# GigaDevice Semiconductor Inc.

# GD32F1x0
# ARM® Cortex™-M3 32-bit MCU

# User Manual

Revision 3.0

( Jun. 2016 )

# Table of Contents

# List of Figures

# List of Tables

# 1. System and memory architecture

The system architecture of the devices of GD32F1x0 series that includes the ARM® Cortex™-M3 processor, bus architecture and memory organization will be described in the following sections. The Cortex™-M3 processor is a next generation processor core which offers many new features. Integrated and advanced features make the Cortex™-M3 processor suitable for market products that require microcontrollers with high performance and low power consumption. In brief, the Cortex™-M3 processor includes three AHB buses known as I-Code, D-Code and System buses. All memory accesses of the Cortex™-M3 processor are executed on the three buses according to the different purposes and the target memory spaces. The memory organization uses a Harvard architecture, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

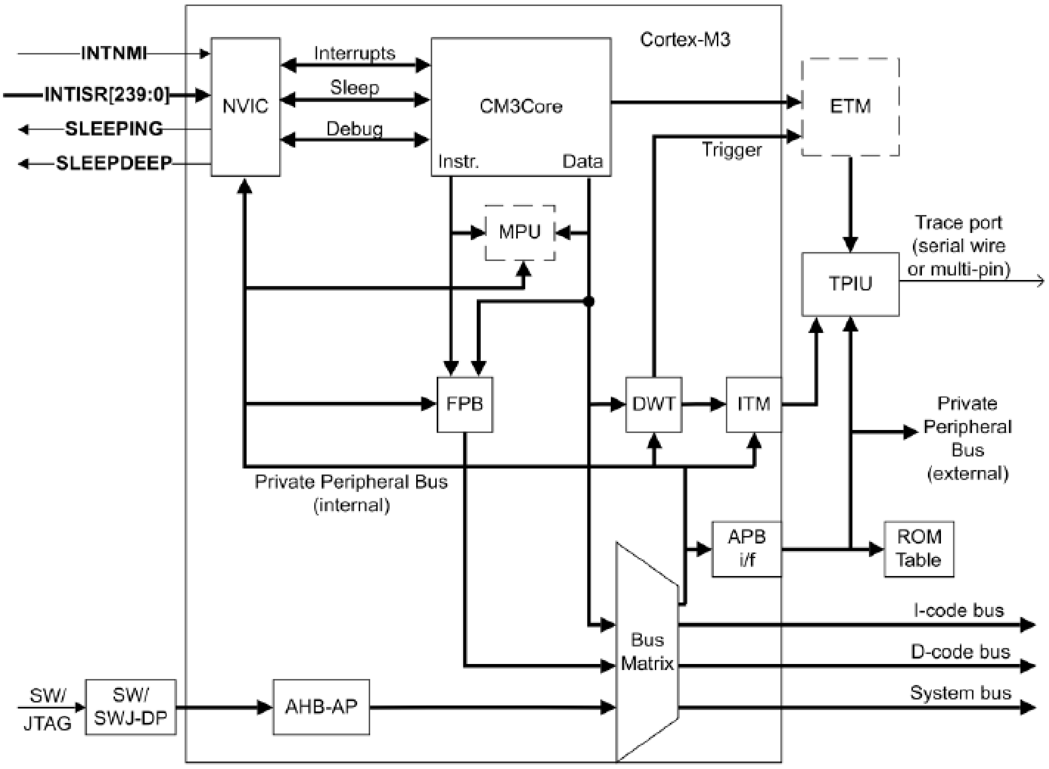
## 1.1. ARM Cortex-M3 processor

The Cortex™-M3 processor is a general-purpose 32-bit processor core especially suitable for products requiring high performance and low power consumption microcontrollers. It offers many new features such as a Thumb-2 instruction sets, hardware divider, low latency of interruption respond time, atomic bit-banding access and multiple buses for simultaneous accesses. The Cortex™-M3 processor is based on the ARMv7 architecture and supports both Thumb and Thumb-2 instruction sets. Some system peripherals listed below are also provided by Cortex™-M3:

- Internal Bus Matrix connected with I-Code bus, D-Code bus, System bus, Private Peripheral Bus (PPB) and debug accesses (AHB-AP)
- Nested Vectored Interrupt Controller (NVIC)
- Flash Patch and Breakpoint (FPB)
- Data Watchpoint and Trace (DWT)
- Instrumentation Trace Macrocell (ITM)
- Serial Wire JTAG Debug Port (SWJ-DP)
- Trace Port Interface Unit (TPIU)

The following figure shows the Cortex™-M3 processor block diagram. For more information, refer to the ARM® Cortex™-M3 Technical Reference Manual.

**Figure 1-1. Cortex™-M3 block diagram**



## 1.2. System architecture

The system architecture of the GD32F1x0 series is shown in the following figure. The AHB matrix based on AMBA 3.0 AHB-LITE is a multi-layer AHB, which enables parallel access paths between multiple masters and slaves in the system. There are four masters on the AHB matrix, including I-Code, D-Code, system bus of the Cortex™-M3 core and DMA. The I-Code bus is the instruction bus and also used for vector fetches from the Code region (0x0000 0000 ~ 0x1FFF FFFF) to the Cortex™-M3 core. The D-Code bus is used for loading/storing data and also for debugging access of the Code region. Similarly, the System bus is used for instruction/vector fetches, data loading/storing and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. The AHB matrix consists of five slaves, including I-Code and D-Code interfaces of 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. Both of the two APB buses are able to operate at full speed up to 72 MHz.

**Figure 1-2. Series system architecture of GD32F130xx and GD32F150xx devices**

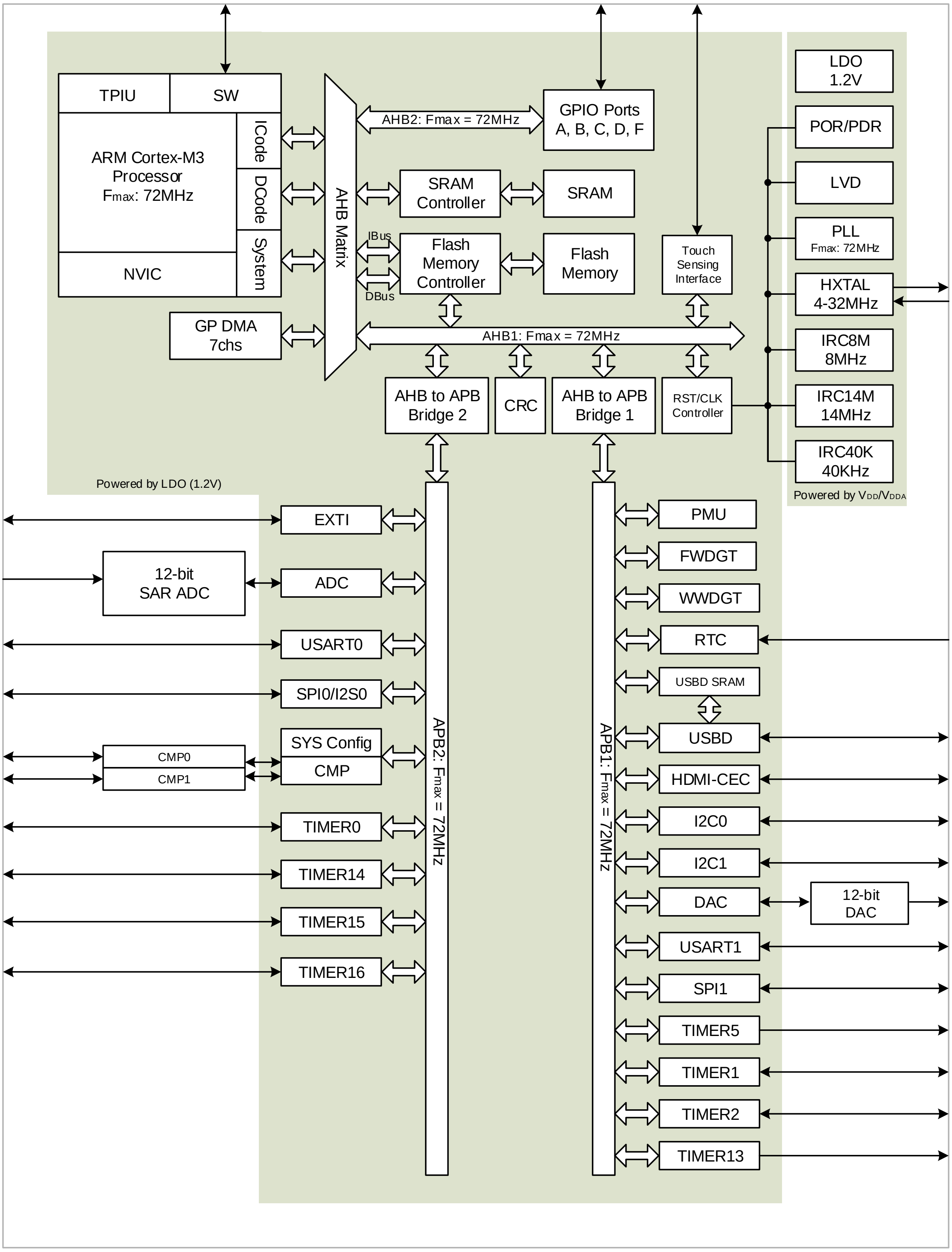
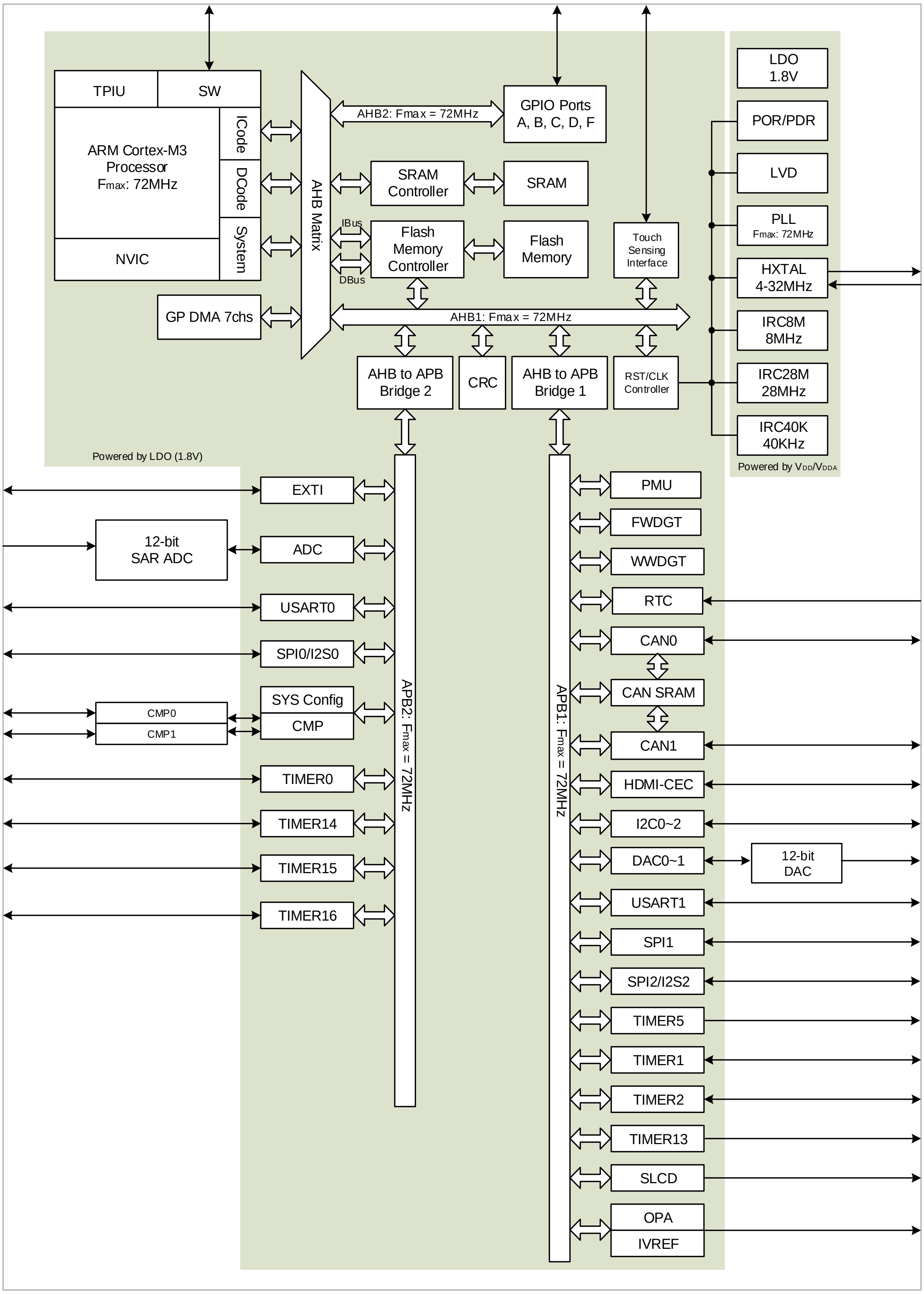


**Figure 1-3. Series system architecture of GD32F170xx and GD32F190xx devices**



## 1.3. Memory map

The ARM® Cortex™-M3 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. The instruction code and data are both located in the same memory address space but in different address ranges. 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™-M3 since it has a 32-bit bus address width. Additionally, a pre-defined memory map is provided by the Cortex™-M3 processor to reduce the software complexity of repeated implementation of different device vendors. However, some regions are used by the ARM® Cortex™-M3 system peripherals. The following figure shows the memory map of the GD32F1x0 series of devices, 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. The APB1 peripherals are located at the address region from 0x4000 0000 to 0x4000 FFFF, while the APB2 peripherals are located from 0x4001 0000 to 0x4001 7FFF. The address region from 0x4002 0000 to 0x4002 FFFF is used for AHB1 peripherals, and the address region from 0x4800 0000 to 0x4800 FFFF is used for AHB2 peripherals.

**Figure 1-4. Memory map of GD32F130xx and GD32F150xx devices**

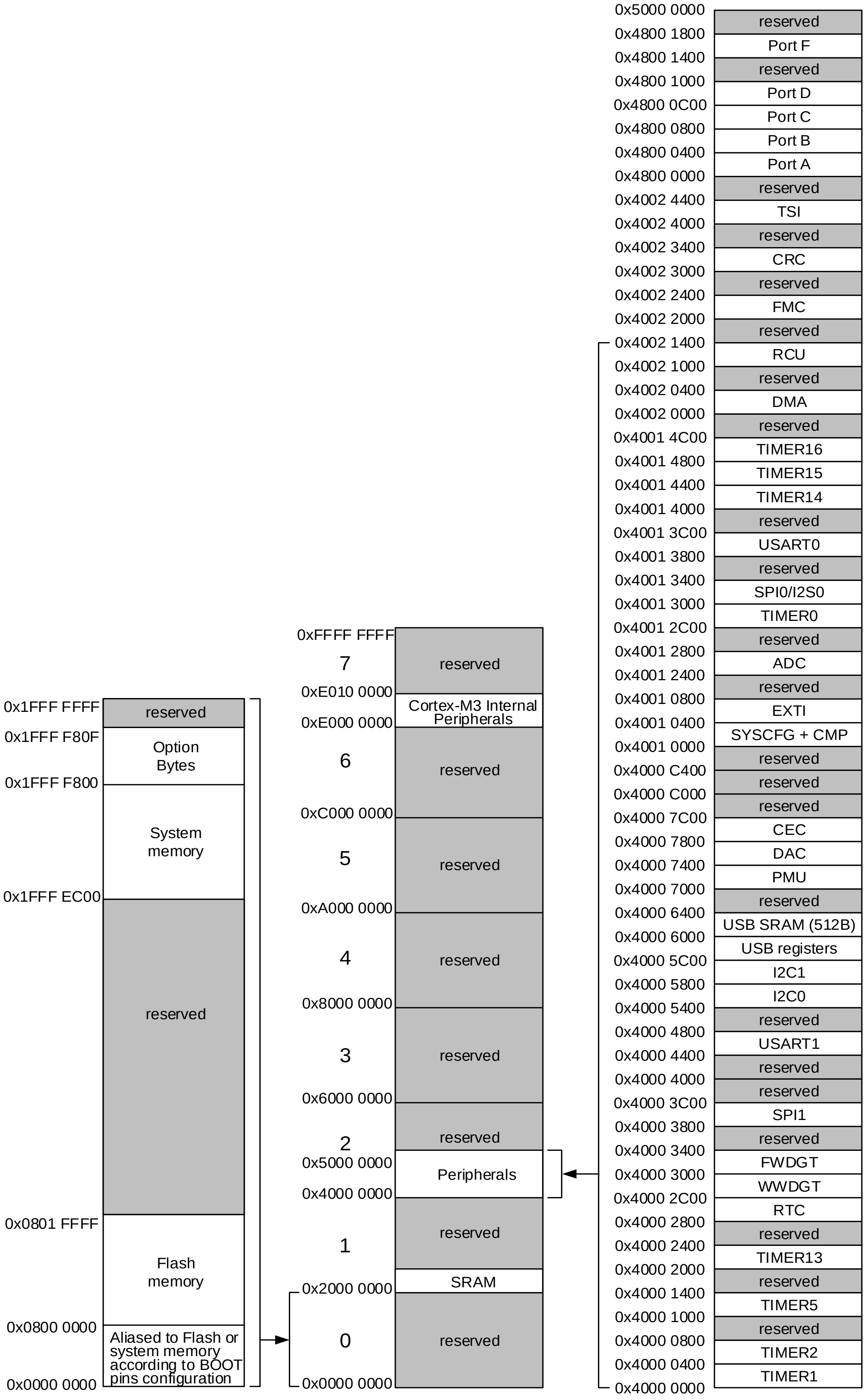
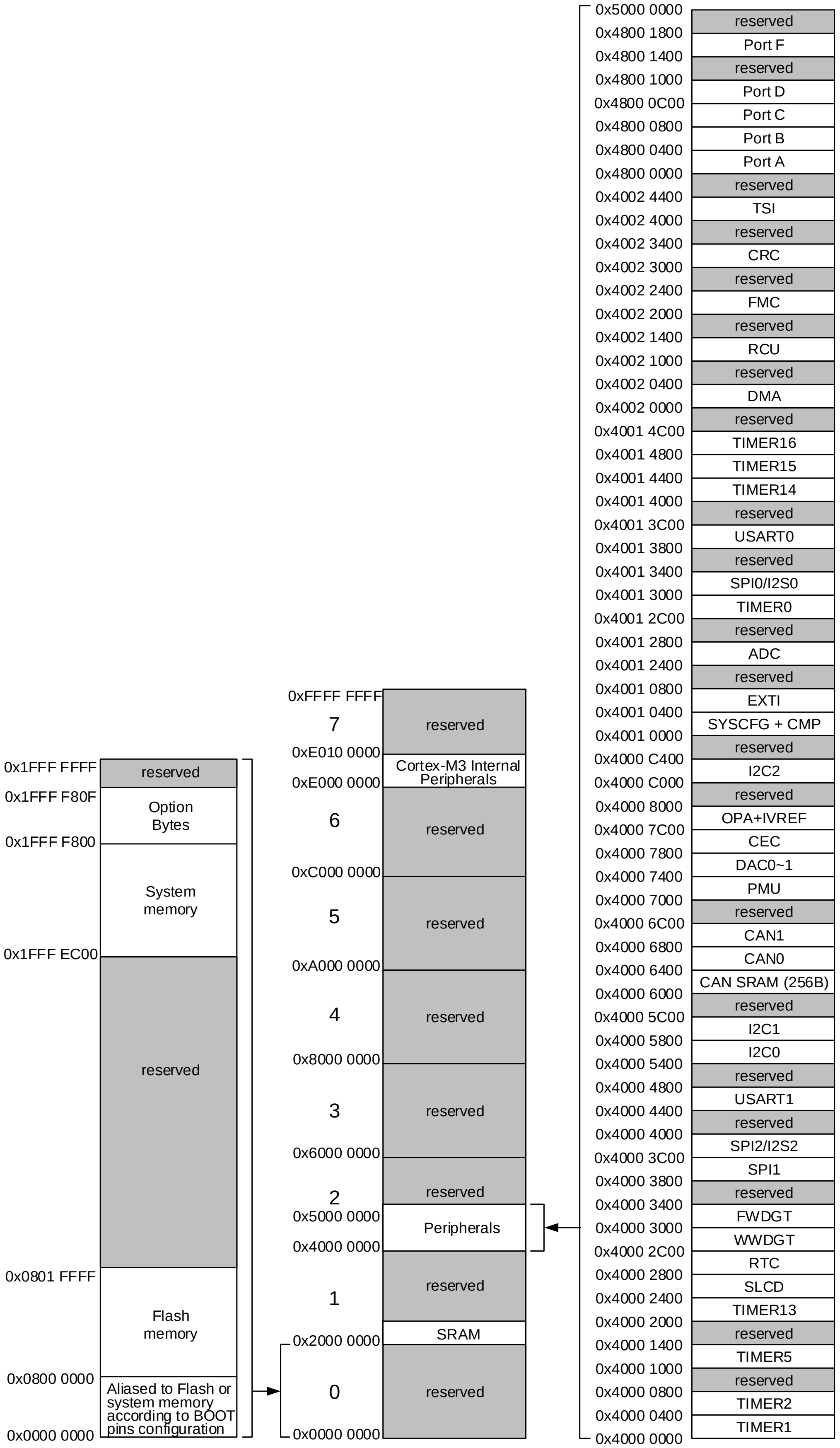


**Figure 1-5. Memory map of GD32F170xx and GD32F190xx devices**



### 1.3.1. Bit-banding

In order to reduce the time of read-modify-write operations, the Cortex™-M3 processor provides a bit-banding function to perform a single atomic bit operation. The memory map includes two bit-band regions. These occupy the SRAM and Peripherals respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

A mapping formula shows how to reference each word in the alias region to a corresponding bit, or target bit, in the bit-band region. The mapping formula is:

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

where:

- Bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- Bit_band_base is the starting address of the alias region.
- Byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- Bit_number is the bit position (0-7) of the targeted bit.

For example, to access bit 7 of address 0x2000 0200, the bit-band alias is:

bit_word_addr = 0x2200 0000 + (0x200 * 32) + (7 * 4) = 0x2200 401C

Writing to address 0x2200 401C will cause bit 7 of address 0x2000 0200 change while a read to address 0x2200 401C will return 0x01 or 0x00 according to the value of bit 7 at the SRAM address 0x2000 0200.

### 1.3.2. On-chip SRAM memory

The devices of GD32F1x0 series contain up to 8 KB 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 OB_SRAM_PARITY_CHECK in the user option byte (refer to Chapter 3.3.9 Option bytes). 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 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.

### 1.3.3. On-chip Flash memory

**For GD32F130xx and GD32F150xx devices**

The devices provide up to 64 KB of on-chip flash memory. The flash memory consists of up to 64 KB main flash organized into 64 pages with 1 KB capacity per page and a 3 KB information block for the boot loader. The following table shows details.

**Table 1-1. Flash module organization**

| Block | Name | Address | Size |
|---|---|---|---|
| Main Flash Block | Page 0 | 0x0800 0000 - 0x0800 03FF | 1 Kbytes |
| | Page 1 | 0x0800 0400 - 0x0800 07FF | 1 Kbytes |
| | Page 2 | 0x0800 0800 - 0x0800 0BFF | 1 Kbytes |
| | . | . | . |
| | . | . | . |
| | Page 63 | 0x0800 FC00 - 0x0800 FFFF | 1 Kbytes |
| Information Block | System memory | 0x1FFF EC00 - 0x1FFF F7FF | 3 Kbytes |
| | Option Bytes | 0x1FFF F800 - 0x1FFF F80F | 16 bytes |

Read accesses to the preceding 32 pages can be performed 32 bits per cycle without any wait state. All of byte, half-word (16 bits) and word (32 bits) read accesses are supported. The flash memory can be programmed half-word (16 bits) or word (32 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.

**For GD32F170xx and GD32F190xx devices**

The devices provide up to 128 KB of on-chip flash memory. 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 following table shows details.

**Table 1-2. Flash module organization**

| Block | Name | Address | Size |
|---|---|---|---|
| Main Flash Block | Page 0 | 0x0800 0000 - 0x0800 03FF | 1 Kbytes |
| | Page 1 | 0x0800 0400 - 0x0800 07FF | 1 Kbytes |
| | Page 2 | 0x0800 0800 - 0x0800 0BFF | 1 Kbytes |
| | . | . | . |
| | . | . | . |
| | Page 127 | 0x0801 FC00 - 0x0801 FFFF | 1 Kbytes |
| Information Block | System memory | 0x1FFF EC00 - 0x1FFF | 3 Kbytes |

| | | | |
|---|---|---|---|
| | | F7FF | |
| | Option Bytes | 0x1FFF F800 - 0x1FFF F80F | 16 bytes |

Read accesses to the preceding 32 pages can be performed 32 bits per cycle without any wait state. All of byte, half-word (16 bits) and word (32 bits) read accesses are supported. The flash memory can be programmed half-word (16 bits) or word (32 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 devices of GD32F1x0 series provides three kinds of boot sources which can be selected using the bit OB_BOOT1_n in the user option byte (refer to Chapter 3.3.9 Option bytes) 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 OB_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.

**Table 1-3. 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 OB_BOOT1_n value.

After power-on sequence or a system reset, the ARM® Cortex™-M3 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. In GD32F1x0 devices, the boot loader can be activated through one of the following serial interfaces: USART0 or USART1.

# 1.5. System configuration registers (SYSCFG)

## 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 OB_BOOT1_n option bit after reset)

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 | | | | | | | | | | | | 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 | | | | | | BOOT_MODE[1:0] | |
| | | | 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 disabled.<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_CH0 and TIMER16_UP DMA requests are mapped on DMA channel 0)<br>1: Remap (TIMER16_CH0 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_CH0 and TIMER15_UP DMA requests are mapped on DMA channel 2)<br>1: Remap (TIMER15_CH0 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)<br>1: Remap (USART0_RX DMA requests are mapped on DMA channel 4) |
| 9 | USART0_TX_DMA_RMP | USART0_TX DMA request remapping enable |

| | | |
|---|---|---|
| | | 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) |
| 7:2 | Reserved | Must be kept at reset value |
| 1:0 | BOOT_MODE<br>[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 OB_BOOT1_n option bit value.<br>x0: Boot from the Main Flash<br>01: Boot from the system memory<br>11: Boot from the embedded SRAM |

### 1.5.2. System configuration register 1 (SYSCFG_CFG1)

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 | | | | | | | | | | | | SLCD_DECA | | | Reserved |
| | | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:1 | SLCD_DECA | Decoupling capacitance connection for SLCD<br>Bit1: Decoupling capacitance connection to PB2 or not<br>Bit2: Decoupling capacitance connection to PB12 or not<br>Bit3: Decoupling capacitance connection to PB0 or not |
| 0 | Reserved | Must be kept at reset value |

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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[3:0] | EXTI 3 sources selection<br>X000: PA3 pin<br>X001: PB3 pin<br>X010: PC3 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 11:8 | EXTI2_SS[3:0] | EXTI 2 sources selection<br>X000: PA2 pin<br>X001: PB2 pin<br>X010: PC2 pin<br>X011: PD2 pin<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 7:4 | EXTI1_SS[3:0] | EXTI 1 sources selection<br>X000: PA1 pin<br>X001: PB1 pin<br>X010: PC1 pin<br>X011: Reserved<br>X100: Reserved<br>X101: PF1 pin<br>X110: Reserved<br>X111: Reserved |
| 3:0 | EXTI0_SS[3:0] | EXTI 0 sources selection<br>X000: PA0 pin<br>X001: PB0 pin<br>X010: PC0 pin<br>X011: Reserved<br>X100: Reserved |

X101: PF0 pin

X110: Reserved

X111: Reserved

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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[3:0] | EXTI 7 sources selection<br>X000: PA7 pin<br>X001: PB7 pin<br>X010: PC7 pin<br>X011: Reserved<br>X100: Reserved<br>X101: PF7 pin<br>X110: Reserved<br>X111: Reserved |
| 11:8 | EXTI6_SS[3:0] | EXTI 6 sources selection<br>X000: PA6 pin<br>X001: PB6 pin<br>X010: PC6 pin<br>X011: Reserved<br>X100: Reserved<br>X101: PF6 pin<br>X110: Reserved<br>X111: Reserved |
| 7:4 | EXTI5_SS[3:0] | EXTI 5 sources selection<br>X000: PA5 pin<br>X001: PB5 pin<br>X010: PC5 pin<br>X011: Reserved |

| | | |
|---|---|---|
| | | X100: Reserved<br>X101: PF5 pin<br>X110: Reserved<br>X111: Reserved |
| 3:0 | EXTI4_SS[3:0] | EXTI 4 sources selection<br>X000: PA4 pin<br>X001: PB4 pin<br>X010: PC4 pin<br>X011: Reserved<br>X100: Reserved<br>X101: PF4 pin<br>X110: Reserved<br>X111: Reserved |

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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[3:0] | EXTI 11 sources selection<br>X000: PA11 pin<br>X001: PB11 pin<br>X010: PC11 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 11:8 | EXTI10_SS[3:0] | EXTI 10 sources selection<br>X000: PA10 pin<br>X001: PB10 pin<br>X010: PC10 pin |

| | | |
|---|---|---|
| | | X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 7:4 | EXTI9_SS[3:0] | EXTI 9 sources selection<br>X000: PA9 pin<br>X001: PB9 pin<br>X010: PC9 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 3:0 | EXTI8_SS[3:0] | EXTI 8 sources selection<br>X000: PA8 pin<br>X001: PB8 pin<br>X010: PC8 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI15_SS [3:0] | | | | EXTI14_SS [3:0] | | | | EXTI13_SS [3:0] | | | | EXTI12_SS [3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:12 | EXTI15_SS[3:0] | EXTI 15 sources selection<br>X000: PA15 pin<br>X001: PB15 pin |

| | | |
|---|---|---|
| | | X010: PC15 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |
| 11:8 | EXTI14_SS[3:0] | 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[3:0] | EXTI 13 sources selection<br>X000: PA13 pin<br>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[3:0] | EXTI 12 sources selection<br>X000: PA12 pin<br>X001: PB12 pin<br>X010: PC12 pin<br>X011: Reserved<br>X100: Reserved<br>X101: Reserved<br>X110: Reserved<br>X111: Reserved |

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

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 | | | | | | | SRAM_PCEF | Reserved | | | | | LVD_<br>LOCK | SRAM_<br>PARITY_<br>ERROR_<br>LOCK | LOCK<br>UP_<br>LOCK |
| | | | | | | | rw | | | | | | 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.<br>0: The LVD interrupt is disconnected from the break input of TIMER0/14/15/16. LVDEN and LVDT[2:0] in the PMU_CTL register can be programmed.<br>1: The LVD interrupt is connected from the break input of TIMER0/14/15/16. LVDEN and LVDT[2:0] in the PMU_CTL register are read only. |
| 1 | SRAM_PARITY_<br>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-M3 LOCKUP output lock<br>This bit is set by software and cleared by a system reset.<br>0: The Cortex-M3 LOCKUP output is disconnected from the break input of TIMER0/14/15/16<br>1: The Cortex-M3 LOCKUP output is connected from the break input of TIMER0/14/15/16 |

## 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<br>[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<br>[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. Power management unit (PMU)

## 2.1. Introduction

The power consumption is regarded as one of the most important issues for the devices of GD32F1x0 series. Accordingly the Power management unit (PMU), provide 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 GD32F130xx and GD32F150xx devices, there are three power domains, including $V_{DD}$ domain, 1.2V domain, and Backup domain, as is shown in the following figure. For GD32F170xx and GD32F190xx devices, there are three power domains, including $V_{DD}$ domain, 1.8V domain, and Backup domain, as is shown in the following figure. The power of the $V_{DD}$ domain is supplied directly by $V_{DD}$. An embedded LDO in the $V_{DD}$ domain is used to supply the 1.2V/1.8V domain power. A power switch is implemented for the Backup domain. It can be powered from the $V_{BAT}$ voltage when the main $V_{DD}$ supply is shut down.

