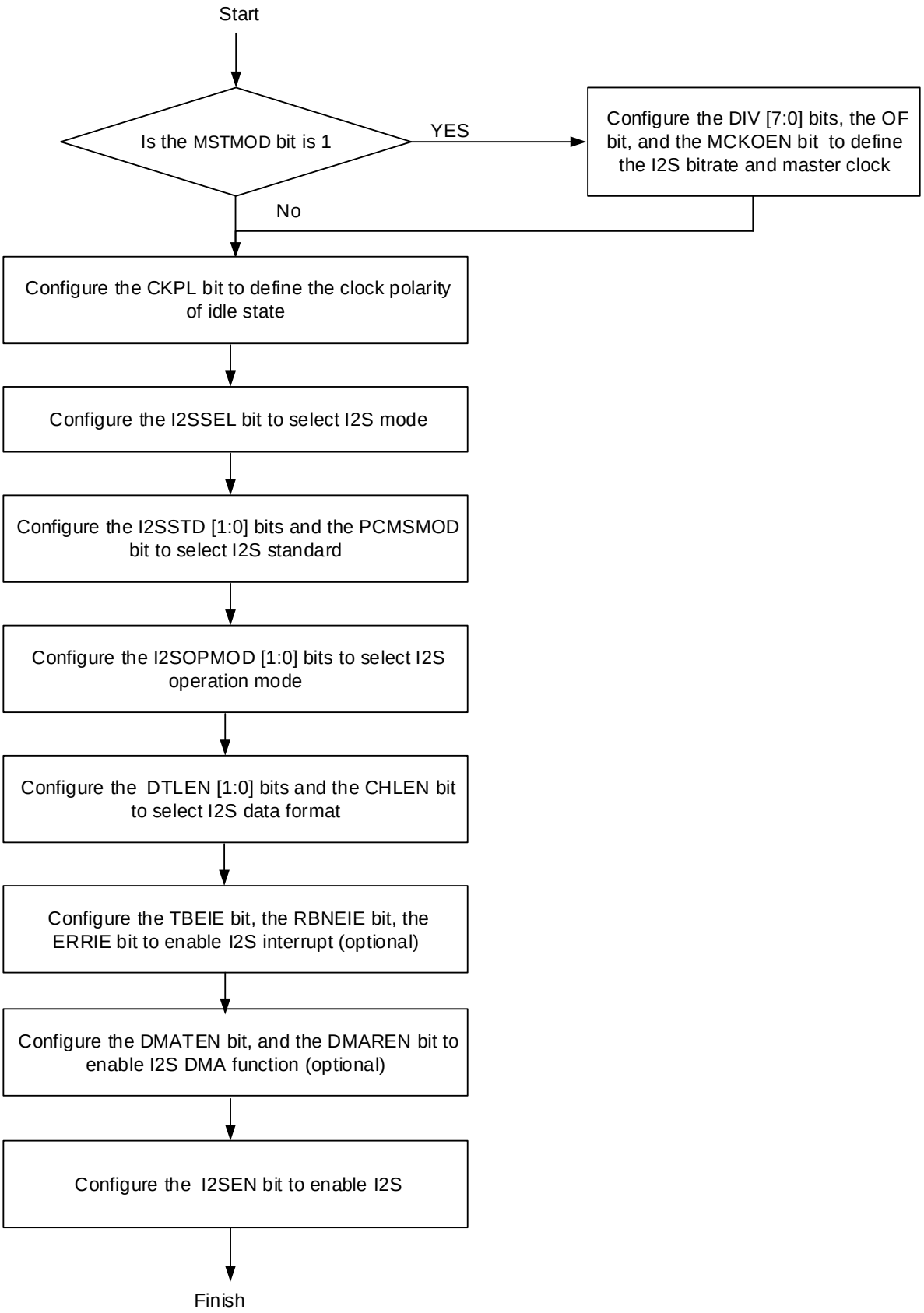
| Operation mode | I2S_MCK | I2S_CK | I2S_WS | I2S_SD |
|---|---|---|---|---|
| Master transmission | Output or NU(1) | Output | Output | Output |
| Master reception | Output or NU(1) | Output | Output | Input |
| Slave transmission | Input or NU(1) | Input | Input | Output |
| Slave reception | Input or NU(1) | Input | Input | Input |

1. NU means the pin is not used by I2S and can be used by other functions.

### I2S initialization sequence

I2S initialization sequence is shown as below ***Figure 18-55. I2S initialization sequence***.

**Figure 18-55. I2S initialization sequence**



### I2S master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmission buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI_CTL1 register is set. At the beginning, the transmission buffer is empty (TBE is high) and no transmission sequence is processing in the shift register. When a half word is written to the SPI_DATA register (TBE goes low), the data is transferred from the transmission buffer to the shift register (TBE goes high) immediately. At the moment, the transmission sequence begins.

The data is parallel loaded into the 16-bit shift register, and shifted out serially to the I2S_SD pin, MSB first. The next data should be written to the SPI_DATA register, when the TBE flag is high. After a write operation to the SPI_DATA register, the TBE flag goes low. When the current transmission finishes, the data in the transmission buffer is loaded into the shift register, and the TBE flag goes back high. Software should write the next audio data into SPI_DATA register before the current data finishes, otherwise, the audio data transmission is not continuous.