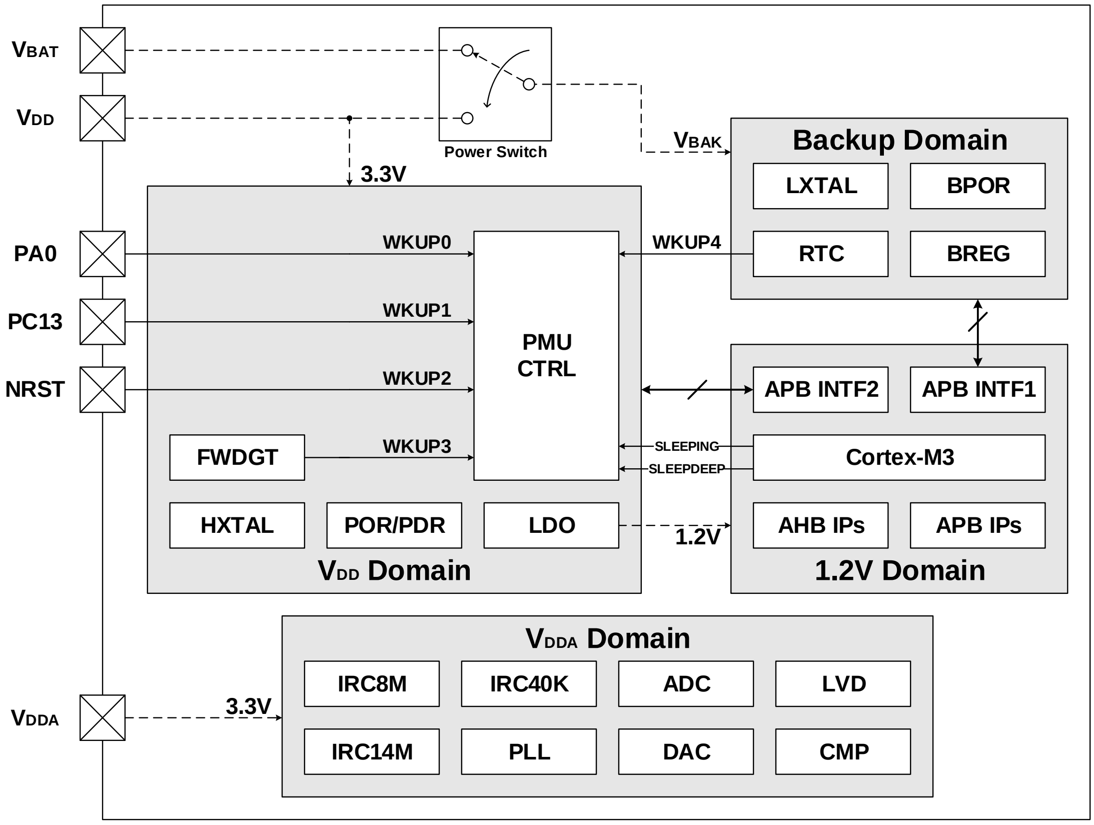
## 2.2. Main features

- Three power domains: $V_{BAT}$, $V_{DD}$ and 1.2V power domains for GD32F130xx and GD32F150xx devices or 1.8V power domains for GD32F170xx and GD32F190xx devices
- Three power saving modes: Sleep, Deep-sleep and Standby modes
- Internal Voltage regulator supplies 1.2V voltage source for GD32F130xx and GD32F150xx devices or 1.8 V voltage source for GD32F170xx and GD32F190xx devices
- 20 bytes of backup register powered by $V_{BAK}$ for data protection of user application data when in the Standby mode
- Low Voltage Detector can issue an interrupt or event when the power is lower than a programmed threshold
- Battery power ($V_{BAT}$) for Backup domain when $V_{DD}$ is shut down
- Calibration register for storing the RTC calibration value

## 2.3. Function description

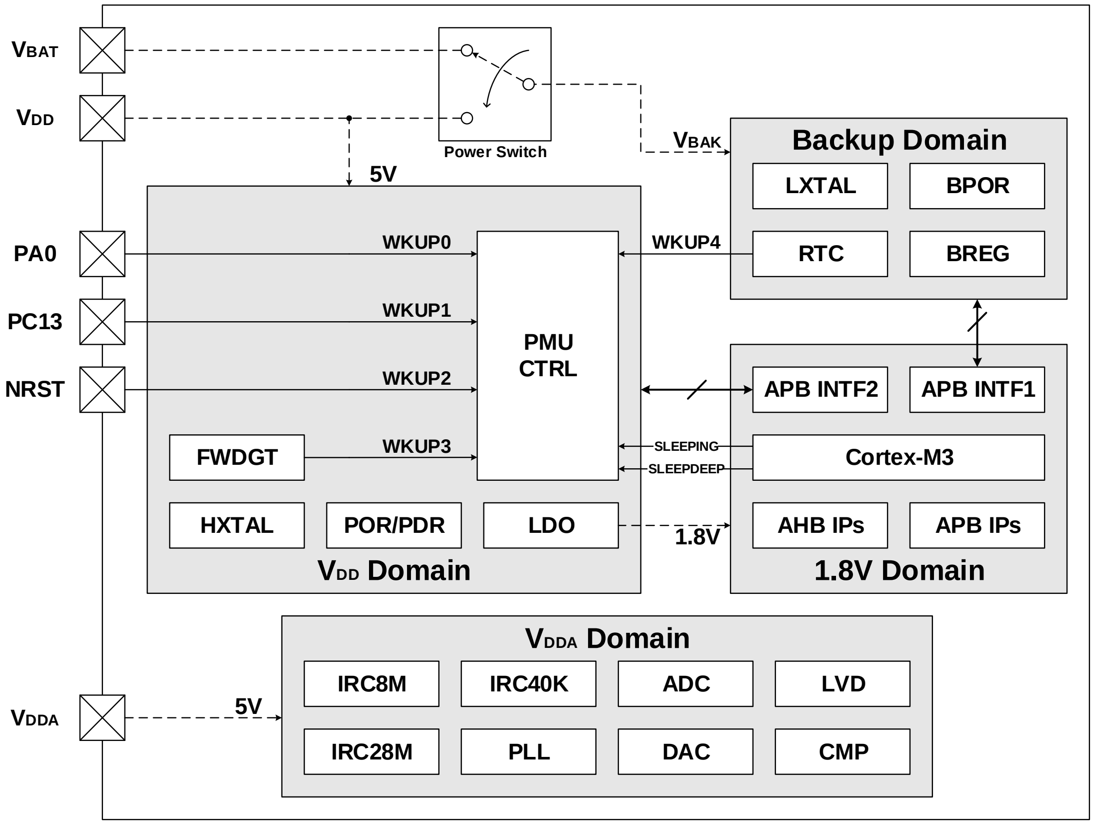
Figure below provides details on the internal configuration of the PMU and the relevant power domains.

**Figure 2-1. Power supply overview of GD32F130xx and GD32F150xx devices**



| | | |
|---|---|---|
| LVD: Low Voltage Detector | LDO: Voltage Regulator | BREG: Backup Registers |
| POR: Power On Reset | PDR: Power Down Reset | BPOR: VBAK Power On Reset |

**Figure 2-2. Power supply overview of GD32F170xx and GD32F190xx devices**



LVD: Low Voltage Detector
LDO: Voltage Regulator
BREG: Backup Registers
POR: Power On Reset
PDR: Power Down Reset
BPOR: VBAK Power On Reset

### 2.3.1. Battery backup domain

The Backup Domain contains BPOR (Backup Domain Power on Reset), BREG (Backup Registers), RTC (Real Time Clock), LXTAL (Low Speed Crystal Oscillator), and three pads, including PC13, PC14, and PC15. It is powered by the $V_{DD}$ or the battery power source ($V_{BAT}$) selected by the internal power switch. To retain the content of the Backup registers and supply the RTC function, when $V_{DD}$ is turned off, $V_{BAT}$ pin can be connected to an optional standby voltage supplied by a battery or by another source. The switch is controlled by the Power Down Reset circuit in the $V_{DD}$ domain. If no external battery is used in the application, it is recommended to connect $V_{BAT}$ externally to $V_{DD}$ with a 100nF external ceramic decoupling capacitor.

The Backup Domain reset sources include BPOR and the Backup Domain software reset. The BPOR signal forces the device to stay in the reset mode until $V_{BAK}$ is completely powered up. Also 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 RTC circuit can be derived from the Internal 40KHz RC oscillator (IRC40K), the Low Speed Crystal oscillator (LXTAL) or HXTAL clock divided by 32. When $V_{DD}$ is shut down, only LXTAL is valid for RTC. Before entering the power saving mode by executing the WFI/WFE instruction, the Cortex™-M3 needs to setup the RTC register with an expected wakeup time and enable the wakeup function to achieve the RTC timer wakeup event. After entering the power saving mode for a certain amount of time, the RTC alarm will be asserted to wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the RTC chapter.

Five 32-bit registers, up to 20 bytes, are provided located in the Backup Domain for user application data storage. These registers are powered by $V_{BAK}$ which constantly supplies power when the $V_{DD}$ power is switched off. The Backup Registers are only reset by the Backup Domain power-on-reset or the Backup Domain software reset.

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, TAMPER pin, RTC Calibration, RTC Alarm or second output. (refer to Chapter 20: Real-time Clock (RTC))
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

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

- PC13 can be used as TAMPER pin, RTC Alarm or second output. (refer to Chapter 20: Real-time Clock (RTC))
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

**Note:** Since PC13, PC14 and PC15 are supplied through the Power Switch, which 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.

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

Most of analog modules are located in the $V_{DDA}$ domain, including IRC8M (Internal 8MHz RC oscillator), IRC14M (Internal 14MHz RC oscillator) for GD32F130xx and GD32F150xx devices or IRC28M (Internal 28MHz RC oscillator) for GD32F170xx and GD32F190xx devices, IRC40K (Internal 40KHz RC oscillator), PLL (Phase Locking Loop), ADC (Analog to Digital Converter), DAC (Digital to Analog Converter), LVD (Low Voltage Detector), and Comparator. Besides, LDO (Voltage Regulator), POR/PDR (Power On/Down Reset), HXTAL (High Speed Crystal oscillator), FWDGT (Free Watchdog Timer), the power control logic, and pads (except PC13, PC14, and PC15) are implemented in the $V_{DD}$ domain. The LDO is implemented to supply power for the 1.2V domain in GD32F130xx and GD32F150xx devices or 1.8V domain in GD32F170xx and GD32F190xx devices. Generally, digital circuits are powered by $V_{DD}$, while most of analog circuits are powered by $V_{DDA}$. The independent power supply $V_{DDA}$ is implemented to achieve better performance of analog circuits, especially to improve the ADC and the DAC conversion accuracy. The voltage of $V_{DDA}$ can be equal or higher than that of $V_{DD}$. $V_{DDA}$ can be externally connected to $V_{DD}$ through the external filtering circuit that avoids noise on $V_{DDA}$. Or, if $V_{DDA}$ is different from $V_{DD}$, $V_{DDA}$ must always be higher. In this case, an external Shottky diode may be implemented between $V_{DD}$ and $V_{DDA}$ for safety.

The LVD is used to detect whether the $V_{DD}$ or $V_{DDA}$ supply voltage is lower than the low voltage detector threshold. The function is configured by the Control Register (PMU_CTL). The LVDEN bit controls whether LVD is enabled or not, while the LVDT[2:0] bits select the low voltage detector threshold from 2.2V to 2.9V for GD32F130xx and GD32F150xx devices or 2.4V to 4.5V for GD32F170xx and GD32F190xx devices. The low power detector status flag (LVDF) is located in the Power Control/Status Register (PMU_CS). Besides, the low voltage detected event is also connected to the EXTI line 16. Users can generate a corresponding interrupt through configuring the EXTI line 16.

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 following figure shows the relationship between the supply voltage and the power reset signal. $V_{POR}$ indicates the threshold of power on reset, while $V_{PDR}$ means the threshold of power down reset. For GD32F130xx and GD32F150xx devices, $V_{PDR}$ is configurable. There are two $V_{PDR}$ values which can be selected by the PDVSEL bit in the RCU_PDVSEL registers (Refer to Chapter4 RCU registers). When the lower one is selected, it is strongly recommended that neither programming nor erasing is performed to the flash memory since it may fail when the voltage is low enough nearly the threshold.

**Figure 2-3. Waveform of the power reset**

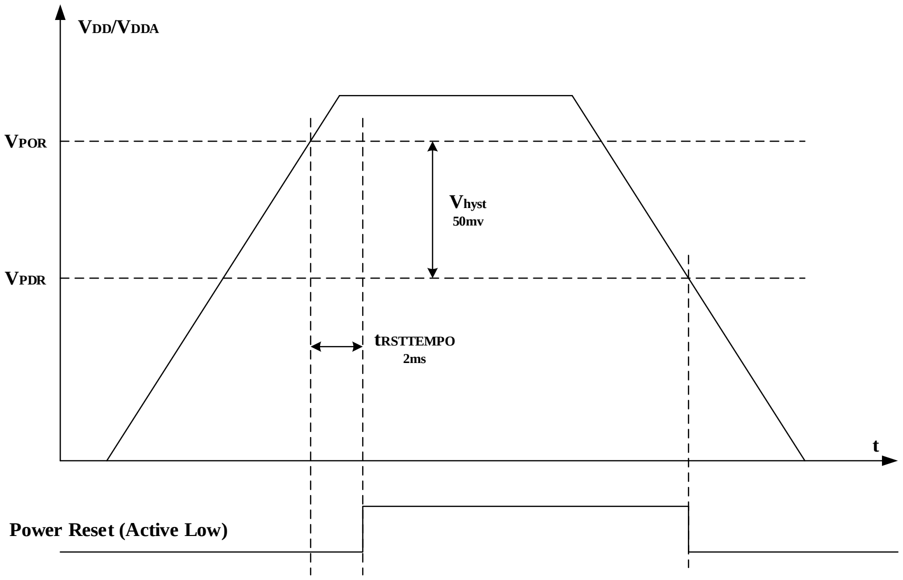
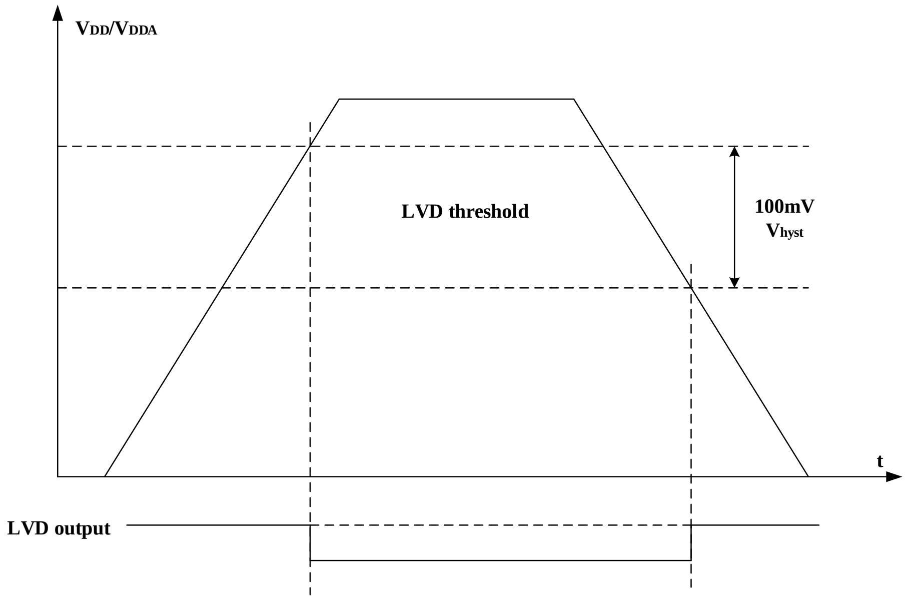


Figure 2-4. Waveform of LVD threshold



### 2.3.3. 1.2V power domain for GD32F130xx and GD32F150xx devices

The main functions that include Cortex™-M3 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the $V_{DD}$ domain, etc., are located in the 1.2V power domain. Once the 1.2V is powered up, the POR will generate a reset sequence on the 1.2V power domain.

### 2.3.4. 1.8V power domain for GD32F170xx and GD32F190xx devices

The main functions that include Cortex™-M3 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the $V_{DD}$ domain, etc., are located in the 1.8V power domain. Once the 1.8V is powered up, the POR will generate a reset sequence on the 1.8V power domain.

### 2.3.5. Power saving modes

After a system reset or a power reset, the GD32F1x0 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. 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™-M3. In Sleep mode, only clock of Cortex™-M3 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex™-M3 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. 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™-M3 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™-M3. In Deep-sleep mode, all clocks in the 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices are off, and all of IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL are disabled. 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™-M3 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. Any interrupt or wakeup event from EXTI lines can wake up the system from the Deep-sleep mode. 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 RTC Alarm must be reset. If not, the program will skip the entry process of Deep-sleep mode to continue to executive the following procedure.

### Standby Mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex™-M3, too. In Standby mode, the whole 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices is power off, the LDO is shut down, and all of IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and set the STBMOD bit in the PMU_CTL register, and clear the WUF bit in the PMU_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed. There are five wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC alarm, the FWDGT reset, and the rising edge on WKUP0 or WKUP1 pin. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers (except Backup Registers) are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex™-M3 will execute instruction code from the 0x0000 0000 address.

**Table 2-1. Power saving mode summary**

| Mode | Sleep | Deep-sleep | Standby |
|---|---|---|---|
| **Description** | Only CPU clock is off | 1. All clocks in the 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices are off<br>2. Disable IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL | 1. 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices is power off<br>2. Disable IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL |
| **LDO Status** | On | On or in 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 for WFE | Any interrupt or event from EXTI lines | 1. NRST pin<br>2. RTC alarm<br>3. FWDGT reset<br>4. WKUP0 pin<br>5. WKUP1 pin |
| **Wakeup Latency** | None | IRC8M wakeup time<br>LDO wakeup time if LDO is in low power mode | Power on sequence |

## 2.4. PMU registers

### 2.4.1. Control register (PMU_CTL)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0x00

Reset value: 0x0000 0000 (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 | | | | | | | BKPWEN | LVDT[2:0] | | | LVDEN | STBRST | WURST | STBMOD | LDOLP |

rw rw rw rc_w1 rc_w1 rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | 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: 2.2V<br>001: 2.3V<br>010: 2.4V<br>011: 2.5V<br>100: 2.6V<br>101: 2.7V<br>110: 2.8V<br>111: 2.9V |
| 4 | LVDEN | Low Voltage Detector Enable<br>0: Disable Low Voltage Detector<br>1: Enable Low Voltage Detector<br>**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™-M3 enters SLEEPDEEP mode<br>1: Enter the Standby mode when the Cortex™-M3 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. |

### For GD32F170xx and GD32F190xx devices

Address offset: 0x00

Reset value: 0x0000 0000 (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 | | | | | | | BKPWEN | LVDT[2:0] | | | LVDEN | STBRST | WURST | STBMOD | LDOLP |
| | | | | | | | rw | rw | | | rw | rc_w1 | rc_w1 | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | 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: 2.4V<br>001: 2.7V<br>010: 3.0V<br>011: 3.3V<br>100: 3.6V<br>101: 3.9V<br>110: 4.2V<br>111: 4.5V |
| 4 | LVDEN | Low Voltage Detector Enable<br>0: Disable Low Voltage Detector<br>1: Enable Low Voltage Detector<br>**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 |

| | | |
|---|---|---|
| | | This bit is always read as 0. |
| 1 | STBMOD | Standby Mode<br>0: Enter the Deep-sleep mode when the Cortex™-M3 enters SLEEPDEEP mode<br>1: Enter the Standby mode when the Cortex™-M3 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. |

### 2.4.2. Power control/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).

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td>WUPEN1</td><td>WUPEN0</td><td colspan="5">Reserved</td><td>LVDF</td><td>STBF</td><td>WUF</td></tr>
<tr><td colspan="6"></td><td>rw</td><td>rw</td><td colspan="5"></td><td>r</td><td>r</td><td>r</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9 | WUPEN1 | WKUP1 Pin Enable<br>0: Disable WKUP1 pin function<br>1: Enable WKEP1 pin function<br>If WUPEN1 is set before entering the power saving mode, a rising edge on the WKUP1 pin wakes up the system from the power saving mode. As the WKUP1 pin is active high, this bit will setup an input pull down mode when the bit is high. |
| 8 | WUPEN0 | WKUP0 Pin Enable<br>0: Disenable WKUP0 pin function<br>1: Enable WKUP0 pin function<br>If WUPEN0 is set before entering the power saving mode, a rising edge on the WKUP0 pin wakes up the system from the power saving mode. As the WKUP0 pin is active high, this bit will setup an input pull down mode when the bit is high. |
| 7:3 | Reserved | Must be kept at reset value |
| 2 | LVDF | Low Voltage Detector Status Flag |

| | | |
|---|---|---|
| | | 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 been in Standby mode<br>1: The device has been in 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: A wakeup event has been received from the WKUP pin or the RTC alarm<br>This bit is cleared only by a POR/PDR or by setting the WURST bit in the PMU_CTL register. |

# 3. Flash memory controller (FMC)

## 3.1. Introduction

The Flash Memory Controller, FMC, provides all the necessary functions for the on-chip flash memory. There is no waiting time within 32K bytes while CPU executes instruction. It also provides page erase, mass erase, and word/half word program for flash memory.

## 3.2. Main features

**For GD32F130xx and GD32F150xx devices：**

- Up to 64 KB of on-chip flash memory for storing instruction/data
- No waiting time within 32K bytes when CPU executes instruction
- A long delay when fetch 32K ~ 64K bytes date from flash
- 3K bytes information block for boot loader
- 16 bytes option bytes block for user application requirements
- 1K bytes page size
- Word or half 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

**For GD32F170xx and GD32F190xx devices：**

- Up to 128 KB of on-chip flash memory for storing instruction/data
- No waiting time within 32K bytes when CPU executes instruction
- A long delay when fetch 32K ~ 128K bytes date from flash
- 3K bytes information block for boot loader
- 16 bytes option bytes block for user application requirements
- 1K bytes page size
- Word or half 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

## 3.3. Function description

### 3.3.1. Flash memory architecture

**For GD32F130xx and GD32F150xx devices**

The flash memory consists of up to 64 KB main flash organized into 64 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 64 pages which can be erased individually. The following table shows the base address, size.

**Table 3-1. 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 |
| | .<br>.<br>. | .<br>.<br>. | .<br>.<br>. |
| | Page 63 | 0x0800 FC00 - 0x0800 FFFF | 1KB |
| Information Block | Boot Loader | 0x1FFF EC00- 0x1FFF F7FF | 3KB |
| Option byte Block | Option byte | 0x1FFF F800 - 0x1FFF F80F | 16B |

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

**For GD32F170xx and GD32F190xx devices**

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 following table shows the base address, size.

**Table 3-2 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 |
| | .<br>.<br>. | .<br>.<br>. | .<br>.<br>. |
| | 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 |

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

### 3.3.2. 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 IBUS or DBUS from the CPU.

### 3.3.3. 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 OBWE bit in FMC_CTL register to 1. The software can set OBWE bit to 0 to protect the OBPG bit and OBER bit in FMC_CTL register again.

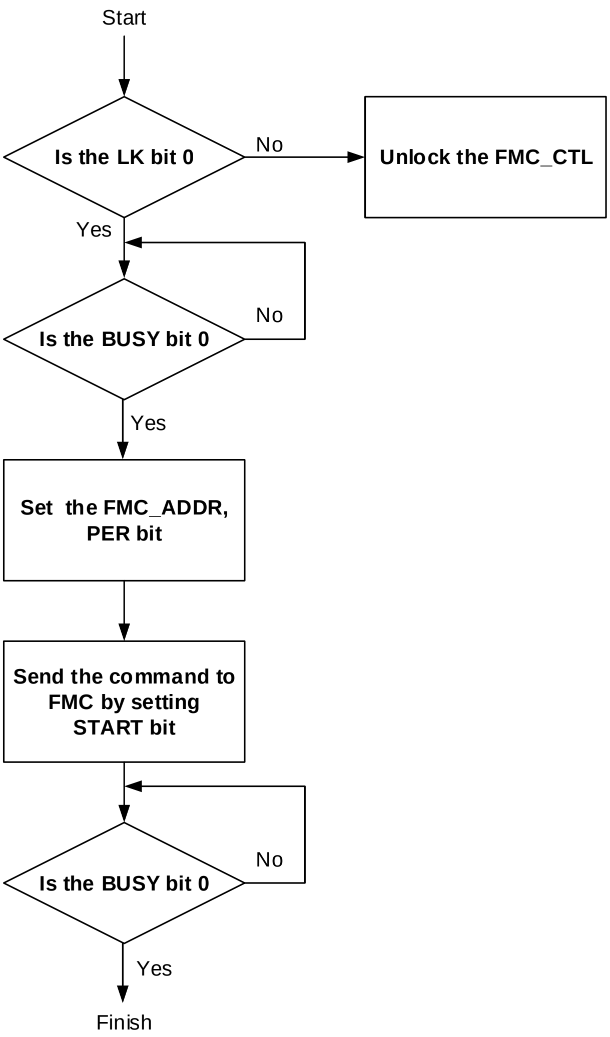
### 3.3.4. Page erase

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 DBUS 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 END in FMC_STAT register is set. Note that a correct target page address must be confirmed. 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 END bit in the FMC_STAT register. The following figure shows the page erase operation flow.

Figure 3-1. Proccess of page erase operation



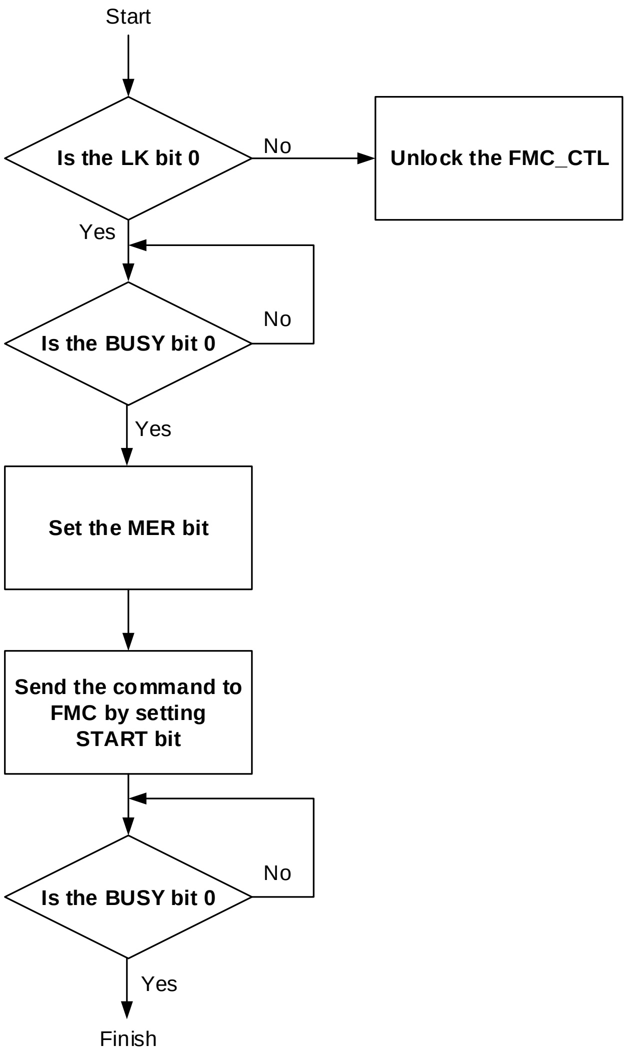
### 3.3.5. Mass erase

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 DBUS 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 END 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 END bit in the FMC_STAT register. (The starting address of programming operation should be 0x08000000) The following figure indicates the mass erase operation flow.

**Figure 3-2. Process of the mass erase operation**



### 3.3.6. Main flash programming

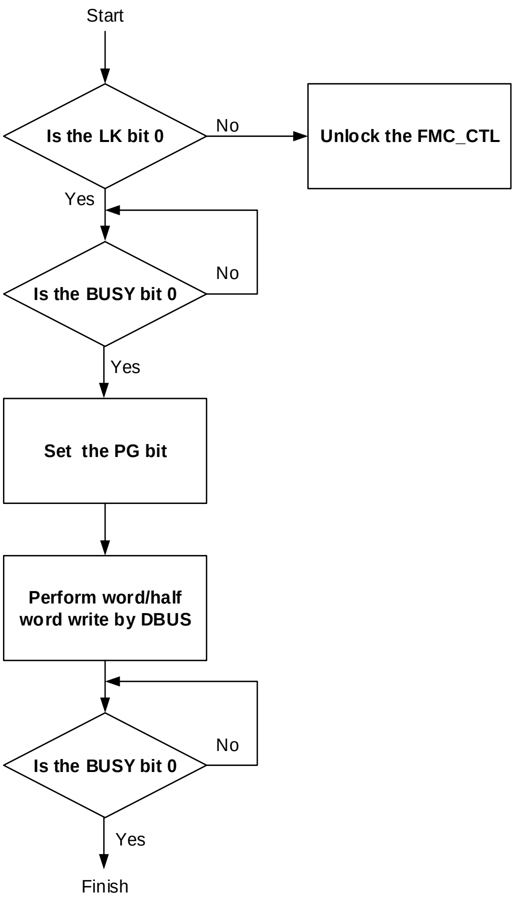
The FMC provides a 32-bit word/16-bit half word programming function which is used to modify the Main Flash memory contents. The following steps show the word programming operation register access sequence.

- 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 word program command into the PG bit in FMC_CTL register.

- A 32-bit word/16-bit half word write at desired address by DBUS.
- 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 DBUS 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 END 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 except programming 0x0. 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 END bit in the FMC_STAT register. The following figure displays the word programming operation flow.

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



## 3.3.7. 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 OBWE 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 DBUS 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 END in FMC_STAT register is set. The end of this operation is indicated by the END bit in the FMC_STAT register.

### 3.3.8. Option byte programming

The FMC provides a 32-bit word/16-bit half 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 OBWE 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/16-bit half word write at desired address by DBUS.
- 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 DBUS 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 END 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 except programming 0x0. The end of this operation is indicated by the END bit in the FMC_STAT register.

### 3.3.9. 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 OBER bit in FMC_OBSTAT register is set, and the option byte is set to 0xFF. The following table is the detail of option bytes.