For all standards except PCM, the I2SCH flag is used to distinguish which channel side the data to transfer belongs to. The I2SCH flag is refreshed at the moment when the TBE flag goes high. At the beginning, the I2SCH flag is low, indicating the left channel data should be written to the SPI_DATA register.

In order to disable I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

### I2S master reception sequence

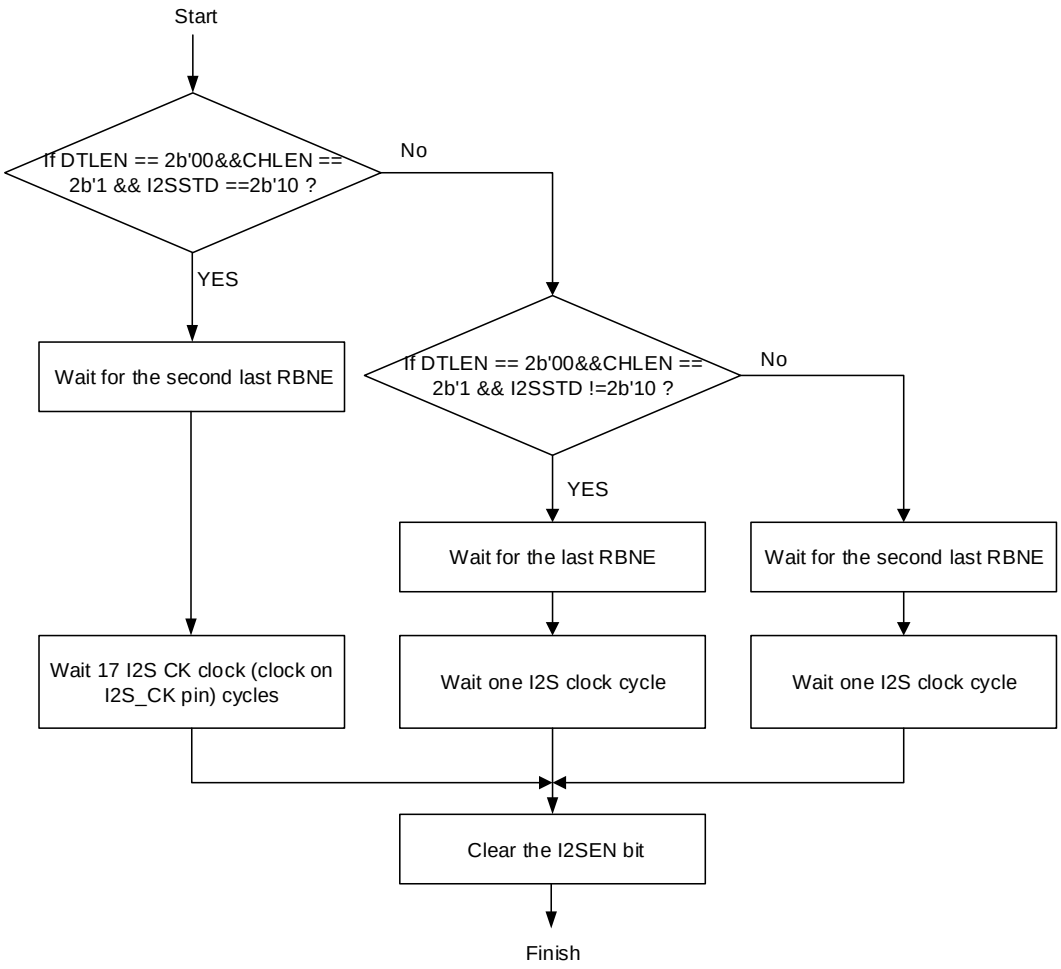
The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the reception buffer is not empty, and an interrupt will be generated if the RBNEIE bit in the SPI_CTL1 register is set. The reception sequence begins immediately when the I2SEN bit in the SPI_I2SCTL register is set. At the beginning, the reception buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the reception buffer (RBNE goes high). The data should be read from the SPI_DATA register, when the RBNE flag is high. After a read operation to the SPI_DATA register, the RBNE flag goes low. It is mandatory to read the SPI_DATA register before the end of the next reception. Otherwise, reception overrun error occurs. The RXORERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is necessary to disable and then enable I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side which the received data belongs to. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

Different sequences are used to disable the I2S in different standards, data length and

channel length. The sequences for each case are shown as below ***Figure 18-56. I2S master reception disabling sequence***.

**Figure 18-56. I2S master reception disabling sequence**




## I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The differences between them are described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The transmission sequence begins when the external master sends the clock and when the I2S_WS signal requests the transfer of data. The data has to be written to the SPI_DATA register before the master initiates the communication. Software should write the next audio data into SPI_DATA register before the current data finishes. Otherwise, transmission underrun error occurs. The TXURERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is mandatory to disable and enable I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to disable I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

#### I2S slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The differences between them are described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The reception sequence begins when the external master sends the clock and when the I2S_WS signal indicates a start of the data transfer. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to disable I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

### 18.4.6. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

### 18.4.7. I2S interrupts

#### Status flags

There are four status flags implemented in the SPI_STAT register, including TBE, RBNE, TRANS and I2SCH. The user can use them to fully monitor the state of the I2S bus.

- **Transmission buffer empty flag (TBE)**

This bit is set when the transmission buffer is empty, the software can write the next data to the transmission buffer by writing the SPI_DATA register.

- **Reception buffer not empty flag (RBNE)**

This bit is set when reception buffer is not empty, which means that one data is received and stored in the reception buffer, and software can read the data by reading the SPI_DATA register.

- **I2S transmitting ongoing flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag will not generate any interrupt.

- **I2S channel side flag (I2SCH)**

This flag indicates the channel side information of the current transfer and has no meaning in PCM mode. In the transmission mode, the I2SCH flag is updated every time TBE changes from 0 to 1. In the reception mode, the I2SCH flag is updated every time RBNE changes

from 0 to 1. This flag will not generate any interrupt.

### Error conditions

There are three error flags:

- **Transmission underrun error flag (TXURERR)**

This situation occurs when the transmission buffer is empty and the valid SCK signal starts in slave transmission mode.

- **Reception overrun error flag (RXORERR)**

This situation occurs when the reception buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in reception buffer is not updated and the newly incoming data is lost.

- **Format error (FERR)**

In slave I2S mode, the I2S monitors the I2S_WS signal and an error flag will be set if I2S_WS toggles at an unexpected position.

I2S interrupt events and corresponding enabled bits are summed up in the ***Table 18-10. I2S interrupt.***

**Table 18-10. I2S interrupt**

| Interrupt flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| TBE | Transmission buffer empty | Write SPI_DATA register | TBEIE |
| RBNE | Reception buffer not empty | Read SPI_DATA register | RBNEIE |
| TXURERR | Transmission underrun error | Read SPI_STAT register | ERRIE |
| RXORERR | Reception overrun error | Read SPI_DATA register and then read SPI_STAT register. | |
| FERR | I2S format error | Read SPI_STAT register | |

## 18.5. Register definition

SPI0/I2S0 base address: 0x4001 3000

SPI1 base address: 0x4000 3800

### 18.5.1. Control register 0 (SPI_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

This register has no meaning in I2S 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BDEN | BDOEN | CRCEN | CRCNT | FF16<br>CRCL | RO | SWNSS EN | SWNSS | LF | SPIEN | PSC [2:0] | | | MSTMOD | CKPL | CKPH |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | BDEN | Bidirectional enable<br>0: 2 line unidirectional transmit mode<br>1: 1 line bidirectional transmit mode. The data transfers between the MOSI pin of master and the MISO pin of slave. |
| 14 | BDOEN | Bidirectional transmit output enable<br>When BDEN is set, this bit determines the direction of transfer.<br>0: Work in receive-only mode<br>1: Work in transmit-only mode |
| 13 | CRCEN | CRC calculation enable<br>0: Disable CRC calculation.<br>1: Enable CRC calculation. |
| 12 | CRCNT | CRC next transfer<br>0: Next transfer is data<br>1: Next transfer is CRC value<br>When the transfer is managed by DMA, CRC value is transferred by hardware. This bit should be cleared.<br>In full-duplex or transmit-only mode, set this bit after the last data is written to SPI_DATA register. In receive only mode, set this bit after the second last data is |

| | | |
|---|---|---|
| | | received. |
| 11 | FF16 | Data frame format (only for SPI0)<br>0: 8-bit data frame format<br>1: 16-bit data frame format |
| | CRCL | CRC length (only for SPI1)<br>0: 8-bit crc length.<br>1: 16-bit crc length. |
| 10 | RO | Receive only<br>When BDEN is cleared, this bit determines the direction of transfer.<br>0: Full-duplex mode<br>1: Receive-only mode |
| 9 | SWNSSEN | NSS software mode selection<br>0: NSS hardware mode. The NSS level depends on NSS pin.<br>1: NSS software mode. The NSS level depends on SWNSS bit.<br>This bit has no meaning in SPI TI mode. |
| 8 | SWNSS | NSS pin selection in NSS software mode<br>0: NSS pin is pulled low.<br>1: NSS pin is pulled high.<br>This bit has an effect only when the SWNSSEN bit is set.<br>This bit has no meaning in SPI TI mode. |
| 7 | LF | LSB first mode<br>0: Transmit MSB first<br>1: Transmit LSB first<br>This bit has no meaning in SPI TI mode. |
| 6 | SPIEN | SPI enable<br>0: Disable SPI peripheral.<br>1: Enable SPI peripheral. |
| 5:3 | PSC[2:0] | Master clock prescaler selection<br>000: PCLK/2<br>001: PCLK/4<br>010: PCLK/8<br>011: PCLK/16<br>100: PCLK/32<br>101: PCLK/64<br>110: PCLK/128<br>111: PCLK/256<br>PCLK means PCLK2 when using SPI0. PCLK means PCLK1 when using SPI1. |
| 2 | MSTMOD | Master mode enable<br>0: Slave mode |

| | | |
|---|---|---|
| | | 1: Master mode |
| 1 | CKPL | Clock polarity selection<br>0: CLK pin is pulled low when SPI is idle.<br>1: CLK pin is pulled high when SPI is idle. |
| 0 | CKPH | Clock phase selection<br>0: Capture the first data at the first clock transition.<br>1: Capture the first data at the second clock transition. |