Table 3-3. Option byte

| Address | Name | Description |
|---|---|---|
| 0x1fff f800 | OB_SPC | option byte Security Protection Code<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 |
| 0x1fff f802 | OB_USER | option byte which user defined<br>[7]: Reserved<br>[6]: OB_SRAM_PARITY_CHECK<br>0: Enable sram parity check<br>1: Disable sram parity check<br>[5]: OB_VDDA_VISOR<br>0: Disable $V_{DDA}$ monitor<br>1: Enable $V_{DDA}$ monitor<br>[4]: OB_BOOT1_n<br>0: BOOT1 bit is 1<br>1: BOOT1 bit is 0<br>[3]: Reserved<br>[2]: OB_STDBY_RSTn<br>0: Generate a reset instead of entering standby mode<br>1: No reset when entering standby mode<br>[1]: OB_DSLP_RSTn<br>0: Generate a reset instead of entering Deep-sleep mode<br>1: No reset when entering Deep-sleep mode<br>[0]: OB_WDG_SW<br>0: Hardware free watchdog timer<br>1: Software free watchdog timer |
| 0x1fff f803 | OB_USER_N | OB_USER complement value |
| 0x1fff f804 | OB_DATA[7:0] | user defined data bit 7 to 0 |
| 0x1fff f805 | OB_DATA_N[7:0] | OB_DATA complement value bit 7 to 0 |
| 0x1fff f806 | OB_DATA[15:8] | user defined data bit 15 to 8 |
| 0x1fff f807 | OB_DATA_N[15:8] | OB_DATA complement value bit 15 to 8 |
| 0x1fff f808 | OB_WP[7:0] | Page Erase/Program Protection bit 7 to 0 |
| 0x1fff f809 | OB_WP_N[7:0] | OB_WP complement value bit 7 to 0 |
| 0x1fff f80a | OB_WP[15:8] | Page Erase/Program Protection bit 15 to 8 |
| 0x1fff f80b | OB_WP_N[15:8] | OB_WP complement value bit 15 to 8 |

### 3.3.10. 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 [15:0] 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 following table shows which pages are protected by set OB_WP [15:0].

**Table 3-4. OB_WP bit for pages protected**

| OB_WP bit | pages protected |
|---|---|
| OB_WP[0] | page0 ~ page 3 |
| OB_WP[1] | page4 ~ page 7 |
| OB_WP[2] | page8 ~ page 11 |
| .<br>.<br>. | .<br>.<br>. |
| OB_WP[14] | page56 ~ page59 |
| OB_WP[15] for GD32F130xx and GD32F150xx devices | page60 ~ page63 |
| OB_WP[15] for GD32F170xx and GD32F190xx devices | page60 ~ page127 |

### 3.3.11. 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. All operations are from user code. The main flash block is accessible by all operations. The option byte cannot be erased, and the OB_SPC byte and its complement value cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

## 3.4. FMC registers

### 3.4.1. Wait state register (FMC_WS)

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 | | | | | | | | | | | | | WSCNT[2:0] | | |
| | | | | | | | | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value |
| 2:0 | WSCNT[2:0] | Wait state counter register<br>These bits set and reset by software. The WSCNT valid when WSEN bit is set<br>000: 0 wait state added<br>001: 1 wait state added<br>010: 2 wait state added<br>011 ~ 111: Reserved |

### 3.4.2. 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. |

### 3.4.3. 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. |

### 3.4.4. 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 | | | | | | | | | | END | WPERR | Reserved. | PGERR | Reserved | BUSY |
| | | | | | | | | | | rw | rw | | rw | | r |

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

| | | |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value |
| 5 | END | 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 | Reserved | Must be kept at reset value |
| 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. |

### 3.4.5. 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 | OBWE | 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<br>This bit is set or cleared by software.<br>0: No interrupt generated by hardware |

| | | |
|---|---|---|
| | | 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 | OBWE | Option byte erase/program enable bit<br>This bit is set by hardware when right sequence written to FMC_OBKEY register. 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. 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.<br>0: No effect<br>1: Main flash page program command |

### 3.4.6. Address register (FMC_ADDR)

Address offset: 0x14

Reset value: 0x0000 0080

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">ADDR[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ADDR[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

| 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 |

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

Address offset: 0x1C

Reset value: 0xXXXX XX0X

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">OB_DATA[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">OB_USER[7:0]</td><td colspan="5">Reserved</td><td colspan="2">PLEVEL[1:0]</td><td>OBER</td></tr>
<tr><td colspan="8">r</td><td colspan="5"></td><td colspan="2">r</td><td>r</td></tr>
</table>

| 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 |
| 2:1 | PLEVEL[1:0] | Security Protection level<br>00: No protection level<br>01: Protect level low<br>11: Protect level high |
| 0 | OBER | 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. |

### 3.4.8. Write protection register (FMC_WP)

Address offset: 0x20

Reset value: 0x0000 XXXX

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">OB_WP[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | OB_WP[15:0] | Store OB_WP[15:0] of option byte block after system reset<br>0: Protection active<br>1: Unprotected |

### 3.4.9. Wait state enable register (FMC_WSEN)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0xFC

Reset value: 0xXXXX XXXX

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Resrved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>WSEN</td></tr>
<tr><td colspan="15"></td><td>rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |
| 0 | WSEN | FMC wait state enable register<br>This bit set and reset by software. This bit also protected by the FMC_KEY register. There need writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register.<br>0: No wait state added when fetching flash<br>1: Wait state added when fetching flash |

**For GD32F170xx and GD32F190xx devices**

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resrved | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | BPEN | FMC bit program enable register<br>This bit set and reset by software.<br>0: No effect, write page must check if flash is “FF”<br>1: Write page donot check the flash is FF. The FMC can program each bit. |
| 0 | WSEN | FMC wait state enable register<br>This bit set and reset by software. This bit is also protected by the FMC_KEY register. The software need writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register.<br>0: No wait state added when fetching flash<br>1: Wait state added when fetching flash |

### 3.4.10. 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 register1 |

These bits are read only by software.

These bits are unchanged constantly after power on. These bits are one time programmed when the chip product.

# 4. Reset and clock unit (RCU)

## 4.1. Reset control unit (RCTL)

### 4.1.1. Introduction

GD32F1x0 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. The 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 description

**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 1.2V power for GD32F130/150xx devices or 1.8V power for GD32F170/190xx devices power. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map.

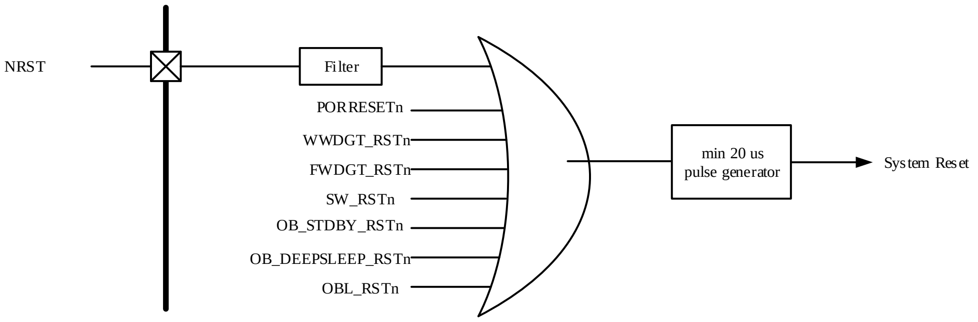
**System Reset**

A system reset is generated by the following events:

- A power on reset (PORRESETn)
- A external pin reset (NRST)
- A window watchdog timer reset (WWDGT_RSTn)
- A free watchdog timer reset (FWDGT_RSTn)
- The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register is set (SW_RSTn)
- Option byte loader reset (OBL_RSTn)
- Reset generated when entering Standby mode when resetting OB_STDBY_RSTn bit in User Option Bytes (OB_STDBY_RSTn)
- Reset generated when entering Deep-sleep mode when resetting OB_DSLP_RSTn bit in User Option Bytes (OB_DSLP_RSTn)

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 ($V_{DD}$ or $V_{BAT}$ power on, if both supplies have previously been powered off).

## 4.2. Clock control unit (CCTL)

### 4.2.1. Introduction

The Clock Control unit provides a range of frequencies and clock functions. These include a Internal 8 MHz RC oscillator (IRC8M), a Internal 14 MHz RC oscillator (IRC14M) for GD32F130xx and GD32F150xx devices or a Internal 28 MHz RC oscillator (IRC28M) for GD32F170xx and GD32F190xx devices, 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™-M3 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 of GD32F130xx and GD32F150xx devices

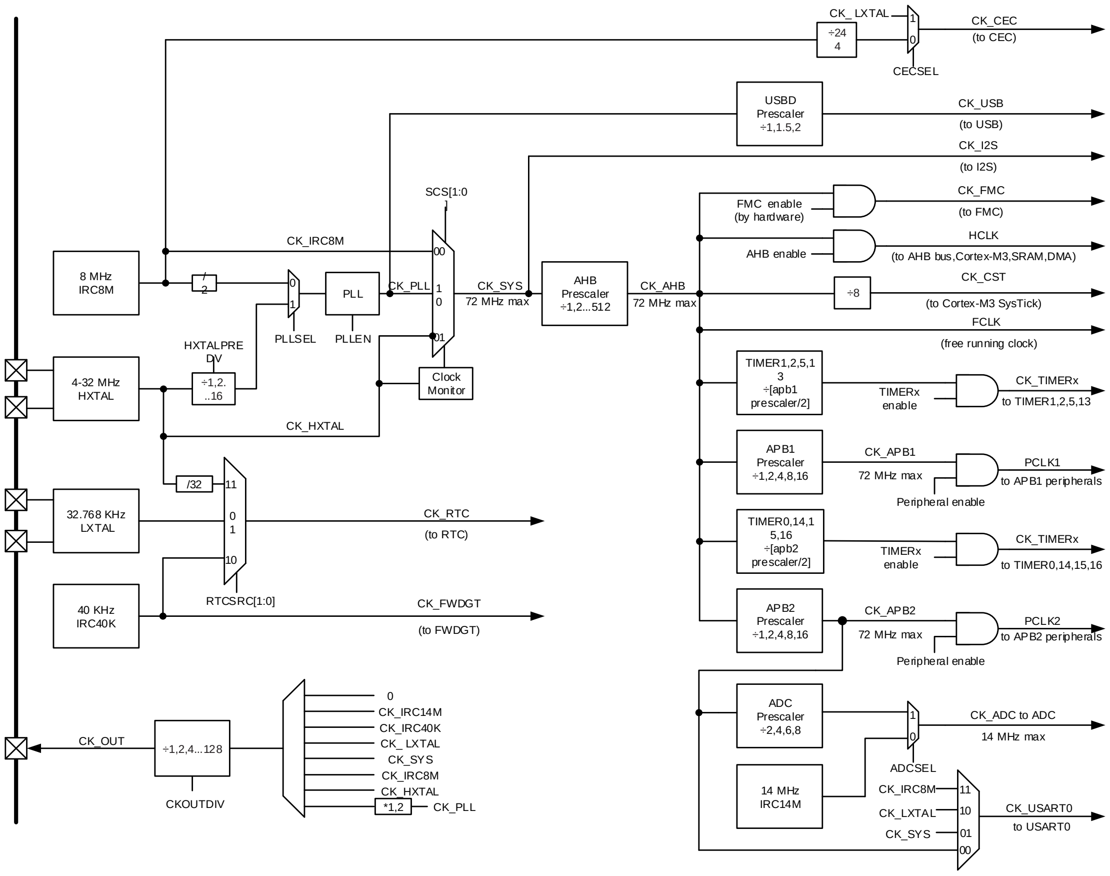
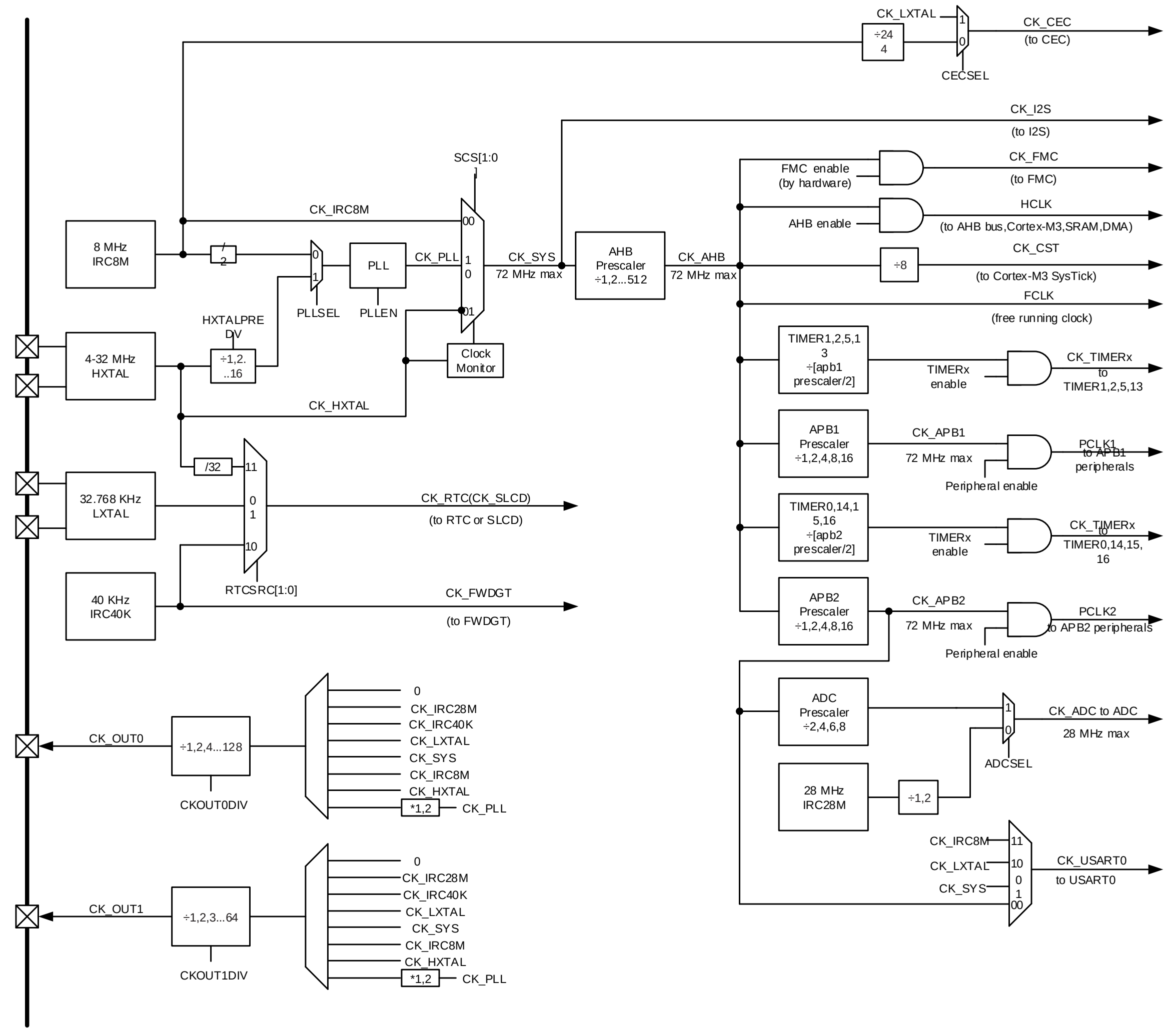


**Figure 4-3. Clock tree of GD32F170xx and GD32F190xx devices**



The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB and the APB2/APB1 domains is 72 MHz. When using I2Cx peripherals, APB1 clock need to ensure that no more than 36MHz.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 Cortex clock (HCLK), configurable in the SysTick Control and Status Register.

The ADC is clocked by the clock of APB2 divided by 2, 4, 6, 8 or IRC14M clock selected for GD32F130xx and GD32F150xx devices or 2, 4, 6, 8 or IRC28M or IRC28M/2 clock for GD32F170xx and GD32F190xx devices 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 CEC clock is clocked by IRC8M divided 244 or LXTAL clock which selected by CECSEL bit in Configuration register 2 (RCU_CFG2).

The RTC or SLCD (for GD32F170xx and GD32F190xx devices only) is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 32 which select by RTCSRC bit 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.

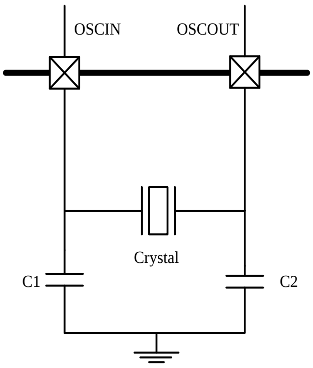
### 4.2.2. Main features

- 4 to 32 MHz High speed crystal oscillator (HXTAL)
- Internal 8 MHz RC oscillator (IRC8M)
- Internal 14 MHz RC oscillator (IRC14M) for GD32F130xx and GD32F150xx devices or Internal 28 MHz RC oscillator (IRC28M) for GD32F170xx and GD32F190xx devices
- 32,768 Hz Low speed crystal oscillator (LXTAL)
- Internal 40KHz RC oscillator (IRC40K)
- PLL clock source can be HXTAL or IRC8M
- HXTAL clock monitor

### 4.2.3. Function description

#### 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.



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 14 MHz RC Oscillator (IRC14M) for GD32F130xx and GD32F150xx devices

The Internal 14 MHz RC Oscillator, IRC14M, has a fixed frequency of 14 MHz and dedicated as ADC clock. The IRC14M RC oscillator can be switched on or off using the IRC14MEN bit in the Control register 1 (RCU_CTL1). The IRC14MSTB flag in the Control register 1 (RCU_CTL1) is used to indicate if the internal 14M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC14MSTBIE, in the Interrupt register, RCU_INT, is set when the IRC14M becomes stable.

### Internal 28 MHz RC Oscillator (IRC28M) for GD32F170xx and GD32F190xx devices

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 Register (RCU_BDCTL). The CK_LXTAL is equal to the external clock which drives the OSC32IN pin.

### Internal 40KHz RC Oscillator (IRC40K)

The Internal 40KHz 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-M3. 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

**For GD32F130xx and GD32F150xx devices**

The clock output capability is ranging from 32 kHz to 54 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_IRC14M |
| 010 | CK_IRC40K |
| 011 | CK_LXTAL |
| 100 | CK_SYS |
| 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).

**For GD32F170xx and GD32F190xx devices**

The clock output capability is ranging from 32 kHz to 54 MHz. There are several clock signals can be selected via the CK_OUT0 Clock Source Selection bits, CKOUT0SEL, in the Configuration register 0 (RCU_CFG0) and CKOUT1SRC in the Configuration register 3 (RCU_CFG3). The corresponding GPIO pin should be configured in the properly Alternate Function I/O (AFIO) mode to output the selected clock signal.

**Table 4-2. Clock source select**

| Clock Source Selection bits | Clock Source |
|---|---|
| 000 | No Clock |
| 001 | CK_IRC28M |
| 010 | CK_IRC40K |
| 011 | CK_LXTAL |
| 100 | CK_SYS |
| 101 | CK_IRC8M |
| 110 | CK_HXTAL |
| 111 | CK_PLL or CK_PLL/2 |

The CK_OUT0 frequency can be reduced by a configurable binary divider, controlled by the CKOUT0DIV[2:0] bits , in the Configuration register 0 (RCU_CFG0).

The CK_OUT1 is seleced by CKOUT1SRC, in the Configuration register 3 (RCU_CFG3). The CK_OUT1 frequency can be reduced by a configurable binary divider, controlled by the CKOUT1DIV[2:0] bits , in the Configuration register 3 (RCU_CFG3).

**Deep-sleep mode clock control**

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

If the HDMI CEC or 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 HDMI CEC or USART0 to wake up the Deep-sleep mode.

**Voltage control**

**For GD32F130xx and GD32F150xx devices**

The power down reset can be controlled by PDR_S bit in the Power down voltage select register (RCU_PDVSEL). If the PDR_S bit is 0, the power down reset assert when the $V_{DD}$ is below 2.6V. If the PDR_S bit is 1, the power down reset assert when the $V_{DD}$ is below 1.8V. When the PDR_S bit is 1 and $V_{DD}$ is below 2.6V, the flash program/erase cannot be used.

The core domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

**Table 4-3. Core domain voltage selected in Deep-sleep mode -**

| DSLPVS[2:0] | Deep-sleep mode voltage(V) |
|---|---|
| 000 | 1.2 |
| 001 | 1.1 |
| 010 | 1.0 |
| 011 | 0.9 |
| 100 ~ 111 | reserved |

The RCU_PDVSEL and RCU_DSV register are protected by Voltage Key register (RCU_VKEY). Only after write 0x1A2B3C4D to the RCU_VKEY register, the RCU_PDVSEL and RCU_DSV register can be write.

**For GD32F170xx and GD32F190xx devices**

The core domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

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

| DSLPVS[2:0] | Deep-sleep mode voltage(V) |
|---|---|
| 000 | 1.8 |
| 001 | 1.6 |
| 010 | 1.4 |
| 011 | 1.2 |
| 100 ~ 111 | reserved |

The RCU_PDVSEL and RCU_DSV register are protected by Voltage Key register (RCU_VKEY). Only after write 0x1A2B3C4D to the RCU_VKEY register, the RCU_PDVSEL and RCU_DSV register can be write.

# 4.3. RCU registers

## 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)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | PLLSTB | PLL<br>EN | Reserved | | | | CKMEN | HXTALBPS | HXTALSTB | HXTALEN |
| | | | | | | r | rw | | | | | rw | rw | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC8MCALIB[7:0] | | | | | | | | IRC8MADJ[4:0] | | | | | Reserved. | IRC8MSTB | IRC8MEN |
| r | | | | | | | | rw | | | | | | r | rw |

| 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. |

| | | |
|---|---|---|
| | | 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] | High Speed Internal Oscillator calibration value register<br>These bits are load automatically at power on. |
| 7:3 | IRC8MADJ[4:0] | High Speed Internal 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)

#### For GD32F130xx and GD32F150xx devices

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] | | | USBDPSC[1:0] | | PLLMF[3:0] | | | | PLLPREDV | PLLSEL |
| rw | 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<br>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 14M 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 | USBDPSC[1:0] | USBD clock prescaler selection<br>Set and reset by software to control the USBD clock prescaler value. The USBD clock must be 48MHz. These bits can’t be reset if the USBD clock is enabled.<br>00: (CK_PLL / 1.5) selected<br>01: CK_PLL selected<br>10: (CK_PLL / 2.5) selected<br>11: (CK_PLL / 2) selected |

| | | |
|---|---|---|
| 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)<br>00010: (PLL source clock x 4)<br>00011: (PLL source clock x 5)<br>00100: (PLL source clock x 6)<br>00101: (PLL source clock x 7)<br>00110: (PLL source clock x 8)<br>00111: (PLL source clock x 9)<br>01000: (PLL source clock x 10)<br>01001: (PLL source clock x 11)<br>01010: (PLL source clock x 12)<br>01011: (PLL source clock x 13)<br>01100: (PLL source clock x 14)<br>01101: (PLL source clock x 15)<br>01110: (PLL source clock x 16)<br>01111: (PLL source clock x 16)<br>10000: (PLL source clock x 17)<br>10001: (PLL source clock x 18)<br>10010: (PLL source clock x 19)<br>10011: (PLL source clock x 20)<br>10100: (PLL source clock x 21)<br>10101: (PLL source clock x 22)<br>10110: (PLL source clock x 23)<br>10111: (PLL source clock x 24)<br>11000: (PLL source clock x 25)<br>11001: (PLL source clock x 26)<br>11010: (PLL source clock x 27)<br>11011: (PLL source clock x 28)<br>11100: (PLL source clock x 29)<br>11101: (PLL source clock x 30)<br>11110: (PLL source clock x 31)<br>11111: (PLL source clock x 32)<br>**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 HXTALPREDV[0] from RCU_CFG1. Refer to RCU_CFG1 HXTALPREDV bits description.<br>Set and cleared by software to divide HXTAL or not which is selected to PLL.<br>0: HXTAL clock selected |

| | | |
|---|---|---|
| | | 1: (CK_HXTAL / 2) clock selected |
| 16 | PLLSEL | PLL Clock Source Selection<br>Set and reset by software to control the PLL clock source.<br>0: (CK_IRC8M / 2) selected as PLL source clock<br>1: HXTAL selected as PLL source clock |
| 15:14 | ADCPSC[1:0] | ADC clock prescaler selection<br>Set and cleared by software.<br>00: (CK_APB2 / 2) selected<br>01: (CK_ APB2 / 4) selected<br>10: (CK_ APB2 / 6) selected<br>11: (CK_ APB2 / 8) 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 |

| | | 11: reserved |
|---|---|---|
| 1:0 | SCS[1:0] | System clock switch<br>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.<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 |

### For GD32F170xx and GD32F190xx devices

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 | CKOUT0DIV[2:0] | | | PLLMF[4] | CKOUT0SEL [2:0] | | | Reserved | | PLLMF[3:0] | | | | PLLPREDV | 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_OUT0<br>0: CK_PLL divide by 2 for CK_OUT0<br>1: CK_PLL divide by 1 for CK_OUT0 |
| 30:28 | CKOUT0DIV[2:0] | The CK_OUT0 divider which the CK_OUT0 frequency can be reduced<br>see bits 26:24 of RCU_CFG0 for CK_OUT0<br>000: The CK_OUT0 is divided by 1<br>001: The CK_OUT0 is divided by 2<br>010: The CK_OUT0 is divided by 4<br>011: The CK_OUT0 is divided by 8<br>100: The CK_OUT0 is divided by 16<br>101: The CK_OUT0 is divided by 32<br>110: The CK_OUT0 is divided by 64<br>111: The CK_OUT0 is divided by 128 |
| 27 | PLLMF[4] | Bit 4 of PLLMF register<br>see bits 21:18 of RCU_CFG0. |

| | | |
|---|---|---|
| 26:24 | CKOUT0SEL[2:0] | CKOUT0 Clock Source Selection<br>Set and reset by software.<br>000: No clock selected<br>001: Internal 28MHz 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)<br>00010: (PLL source clock x 4)<br>00011: (PLL source clock x 5)<br>00100: (PLL source clock x 6)<br>00101: (PLL source clock x 7)<br>00110: (PLL source clock x 8)<br>00111: (PLL source clock x 9)<br>01000: (PLL source clock x 10)<br>01001: (PLL source clock x 11)<br>01010: (PLL source clock x 12)<br>01011: (PLL source clock x 13)<br>01100: (PLL source clock x 14)<br>01101: (PLL source clock x 15)<br>01110: (PLL source clock x 16)<br>01111: (PLL source clock x 16)<br>10000: (PLL source clock x 17)<br>10001: (PLL source clock x 18)<br>10010: (PLL source clock x 19)<br>10011: (PLL source clock x 20)<br>10100: (PLL source clock x 21)<br>10101: (PLL source clock x 22)<br>10110: (PLL source clock x 23)<br>10111: (PLL source clock x 24)<br>11000: (PLL source clock x 25)<br>11001: (PLL source clock x 26)<br>11010: (PLL source clock x 27)<br>11011: (PLL source clock x 28)<br>11100: (PLL source clock x 29) |

| | | |
|---|---|---|
| | | 11101: (PLL source clock x 30)<br>11110: (PLL source clock x 31)<br>11111: (PLL source clock x 32)<br>**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 HXTALPREDV[0] from RCU_CFG1. Refer to RCU_CFG1 HXTALPREDV bits description.<br>Set and cleared by software to divide HXTAL or not which is selected to PLL.<br>0: HXTAL clock selected<br>1: (CK_HXTAL / 2) clock selected |
| 16 | PLLSEL | PLL Clock Source Selection<br>Set and reset by software to control the PLL clock source.<br>0: (CK_IRC8M / 2) selected as PLL source clock<br>1: HXTAL selected as PLL source clock |
| 15:14 | ADCPSC[1:0] | ADC clock prescaler selection<br>Set and cleared by software.<br>00: (CK_APB2 / 2) selected<br>01: (CK_APB2 / 4) selected<br>10: (CK_APB2 / 6) selected<br>11: (CK_APB2 / 8) 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 |

| | | |
|---|---|---|
| | | 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<br>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.<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 |

## 4.3.3. Interrupt register (RCU_INT)

### For GD32F130xx and GD32F150xx devices

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 | IRC14M<br>STBIC | PLL<br>STBIC | HXTAL<br>STBIC | IRC8M<br>STBIC | LXTAL<br>STBIC | IRC40K<br>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 | | IRC14M<br>STBIE | PLL<br>STBIE | HXTAL<br>STBIE | IRC8M<br>STBIE | LXTAL<br>STBIE | IRC40K<br>STBIE | CKMIF | Reserved | IRC14M<br>STBIF | PLL<br>STBIF | HXTAL<br>STBIF | IRC8M<br>STBIF | LXTAL<br>STBIF | IRC40K<br>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 |

| | | |
|---|---|---|
| | | 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 | IRC14MSTBIC | IRC14M stabilization Interrupt Clear<br>Write 1 by software to reset the IRC14MSTBIF flag.<br>0: Not reset IRC14MSTBIF flag<br>1: Reset IRC14MSTBIF 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 LXTALRDYF flag |
| 16 | IRC40KSTBIC | IRC40K Stabilization Interrupt Clear<br>Write 1 by software to reset the IRC40KRDYF flag.<br>0: Not reset IRC40KSTBIF flag<br>1: Reset IRC40KRDYF flag |
| 15:14 | Reserved | Must be kept at reset value |
| 13 | IRC14MSTBIE | IRC14M Stabilization Interrupt Enable<br>Set and reset by software to enable/disable the IRC14M stabilization interrupt.<br>0: Disable the IRC14M stabilization interrupt<br>1: Enable the IRC14M 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 |

| | | |
|---|---|---|
| | | 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<br>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 | IRC14MSTBIF | IRC14M stabilization interrupt flag<br>Set by hardware when the IRC14M is stable and the IRC14MSTBIE bit is set.<br>Reset by software when setting the IRC14MSTBIC bit.<br>0: No IRC14M stabilization interrupt generated<br>1: IRC14M 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 32kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set.<br>Reset by software when setting the IRC40KSTBIC bit.<br>0: No IRC40K stabilization clock ready interrupt generated<br>1: IRC40K stabilization interrupt generated |

### For GD32F170xx and GD32F190xx devices

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<br>STBIC | PLL<br>STBIC | HXTAL<br>STBIC | IRC8M<br>STBIC | LXTAL<br>STBIC | IRC40K<br>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<br>STBIE | PLL<br>STBIE | HXTAL<br>STBIE | IRC8M<br>STBIE | LXTAL<br>STBIE | IRC40K<br>STBIE | CKMIF | Reserved | IRC28M<br>STBIF | PLL<br>STBIF | HXTAL<br>STBIF | IRC8M<br>STBIF | LXTAL<br>STBIF | IRC40K<br>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. |

| | | |
|---|---|---|
| | | 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 LXTALRDYF flag |
| 16 | IRC40KSTBIC | IRC40K Stabilization Interrupt Clear<br>Write 1 by software to reset the IRC40KRDYF flag.<br>0: Not reset IRC40KSTBIF flag<br>1: Reset IRC40KRDYF 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 |

| | | |
|---|---|---|
| | | 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<br>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 32.768KHz crystal oscillator clock is stable and |

| Bits | Fields | Descriptions |
|---|---|---|
| | | 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.<br>Reset by software when setting the IRC40KSTBIC bit.<br>0: No IRC40K stabilization clock ready interrupt generated<br>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 | | | | | | | | | | | | | TIMER16RST | TIMER15RST | TIMER14RST |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USART0RST | Res. | SPI0RST | TIMER0RST | Res. | ADCRST | Reserved | | | | | | | | CFGRST |
| | 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 |

14 USART0RST USART0 Reset

This bit is set and reset by software.

0: No reset

1: Reset the USART0

13 Reserved Must be kept at reset value

12 SPI0RST SPI0 Reset

This bit is set and reset by software.

0: No reset

1: Reset the SPI0

11 TIMER0RST TIMER0 reset

This bit is set and reset by software.

0: No reset

1: Reset the TIMER0

10 Reserved Must be kept at reset value

9 ADCRST ADC reset

This bit is set and reset by software.

0: No reset

1: Reset the ADC

8:1 Reserved Must be kept at reset value

0 CFGRST System configuration reset

This bit is set and reset by software.