### 18.5.2. Control register 1 (SPI_CTL1)

Address offset: 0x04

Reset value: SPI0: 0x0000 0000

SPI1: 0x0000 0700

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TXDMA_ODD | RXDMA_ODD | BYTEN | DZ[3:0] | | | | TBEIE | RBNEIE | ERRIE | TMOD | NSSP | NSSDRV | DMATEN | DMAREN |
| | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | TXDMA_ODD | Odd bytes in TX DMA channel (only for SPI1)<br>In data merging mode, this bit is set if the total number of data to transmit by DMA is odd. It has effect only when DMATEN is set and data merging mode enable (data size is less than or equal to 8-bit and write access to SPI_DATA is 16-bit wide).<br>This field can be written only when SPI is disabled.<br>0: The total number of data to transmit by DMA is even.<br>1: The total number of data to transmit by DMA is odd. |
| 13 | RXDMA_ODD | Odd bytes in RX DMA channel (only for SPI1)<br>In data merging mode, this bit is set if the total number of data to receive by DMA is odd. It has effect only when DMAREN is set and data merging mode enable (data size is less than or equal to 8-bit and write access to SPI_DATA is 16-bit wide).<br>This field can be written only when SPI is disabled.<br>0: The total number of data to receive by DMA is even.<br>1: The total number of data to receive by DMA is odd. |
| 12 | BYTEN | Byte access enable (only for SPI1)<br>This bit is used to indicate the access size to FIFO, and set the threshold of the |

| | | |
|---|---|---|
| | | RXFIFO that generate RBNE.<br>0: Half-word access, and RBNE is generated when RXLVL >= 2.<br>1: Byte access, and RBNE is generated when RXLVL >= 1. |
| 11:8 | DZ[3:0] | Date size (only for SPI1)<br>This field indicates the data size for transfer.<br>0000: Force to “0111”<br>0001: Force to “0111”<br>0010: Force to “0111”<br>0011: 4-bit<br>0100: 5-bit<br>……<br>1111: 16-bit |
| 7 | TBEIE | Transmit buffer / TXFIFO empty interrupt enable<br>0: Disable TBE interrupt<br>1: Enable TBE interrupt. An interrupt is generated when the TBE bit is set. |
| 6 | RBNEIE | Receive buffer / RXFIFO not empty interrupt enable<br>0: Disable RBNE interrupt.<br>1: Enable RBNE interrupt. An interrupt is generated when the RBNE bit is set. |
| 5 | ERRIE | Errors interrupt enable.<br>0: Disable error interrupt<br>1: Enable error interrupt. An interrupt is generated when the CRCERR bit or the CONFERR bit or the FERR bit or the RXORERR bit or the TXURERR bit is set. |
| 4 | TMOD | SPI TI mode enable.<br>0: Disable SPI TI mode<br>1: Enable SPI TI mode |
| 3 | NSSP | SPI NSS pulse mode enable.<br>0: Disable SPI NSS pulse mode<br>1: Enable SPI NSS pulse mode |
| 2 | NSSDRV | Drive NSS output<br>0: Disable master NSS output<br>1: Enable master NSS output |
| 1 | DMATEN | Transmit buffer / TXFIFO DMA enable<br>0: Disable transmit buffer / TXFIFO DMA.<br>1: Enable transmit buffer / TXFIFO DMA, when the TBE bit in SPI_STAT is set, it will be a DMA request on corresponding DMA channel. |
| 0 | DMAREN | Receive buffer / RXFIFO DMA enable<br>0: Disable receive buffer / RXFIFO DMA<br>1: Enable receive buffer / RXFIFO DMA, when the RBNE bit in SPI_STAT is set, it |

will be a DMA request on corresponding DMA channel.

### 18.5.3. Status register (SPI_STAT)

Address offset: 0x08

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

| 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 | | | TXLVL[1:0] | | RXLVL[1:0] | | FERR | TRANS | RXORERR | CONFERR | CRCERR | TXURERR | I2SCH | TBE | RBNE |
| | | | r | | r | | rc_w0 | r | r | r | rc_w0 | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:11 | TXLVL[1:0] | TXFIFO level (only for SPI1)<br>00: Empty<br>01: 1/4 full<br>10: 1/2 full<br>11: Full<br>**Note:** The FIFO level here refers to the current actual storage of the FIFO. Here, the FIFO is considered full when the FIFO level is greater than 1/2. |
| 10:9 | RXLVL[1:0] | RXFIFO level (only for SPI1)<br>00: Empty<br>01: 1/4 full<br>10: 1/2 full<br>11: Full<br>This field has no meaning when SPI is in receive-only mode with CRC function enabled.<br>**Note:** The FIFO level here refers to the current actual storage of the FIFO. Here, the FIFO is considered full when the FIFO level is greater than 1/2. |
| 8 | FERR | Format error<br>SPI TI Mode:<br>0: No TI mode format error<br>1: TI mode format error occurs.<br>I2S Mode:<br>0: No I2S format error<br>1: I2S format error occurs.<br>This bit is set by hardware and is able to be cleared by writing 0. |

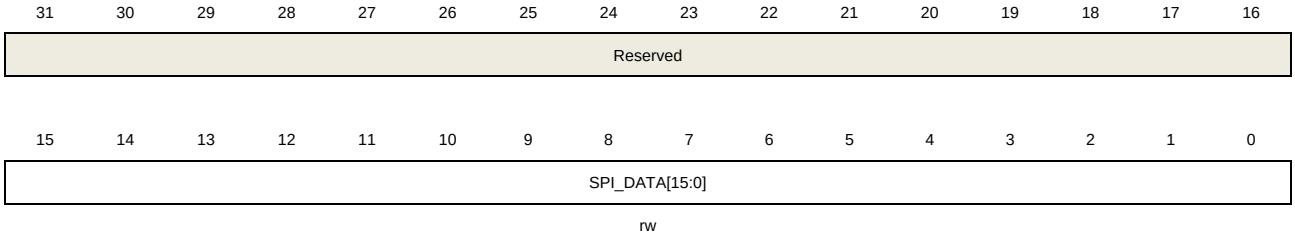
| | | |
|---|---|---|
| 7 | TRANS | Transmitting ongoing bit<br>0: SPI or I2S is idle.<br>1: SPI or I2S is currently transmitting and/or receiving a frame.<br>This bit is set and cleared by hardware. |
| 6 | RXORERR | Reception overrun error bit<br>0: No reception overrun error occurs.<br>1: Reception overrun error occurs.<br>This bit is set by hardware and cleared by a read operation on the SPI_DATA register followed by a read access to the SPI_STAT register. |
| 5 | CONFERR | SPI configuration error<br>0: No configuration fault occurs.<br>1: Configuration fault occurred. (In master mode, the NSS pin is pulled low in NSS hardware mode or SWNSS bit is low in NSS software mode.)<br>This bit is set by hardware and cleared by a read or write operation on the SPI_STAT register followed by a write access to the SPI_CTL0 register.<br>This bit is not used in I2S mode. |
| 4 | CRCERR | SPI CRC error bit<br>0: The SPI_RCRC value is equal to the received CRC data at last.<br>1: The SPI_RCRC value is not equal to the received CRC data at last.<br>This bit is set by hardware and is able to be cleared by writing 0.<br>This bit is not used in I2S mode. |
| 3 | TXURERR | Transmission underrun error bit<br>0: No transmission underrun error occurs.<br>1: Transmission underrun error occurs.<br>This bit is set by hardware and cleared by a read operation on the SPI_STAT register.<br>This bit is not used in SPI mode. |
| 2 | I2SCH | I2S channel side<br>0: The next data needs to be transmitted or the data just received is channel left.<br>1: The next data needs to be transmitted or the data just received is channel right.<br>This bit is set and cleared by hardware.<br>This bit is not used in SPI mode, and has no meaning in the I2S PCM mode. |
| 1 | TBE | Transmit buffer / TXFIFO empty<br>0: Transmit buffer / TXFIFO is not empty.<br>1: Transmit buffer / TXFIFO is empty. |
| 0 | RBNE | Receive buffer / RXFIFO not empty<br>0: Receive buffer / RXFIFO is empty.<br>1: Receive buffer / RXFIFO is not empty. |

### 18.5.4. Data register (SPI_DATA)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).



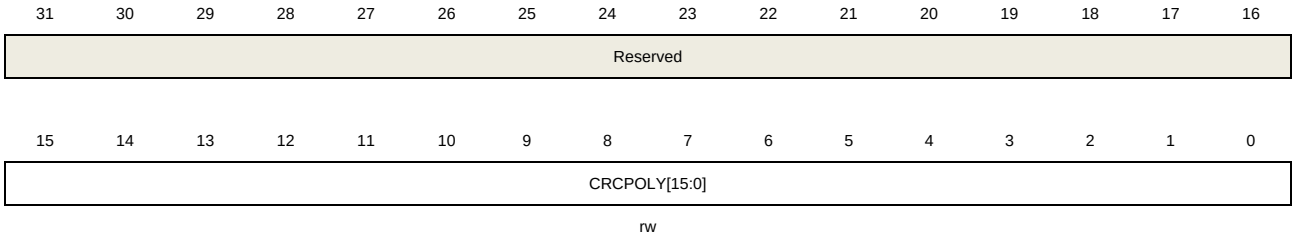

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | SPI_DATA[15:0] | Data transfer register.<br>For SPI0, the hardware has two buffers, including transmission buffer and reception buffer. Write data to SPI_DATA will save the data to transmission buffer and read data from SPI_DATA will get the data from reception buffer. When the data frame format is set to 8-bit data, the SPI_DATA [15:8] is forced to 0 and the SPI_DATA [7:0] is used for transmission and reception, transmission buffer and reception buffer are 8-bits. If the Data frame format is set to 16-bit data, the SPI_DATA [15:0] is used for transmission and reception, transmission buffer and reception buffer are 16-bit.<br>For SPI1, the hardware has two FIFOs, including TXFIFO and RXFIFO. Write data to SPI_DATA will save the data to TXFIFO and read data from SPI_DATA will get the data from RXFIFO.<br>**Note:** In fact, SPI1 hardware determines the size of each access to SPI_DATA only based on the BYTEN bit in SPI_CTL1, regardless of the size of the software's current operation. |

### 18.5.5. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x10

Reset value: 0x0000 0007

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CRCPOLY[15:0] | CRC polynomial register<br>This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h. |

### 18.5.6. RX CRC register (SPI_RCRC)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RCRC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RCRC[15:0] | RX CRC value<br>When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCRC register. For SPI0, if the data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in RCRC[7:0], when the data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in RCRC[15:0]. For SPI1, CRC function is valid only when the data length is 8 bits or 16 bits. And if the CRC length is set to 8-bit and the data size is equal to 8-bit, the CRC calculation is based on CRC8 standard, and saves the value in RCRC [7:0]. In addition to this, the calculation is based on CRC16 standard, and saves the value in RCRC [15:0].<br>The hardware computes the CRC value after each received bit, when the TRANS is set, a read to this register could return an intermediate value.<br>This register is reset when the CRCEN bit in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set. |