0: No reset

1: Reset System configuration

### 4.3.5. APB1 reset register (RCU_APB1RST)

#### For GD32F130xx and GD32F150xx devices

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 | CEC<br>RST | DAC<br>RST | PMU<br>RST | Reserved | | | | USBD<br>RST | I2C1<br>RST | I2C0<br>RST | Reserved | | | USART<br>1RST | Res. |
| | rw | rw | rw | | | | | rw | rw | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPI2 | SPI1 | Reserved | | WWDGT | Reserved | | TIMER13 | Reserved | | | TIMER5 | Reserved | | TIMER2 | TIMER1 |

|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| RST | RST |  | RST |  | RST |  | RST |  | RST | RST |
| rw | rw |  | rw |  | rw |  | rw |  | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value |
| 30 | CECRST | HDMI CEC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset hdmi cec unit |
| 29 | DACRST | DAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset DAC unit |
| 28 | PMURST | Power control reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset power control unit |
| 27:24 | Reserved | Must be kept at reset value |
| 23 | USBDRST | USBD reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset USBD |
| 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 | Reserved | Must be kept at reset value |
| 15 | SPI2RST | SPI2 reset<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No reset<br>1: Reset SPI2 |
| 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<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER2 timer |
| 0 | TIMER1RST | TIMER1 timer reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER1 timer |

### For GD32F170xx and GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPAIVREF | CEC | DAC | PMU | Reserved | CAN1 | CAN0 | Reserved | | I2C1 | I2C0 | Reserved | | | USART | Reserved |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | RST | RST | RST | | RST | RST | | | RST | RST | | | | 1RST | |
| 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 |
| SPI2<br>RST | SPI1<br>RST | Reserved | | WWDGT<br>RST | Reserved. | SLCDRST | TIMER13<br>RST | Reserved | | TIMER5<br>RST | Reserved | | | TIMER2<br>RST | TIMER1<br>RST |
| rw | rw | | | rw | | rw | rw | | | | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | OPAIVREFRST | OPA and IVREF reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset OPA unit |
| 30 | CECRST | HDMI CEC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset hdmi cec unit |
| 29 | DACRST | DAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset DAC unit |
| 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 | Reserved | Must be kept at reset value |
| 26 | CAN1RST | CAN1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN1 |
| 25 | CAN0RST | CAN0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN0 |
| 24:23 | 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 | Reserved | Must be kept at reset value |
| 15 | SPI2RST | SPI2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset SPI2 |
| 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 | Reserved | Must be kept at reset value |
| 9 | SLCDRST | SLCD reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset SLCD Timer |
| 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<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER2 timer |
| 0 | TIMER1RST | TIMER1 timer reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset TIMER1 timer |

### 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 | | | | | | | TSIEN | Res. | PFEN | Res. | PDEN | PCEN | PBEN | PAEN | Res. |
| | | | | | | | rw | | rw | | rw | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved. | | | | | | | | | CRCEN | Res. | FMCEN | Res. | SRAMEN | Res. | DMAEN |
| | | | | | | | | | rw | | rw | | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value |
| 24 | TSIEN | TSI clock enable<br>This bit is set and reset by software.<br>0: Disabled TSI clock<br>1: Enabled TSI clock |
| 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 | Reserved | Must be kept at reset value |
| 20 | PDEN | GPIO port D clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port D clock |

| | | |
|---|---|---|
| | | 1: Enabled GPIO port D clock |
| 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 | FMCEN | 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 | SRAMEN | 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 | | | | | | | | | | | | | TIMER16EN | TIMER15EN | TIMER14EN |
| | | | | | | | | | | | | | 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 | | | | | | | | CFGCMP<br>EN |
| | rw | | rw | rw | | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | 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 |

| | | |
|---|---|---|
| | | 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<br>This bit is set and reset by software.<br>0: Disabled System configuration and comparator clock<br>1: Enabled System configuration and comparator clock |

## 4.3.8. APB1 enable register (RCU_APB1EN)

### For GD32F130xx and GD32F150xx devices

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 | CECEN | DACEN | PMUEN | Reserved | | | | USBDEN | I2C1EN | I2C0EN | Reserved | | | USART1EN | Reserved |
| | rw | rw | rw | | | | | rw | rw | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPI2EN | SPI1EN | Reserved | | WWDGTEN | Reserved | | TIMER13EN | Reserved | | | TIMER5EN | Reserved | | TIMER2EN | TIMER1EN |
| rw | rw | | | rw | | | rw | | | | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value |
| 30 | CECEN | Hdmi cec interface clock enable<br>This bit is set and reset by software.<br>0: Disabled Hdmi cec interface clock<br>1: Enabled Hdmi cec interface clock |
| 29 | DACEN | DAC interface clock enable<br>This bit is set and reset by software.<br>0: Disabled DAC interface clock<br>1: Enabled DAC interface clock |

| | | |
|---|---|---|
| 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 |
| 27:24 | Reserved | Must be kept at reset value |
| 23 | USBDEN | USBD clock enable<br>This bit is set and reset by software.<br>0: Disabled USBD clock<br>1: Enabled USBD clock |
| 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 | Reserved | Must be kept at reset value |
| 15 | SPI2EN | SPI2 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI2 clock<br>1: Enabled SPI2 clock |
| 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 | TIMER1EN | TIMER1 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER1 timer clock<br>1: Enabled TIMER1 timer clock |

### For GD32F170xx and GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPAIVREFEN | CECEN | DACEN | PMUEN | Reserved | CAN1EN | CAN0EN | Reserved | | I2C1EN | I2C0EN | Reserved | | | USART1EN | Reserved |
| rw | rw | rw | rw | | rw | rw | | | rw | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPI2EN | SPI1EN | Reserved | | WWDGTEN | Res. | SLCDEN | TIMER13EN | Reserved | | | TIMER5EN | Reserved | | TIMER2EN | TIMER1EN |
| rw | rw | | | rw | | | rw | | | | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | OPAIVREFEN | OPA and IVREF clock enable<br>This bit is set and reset by software.<br>0: Disabled OPA and IVREF interface clock<br>1: Enabled OPA and IVREF interface clock |
| 30 | CECEN | Hdmi cec interface clock enable<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled Hdmi cec interface clock<br>1: Enabled Hdmi cec interface clock |
| 29 | DACEN | DAC interface clock enable<br>This bit is set and reset by software.<br>0: Disabled DAC interface clock<br>1: Enabled DAC interface clock |
| 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 | Reserved | Must be kept at reset value |
| 26 | CAN1EN | CAN1 clock enable<br>This bit is set and reset by software.<br>0: Disabled CAN1 clock<br>1: Enabled CAN1 clock |
| 25 | CAN0EN | CAN0 clock enable<br>This bit is set and reset by software.<br>0: Disabled CAN0 clock<br>1: Enabled CAN0 clock |
| 24:23 | 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 | Reserved | Must be kept at reset value |
| 15 | SPI2EN | SPI2 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI2 clock |

| | | |
|---|---|---|
| | | 1: Enabled SPI2 clock |
| 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 | Reserved | Must be kept at reset value |
| 9 | SLCDEN | SLCD clock enable<br>This bit is set and reset by software.<br>0: Disabled SLCD clock<br>1: Enabled SLCD clock |
| 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 | TIMER1EN | TIMER1 timer clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER1 timer clock<br>1: Enabled TIMER1 timer clock |

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

#### For GD32F130xx and GD32F150xx devices

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 |
| | | | | | | | | | | | | | | | rw |

| 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 |

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

### For GD32F170xx and GD32F190xx devices

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 |
| | | | | | | | | | | | | | | | rw |

| 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 |

| | | |
|---|---|---|
| | | 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 and SLCD clock entry selection<br>Set and reset by software to control the RTC and SLCD clock source. Once the RTC and SLCD clock source has been selected, it cannot be changed anymore unless the Backup domain is reset.<br>00: No clock selected<br>01: CK_LXTAL selected as RTC/SLCD source clock<br>10: CK_IRC40K selected as RTC/SLCD source clock<br>11: (CK_HXTAL / 32) selected as RTC/SLCD 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<br>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<br>RSTF | WWDGT<br>RSTF | FWDGT<br>RSTF | SW<br>RSTF | POR<br>RSTF | EP<br>RSTF | OBL<br>RSTF | RSTFC | V12<br>RSTF | Reserved | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | IRC40K<br>STB | IRC40K<br>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.<br>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 | | | | | | | TSIRST | Reserved | PFRST | Reserved | PDRST | PCRST | PBRST | PARST | Reserved |
| | | | | | | | rw | | rw | | rw | rw | rw | rw | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value |
| 24 | TSIRST | TSI unit reset |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: No reset TSI unit<br>1: Reset TSI unit |
| 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 | Reserved | Must be kept at reset value |
| 20 | PDRST | GPIO port D reset<br>This bit is set and reset by software.<br>0: No reset GPIO port D<br>1: Reset GPIO port D |
| 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 | | | | | | | | | | | | HXTALPREDV[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:0 | HXTALPREDV[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 HXTALPREDV is same as bit 17 of RCU_CFG0, so modifying bit 17 of RCU_CFG0 aslo modifies bit 0 of RCU_CFG2.<br>The CK_HXTAL is divided by (HXTALPREDV + 1).<br>0000: HXTAL input to PLL not divided<br>0001: HXTAL input to PLL divided by 2<br>0010: HXTAL input to PLL divided by 3<br>0011: HXTAL input to PLL divided by 4<br>0100: HXTAL input to PLL divided by 5<br>0101: HXTAL input to PLL divided by 6<br>0110: HXTAL input to PLL divided by 7<br>0111: HXTAL input to PLL divided by 8<br>1000: HXTAL input to PLL divided by 9<br>1001: HXTAL input to PLL divided by 10<br>1010: HXTAL input to PLL divided by 11<br>1011: HXTAL input to PLL divided by 12<br>1100: HXTAL input to PLL divided by 13<br>1101: HXTAL input to PLL divided by 14<br>1110: HXTAL input to PLL divided by 15<br>1111: HXTAL input to PLL divided by 16 |

## 4.3.13. Configuration register 2 (RCU_CFG2)

### For GD32F130xx and GD32F150xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | ADCSEL | Reserved | CECSEL | Reserved | | | | USART0SEL[1:0] | |
| | | | | | | | rw | | rw | | | | | rw | |

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

| | | |
|---|---|---|
| 31:9 | 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_IRC14M<br>1: CK_ADC select CK_APB2 which is divided by 2/4/6/8. |
| 7 | Reserved | Must be kept at reset value |
| 6 | CECSEL | CK_CEC clock source selection<br>This bit is set and reset by software.<br>0: CK_CEC select CK_IRC8M divided by 244<br>1: CK_CEC select CK_LXTAL |
| 5:2 | 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 |

### For GD32F170xx and GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | IRC28MDIV |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | ADCSEL | Reserved | CECSEL | Reserved | | | | USART0SEL[1:0] | |
| | | | | | | | rw | | rw | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |
| 16 | IRC28MDIV | CK_IRC28M divider 2 or not<br>This bit is set and reset by software.<br>0: IRC28M/2 select to ADC clock<br>1: IRC28M select to 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. |

| | | |
|---|---|---|
| | | 0: CK_ADC select CK_IRC28M or CK_IRC28M/2 set by IRC28MDIV<br>1: CK_ADC select CK_APB2 which is divided by 2/4/6/8. |
| 7 | Reserved | Must be kept at reset value |
| 6 | CECSEL | CK_CEC clock source selection<br>This bit is set and reset by software.<br>0: CK_CEC select CK_IRC8M divided by 244<br>1: CK_CEC select CK_LXTAL |
| 5:2 | 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)

### For GD32F130xx and GD32F150xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC14MCALIB[7:0] | | | | | | | | IRC14MADJ[4:0] | | | | | Reserved | IRC14MSTB | IRC14MEN |
| r | | | | | | | | rw | | | | | | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | IRC14MCALIB[7:0] | Internal 14M RC Oscillator calibration value register<br>These bits are load automatically at power on. |
| 7:3 | IRC14MADJ[4:0] | Internal 14M RC Oscillator clock trim adjust value<br>These bits are set by software. The trimming value is there bits (IRC14MADJ) added to the IRC14MCALIB[7:0] bits. The trimming value should trim the IRC14M to 14 MHz ± 1%. |
| 2 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 1 | IRC14MSTB | IRC14M Internal 14M RC Oscillator stabilization Flag<br>Set by hardware to indicate if the IRC14M oscillator is stable and ready for use.<br>0: IRC14M oscillator is not stable<br>1: IRC14M oscillator is stable |
| 0 | IRC14MEN | IRC14M Internal 14M RC oscillator Enable<br>Set and reset by software.<br>0: Internal 14 MHz RC oscillator disabled<br>1: Internal 14 MHz RC oscillator enabled |

### For GD32F170xx and GD32F190xx devices

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 28MHz RC Oscillator calibration value register<br>These bits are load automatically at power on. |
| 7:3 | IRC28MADJ[4:0] | Internal 28MHz 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 28 MHz ± 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 28MHz RC oscillator disabled<br>1: Internal 28 MHz RC oscillator enabled |

### 4.3.15. Configuration register 3 (RCU_CFG3) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

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 | | CKOUT1DIV[5:0] | | | | | | Reserved | | | | | CKOUT1SRC[2:0] | | |
| | | rw | | | | | | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13:8 | CKOUT1DIV[5:0] | The CK_OUT1 divider which the CK_OUT1 frequency can be reduced<br>see bits 2:0 of RCU_CFG3 for CK_OUT1<br>0: The CK_OUT1 is divided by 1<br>1: The CK_OUT1 is divided by 2<br>2: The CK_OUT1 is divided by 3<br>…<br>63: The CK_OUT1 is divided by 64 |
| 7:3 | Reserved | Must be kept at reset value |
| 2:0 | CKOUT1SRC[2:0] | CKOUT1 Clock Source Selection<br>Set and reset by software.<br>000: No clock selected<br>001: Internal 28 MHz RC Oscillator clock selected<br>010: Internal 40KHz RC Oscillator clock selected<br>011: Low Speed Crystal Oscillator clock selected<br>100: System clock selected<br>101: Internal 8 MHz RC Oscillator clock selected<br>110: High Speed Crystal Oscillator clock selected<br>111: (CK_PLL / 2) or CK_PLL selected depend on PLLDV |

### 4.3.16. Additional enable register (RCU_ADDEN)

Address offset: 0xF8

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 | | | | | | | | | | | | | | | I2C2EN |
| | | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |
| 0 | I2C2EN | I2C2 unit clock enable<br>This bit is set and reset by software<br>0: Disable I2C2 unit clock<br>1: Enable I2C2 unit clock |

### 4.3.17. Additional reset register (RCU_ADDRST)

Address offset: 0xFC

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 | | | | | | | | | | | | | | | I2C2RST |
| | | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |
| 0 | I2C2RST | I2C2 unit reset<br>This bit is set and reset by software<br>0: Not reset I2C2 unit<br>1: Reset I2C2 unit |

### 4.3.18. 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_PDVSEL and 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_PDVSEL and RCU_DSV register can be written. |

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

#### For GD32F130xx and GD32F150xx devices

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[2:0] | | |
| | | | | | | | | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value |
| 2:0 | DSLPVS[2:0] | Deep-sleep mode voltage select<br>These bits is set and reset by software<br>000 : The core voltage is 1.2V in Deep-sleep mode<br>001 : The core voltage is 1.1V in Deep-sleep mode<br>010 : The core voltage is 1.0V in Deep-sleep mode<br>011 : The core voltage is 0.9V in Deep-sleep mode<br>100~111 : Reserved |

#### For GD32F170xx and GD32F190xx devices

Address 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[2:0] | | |
| | | | | | | | | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value |
| 2:0 | DSLPVS[2:0] | Deep-sleep mode voltage select<br>These bits is set and reset by software<br>000 : The core voltage is 1.8V in Deep-sleep mode<br>001 : The core voltage is 1.6V in Deep-sleep mode<br>010 : The core voltage is 1.4V in Deep-sleep mode<br>011 : The core voltage is 1.2V in Deep-sleep mode<br>100~111 : Reserved |

### 4.3.20. Power down voltage select register (RCU_PDVSEL) of GD32F130xx and GD32F150xx devices

Address offset: 0x138

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 | | | | | | | | | | | | | | | PDRVS |
| | | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |
| 0 | PDRVS | Power down voltage select<br>This bit is set and reset by software<br>0: The Power down voltage is 2.6V<br>1: The Power down voltage is 1.8V |

# 5. General-purpose and alternate-function I/Os (GPIO and AFIOs)

## 5.1. Introduction

There are up to 55 general purpose I/O pins, (GPIO), named PA0 ~ PA15 and PB0 ~ PB15, PC0 ~ PC15, PD2, PF0/PF1, PF4 ~ PF7 for the 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 regardless of the AF input or output pins.

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), as input, as peripheral alternate function or as analog mode. Each GPIO pin can be configured as pull-up, pull-down or no pull-up/pull-down. All GPIOs are high-current capable except for analog mode.

## 5.2. Main features

- 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 configurations
- Alternate function input/output configurations
- Port configuration lock

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Single cycle toggle output capability

## 5.3. Function description

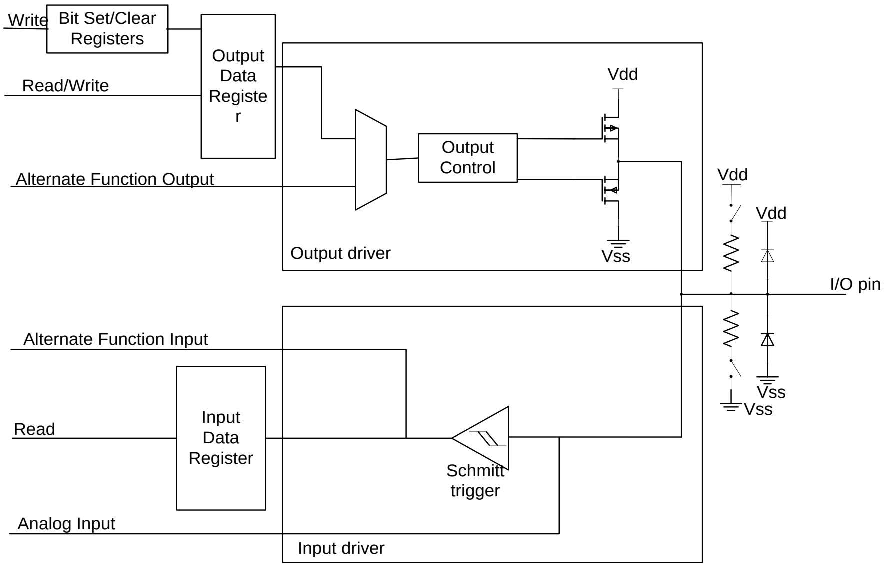
Each of the general-purpose I/O ports can be configured as GPIO inputs, GPIO outputs, AF function or analog mode by GPIO 32-bit configuration registers (GPIOx_CTL). AFIO input or output is selected by AFIO output enable. When the port is output (GPIO output or AFIO output), it can be configured as push-pull 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 5-1. 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 below shows the basic structure of an I/O Port bit.

**Figure 5-1. Basic structure of a standard I/O port bit**



### 5.3.1. GPIO pin configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured into the input floating mode that input disabled without Pull-Up(PU)/Pull-Down(PD) resistors. But the Serial-Wired Debug pins are in input PU/PD mode after reset:

PA14: SWCLK in input pull-down mode

PA13: SWDIO in input pull-up mode

The GPIO pins can be configured as inputs or outputs. And 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 AHB2 clock cycle to the port input status register (GPIO_ISTAT).

When the GPIO pins are configured as output pins, 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 (GPIO_OCTL) is output on the I/O pin.

When programming the GPIO_OCTL at bit level no need to disable interrupts, user can modify only one or several bits in a single atomic AHB2 write access by programming ‘1’ to the Bit Operate Register (GPIO_BOP, or for clearing only GPIO_BC , or for toggle only GPIO_TG). The other bits will not be affected.

### 5.3.2. Alternate functions (AF)

When the port is configured as AFIO (set CTLn to “10” bits, 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_AFSEL[1:0]). The detail alternate function assignments for each port are in the device datasheet.

### 5.3.3. Additional functions

Some pins have additional functions, which have priority over the configuration in the standard GPIO registers. When for ADC or DAC 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.

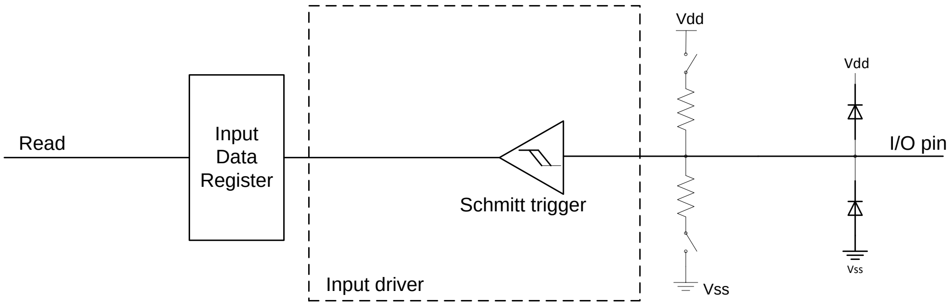
### 5.3.4. Input configuration

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 AHB2 clock cycle the data present on the I/O pad is got to the Data Input Register
- The Output Buffer is disabled

The figure below shows the Configuration of the I/O Port bit.

**Figure 5-2. Input floating/pull up/pull down configurations**



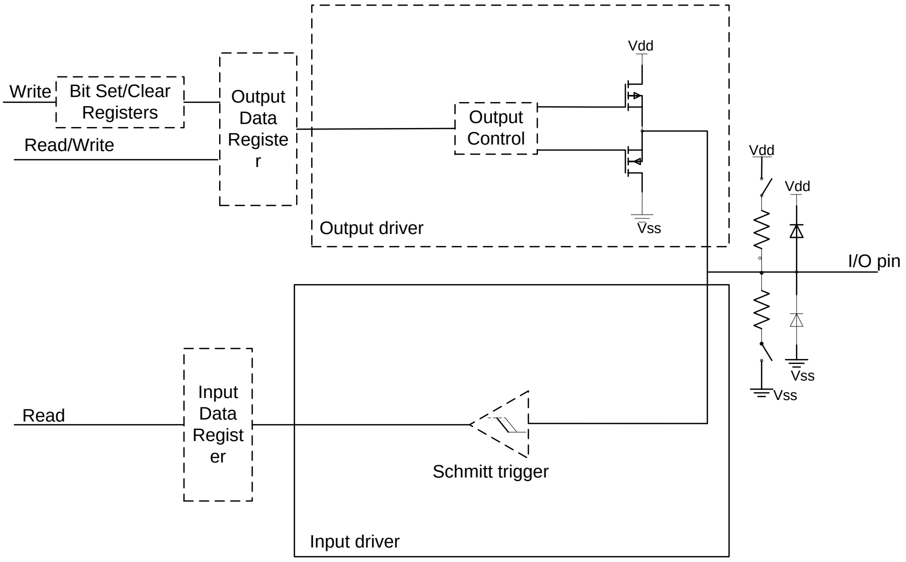
## 5.3.5. Output configuration

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: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z.
  - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in theOutput register activates the P-MOS.
- A read access to the Data Output register gets the last written value in Push-Pull mode
- A read access to the Data Input Register gets the I/O state in open drain mode

The figure below shows the Output configuration of the I/O port bit.

**Figure 5-3. Output configuration**



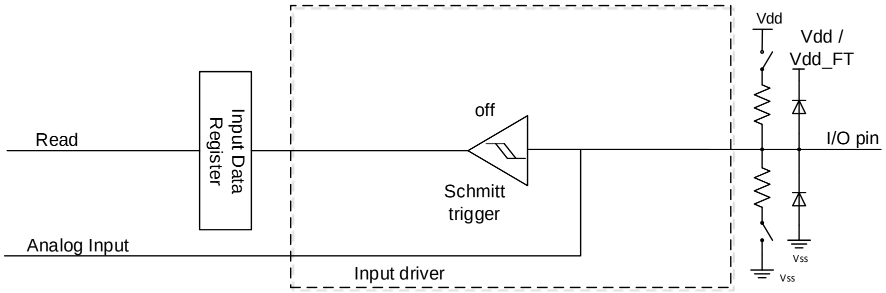
## 5.3.6. Analog configuration

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 Data Input Register gets the value “0”.

The figure below shows the high impedance-analog configuration

**Figure 5-4. High impedance-analog configuration**



## 5.3.7. Alternate function (AF) configuration

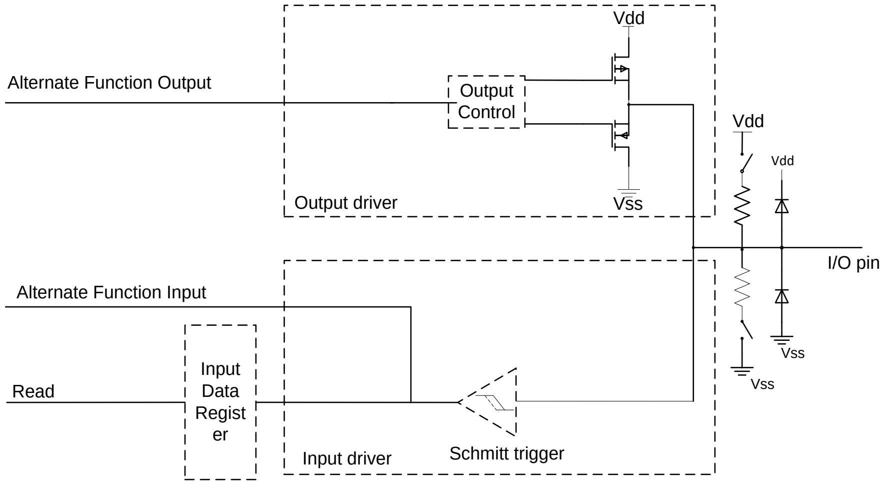
To suit for different device packages, the GPIO supports some alternate functions to some other pins by software.

When be configured as Alternate Function:

- The Output Buffer is turned on 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 Data Input Register every AHB2 clock cycle
- A read access to the Data Input Register gets the I/O state in open drain mode
- A read access to the Data Output register gets the last written value in Push-Pull mode

The figure below shows the Alternate Function Configuration of the I/O Port bit.

**Figure 5-5. Alternate function configuration**



### 5.3.8. 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_AFSEL[1:0]. It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx_LOCK). When the 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.

## 5.4. GPIO registers

### 5.4.1. Port control register (GPIOx_CTL) (x=A..D,F)

Address offset: 0x00

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

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 |
|---|---|---|---|---|---|---|---|
| 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<br>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. |

| | | |
|---|---|---|
| | | 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 state)<br>01: GPIO output mode<br>10: Alternate function mode.<br>11: Analog mode |

### 5.4.2. Port output mode register (GPIOx_OMODE) (x=A..D,F)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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. |

| | | |
|---|---|---|
| | | 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.<br>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)<br>1: Output open-drain mode |

### 5.4.3. Port output speed register (GPIOx_OSPD) (x=A..D,F)

Address offset: 0x08

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

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 |
|---|---|---|---|---|---|---|---|
| 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.<br>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. |

| | | |
|---|---|---|
| | | 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 state)<br>01: Output max speed 10M<br>11: Output max speed 50M |

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

Address offset: 0x0C

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

| | | |
|---|---|---|
| | | 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.<br>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. |

| | | |
|---|---|---|
| | | 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 state)<br>01: With pull-up mode<br>10: With pull-down mode<br>11: Reserved |

### 5.4.5. Port input status register (GPIOx_ISTAT) (x=A..D,F)

Address offset: 0x10
eset value: 0x0000 XXXX.

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISTAT15 | ISTAT14 | ISTAT13 | ISTAT12 | ISTAT11 | ISTAT10 | ISTAT9 | ISTAT8 | ISTAT7 | ISTAT6 | ISTAT5 | ISTAT4 | ISTAT3 | ISTAT2 | ISTAT1 | ISTAT0 |
| 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 | ISTAT[15:0] | Port input data<br>These bits are set and cleared by hardware.<br>0: Input signal low<br>1: Input signal high |

### 5.4.6. Port output control register (GPIOx_OCTL) (x=A..D,F)

Address offset: 0x14
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | OCTL[15:0] | Port output data<br>These bits are set and cleared by software.<br>0: Pin output low<br>1: Pin output high |

### 5.4.7. Port bit operate register (GPIOx_BOP) (x=A..D,F)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | CRx | Port Clear bit<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLx bit<br>1: Clear the corresponding OCTLx bit |
| 15:0 | BOPx[15:0] | Port Set bit<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLx bit<br>1: Set the corresponding OCTLx bit |

### 5.4.8. 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 setted using the Lock Key Writing Sequence.And can always be read.<br>0: Port configuration lock key not active<br>1: Port configuration lock key active.<br>GPIO_LOCK register is locked until an MCU reset..<br><br>LOCK key writing sequence<br>Write 1→Write 0→Write 1→ Read 0→ Read 1<br>**Note:** The value of LKx[15:0] must hold during the LOCK Key Writing sequence. |
| 15:0 | LKx[15:0] | Port Lock bit 0 ~ 15<br>These bits are set and cleared by software<br>0: Port configuration not locked<br>1: Port configuration locked |

### 5.4.9. Alternate function selected register0 (GPIOx_AFSEL0) (x=A, B, C)

Address offset: 0x20

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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. |

| | | |
|---|---|---|
| | | 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)<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>1000 ~ 1111: Reserved |

### 5.4.10. Alternate function selected register1 (GPIOx_AFSEL1) (x=A,B,C)

Address offset: 0x24

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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.<br>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>1000 ~ 1111: Reserved |

### 5.4.11. Bit clear register (GPIOx_BC) (x=A..D,F)

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 | | | | | | | | | | | | | | | |

| 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 | CRx[15:0] | Port Clear bit<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLx bit<br>1: Clear the corresponding OCTLx bit |

### 5.4.12. Port bit toggle register (GPIOx_TGx) (x=A..D,F) of GD32F170xx and GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | TGx[15:0] | Port Toggle bit<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLx bit<br>1: Toggle the corresponding OCTLx bit |

# 6. CRC calculation unit

## 6.1. Introduction

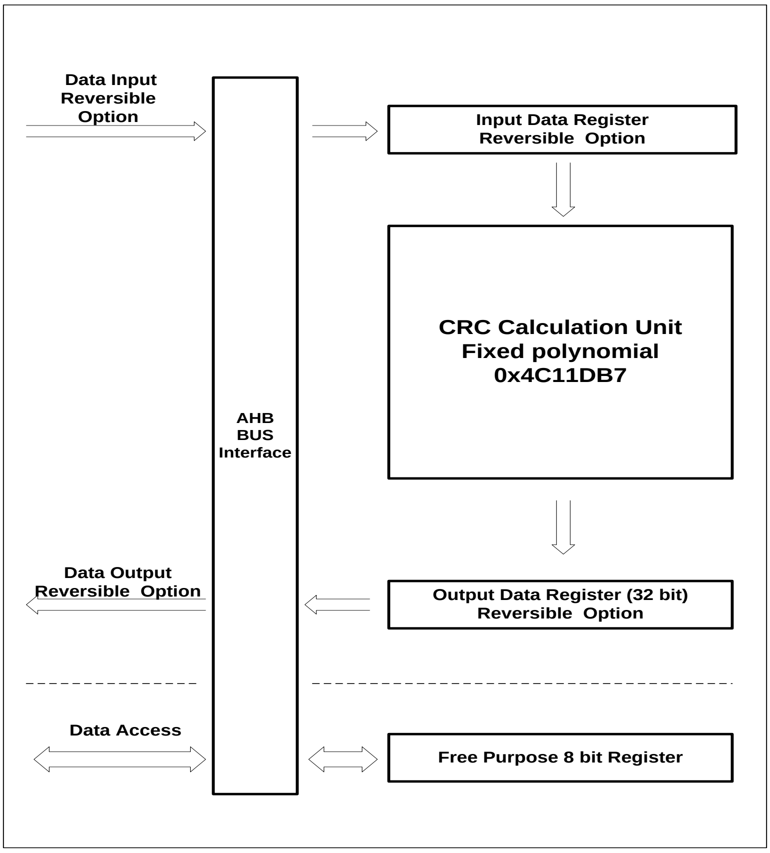
A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

This CRC calculation unit can be used to calculate 32/16/8 bit CRC code within fixed polynomial.