### 18.5.7. TX CRC register (SPI_TCRC)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCRC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | TCRC[15:0] | TX CRC value<br>When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in TCRC register. For SPI0, if the data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in TCRC[7:0], when the data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in TCRC[15:0]. For SPI1, CRC function is valid only when the data length is 8 bits or 16 bits. And if the CRC length is set to 8-bit and the data size is equal to 8-bit, the CRC calculation is based on CRC8 standard, and saves the value in TCRC[7:0]. In addition to this, the calculation is based on CRC16 standard, and saves the value in TCRC[15:0].<br>The hardware computes the CRC value after each transmitted bit, when the TRANS is set, a read to this register could return an intermediate value. The different frame formats (LF bit of the SPI_CTL0) will get different CRC values.<br>This register is reset when the CRCEN bit in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set. |

## 18.5.8. I2S control register (SPI_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

| 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 | | | | I2SSEL | I2SEN | I2SOPMOD[1:0] | | PCMSMOD | Reserved | I2SSTD[1:0] | | CKPL | DTLEN[1:0] | | CHLEN |
| | | | | rw | rw | rw | | rw | | rw | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | I2SSEL | I2S mode selection |

| | | |
|---|---|---|
| | | 0: SPI mode<br>1: I2S mode<br>This bit should be configured when SPI mode or I2S mode is disabled. |
| 10 | I2SEN | I2S enable<br>0: Disable I2S<br>1: Enable I2S<br>This bit is not used in SPI mode. |
| 9:8 | I2SOPMOD[1:0] | I2S operation mode<br>00: Slave transmission mode<br>01: Slave reception mode<br>10: Master transmission mode<br>11: Master reception mode<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 7 | PCMSMOD | PCM frame synchronization mode<br>0: Short frame synchronization<br>1: long frame synchronization<br>This bit has a meaning only when PCM standard is used.<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 6 | Reserved | Must be kept at reset value. |
| 5:4 | I2SSTD[1:0] | I2S standard selection<br>00: I2S Philips standard<br>01: MSB justified standard<br>10: LSB justified standard<br>11: PCM standard<br>These bits should be configured when I2S mode is disabled.<br>These bits are not used in SPI mode. |
| 3 | CKPL | Idle state clock polarity<br>0: The idle state of I2S_CK is low level.<br>1: The idle state of I2S_CK is high level.<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 2:1 | DTLEN[1:0] | Data length<br>00: 16 bits<br>01: 24 bits<br>10: 32 bits<br>11: Reserved<br>These bits should be configured when I2S mode is disabled. |

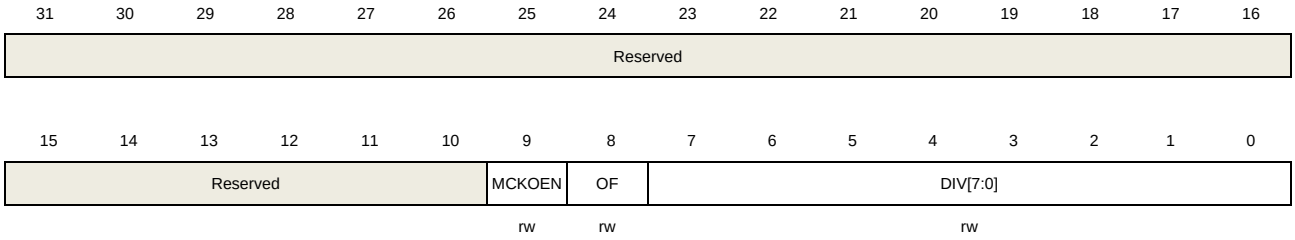
| | | |
|---|---|---|
| | | These bits are not used in SPI mode. |
| 0 | CHLEN | Channel length<br>0: 16 bits<br>1: 32 bits<br>The channel length must be equal to or greater than the data length.<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |

### 18.5.9. I2S clock prescaler register (SPI_I2SPSC)

Address offset: 0x20

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | MCKOEN | I2S_MCK output enable<br>0: Disable I2S_MCK output<br>1: Enable I2S_MCK output<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 8 | OF | Odd factor for the prescaler<br>0: Real divider value is DIV * 2<br>1: Real divider value is DIV * 2 + 1<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 7:0 | DIV[7:0] | Dividing factor for the prescaler<br>Real divider value is DIV * 2 + OF.<br>DIV must not be 0.<br>These bits should be configured when I2S mode is disabled.<br>These bits are not used in SPI mode. |

### 18.5.10. Quad-SPI mode control register (SPI_QCTL) of SPI1

Address offset: 0x80

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

| 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 | | | | | | | | | | | | | | QRD | QMOD |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | QRD | Quad-SPI mode read select.<br>0: SPI is in quad wire write mode.<br>1: SPI is in quad wire read mode.<br>This bit can only be configured when the SPI is not busy (the TRANS bit is cleared).<br>This bit is only available in SPI1. |
| 0 | QMOD | Quad-SPI mode enable.<br>0: SPI is in single wire mode.<br>1: SPI is in Quad-SPI mode.<br>This bit can only be configured when the SPI is not busy (the TRANS bit is cleared).<br>This bit is only available in SPI1. |

# 19. Operational amplifiers (OPA)

**This chapter applies to GD32E231K8Q6/GD32E231T8O6series.**

## 19.1. Overview

The OPA is a low noise, low voltage and low power operational amplifier with high gain-bandwidth product of 6.5MHz and slew rate of 5V/μs. The maximum input offset voltage is only 3.5mV and the input common mode range extends beyond the supply rails.

## 19.2. Characteristics

- Combinatorial work with ADC
- Low offset voltage: 1mV (typ)
- High gain: 104dB (typ)
- High gain bandwidth: 6.5MHz
- Rail-to-rail input/output
- Low supply voltage: +2.7 V to +3.6V

## 19.3. Function overview

GD32E231K8Q6 product supports 1-way OPA, where the OPA input is INPA/INNA and the output is VOUTA. GD32E231T8O6 product supports two channels of OPA, one OPA input is INPA/INNA, the output is VOUTA; The other OPA input is INPB/INNB and the output is VOUTB. When the MCU is powered on, the OPA starts working.

# 20. Appendix

## 20.1. List of abbreviations used in register

**Table 20-1. List of abbreviations used in register**

| abbreviations for registers | Descriptions |
|---|---|
| read/write (rw) | Software can read and write to this bit. |
| read-only (r) | Software can only read this bit. |
| write-only (w) | Software can only write to this bit. Reading this bit returns the reset value. |
| read/clear write 1 (rc_w1) | Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value. |
| read/clear write 0 (rc_w0) | Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value. |
| toggle (t) | The software can toggle this bit by writing 1. Writing 0 has no effect. |
| read/set (rs) | Software can read as well as set this bit to 1. Writing '0' has no effect on the bit value. |
| read/clear by read (rc_r) | Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value. |

## 20.2. List of terms

**Table 20-2. List of terms**

| Glossary | Descriptions |
|---|---|
| Word | Data of 32-bit length. |
| Half-word | Data of 16-bit length. |
| Byte | Data of 8-bit length. |
| IAP (in-application programming) | Writing 0 has no effect IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running. |
| ICP (in-circuit programming) | ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the boot loader while the device is mounted on the user application board. |
| Option bytes | Product configuration bits stored in the Flash memory. |
| AHB | Advanced high-performance bus. |
| APB | Advanced peripheral bus. |
| RAZ | Read-as-zero. |
| WI | Writes ignored. |
| RAZ/WI | Read-as-zero, writes ignored. |

## 20.3. Available peripherals

For availability of peripherals and their number across all MCU series types, refer to the corresponding device data datasheet.

# 21. Revision history

**Table 21-1. Revision history**

| Revision No. | Description | Date |
|---|---|---|
| 1.0 | Initial Release | Mar.8, 2019 |
| 1.1 | 1. Modify access mode and reset value of TIMER register.<br>2. Delete the I-BUS and D-BUS descriptions in Figure 1-2.<br>3. Add the descriptions of [31:16] bit domain in TIMERs register.<br>4. Modify the reset value of I2C_FMPCMG register. | Oct.8, 2019 |
| 1.2 | 1. Modify the FMC_WS register reset value to 0x00000030.<br>2. Modify the figures for Timing of EAPWM and Timing of CAPWM in TIMER chapter. | Mar.21, 2020 |
| 1.3 | 1. Modify the clock tree in the RCU chapter.<br>2. Modify Figure 17-1 of the I2C chapter.<br>3. Replace the SMBTYPE of chapter 17.3.11 in the I2C module with SMBSEL.<br>4. In chapter 12.1.3 of the WDGT module, add notes about entering deepsleep/standby mode immediately after feeding the WDGT.<br>5. In chapter 10.4.3 of the ADC module, add a description about the delay after the ADC enable. | Jul.2, 2020 |
| 1.4 | 1. Modify the description of the CKEN bit field of ***Control register 1 (USART_CTL1)*** in the USRT chapter.<br>2. In the ***VDD domain*** of PMU chapter, add the description of VDDA monitor.<br>3. Update ***Table 5-1. NVIC exception types in Cortex-M23***.<br>4. Modified the description of bit 15 of ***Transfer status register 0 (I2C_STAT0)*** in the I2C chapter. | Jun.7, 2021 |
| 1.5 | 1. Update ***Power management unit (PMU)*** chapter.<br>2. Update ***Serial peripheral interface/Inter-IC sound (SPI/I2S)*** chapter.<br>3. Update ***Inter-integrated circuit interface (I2C)*** chapter.<br>4. Update ***General-purpose I/Os (GPIO)*** chapter.<br>5. Update ***Reset and clock unit (RCU)*** chapter.<br>6. Update ***Analog to digital converter (ADC)*** chapter.<br>7. Update ***Comparator (CMP)*** chapter.<br>8. Update ***Timer (TIMERx)*** chapter.<br>9. Update ***Universal synchronous asynchronous receiver transmitter (USART)*** chapter.<br>10. Modify the timeout of ***Table 12-1. Min/max FWDGT timeout period at 40 kHz (IRC40K).***<br>11. Delete the OPA chapter. | Jul.18, 2022 |
| 1.6 | 1. Update ***Interrupt / event controller (EXTI)*** chapter. | Jun.9, 2023 |