## 6.2. Main features

- 32/16/8 bit data input and 32-bit data output. Calculation period is 4/2/1 AHB Clock cycles for 32/16/8 bit input data size from data entered to the calculation result vailable.
- Free 8-bit register is unrelated for calculation and can be used for any other goals by any other peripheral devices.
- Fixed polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} +X^{8} + X^{7} + X^{5} + X^{4} + X^{2}+ X + 1$
  This 32-bit CRC polynomial is a common polynomial used in Ethernet.
- One 32-bit Input buffer for higher bus efficiency
- Reversible option for input and output data
- User setting initial value for flexible calculation

Figure 6-1.Block Diagram of CRC Calculation Unit



## 6.3. Function description

There are five functions in this unit:

- CRC calculation 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 do not clear the CRC_DATA register by software setting CRC_CTL register, the new input raw data will calculate based on the result of previous value of CRC_DATA.
  CRC calculation will spend 4/2/1 AHB clock cycles for 32/16/8 bit data size, during this period AHB will not be hanged because the existence of the 32-bit 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 0x1A2B3C4D:
  1) byte reverse:
  32-bit data is divided into 4 groups and reverse implement in group inside. Reversed data: 0x58D43CB2.
  2) half-word reverse:

32-bit data is divided in to 2 groups and reverse implement in group inside. Reversed data: 0xD458B23C.

3) word reverse:

32-bit data is divided in to 1 group and reverse implement in group inside. Reversed data: 0xB23CD458

For output data, reverse type is word reverse.

For example: when REV_O=1, calculation result 0x22CC4488 will be converted to 0x11223344.

- Multiple input data size support function will make users have the flexibility to adjust the combination of the calculation data.
  For example: 6 bytes input data can be combined by 1 word and 1 half-word, also it can be combined by 3 half-word.
- User configurable initial calculation data function will support user calculate CRC data value under any initial value.

# 6.4. CRC Registers

## 6.4.1. Data Register (CRC_DATA)

Address offset: 0x00

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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] | 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 CRC calculation result. If the input calculation data is less than 4 byte, the actual valid byte is the corresponding least significant byte. |

## 6.4.2. Free Data Register (CRC_FDATA)

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 | | | | | | | | FDATA[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | FDATA[7:0] | Free Data Register Bits<br>Software write and read.<br>These bits are unrelated with CRC calculation. This byte can be used for any goals by any other peripheral. The CRC_CTL register will generate no effect to the byte. |

### 6.4.3. Control Register (CRC_CTL)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | REV_O | REV_I[1:0] | | Reserved | | | | RST |
| | | | | | | | | rw | rw | | | | | | rs |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | REV_O | Output Data Reverse Function<br>0: Output data not reverse<br>1: Output data reversed by 32-bit |
| 6:5 | REV_I[1:0] | Input Data Reverse Function<br>0x0: Input data not reverse<br>0x1: Input data reversed by byte type<br>0x2: Input data reversed by half-word type<br>0x3: Input data reversed by word type |
| 4:1 | Reserved | Must be kept at reset value |
| 0 | RST | This bit can reset the CRC_DATA register to the value of 0xFFFFFFFF then |

automatically cleared itself to 0 by hardware. This bit will generate no effect to CRC_FDATA.

Software writes and reads.

### 6.4.4. 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<br>. |

# 7. Interrupts and events

## 7.1. Introduction

Cortex-M3 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and controls power management. It's tightly coupled to the processor core. You can read the Technical Reference Manual of Cortex-M3 for more details about NVIC.

An external interrupt/event controller (EXTI) is provided in chip which contains 23 independent edge detectors and generates interrupt requests or events to the processor. 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.

## 7.2. Main features

- Cortex-M3 system exception
- Up to 52 maskable peripheral interrupts for GD32F130xx and GD32F150xx devices or 74 maskable peripheral interrupts for GD32F170xx and GD32F190xx devices
- 4 bits interrupt priority configuration-16 priority levels
- Efficient interrupt processing
- Support exception pre-emption and tail-chaining
- Wake up system from power saving mode
- Up to 23 independent edge detectors in EXTI
- 3 trigger types: rising, falling and both edges
- Software interrupt or event trigger
- Trigger sources configurable

## 7.3. Function description

### 7.3.1. NVIC and exception/interrupt processing

The ARM Cortex-M3 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.

Table 7-1. NVIC exception types in Cotrex-M3

| 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 |
| **MemManage** | 4 | Programmable | 0x0000_0010 | Memory management |
| **BusFault** | 5 | Programmable | 0x0000_0014 | Prefetch fault, memory access fault |
| **UsageFault** | 6 | Programmable | 0x0000_0018 | Undefined instruction or illegal state |
| - | 7-10 | - | 0x0000_001C - 0x0000_002B | Reserved |
| **SVCall** | 11 | Programmable | 0x0000_002C | System service call via SWI instruction |
| **Debug Monitor** | 12 | Programmable | 0x0000_0030 | Debug Monitor |
| **-** | 13 | - | 0x0000_0034 | Reserved |
| **PendSV** | 14 | Programmable | 0x0000_0038 | Pendable request for system service |
| **SysTick** | 15 | Programmable | 0x0000_003C | System tick timer(b) |
| **Interrupts** | 16-89 | Programmable | 0x0000_0040 - 0x0000_0164 | Peripheral Interrupts (See below table for detail) |

Table 7-2. Interrupt vector table of GD32F130xx and GD32F150xx devices

| Interrupt Number | Vector Number | Peripheral Interrupt Description | Vector Address |
|---|---|---|---|
| **IRQ 0** | 16 | Window watchdog interrupt | 0x0000_0040 |
| **IRQ 1** | 17 | LVD through EXTI Line detection interrupt | 0x0000_0044 |
| **IRQ 2** | 18 | RTC global interrupt | 0x0000_0048 |
| **IRQ 3** | 19 | Flash 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 | TSI global interrupt | 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 CMP0-1 interrupt | 0x0000_0070 |
| **IRQ 13** | 29 | TIMER0 Break, update, trigger and commutation interrupt | 0x0000_0074 |

| Interrupt Number | Vector Number | Peripheral Interrupt Description | Vector Address |
|---|---|---|---|
| **IRQ 14** | 30 | TIMER0 Capture Compare interrupt | 0x0000_0078 |
| **IRQ 15** | 31 | TIMER1 global interrupt | 0x0000_007C |
| **IRQ 16** | 32 | TIMER2 global interrupt | 0x0000_0080 |
| **IRQ 17** | 33 | TIMER5 and DAC global 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 | CEC global interrupt | 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 | I2C2 event interrupt | 0x0000_00CC |
| **IRQ 36** | 52 | I2C2 error interrupt | 0x0000_00D0 |
| **IRQ 37** | 53 | USBD Low Priority interrupts | 0x0000_00D4 |
| **IRQ 38** | 54 | USBD High Priority interrupts | 0x0000_00D8 |
| **IRQ 39-41** | 55-57 | Reserved | 0x0000_00DC-<br>0x0000_00E4 |
| **IRQ 42** | 58 | USBD Wake Up through EXTI Line18 interrupt | 0x0000_00E8 |
| **IRQ 43-47** | 59-63 | Reserved | 0x0000_00EC-<br>0x0000_00FC |
| **IRQ 48** | 64 | DMA Channel5-6 global interrupt | 0x0000_0100 |
| **IRQ 49-50** | 65-66 | Reserved | 0x0000_0104-<br>0x0000_0108 |
| **IRQ 51** | 67 | SPI2 global interrupt | 0x0000_010C |

**Table 7-3. Interrupt vector table of GD32F170xx and GD32F190xx devices**

| Interrupt Number | Vector Number | Peripheral Interrupt Description | Vector Address |
|---|---|---|---|
| **IRQ 0** | 16 | Window watchdog timer interrupt | 0x0000_0040 |
| **IRQ 1** | 17 | LVD through EXTI Line detection interrupt | 0x0000_0044 |
| **IRQ 2** | 18 | RTC global interrupt | 0x0000_0048 |

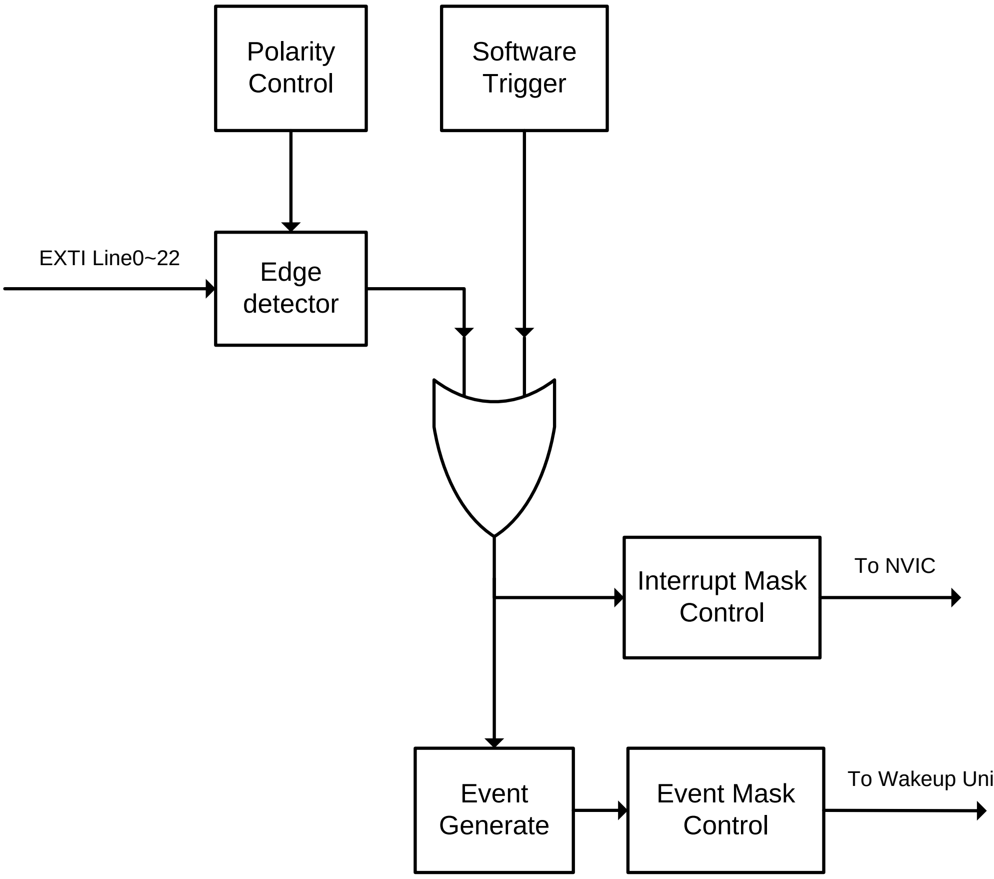
| Interrupt Number | Vector Number | Peripheral Interrupt Description | Vector Address |
|---|---|---|---|
| **IRQ 3** | 19 | Flash global interrupt | 0x0000_004C |
| **IRQ 4** | 20 | RCU global interrupt | 0x0000_0050 |
| **IRQ 5** | 21 | EXTI Line0-1 interrupt | 0x0000_0054 |
| **IRQ 6** | 22 | EXTI Line2-3 interrupt | 0x0000_0058 |
| **IRQ 7** | 23 | EXTI Line4-15 interrupt | 0x0000_005C |
| **IRQ 8** | 24 | TSI global interrupt | 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 CMP0-1 interrupt | 0x0000_0070 |
| **IRQ 13** | 29 | TIMER0 Break, update, trigger and commutation interrupt | 0x0000_0074 |
| **IRQ 14** | 30 | TIMER0 Capture Compare interrupt | 0x0000_0078 |
| **IRQ 15** | 31 | TIMER1 global interrupt | 0x0000_007C |
| **IRQ 16** | 32 | TIMER2 global interrupt | 0x0000_0080 |
| **IRQ 17** | 33 | TIMER5 and DAC global 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 | CEC global interrupt | 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 | I2C2 event interrupt | 0x0000_00CC |
| **IRQ 36** | 52 | I2C2 error interrupt | 0x0000_00D0 |
| **IRQ 37-42** | 53-58 | Reserved | 0x0000_00D4-<br>0x0000_00E8 |
| **IRQ43** | 59 | CAN0_TX | 0x0000_00EC |
| **IRQ44** | 60 | CAN0_RX0 | 0x0000_00F0 |
| **IRQ45** | 61 | CAN0_RX1 | 0x0000_00F4 |

| Interrupt Number | Vector Number | Peripheral Interrupt Description | Vector Address |
|---|---|---|---|
| **IRQ46** | 62 | CAN0_SCE | 0x0000_00F8 |
| **IRQ 47** | 63 | SLCD | 0x0000_00FC |
| **IRQ 48** | 64 | DMA Channe5-6 global interrupt | 0x0000_0100 |
| **IRQ 49-50** | 65-66 | Reserved | 0x0000_0104-<br>0x0000_0108 |
| **IRQ 51** | 67 | SPI2 global interrupt | 0x0000_010C |
| **IRQ52-69** | 68-85 | Reserved | 0x0000_0110-<br>0x0000_0154 |
| **IRQ70** | 86 | CAN1_TX | 0x0000_0158 |
| **IRQ71** | 87 | CAN1_RX0 | 0x0000_015C |
| **IRQ72** | 88 | CAN1_RX1 | 0x0000_0160 |
| **IRQ73** | 89 | CAN1_SCE | 0x0000_0164 |

### 7.3.2. External Interrupt and Event (EXTI)

The EXTI contains 23 independent edge detectors and generates 28 interrupt requests or events to the processor. The EXTI has three trigger types for the 23 edge detectors: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently. The figure below is the block diagram of EXTI.

Figure 7-1. Block diagram of EXTI.



The EXTI trigger source includes 16 lines from I/O pins and 8 lines for GD32F130xx and GD32F150xx devices (including LVD, RTC, USBD, USART, CEC and CMP, please refer to Table 7-4 for detail) or 7 lines for GD32F170xx and GD32F190xx devices (including LVD, RTC, USART, CEC and CMP, please refer to Table 7-5 for detail) from the internal modules. All GPIO pins can be selected as an EXTI trigger source by configuring SYSCFG_EXTISSx registers in SYSCFG module (please refer to System section for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex-M3 processor fully implements the Wait-For-Interrupt (WFI), Wait-For-Event (WFE) and the Send Event (SEV) instructions. There is a Wake-up Interrupt Controller (WIC) in chip, which enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and events. EXTI can be used to wake up the processor and the whole system when some expected events occur, such as a special I/O pin toggling or RTC alarm.

The internal trigger sources from CEC and USART are able to generate event for waking up the system. However, the two modules are also requested to generate a synchronous interrupt for the processor to wake up the CPU from Deep-sleep mode. Both internal trigger lines can be masked by setting corresponding INTEN and EVEN registers in EXTI module.

**Hardware Trigger**

Hardware trigger may be used to detect the voltage change of external or internal signals. The software should follow these steps to use this function:

1. Configure EXTI sources in SYSCFG module based on application requirement.
2. Configure EXTI_RTEN and EXTI_FTEN to enable the rising or falling detection on related pins. (Software may set both RTENx and FTENx for a pin at the same time to detect both rising and falling changes on this pin).
3. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
4. EXTI starts to detect changes on the configured pins. The related PDx bits in EXTI_PD will be set when desired change is detected on these pins and thus, trigger interrupt or event for software. The software should response to the interrupts or events and clear these PDx bits.

**Software Trigger**

Software may also trigger EXTI interrupts or events following these steps:

1. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
2. Set SWIEN**x** bits in EXTI_SWIEN register. The related PD bits will be set immediately and thus, trigger interrupts or events. Software should response to these interrupts, and clear related PDx bits.

**Table 7-4. EXTI source of GD32F130xx and GD32F150xx devices**

| EXTI Line Number | Source | Attrubute |
|---|---|---|
| 0 | PA0 / PB0 / PC0 / PF0 | External |
| 1 | PA1 / PB1 / PC1 / PF1 | External |
| 2 | PA2 / PB2 / PC2 / PD2 | External |
| 3 | PA3 / PB3 / PC3 | External |
| 4 | PA4 / PB4 / PC4 / PF4 | External |
| 5 | PA5 / PB5 / PC5 / PF5 | External |
| 6 | PA6 / PB6 / PC6 / PF6 | External |
| 7 | PA7 / PB7 / PC7 / PF7 | External |
| 8 | PA8 / PB8 / PC8 | External |
| 9 | PA9 / PB9 / PC9 | External |
| 10 | PA10 / PB10 / PC10 | External |
| 11 | PA11 / PB11 / PC11 | External |
| 12 | PA12 / PB12/ PC12 | External |
| 13 | PA13 / PB13 / PC13 | External |
| 14 | PA14 / PB14 / PC14 | External |
| 15 | PA15 / PB15 / PC15 | External |
| 16 | LVD | External |
| 17 | RTC Alarm | External |
| 18 | USBD Wakeup | External |
| 19 | RTC tamper and Timestamp | External |
| 20 | Reserved | Reserved |
| 21 | CMP0 output | External |
| 22 | CMP1 output | External |

| EXTI Line Number | Source | Attrubute |
|---|---|---|
| 23 | Reserved | Reserved |
| 24 | Reserved | Reserved |
| 25 | USART0 Wakeup | Internal |
| 26 | Reserved | Reserved |
| 27 | CEC Wakeup | Internal |

**Table 7-5. EXTI source of GD32F170xx and GD32F190xx devices**

| EXTI Line Number | Source | Attrubute |
|---|---|---|
| 0 | PA0 / PB0 / PC0 / PF0 | External |
| 1 | PA1 / PB1 / PC1 / PF1 | External |
| 2 | PA2 / PB2 / PC2 / PD2 | External |
| 3 | PA3 / PB3 / PC3 | External |
| 4 | PA4 / PB4 / PC4 / PF4 | External |
| 5 | PA5 / PB5 / PC5 / PF5 | External |
| 6 | PA6 / PB6 / PC6 / PF6 | External |
| 7 | PA7 / PB7 / PC7 / PF7 | External |
| 8 | PA8 / PB8 / PC8 | External |
| 9 | PA9 / PB9 / PC9 | External |
| 10 | PA10 / PB10 / PC10 | External |
| 11 | PA11 / PB11 / PC11 | External |
| 12 | PA12 / PB12/ PC12 | External |
| 13 | PA13 / PB13 / PC13 | External |
| 14 | PA14 / PB14 / PC14 | External |
| 15 | PA15 / PB15 / PC15 | External |
| 16 | LVD | External |
| 17 | RTC Alarm | External |
| 18 | Reserved | Reserved |
| 19 | RTC tamper and Timestamp | External |
| 20 | Reserved | Reserved |
| 21 | CMP0 output | External |
| 22 | CMP1 output | External |
| 23 | Reserved | Reserved |
| 24 | Reserved | Reserved |
| 25 | USART0 Wakeup | Internal |
| 26 | Reserved | Reserved |
| 27 | CEC Wakeup | Internal |

## 7.4. Interrupts and EXTI registers

### 7.4.1. Interrupt Enable register (EXTI_INTEN)

Address offset: 0x00

Reset value: 0x0F90 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 | INTEN**x** | Interrupt mask bit<br>0: Interrupt from Line**x** is disabled<br>1: Interrupt from Line**x** is enabled |

## 7.4.2. 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 | EVEN**x** | Event enable bit<br>0: Event from Line**x** is disabled<br>1: Event from Line**x** is enabled |

## 7.4.3. 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 | | | | | | | | | RTEN22 | RTEN21 | Reserved | RTEN19 | RTEN18 | RTEN17 | RTEN16 |
| | | | | | | | | | rw | 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 |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22:21 | RTEN**x** | Rising edge trigger enable<br>0: Rising edge of Line**x** is not valid<br>1: Rising edge of Line**x** is valid as an interrupt/event request |
| 20 | Reserved | Must be kept at reset value |
| 19 | RTEN19 | Rising edge trigger enable<br>0: Rising edge of Line19 is not valid<br>1: Rising edge of Line19 is valid as an interrupt/event request |
| 18 | RTEN18 | Rising edge trigger enable<br>This bit is valid only for GD32F150xx device.<br>0: Rising edge of Line18 is invalid<br>1: Rising edge of Line18 is valid as an interrupt/event request |
| 17: 0 | RTEN**x** | Rising edge trigger enable<br>0: Rising edge of Line**x** is invalid<br>1: Rising edge of Line**x** is valid as an interrupt/event request |

### 7.4.4. 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 | | | | | | | | | FTEN22 | FTEN21 | Reserved | FTEN19 | FTEN18 | FTEN17 | FTEN16 |
| | | | | | | | | | rw | rw | | 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: 23 | Reserved | Must be kept at reset value |
| 22: 21 | FTEN**x** | Falling edge trigger enable<br>0: Falling edge of Line**x** is invalid<br>1: Falling edge of Line**x** is valid as an interrupt/event request |
| 20 | Reserved | Must be kept at reset value |
| 19 | FTEN19 | Falling edge trigger enable<br>0: Falling edge of Line19 is invalid<br>1: Falling edge of Line19 is valid as an interrupt/event request |
| 18 | FTEN18 | Falling edge trigger enable<br>This bit is valid only for GD32F150xx device.<br>0: Falling edge of Line18 is invalid<br>1: Falling edge of Line18 is valid as an interrupt/event request |
| 17: 0 | FTEN**x** | Falling edge trigger enable<br>0: Falling edge of Line**x** is invalid<br>1: Falling edge of Line**x** is valid as an interrupt/event request |

### 7.4.5. 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 | | | | | | | | | SWIEV22 | SWIEV21 | Reserved | SWIEV19 | SWIEV18 | SWIEV17 | SWIEV16 |
| | | | | | | | | | rw | rw | | 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 |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value |
| 22: 21 | SWIEV**x** | Interrupt/Event software trigger<br>0: Deactivate the EXTI**x** software interrupt/event request<br>1: Activate the EXTI**x** software interrupt/event request |
| 20 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 19: 0 | SWIEV**x** | Interrupt/Event software trigger<br>0: Deactivate the EXTI**x** software interrupt/event request<br>1: Activate the EXTI**x** software interrupt/event request |

### 7.4.6. 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 | | | | | | | | | PD22 | PD21 | Reserved | PD19 | PD18 | PD17 | PD16 |
| | | | | | | | | | rc_w1 | rc_w1 | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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: 23 | Reserved | Must be kept at reset value |
| 22: 21 | PD**x** | Interrupt pending status<br>0: EXIT Line**x** is not triggered<br>1: EXIT Line**x** is triggered<br>This bit is cleared to 0 by writing 1 to it. |
| 20 | Reserved | Must be kept at reset value |
| 19: 0 | PD**x** | Interrupt pending status<br>0: EXIT Line**x** is not triggered<br>1: EXIT Line**x** is triggered<br>This bit is cleared to 0 by writing 1 to it. |

# 8. Direct memory access controller (DMA)

## 8.1. Introduction

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and memory or between memory and 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 7 channels in the DMA controller. Each channel is assigned a specific or multiple target peripheral devices for memory access request management. An arbiter is implemented inside to handle the priority among DMA requests.

Both the DMA controller and the Cortex$^{TM}$-M3 core implement data access through the system bus. An arbitration mechanisim is implemented to solve the competition between these two masters. When the same peripheral is targeted, the CPU access will be suspended for some specific bus cycles. A round-robin scheduling algorithm is utilized in the bus matrix to guaranty at least half the bandwidth to the CPU.

## 8.2. Main features

- Programmable length of data to be transferred, max to 65536
- 7 channels and each channel is configurable
- AHB and APB peripherals, FLASH, SRAM can be accessed as source and destination
- Each channel is connected to fixed hardware DMA request
- The priorities of DMA channel requests are determined by software configuration and hardware channel number
- Support peripheral to memory, memory to peripheral, and memory to memory transfers
- Three types of event flags and one single interrupt request for each channel
- Configurable transfer size of source and destination in a channel

## 8.3. Function description

### 8.3.1. DMA transfers

The handshake mechanism between the DMA controller and peripherals is based on the request signal and the acknowledge signal. The request singal indicates the peripheral is requesting for DMA access, while the acknowledge signal indicates the DMA controller has accessed the peripheral. When the DMA controller receives two or more requests at a time, the requests are served depending on the channel priorities. 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_CHxCTL0 registers. For details, see Chapter 8.3.3. The DMA_CHxCNT register controls how many transfers to be transmitted on the channel. The PWIDTH and MWIDTH bits in the DMA_CHxCTL0 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 following table.

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

| 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[15:0] @0x0<br>2: Write B5B4[15:0] @0x2<br>3: Write B9B8[15:0] @0x4<br>4: Write BDBC[15:0] @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 |

### 8.3.2. Arbitration among channels

When two or more requests are received at a 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 software priority is configured by the PRIO[1:0] bits in the DMA_CHxCTL0 register. There are four levels, including low, medium, high, and ultra high. And the hardware priority is fixed. The lower the channel number is, the higher the channel priority is. For instance, channel 0 has the higher priority over channel 2. The software priority is more significant than the hardware priority. If two channels are configured in different software priorities, the channel with higher software priority is served. If two channels are configured in the same software priority, the channel with higher hardware priority is served.

### 8.3.3. Next address generation algorithm

PWIDTH and MWIDTH bits in the DMA_CHxCTL0 register are used for configuring the transfer data size of peripheral and memory. PNAGA and MNAGA bits in the DMA_CHxCTL0 register are used to configure the next address generation algorithm of peripheral and memory. There are two algorithms including the fixed address mode and the increasing address mode. In the fixed address mode, the next address is equal to the current address. In the increasing address mode, the next address is the current address plus 1 or 2 or 4, depending on the transfer data size.

### 8.3.4. Circulation mode

Circulation mode is implemented for circular data flows (for example, ADC scan mode). The feature can be enabled by configuring the CMEN bit in the DMA_CHxCTL0 register. If enabled, the remain counter of the channel is automatically reloaded with the initial programmed value when it reaches zero. So the DMA requests are always served.

### 8.3.5. Memory to memory mode

The memory to memory mode is enabled by setting the M2M bit in the DMA_CHxCTL0 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_CHxCTL0 register, and stops when the DMA_CHxCNT register reaches zero.

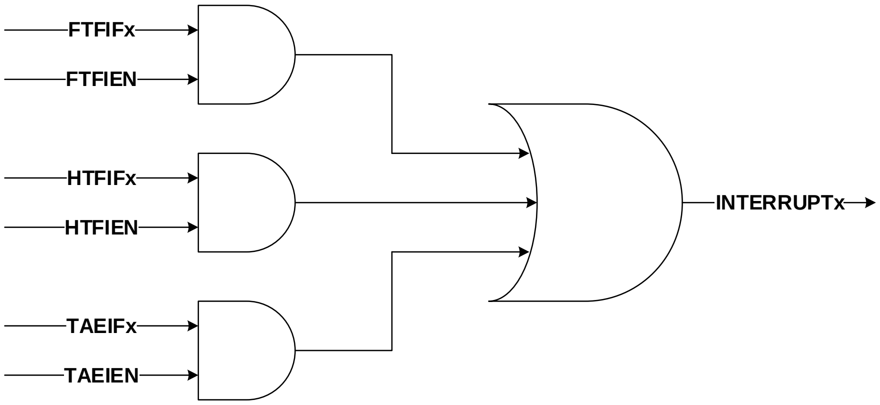
### 8.3.6. Interrupt requests

Each DMA channel has a dedicated interrupt. There are three types of interrupt event, including transfer complete, half transfer complete, and transfer error. An interrupt may be produced when any type of interrupt event occurs on the channel. 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_CHxCTL0 register. The relationship is described in the following table.

**Table 8-2. DMA interrupt event**

| Interrupt event | Flag bit | Clear bit | Enable bit |
|---|---|---|---|
| Transfer complete | FTFIF | FTFIFC | FTFIE |
| Half transfer complete | HTFIF | HTFIFC | HTFIE |
| Transfer error | TAEIF | TAEIFC | TAEIE |

**Figure 8-1. DMA interrupt generation logic**



**Note:** “x” indicates channel number (x=0…6).

The transfer error event occurs when the DMA controller accesses a reserved address space. At the moment, the channel is automatically shut down through hardware clear of the CHEN bit in the DMA_CHxCTL0 register.

### 8.3.7. DMA channel configuration procedure

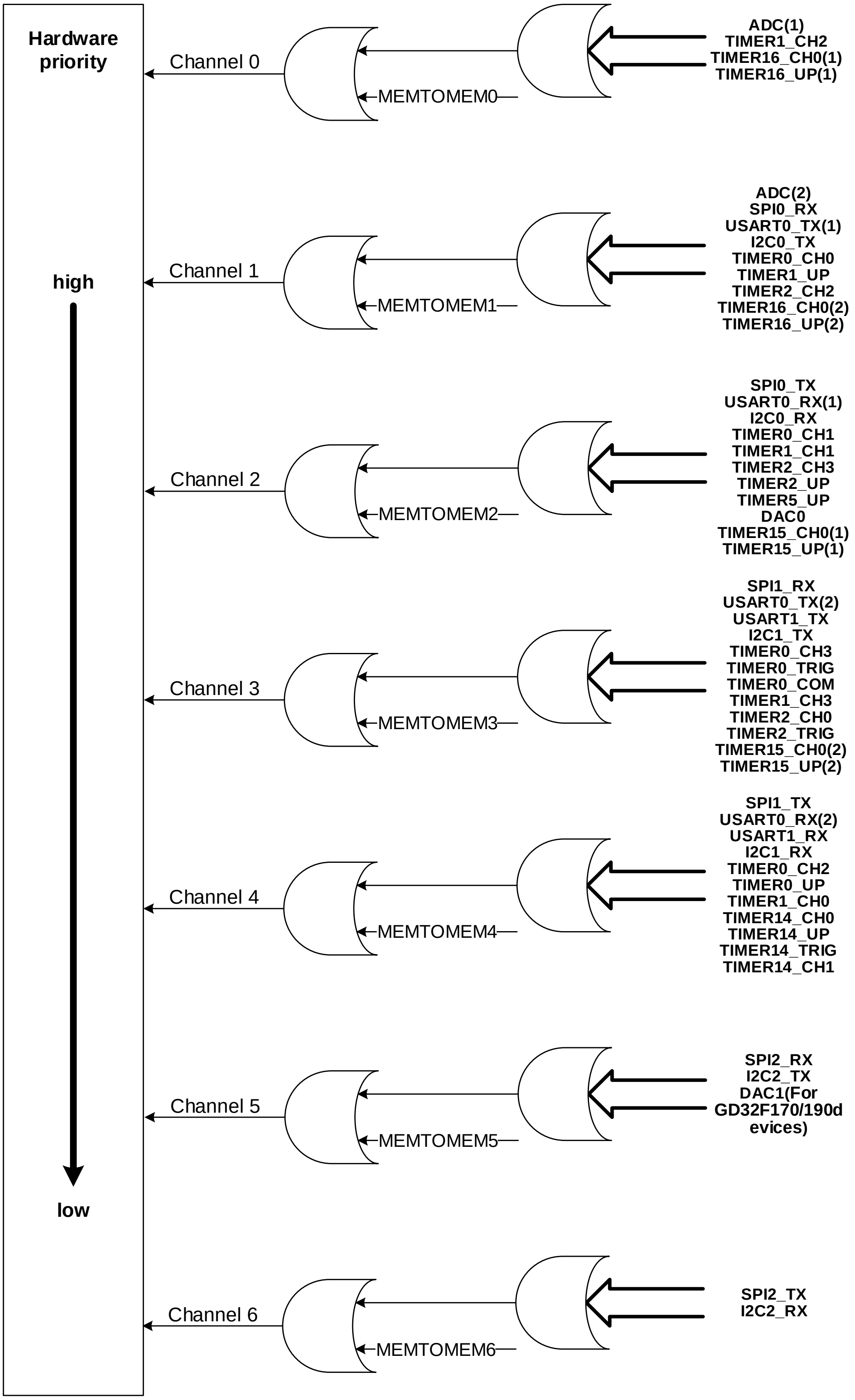
The following sequence should be followed to configure a DMA channel.

1. Configure the DMA_CHxPADDR register for setting the peripheral address.
2. Configure the DMA_CHxMADDR register for setting the memory address.
3. Configure the DMA_CHxCNT register for setting the total transfer data number.
4. Configure the DMA_CHxCTL0 register for channel software priority, transfer direction, mode type, data size and interrupt type.
5. Configure the DMA_CHxCTL0 register for setting the CHEN bit.

### 8.3.8. 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 following figure. 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.

Figure 8-2. DMA request mapping



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.

The following table lists the support request from peripheral of each channel.

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

| Peripheral | Channel 0 | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 |
|---|---|---|---|---|---|---|---|
| **ADC** | ADC(1) | ADC(2) | • | • | • | • | • |
| **SPI/I2S** | • | SPI0_RX | SPI0_TX | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX |
| **USART** | • | USART0_TX(1) | USART0_RX(1) | USART0_TX(2)<br>USART1_TX | USART0_RX(2)<br>USART1_RX | • | • |
| **I$^2$C** | • | I$^2$C0_TX | I$^2$C0_RX | I$^2$C1_TX | I$^2$C1_RX | I$^2$C2_TX | I$^2$C2_RX |
| **TIMER0** | • | TIMER0_CH0 | TIMER0_CH1 | TIMER0_CH3<br>TIMER0_TRIG<br>TIMER0_COM | TIMER0_CH2<br>TIMER0_UP | • | • |
| **TIMER1** | TIMER1_CH2 | TIMER1_UP | TIMER1_CH1 | TIMER1_CH3 | TIMER1_CH0 | • | • |
| **TIMER2** | • | TIMER2_CH2 | TIMER2_CH3<br>TIMER2_UP | TIMER2_CH0<br>TIMER2_TRIG | • | • | • |
| **TIMER5/DAC** | • | • | TIMER5_UP<br>DAC0 | • | • | DAC1 (For GD32F170/190devices) | • |
| **TIMER14** | • | • | • | • | TIMER14_CH0<br>TIMER14_UP<br>TIMER14_TRIG<br>TIMER14_COM | • | • |
| **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) | • | • | • | • | • |

## 8.4. DMA registers

### 8.4.1. Interrupt flag register (DMA_INTF)

Address offset: 0x00

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 | | | | TAEIF6 | HTFIF6 | FTFIF6 | GIF6 | TAEIF5 | HTFIF5 | FTFIF5 | GIF5 | TAEIF4 | HTFIF4 | FTFIF4 | GIF4 |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TAEIF3 | HTFIF3 | FTFIF3 | GIF3 | TAEIF2 | HTFIF2 | FTFIF2 | GIF2 | TAEIF1 | HTFIF1 | FTFIF1 | GIF1 | TAEIF0 | HTFIF0 | FTFIF0 | GIF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

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

| | | |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27/23/19/<br>15/11/7/3 | TAEIFx | Transfer access error flag of channel x (x=0…6)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Error has not occurred on channel x<br>1: Error has occurred on channel x |
| 26/22/18/<br>14/10/6/2 | HTFIFx | Half transfer finish flag of channel x (x=0…6)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Half number of transfer has not completed on channel x<br>1: Half number of transfer has completed on channel x |
| 25/21/17/<br>13/9/5/1 | FTFIFx | Full transfer finish flag of channel x (x=0…6)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: Transfer has not completed on channel x<br>1: Transfer has completed on channel x |
| 24/20/16/<br>12/8/4/0 | GIFx | Global interrupt flag of channel x (x=0…6)<br>Hardware set and software cleared by configuring DMA_INTC register.<br>0: None of TAEIF, HTFIF or FTFIF occurs on channel x<br>1: At least one of TAEIF, HTFIF or FTFIF occurs on channel x |

### 8.4.2. Interrupt flag clear register (DMA_INTC)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TAEIFC6 | HTFIFC6 | FTFIFC6 | GIC6 | TAEIFC5 | HTFIFC5 | FTFIFC5 | GIC5 | TAEIFC4 | HTFIFC4 | FTFIFC4 | GIC4 |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TAEIFC3 | HTFIFC3 | FTFIFC3 | GIC3 | TAEIFC2 | HTFIFC2 | FTFIFC2 | GIC2 | TAEIFC1 | HTFIFC1 | FTFIFC1 | GIC1 | TAEIFC0 | HTFIFC0 | FTFIFC0 | GIC0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27/23/19/<br>15/11/7/3 | TAEIFCx | Clear bit for transfer access error flag of channel x (x=0…6)<br>0: No effect<br>1: Clear TAEIFx bit in the DMA_INTF register |
| 26/22/18/<br>14/10/6/2 | HTFIFCx | Clear bit for half transfer finish flag of channel x (x=0…6)<br>0: No effect<br>1: Clear HTFIFx bit in the DMA_INTF register |
| 25/21/17/ | FTFIFCx | Clear bit for Full transfer finish flag of channel x (x=0…6) |

| | | |
|---|---|---|
| 13/9/5/1 | | 0: No effect<br>1: Clear FTFIFx bit in the DMA_INTF register |
| 24/20/16/<br>12/8/4/0 | GICx | Clear global interrupt flag of channel x (x=0…6)<br>0: No effect<br>1: Clear GIFx, TAEIFx, HTFIFx and FTFIFx bits in the DMA_INTF register |

### 8.4.3. Channel x control registers (DMA_CHxCTL0)

x = 0..6, where x is channel number

Address offset: 0x08 + 0d20 × (x)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | M2M | PRIO[1:0] | | MWIDTH[1:0] | | PWIDTH[1:0] | | MNAGA | PNAGA | CMEN | TM | TAEIE | 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<br>0: Disable Memory to Memory Mode<br>1: Enable Memory to Memory Mode |
| 13:12 | PRIO[1:0] | Priority Level of this channel<br>00: Low<br>01: Medium<br>10: High<br>11: Ultra high |
| 11:10 | MWIDTH[1:0] | Transfer data size of memory<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved |
| 9:8 | PWIDTH[1:0] | Transfer data size of peripheral<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved |
| 7 | MNAGA | Next address generation algorithm of memory |

| | | |
|---|---|---|
| | | 0: Fixed address mode<br>1: Increasing address mode |
| 6 | PNAGA | Next address generation algorithm of peripheral<br>0: Fixed address mode<br>1: Increasing address mode |
| 5 | CMEN | Circulation mode enable<br>0: Disable circulation mode.<br>1: Enable circulation mode |
| 4 | TM | Transfer mode<br>0: Read from peripheral and write to memory<br>1: Read from memory and write to peripheral |
| 3 | TAEIE | Enable bit for tranfer access error interrupt<br>0: Disable the channel error interrupt<br>1: Enable the channel error interrupt |
| 2 | HTFIE | Enable bit for half transfer finish interrupt<br>0: Disable HT interrupt<br>1: Enable HT interrupt |
| 1 | FTFIE | Enable bit for full transfer finish interrupt<br>0: Disable TC interrupt<br>1: Enable TC interrupt |
| 0 | CHEN | Channel enable<br>0: Disable channel<br>1: Enable channel |

### 8.4.4. Channel x counter registers (DMA_CHxCNT)

x = 0..6, where x is channel number
Address offset: 0x0C + 0d20 × (x)
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** |
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 15:0 | CNT[15:0] | Transfer counter<br>This register signifies the number of remaining data to be transferred by the DMA. It can be modified only when the channel is disabled. Once the data transfer is started, this register is read-only. After each DMA transfer, CNT is decreamented by 1. When CNT = 0, no DMA transfer can be issued whether the corresponding channel is enabled or not. CNT can be reloaded automatically to the original value after the overall DMA transfer operation depending on the channel reload configuration previously specified. |

### 8.4.5. Channel x peripheral base address registers (DMA_CHxPADDR)

x = 0..6, where x is channel number
Address offset: 0x10 + 0d20 × (x)
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit) .
**Note:** Do not configure this register when channel is enabled.

| 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>When PWIDTH is 01 (16-bit), PADDR[0] is ignored. Access is automatically aligned to a half word address.<br>When PWIDTH is 10 (32-bit), PADDR[1:0] is ignored. Access is automatically aligned to a word address. |

### 8.4.6. Channel x memory base address registers (DMA_CHxMADDR)

x = 0..6, where x is channel number
Address offset: 0x14 + 0d20 × (x)
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit) .
**Note: Do not configure this register when channel is enabled.**

| 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<br>When MWIDTH is 01 (16-bit), MADDR[0] is ignored. Access is automatically aligned to a half word address.<br>When MWIDTH is 10 (32-bit), MADDR[1:0] is ignored. Access is automatically aligned to a word address. |

# 9. Timer (TIMERx)

Timers (TIMERx) are devided into six sorts.

| TIMER | TIMER0 | TIMER1/2 | TIMER5 | TIMER13 | TIMER14 | TIMER15/16 |
|---|---|---|---|---|---|---|
| TYPE | Advanced | General(1) | Basic | General(2) | General(3) | General(4) |
| Prescaler | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit |
| Counte mode | UP,DOWN, Center-aligned | UP,DOWN, Center-aligned | UP ONLY | UP ONLY | UP ONLY | UP ONLY |
| Repetition | ● | × | × | × | ● | ● |
| Capture/ compare CH | 4 | 4 | 0 | 1 | 2 | 1 |
| Output Complementary & Deadtime | ● | × | × | × | ● | ● |
| Break | ● | × | × | × | ● | ● |
| Single Pulse | ● | ● | × | × | ● | ● |
| Quadrature Decoder | ● | ● | × | × | × | × |
| Slave Controller | ● | ● | × | × | ● | × |
| Inter connection | ●(1) | ●(2) | TRGO TO DAC | × | ●(3) | × |
| Debug Mode | ● | ● | ● | ● | ● | ● |
| DMA | ● | ● | ● | × | ● | ● |

(1) TIMER0: tm14_trgo->it0; tm1_trgo->it1; tm2_trgo->it2; 0->it3

(2) TIMER1: tm0_trgo->it0; tm14_trgo->it1; tm2_trgo->it2; 0->it3;
TIMER2: tm0_trgo->it0; tm1_trgo->it1; tm14_trgo->it2; 0->it3;

(3) TIMER14:tm1_trgo->it0; tm2_trgo->it1; 0->it2; 0->it3

# 9.1. Advanced timer (TIMER0)

## 9.1.1. Introduction

The Advanced Timer, known as TIMER0, may be used for a variety of purposes of advanced control. It consists of one 16-bit up/down-counter; four 16-bit capture/compare registers (TIMER0_CHxCV), one 16-bit counter auto reload register (TIMER0_CAR) and several control registers. They can be used for a variety of purposes including general timer, input signal pulse width measurement or output waveform generation such as single pulse generation or PWM output. The TIMER0 supports an Encoder Interface using a decoder with two inputs.

The advanced (TIMER0) and general (TIMERx) timers are completely independent. They do not share any resources but can be synchronized together.

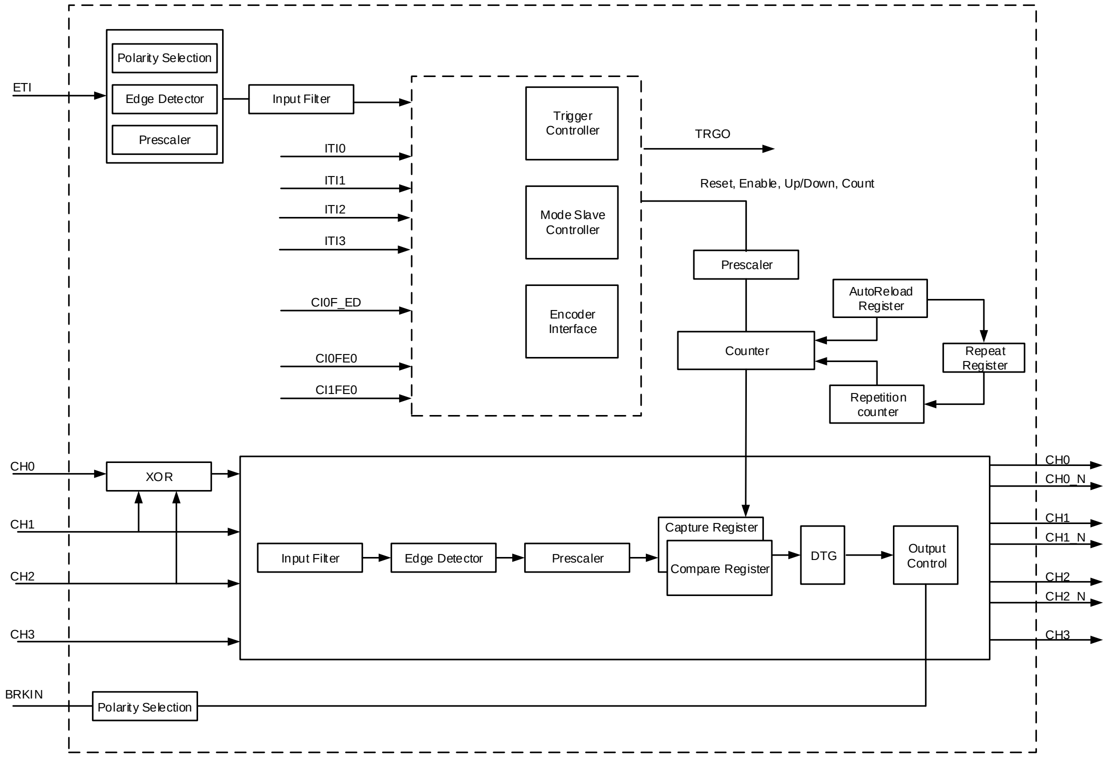
## 9.1.2. Main features

- 16-bit down, up, down/up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels support functions including input capture, compare match output, generation of PWM waveform (edge and center-aligned Mode), and single pulse mode output.
- Counter repetition to update the timer registers only after a given number of cycles of the counter.
- Break input to trigger the timer’s output signals to a known state.
- Interrupt/DMA generation by update, trigger event, input capture event, output compare match event or break input
- Programmable dead-time for output match.
- Synchronization circuit to control TIMER0 with external signals or to interconnect several timers together.
- Encoder interface controller with two inputs using quadrature decoder
- TIMER0 Master/Slave mode controller

## 9.1.3. Function description

Figure below provides details on the internal configuration of the advanced timer.

**Figure 9-1. Advanced timer block diagram**



## Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMER0_PSC) which can be changed on the fly but be taken into account at the next update event.

Figure 9-2. Counter timing diagram with prescaler division change from 1 to 2

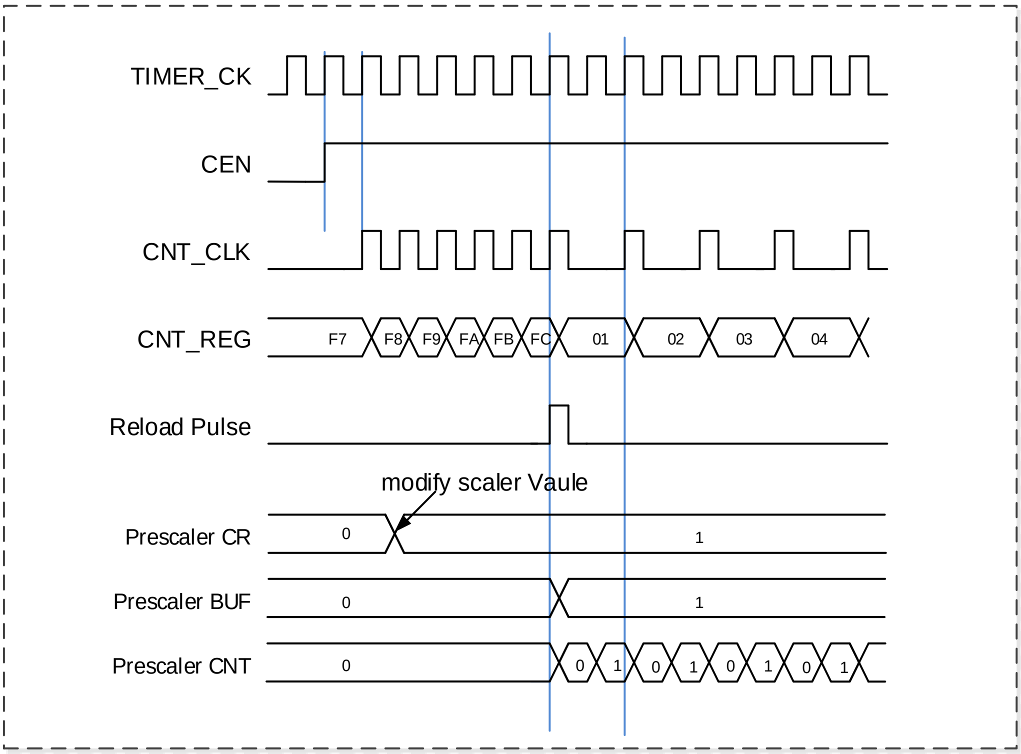
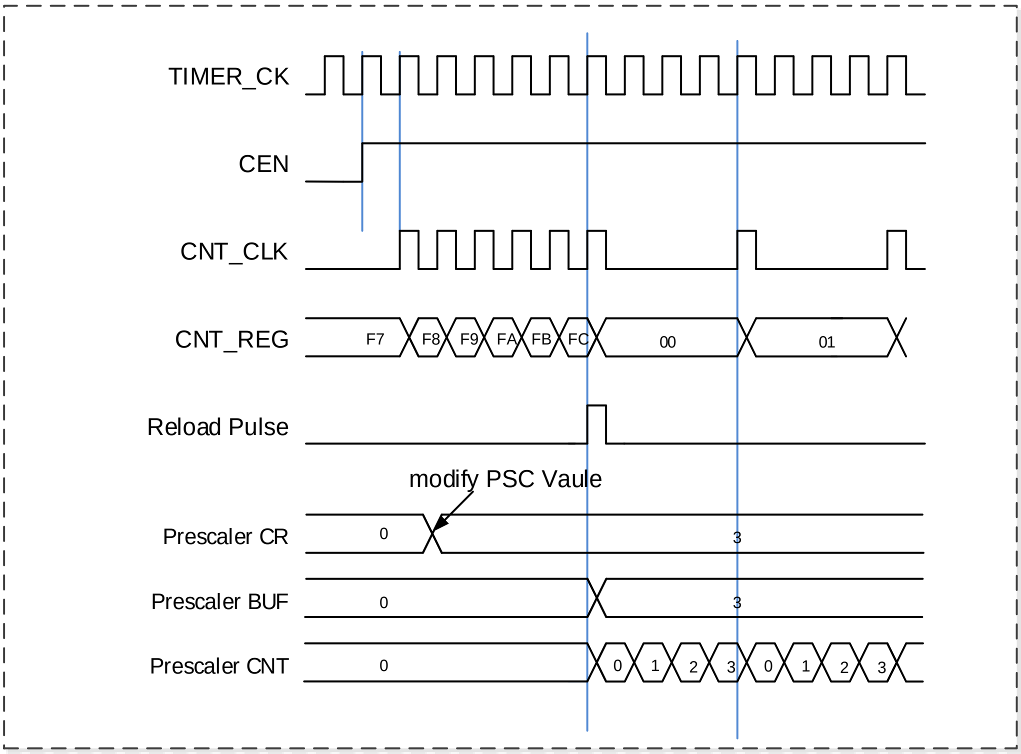


Figure 9-3. Counter timing diagram with prescaler division change from 1 to 4



## Upcounting mode

In this mode the counter counts continuously from 0 to the counter-reload value, which is

defined in the TIMER0_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. If the repetition counter is set, the update events generated after the number (TIMERx_CREP+1) of overflow. Else the update event is generated at each counter overflow.The counting direction bit DIR in the TIMER0_CTL0 register should be set to 0 for the upcounting mode.

When the update event is set by the UPG bit in the TIMER0_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMER0_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

**Figure 9-4. Counter timing diagram, internal clock divided by 1**

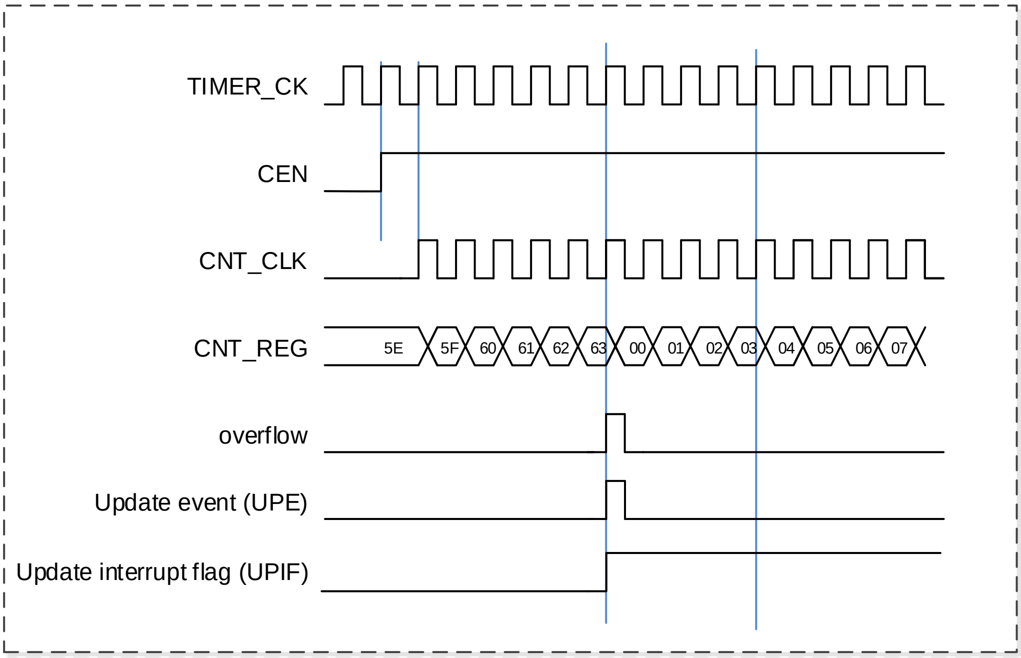


**Figure 9-5. Counter timing diagram, internal clock divided by 2**

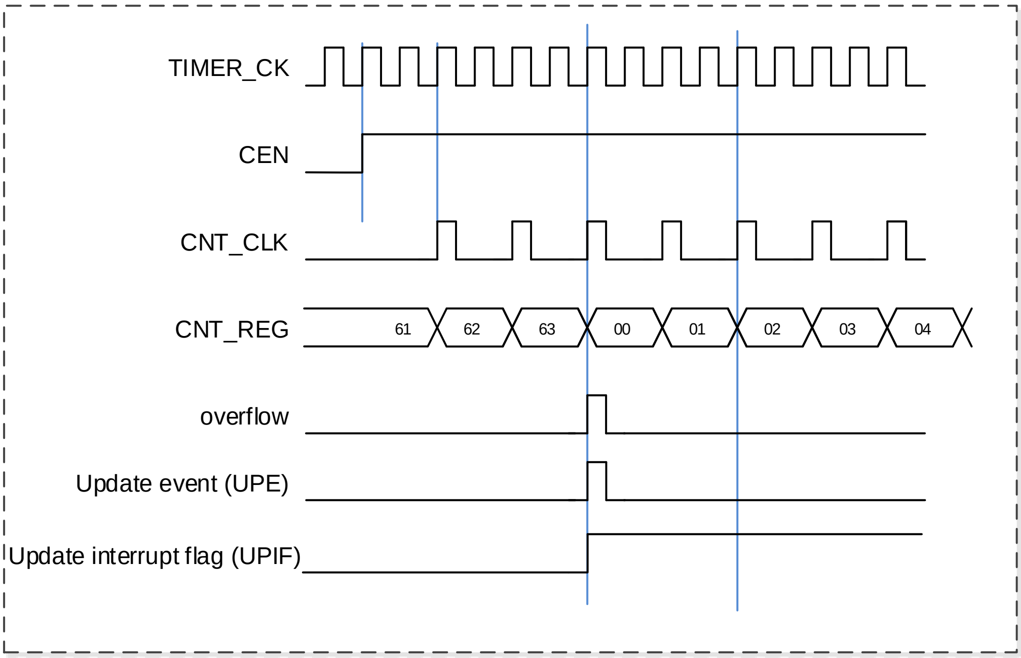


**Figure 9-6. Counter timing diagram, internal clock divided by 4**

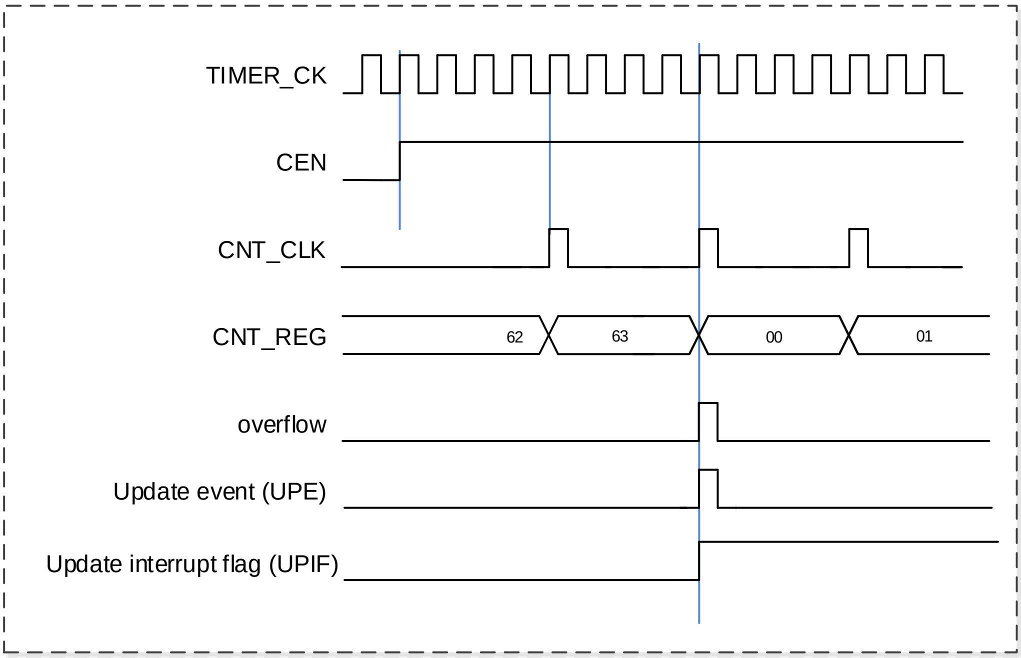


Figure 9-7. Counter timing diagram, internal clock divided by N

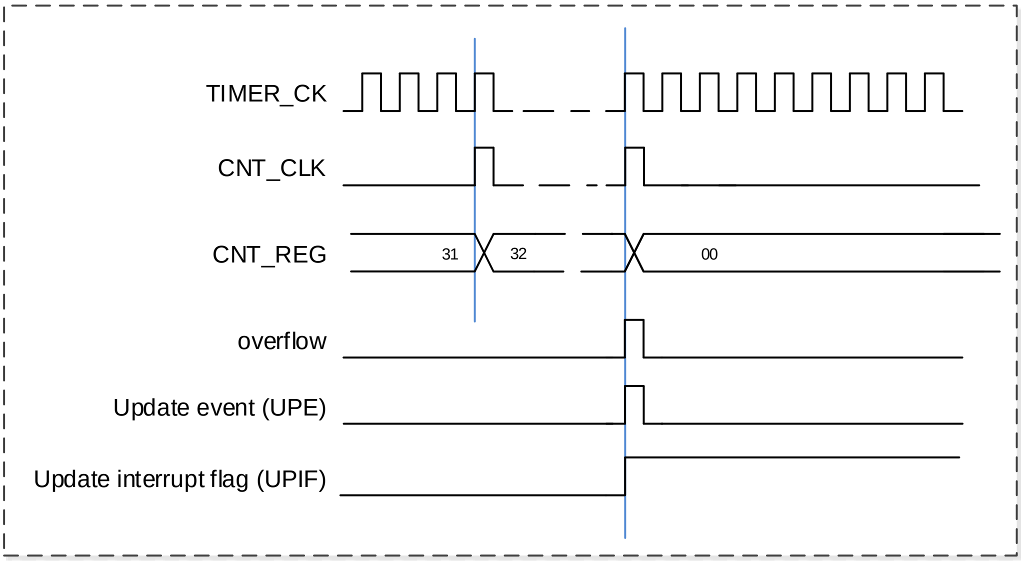


Figure 9-8. Counter timing diagram, update event when ARSE=0

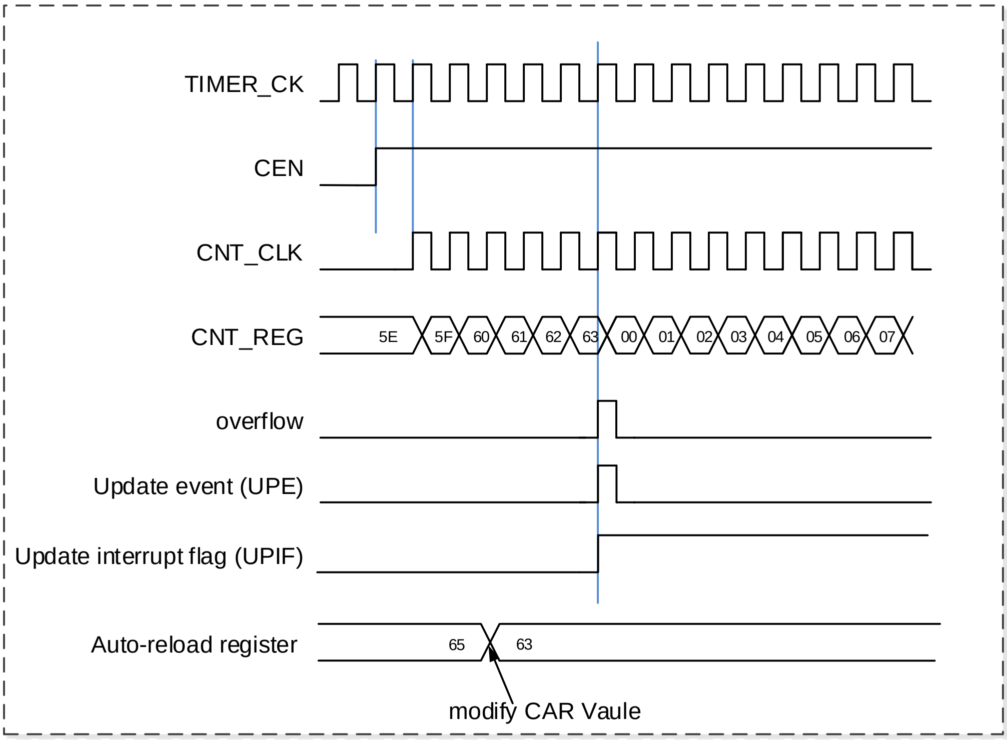
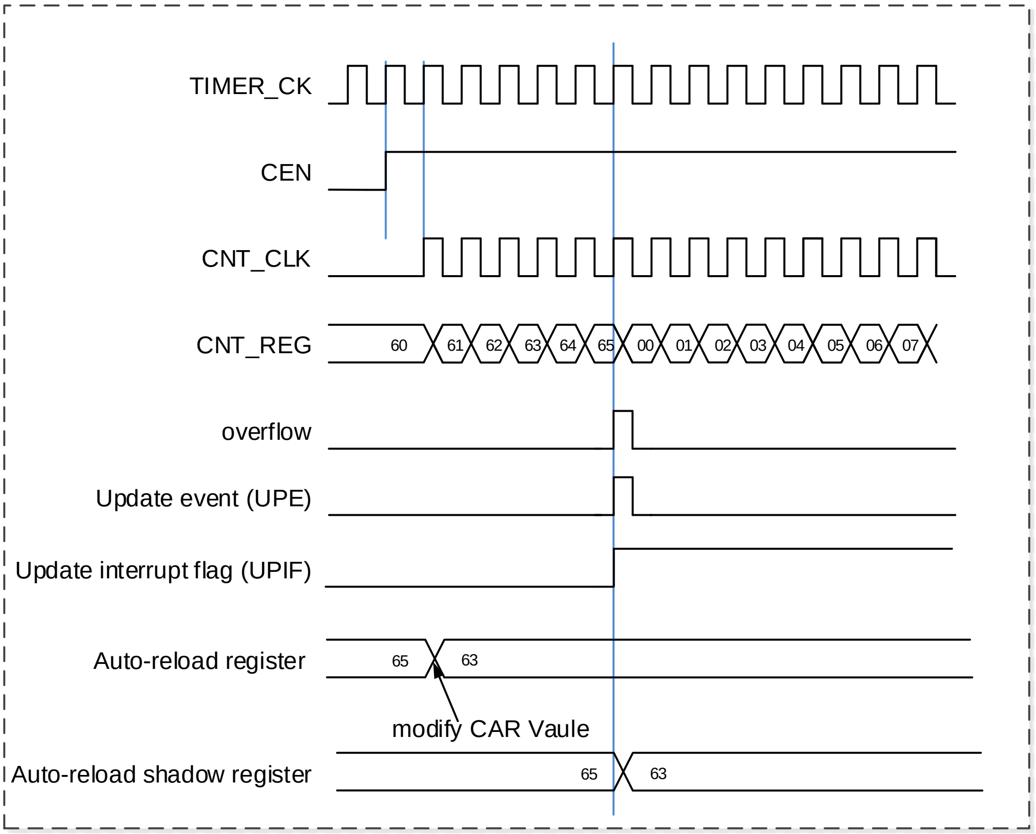


**Figure 9-9. Counter timing diagram, update event when ARSE=1**



## Downcounting mode

In this mode the counter counts continuously from the counter-reload value, which is defined in the TIMER0_CAR register, to 0 in a count-down direction. Once the counter reaches 0, the counter restarts to count once again from the counter-reload value. If the repetition counter is set, the update event generated after the number (TIMERx_CREP+1) of underflow. Else the update event is generated at each counter underflow. The counting direction bit DIR in the TIMER0_CTL0 register should be set to 1 for the down-counting mode.