| | | |
|---|---|---|
| | 2. Delete excess CI0 input sources and description of L2PWM input capture, modify the encoder and description of ROS/IOS in ***TIMER 14.3.4*** chapter.<br>3. Modify the description of starting a new transfer after modifying the DMA configuration in ***DMA 8.4.1*** chapter.<br>4. Delete the 16-bit programming of option byte and add factory value of option byte in ***FMC 2.3.9/2.3.10*** chapter<br>5. Modify the I2SSTDSEL to I2SSTD in Figure 18 55. I2S initialization sequence and Figure 18 56. I2S master reception disabling sequence. | |
| 2.0 | 1. Add GD32E235xx series support and modify the FLASH size to support up to 128KB. | Jul.17, 2024 |
| 2.1 | 1. Update ***Flash memory controller (FMC)*** chapter.<br>2. Update ***System and memory architecture*** chapter. Modify DBUS to AHB BUS. Change 0x0000 FFFF to 0x0001 FFFF and 0x2000 1FFF to 0x2000 3FFFF in ***Table 1 1. Memory map of GD32E23x series***.<br>3. Add OB_WP[31:16] in ***FMC 2.3.11*** chapter.<br>4. Modify the register access method of the GPIO module.<br>5. Modify the option bytes cannot be reprogrammed in ***FMC 2.3.12*** chapter.<br>6. Modification of DDRE bit in USART_CTL2 in USART module.<br>7. Add OPA support for GD32E231 series.<br>8. CMP consistency modification. | Feb.19, 2024 |
| 2.2 | 1. In the ***SPI/I2S chapter***, modify the misspelled word "Phillips" to "Philips".<br>2. In ***sections 6.4.9 and 6.4.10*** of GPIO, modify "x=A,B,C" to "A,B,F", consistent with the datasheet.<br>3. In ***section 11.3.2*** of CMP, modify the description regarding the output of CMP to the pin, and change the double-sided hysteresis in ***Figure 11-2*** to single-sided hysteresis.<br>4. In ***section 13.4.4*** of RTC, modify the description of the Status Register (RTC_STAT) bit fields. | Jul. 24, 2024 |
| 2.3 | 1. In the ***ADC chapter***, add a note regarding directly reading the ADC_RDATA register in the functional description of the run mode.<br>2. Add note in the software to avoid configuring CH0NP/CH0P bits to 0b11 in ***TIMER chapter***.<br>3. In ***sections 14.1.5*** and ***14.2.5***, remove the description "OCPRE_CLR_INT" from the CORC/CHxCOMCEN bit field and replace it with the source description for clearing.<br>4. Add the register bit RBNECM to the I2C_CTL1 register.<br>5. Add selection options for OPA, and modify the different numbers of OPA input/output channels between GD32E231K8Q6 and | Jan. 13, 2025 |

| | | |
|---|---|---|
| | GD32E231T8O6.<br>6. Delete “Module supply requirements: 2.4V to 3.6V, and typical power supply voltage is 3.3V.” in ***section 10.2***.<br>7. Delete “2.4V ≤VDDA≤ 3.6 V” in ***Table 10-2***.<br>8. Modify the on-chip flash description in ***section 1.3.2***.<br>9. Remove the bootloader interface description in ***section 1.4***, referring to the datasheet (DS). | |
| 2.4 | 1. Add support for GD32E225 series.<br>2. Adjust ***Figure 3-1***, delete BPOR from the Backup Domain, and add CMP to the VDDA domain.<br>3. Delete BPOR-related descriptions from ***section 3.3.1***.<br>4. Add CMP related descriptions. | Mar. 21, 2025 |
| 2.5 | 1. Add description of the 24-pin package in ***section 1.5.1***.<br>2. Modify the time out feature to change 35ms to 25ms in ***section 17.3.11***.<br>3. Add a Note in ***section 18.4.4*** of the I2S Clock chapter. | Aug. 8, 2025 |
| 2.6 | 1. ***Section 18.3.9***: Add precautions regarding CRC usage in SPI slave mode.<br>2. Update the description/caption for ***Figure 3-1*** Power supply overview.<br>3. Revise ***Section 3.3.2*** to update the description of the VDD/VDDA power domain.<br>4. Remove the IO2/IO3 pin driver functionality for SPI four-wire (quad) mode in ***Section 18.3.2***, and update the register bit-field descriptions in ***Section 18.5.10*** accordingly.<br>5. Update ***Table 11-1*** — Summary of CMP inputs and outputs.<br>6. Update ***Figure 14-1***: In Abort mode, CMP is used as TIMER0_BRKIN and cannot control polarity.<br>7. ***Section 16.3.12***: For byte mode (T=0), change the number of inter-frame bit periods inserted before a retransmitted data frame from 2.5 bits to 2 bits.<br>8. ***Section 16.3.4***: In the receive-sampling description, remove the clause "rwhatever it is a data bit, parity bit or stop bit," from the start-bit sampling text.<br>9. ***Section 16.4.6*** (RT[23:0]): In the RT[23:0] bit-field description add the note: "Note: When oversampling by 16, RT[23:0] cannot be configured as 0xFFFFFF."<br>10. ***Section 16.3.10***: For infrared IrDA mode — when using the low-power prescaler with a high division setting the waveform may not output correctly; add the note: "USART_BAUD register needs to be configured as 16*PSC[7:0]."<br>11. ***Section 2.2***: Add the feature description: "Flash memory supports 64-btis wide data reading" | Jan. 20, 2026 |

| | | |
|---|---|---|
| | 12. ***Section 12.1.3***: At the end of the functional description notes, add: "2. There must be an interval of 7 or more IRC32K clock cycles between two sonsecutive watchdog reload operations." | |

# Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 GigaDevice Semiconductor Inc. – All rights reserved