When the update event is set by the UPG bit in the TIMER0_SWEVG register, the counter value will be initialized to the counter-reload value and generates an update event.

If set the UPDIS bit in TIMER0_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

Figure 9-10. Counter timing diagram, internal clock divided by 1

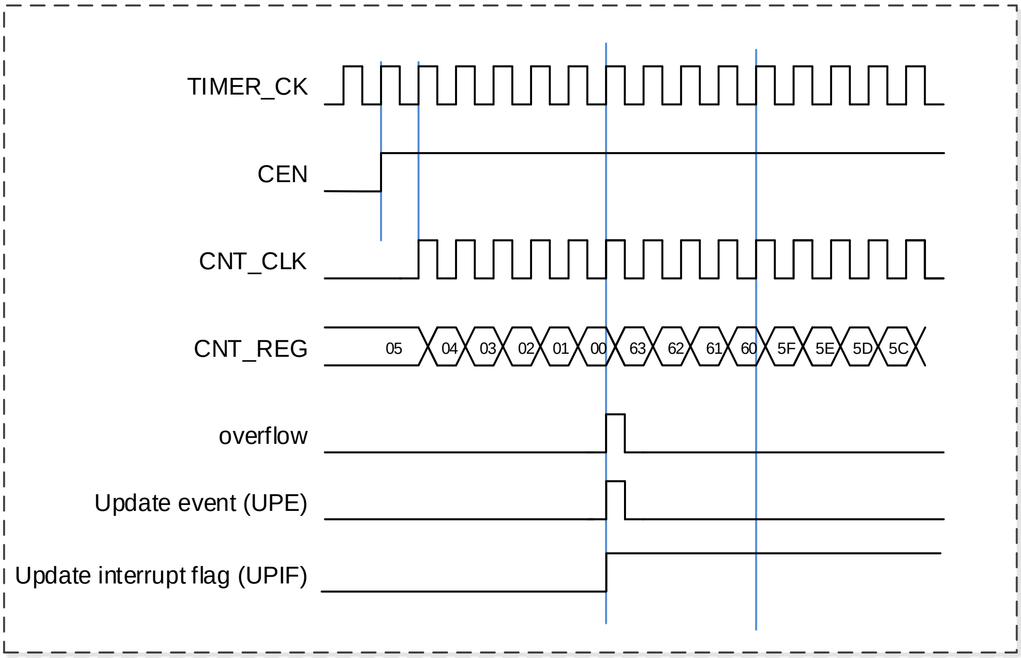


**Figure 9-11. Counter timing diagram, internal clock divided by 2**

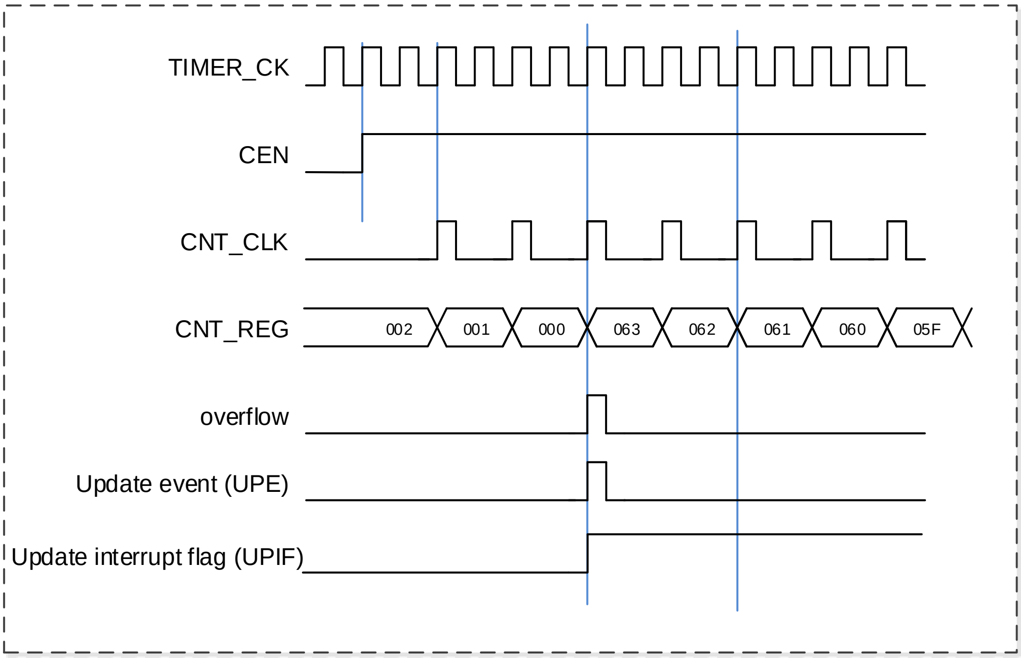


**Figure 9-12. Counter timing diagram, internal clock divided by 4**

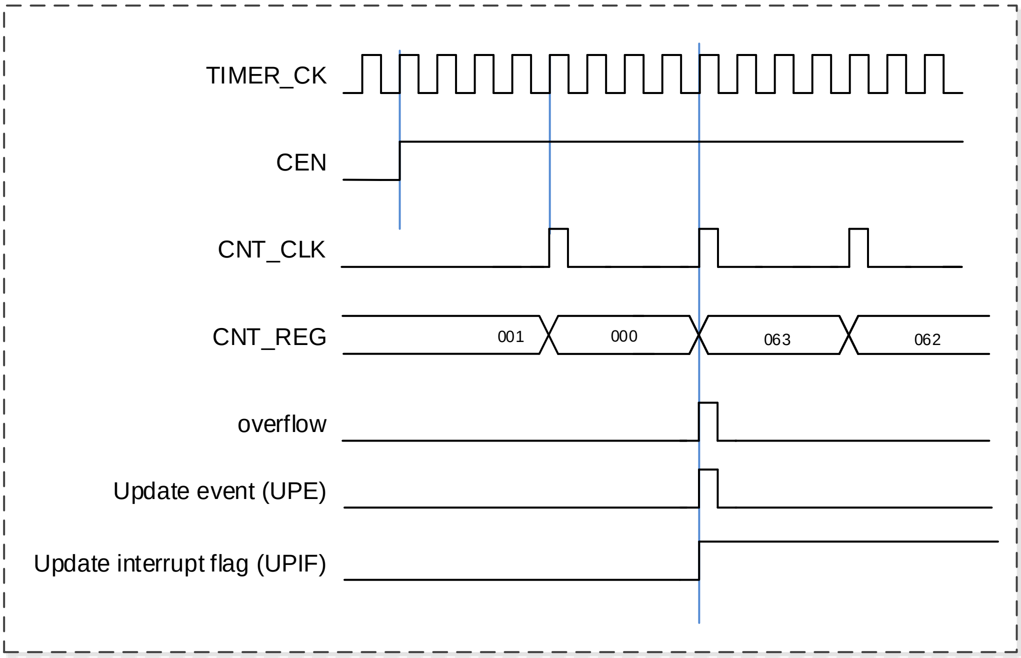


Figure 9-13. Counter timing diagram, internal clock divided by N

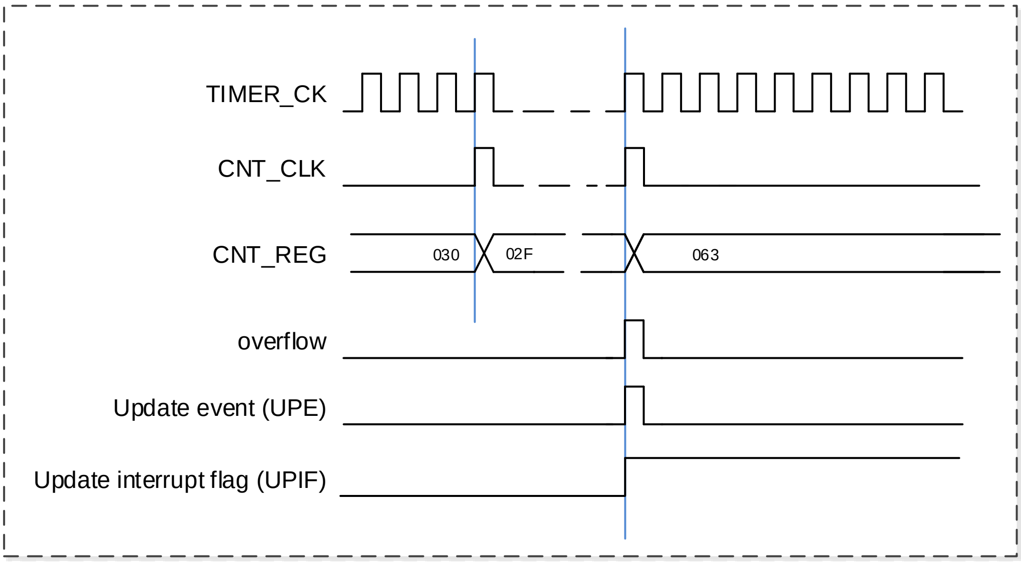
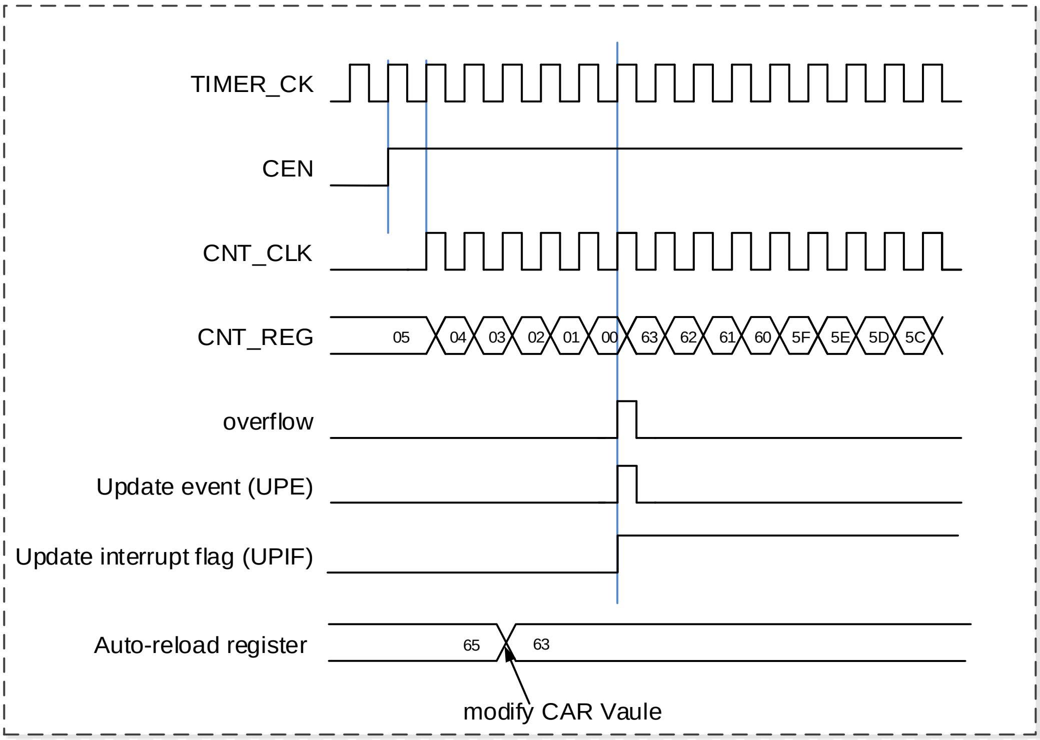


Figure 9-14. Counter timing diagram, update event when repetition counter is not used



## Center-aligned counting mode

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 TIMER0_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMER0_SWEVG register will initialize the counter value to 0 irrespective of whether the counter is counting up or down in the center-align counting mode and generates an update event.

The UPIF bit in the TIMERx_INTF register can be set to 1 when an underflow event at count-down (CAM in TIMER0_CTL0 is “01”), an overflow event at count-up (CAM in TIMER0_CTL0 is “10”) or both of them occur (CAM in TIMER0_CTL0 is “11”).

If set the UPDIS bit in the TIMER0_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x5.

**Figure 9-15. Counter timing diagram, internal clock divided by 1, TIMERx_CAR = 0x5**

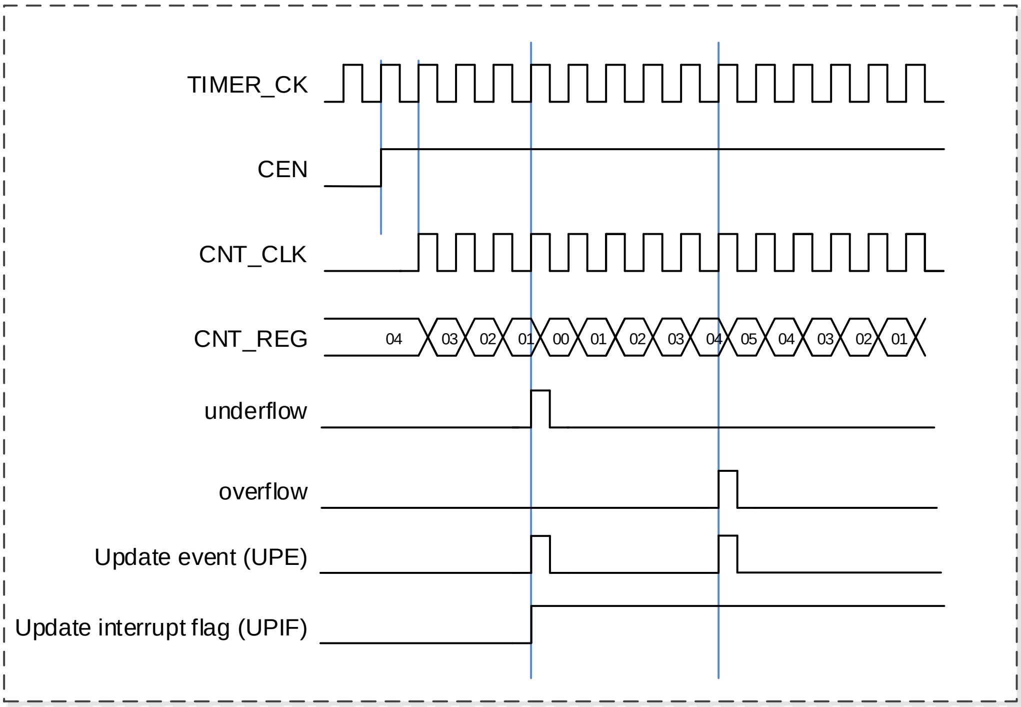


Figure 9-16. Counter timing diagram, internal clock divided by 2

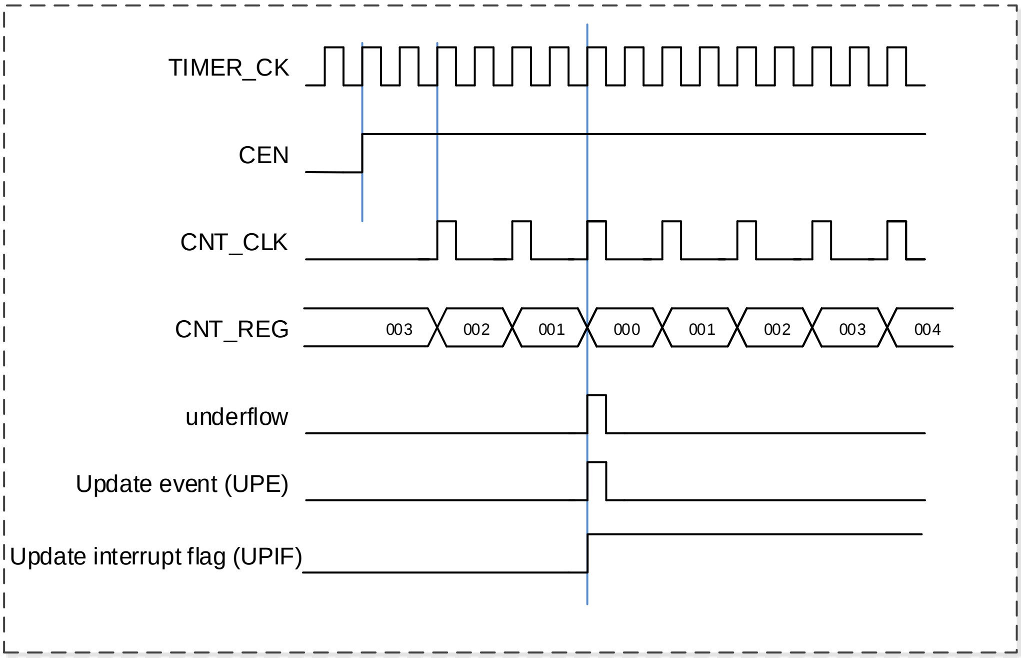


Figure 9-17. Counter timing diagram, internal clock divided by 4, TIMERx_CAR=0x63

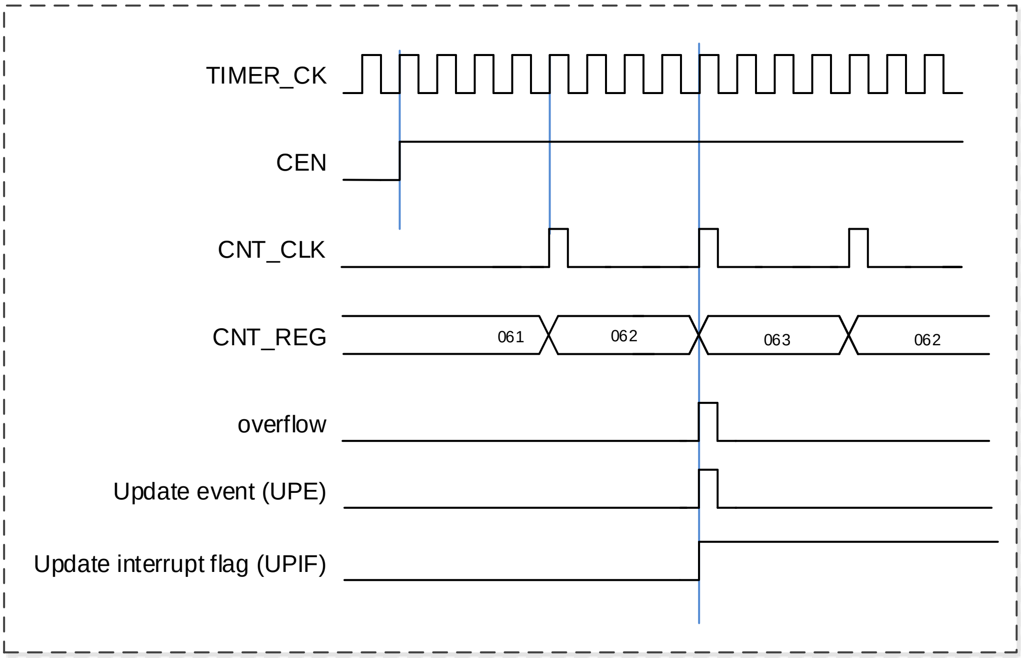


**Figure 9-18. Counter timing diagram, internal clock divided by N**

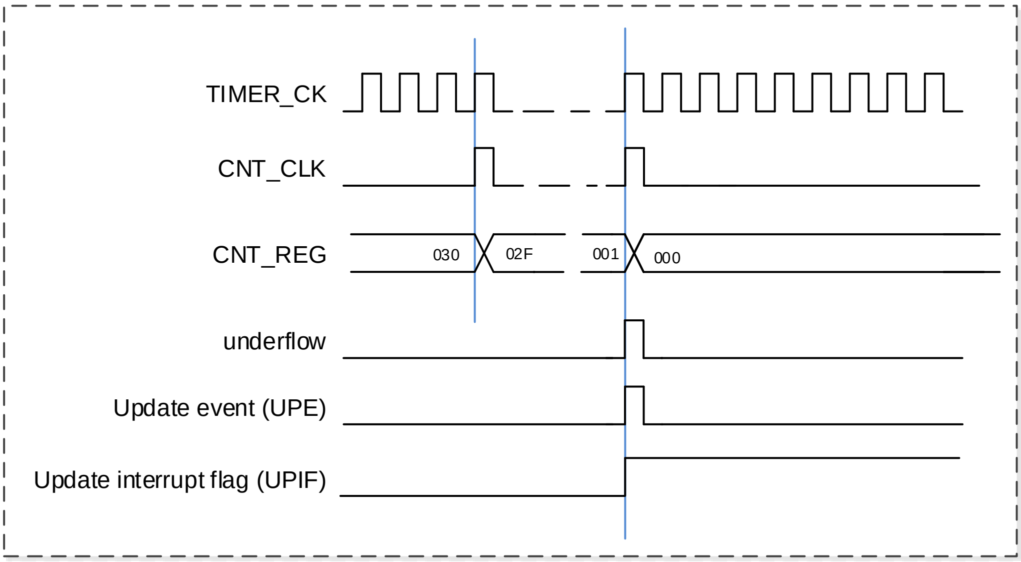


**Figure 9-19. Counter timing diagram, update event with ARSE=1(counter underflow)**

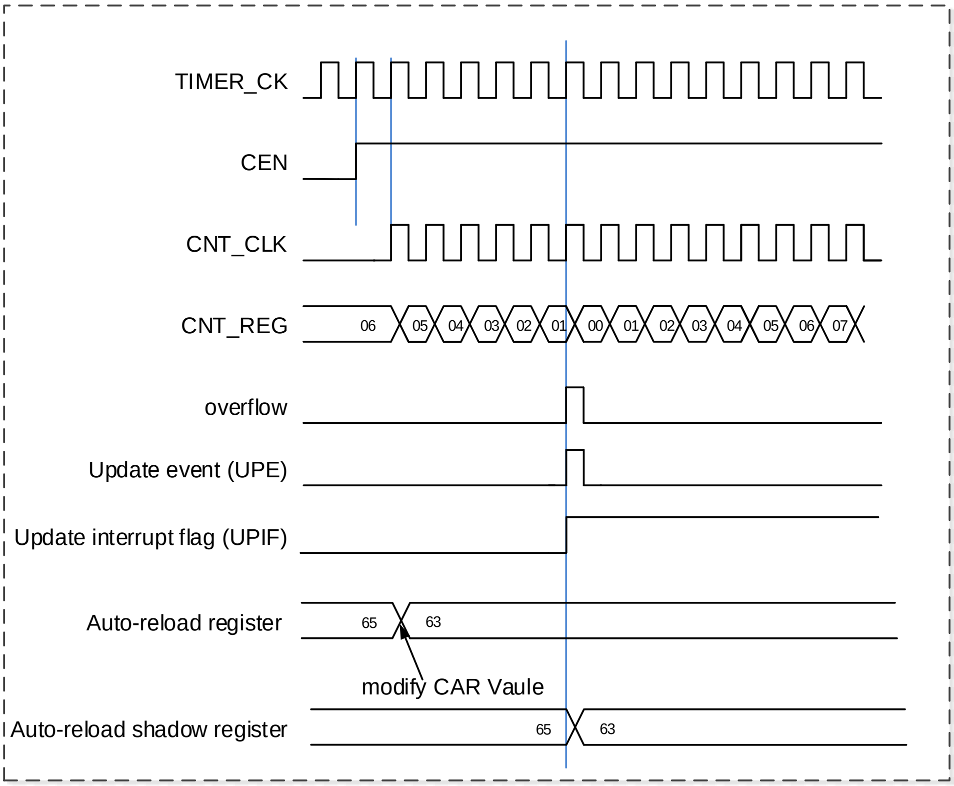
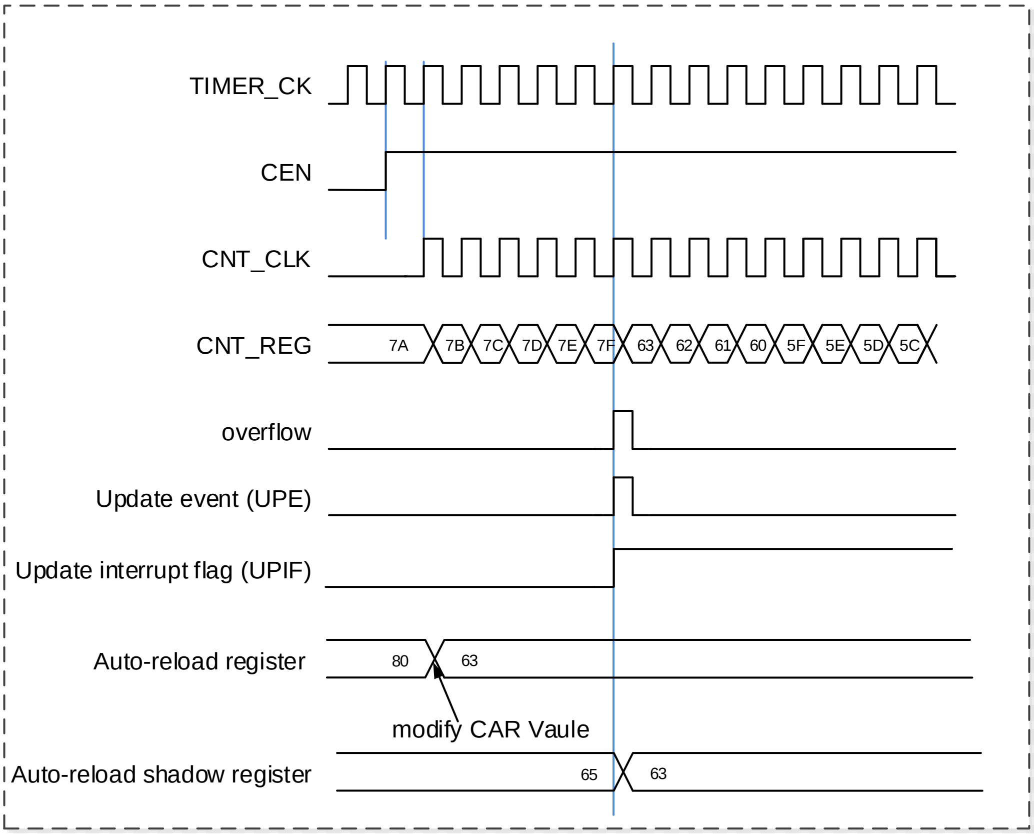


**Figure 9-20. Counter timing diagram, Update event with ARSE=1 (counter overflow)**



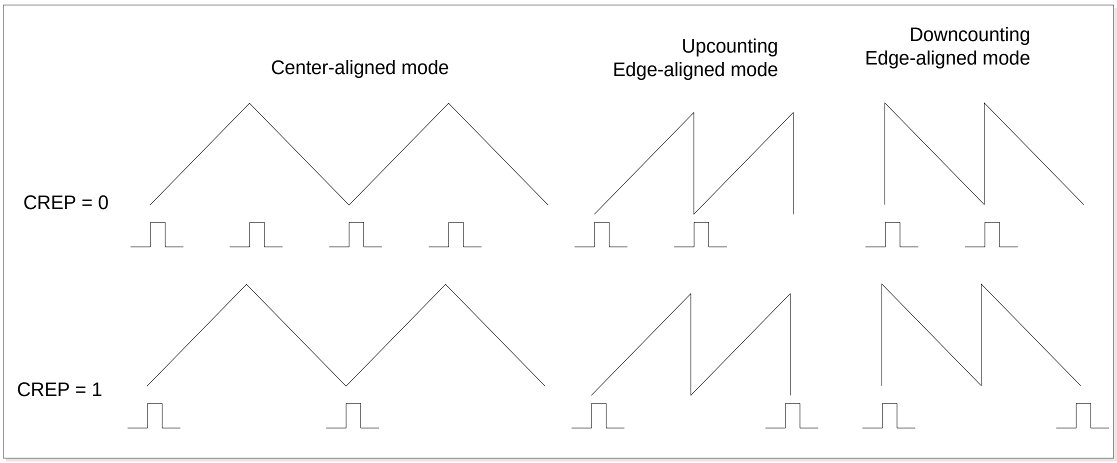
### Counter Repetition

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 TIMER0_CREP register. The repetition counter is decremented at each counter overflow in up-counting mode, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned mode.

Setting the UPG bit in the TIMER0_SWEVG register will reload the content of CREP in TIMER0_CREP register and generator an update event.

For odd values of CREP in center-aligned mode, the update event occurs either on the overflow or on the under depending on when the CREP register was written and when the counter was started. The update event generated at overflow when the CREP was written before starting the counter, and generated at underflow when the CREP was written after starting the counter.

Figure 9-21. Update rate examples depending on mode and TIMERx_CREP



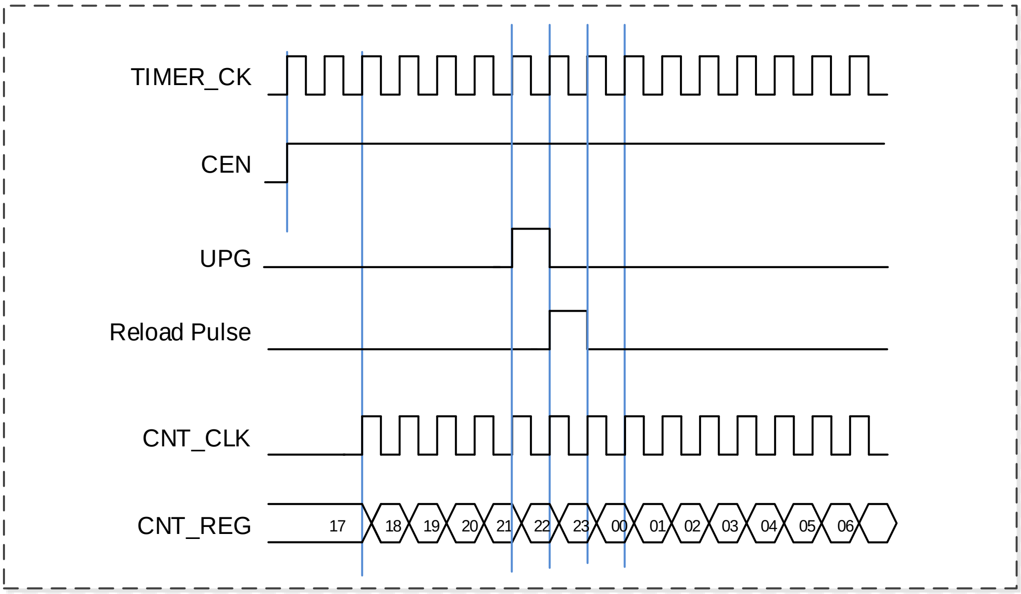
### Clock selection

The following describes the Timer Module clock controller which determines the clock source of the internal prescaler counter.

- Internal timer clock TIMER_CK

The default internal clock source is the APB2 clock CK_APB2 used to drive the counter prescaler when the slave mode is disabled. If the slave mode controller is enabled by setting SMC field in the TIMER0_SMCFG register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS field in the TIMER0_SMCFG register and described as follows. When the slave mode selection bits SMC are set to 0x4, 0x5 or 0x6, the internal clock TIMER_CK is the counter prescaler driving clock source.

Figure 9-22. Control circuit in normal mode, internal clock divided by 1



- Quadrature decoder

To select Quadrature Decoder mode the SMC field should be set to 0x1, 0x2 or 0x3 in the TIMER0_SMCFG register. The Quadrature Decoder function uses two input states of the TIMER0_CH0 and TIMER0_CH1 pins to generate the clock pulse to drive the counter prescaler. The counting direction bit DIR is modified by hardware automatically at each transition on the input source signal. The input source signal can be derived from the TIMER0_CH0 pin only, the TIMER0_CH1 pin only or both TIMER0_CH0 and TIMER0_CH1 pins.

- Internal trigger inputs (ITI)

The counter prescaler can count during each rising or falling edge of the ITI signal. This mode can be selected by setting the SMC field to 0x6 in the TIMER0_SMCFG register, here the counter will act as an event counter. The input event, known as ITI here, can be selected by setting the TRGS field. When the ITI signal is selected as the clock source, the internal edge detection circuitry will generate a clock pulse during each ITI signal rising or falling edge to drive the counter prescaler.

- Channel input pin (CI)

The counter prescaler can be driven to count during each rising or falling edge on the external pin TIMER0_CIx. This mode can be selected by setting SMC field to 0x7 and the TRGS field to 0x4, 0x5 or 0x6. Note that the CIx is derived from the TIMER0_CHx sampled by a digital filter.

- External trigger input (ETIF)

The counter prescaler can be driven to count during each rising or falling edge on the external pin TIMER0_ETI. This mode can be selected by setting the SMC1 bit in the TIMER0_SMCFG register to 1. The other way to select the ETIF signal as the clock source is to set the SMC field to 0x6 and the TRGS field to 0x7 respectively. Note that the ETIF signal is derived from the TIMER0_ETI pin sampled by a digital filter. When the clock source is selected to come from the ETIF signal, the Trigger Controller including the edge detection circuitry will generate a clock pulse during each ETIF signal rising edge to clock the counter prescaler.

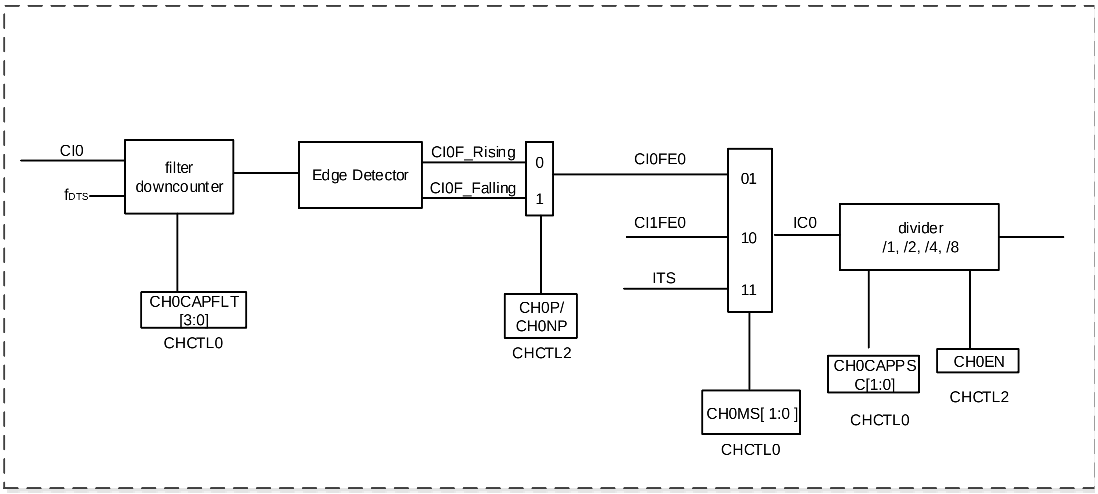
### Capture/compare channels

The TIMER0 has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture/compare value register including an input stage, channel controller and an output stage.

- Input capture stage

The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. The channel 0 input signal (CI0) can be chosen to come from the TIMER0_CH0 signal or the Excusive-OR function of the TIMER0_CH0, TIMER0_CH1 and TIMER0_CH2 signals. The channel input signal (CIx) is sampled by a digital filter to generate a filtered input signal CIxF. Then the channel polarity and the edge detection block can generate a CIxFEx signal for the input capture function. The effective input event number can be set by the channel input prescaler (CHxCAPPSC).

**Figure 9-23. Capture/compare channel (example: channel 0 input stage)**



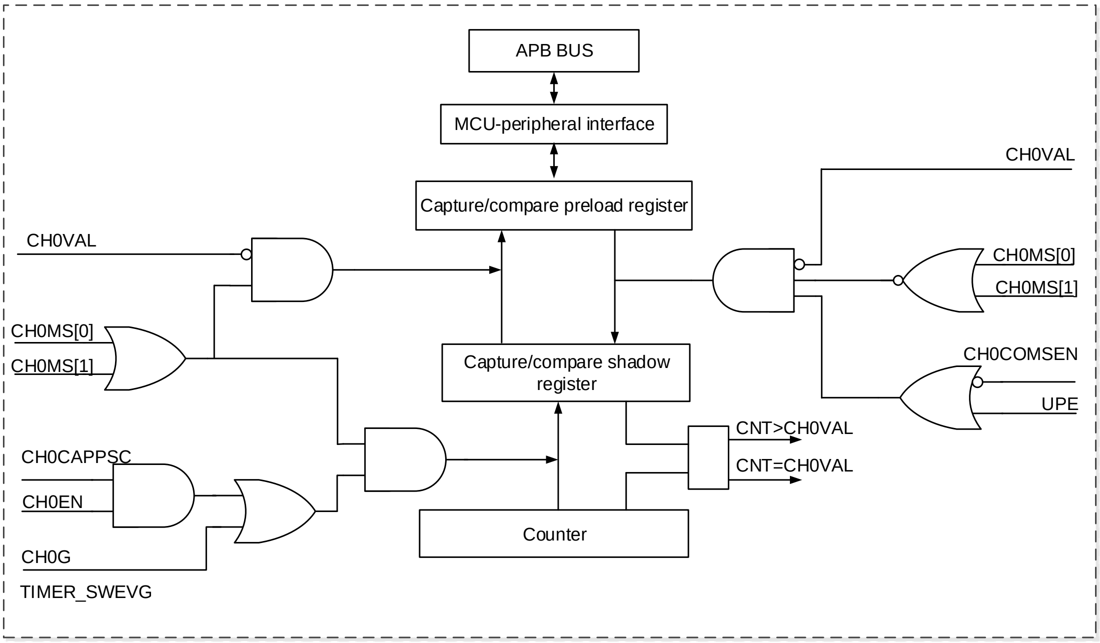
- Channel controller

The GPTM has four independent channels which can be used as capture inputs or compare match outputs.

When used in the input capture mode, the counter value is captured into the TIMER0_CHxCV shadow register first and then transferred into the TIMER0_CHxCV preload register when the capture event occurs.

When used in the compare match output mode, the contents of the TIMER0_CHxCV preload register is copied into the associated shadow register; the counter value is then compared with the register value.

**Figure 9-24. Capture/compare channel 0 main circuit**



- Output stage

The TIMER0 has four channels for compare match, single pulse or PWM output function.

**Figure 9-25. Output stage of capture/compare channel (channel 0 to 2)**

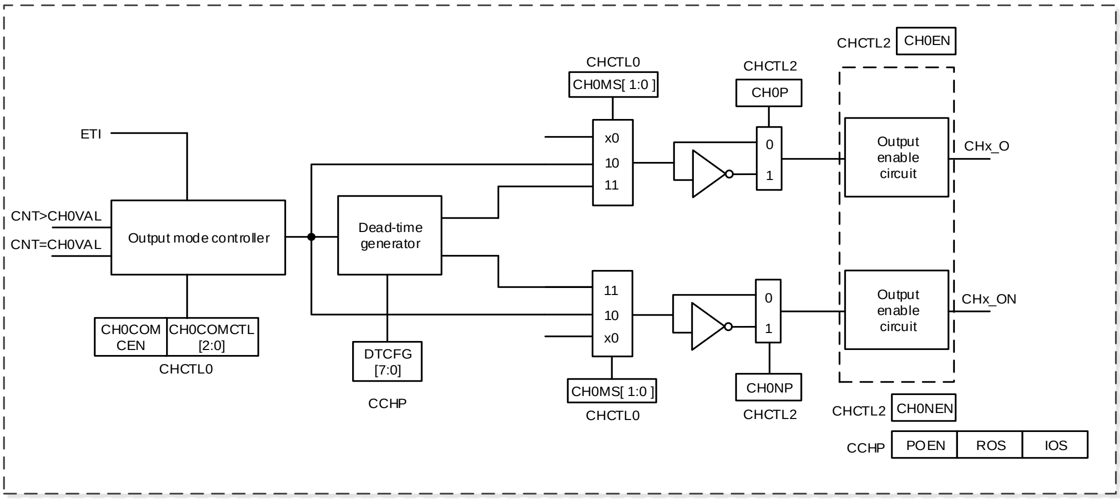
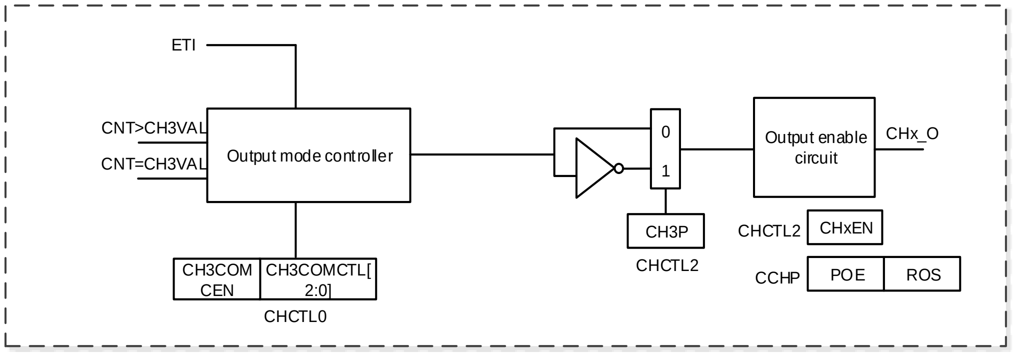


**Figure 9-26. Output stage of capture/compare channel (channel 3)**



### Input Capture Mode

When the channel is used as a capture input, the counter value is captured into the Channel Capture/Compare Register (TIMER0_CHxCV) when an effective input signal transition occurs. Once the capture event occurs, the CHxIF flag in the TIMER0_INTF register is set. If the CHxIF bit is already set, i.e., the flag has not yet been cleared by software, and another capture event on this channel occurs, the corresponding channel Over-Capture flag, named CHxOF, will be set.Once the capture event occurs, a DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit. The fllowing step maybe used to implement input capture mode:

- Select input signal by set CHxMS bits in the channel control register (TIMER0_CHCTLx).
- Add input filter to input signal by set CHxCAPFLT bitst in the the channel control register (TIMER0_CHCTLx).
- Select input capture edge (rising or falling) by set CHxP/CHxNP bit in the channel control register 2 (TIMER0_CHCTL2).
- If input signal need prescaler, set CHxCAPPSC bits in the the channel control register (TIMER0_CHCTLx).
- If need interrupt, set CHxIE bit in DMA and interrupt enable register (TIMER0_DMAINTEN) to 1.
- If need DMA request, set CHxDEN bit in DMA and interrupt enable register (TIMER0_DMAINTEN) to 1.
- Enable the channel by set CHxEN bit in the channel control register 2 (TIMER0_CHCTL2) to 1.

After configure this registers, the input capture event on this channel may occurs at the corresponding edge. And the counter value is captured into the Channel Capture/Compare Register (TIMER0_CHxCV). An interrupt generated if CHxIE bit set. A DMA request generated if CHxDEN set.

The input capture mode can be also used for pulse width measurement from signals on the TIMER0_CHx pins (CIx). For example, PWM signal connect to CI0 input. Select channel 0 capture signal to CI0 by setting CH0MS to 01 in the channel control register (TIMER0_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 10 in the channel control register (TIMER0_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMER0_CH0CV can measure the PWM period and the TIMER0_CH1CV can measure the PWM duty.

### Output Compare Mode/PWM Mode

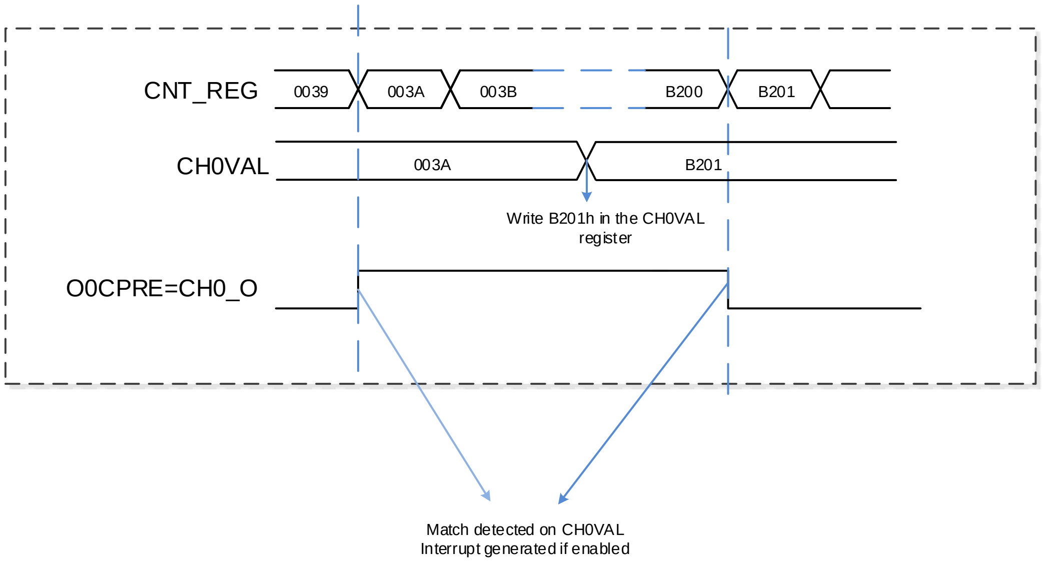
When the channel is used as an output mode by setting the CHxMS in the the channel control register (TIMER0_CHCTLx) to 00, the channel outputs waveform according to the CHxCOMCTL bits in the the the channel control register (TIMER0_CHCTLx), and the comparision between the counter and TIMER0_CHxCV registers. When the comparision equals, the CHxIF flag in the Interrupt flag registers (TIMER0_INTF) set to 1. A DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit.

In output mode, the channel can outputs active level by set the CHxCOMCTL to 101, and outputs inactive level by set the CHxCOMCTL bits to 100.

In ouput mode, the channel output set to active level when the comparision between the counter and TIMER0_CHxCV registers match, by set the CHxCOMCTL to 001. The channel output set to inactive level when the comparision between the counter and TIMER0_CHxCV registers match, by set the CHxCOMCTL to 010.

In output compare toggle mode (by set the CHxCOMCTL to 011), the channel output toggles when the comparision between the counter and TIMER0_CHxCV registers match.

**Figure 9-27. Output compare toggle mode, toggle on CH0_O**



In the output PWM mode (by setting the CHxCOMCTL bits to 110 (PWM mode 0) or to 111 (PWM mode 1)), the channel can outputs PWM waveform according to the TIMER0_CAR registers and TIMER0_CHxCV registers.

Figure 9-28. Output compare PWM mode 0 on CH0_O, upcounting mode

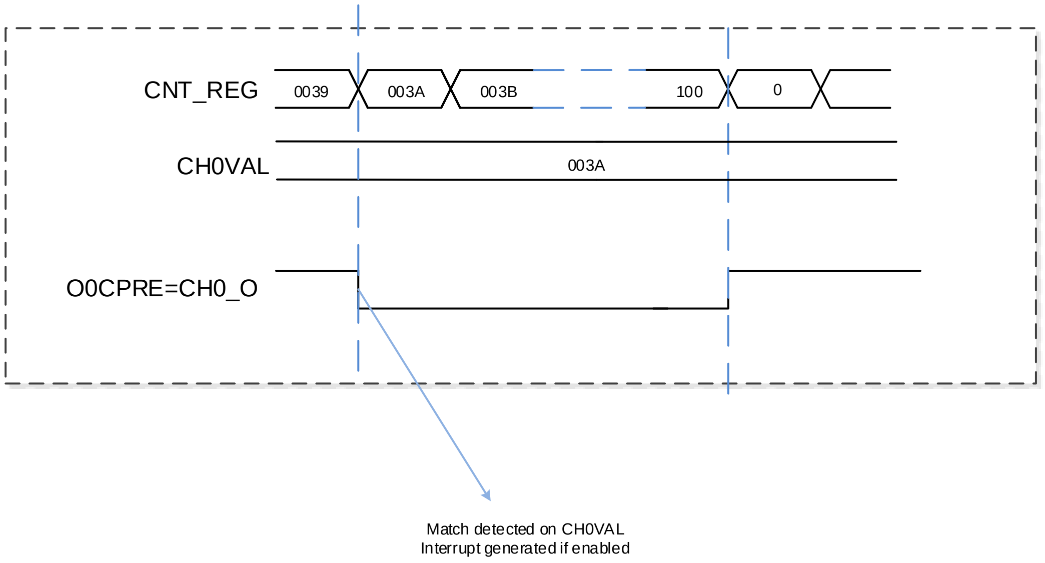
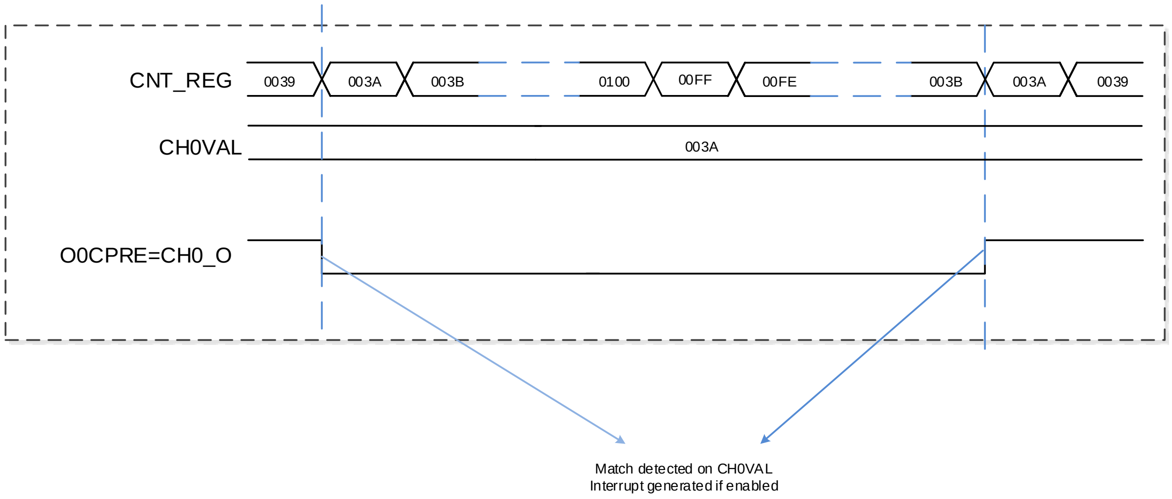


Figure 9-29. Output compare PWM mode 0 on CH0_O, center-aligned counting mode



### Channel Output Reference Signal

When the TIMER0 is used in the compare match output mode, the OxCPRE signal (Channel x Output Reference signal) is defined by setting the CHxCOMCTL bits. 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 TIMER0_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 TIMER0_CHxCV content. With regard to a more detailed 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 TIMER0_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIF signal is derived from the external TIMER0_ ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMER0_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs Complementary and Dead-time

The TIMER0 can output two complementary signals. The complementary signals CHx_O and CHx_ON are activated by a combination of several control bits: the CHxEN and CHxNEN bits in the TIMER0_CHCTL2 register and the POEN, ISOx, ISOxN, IOS and ROS bits in the TIMER0_CCHP and TIMER0_CTL1 registers.The outputs polarity are determined by CHxP and CHxNP bits in the TIMER0_CHCTL2 register.

If CHxEN, CHxNEN and POEN bits are 1, dead-time should insertion. The rising edge of CHx_O output is delayed relative to the rising edge of OxCPRE and the rising edge of CHx_ON output is delayed relative to the falling edge of OxCPRE.The delay value is a 8-bit dead-time counter determined by DTCFG field in TIMER0_CCHP register.If the delay value is greater than the width of the active output (CHx_O or CHx_ON) then the corresponding pulse is not generated.

**Figure 9-30. Complementary output with dead-time insertion.**

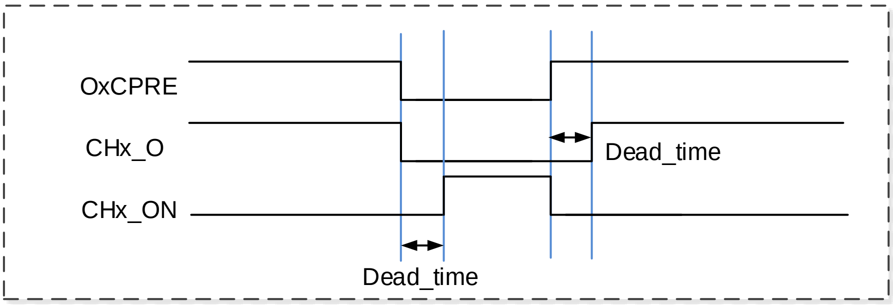


**Figure 9-31. Dead-time waveforms with delay greater than the negative pulse**

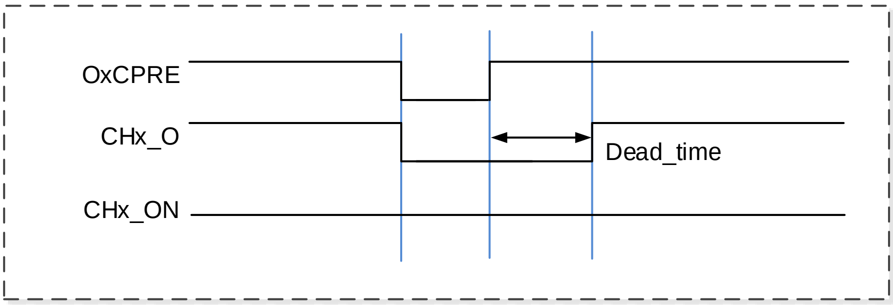
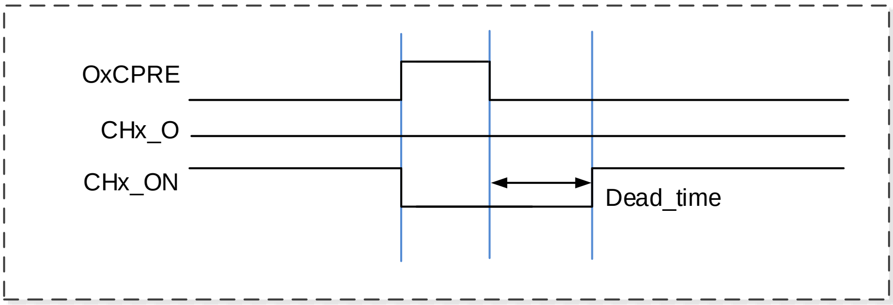


**Figure 9-32. Dead-time waveforms with delay greater than the positive pulse**



### Break function

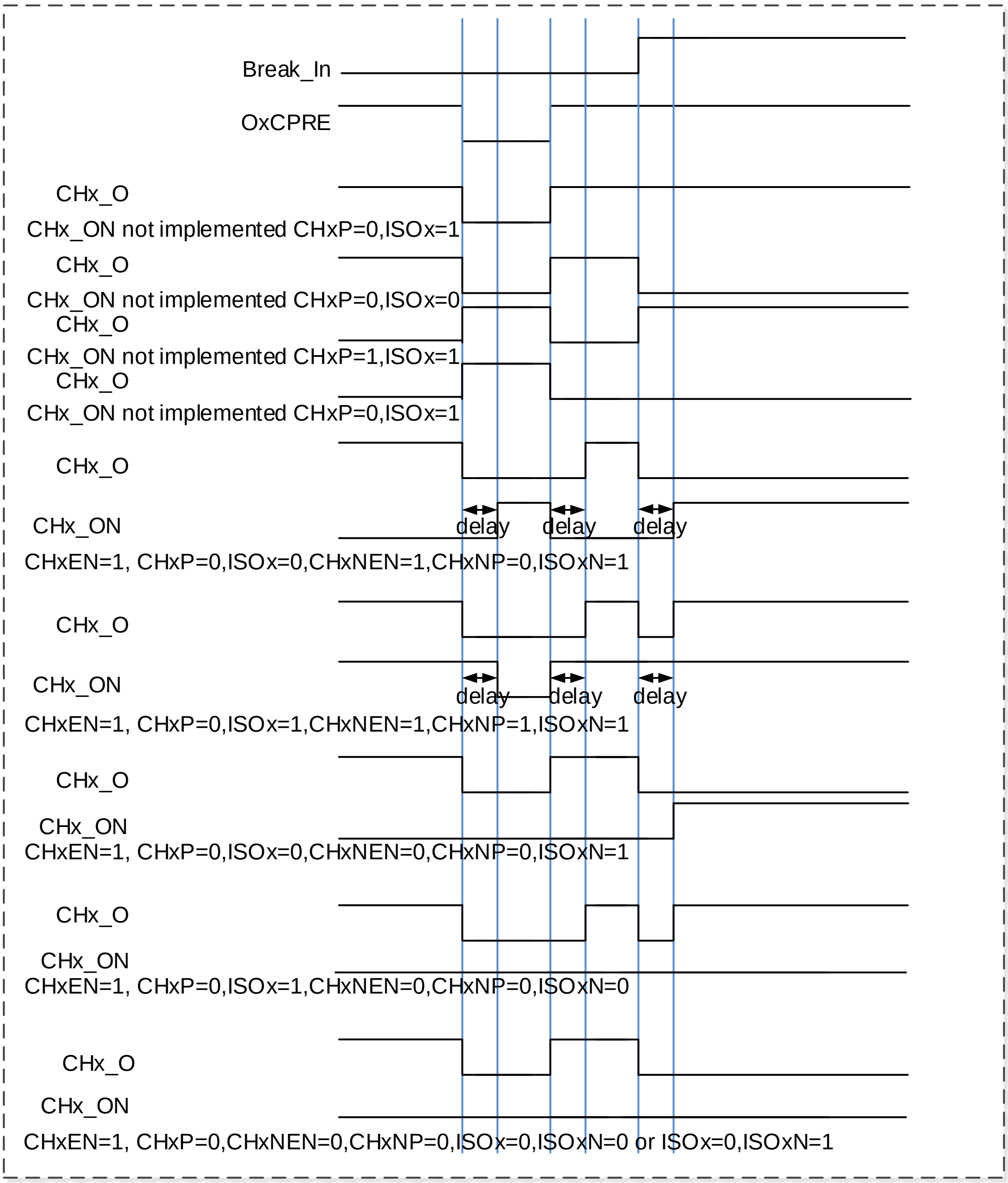
In this function, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMER0_CCHP register, ISOx and ISOxN bits in the TIMER0_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin or HXTAL stack event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMER0_CCHP register. The break input polarity is setting by the BRKP bit in TIMER0_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 in the TIMER0_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 TIMER0_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 9-33. Output behavior in response to a break（The break input is acting on high**

**level）**

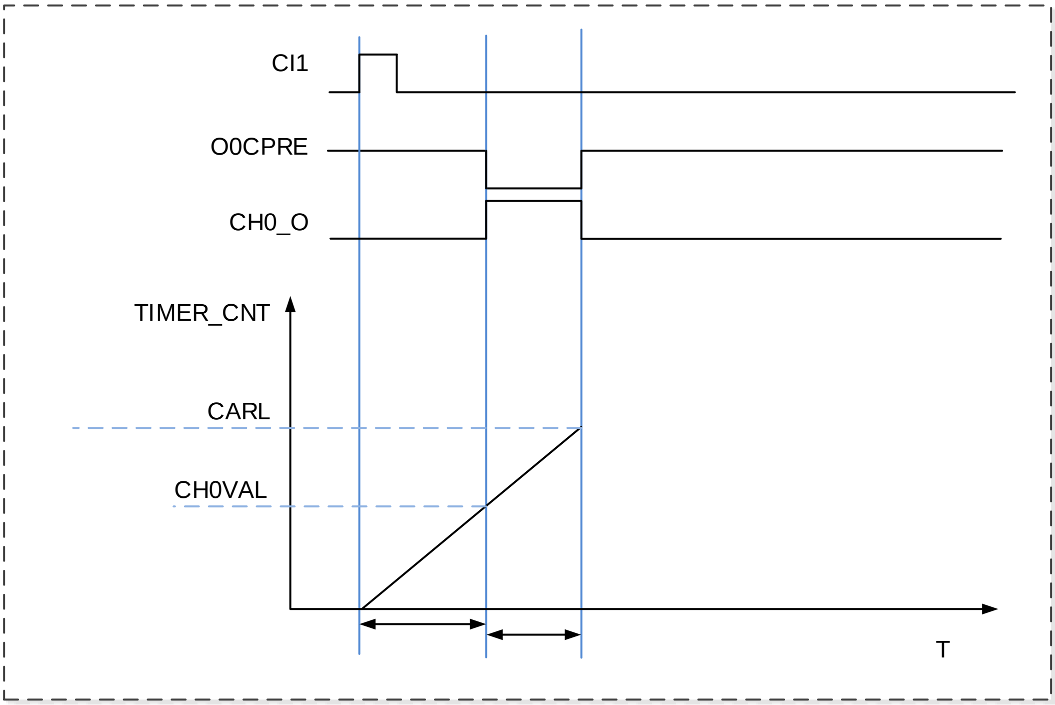

Break_In
OxCPRE
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=0
CHx_O
CHx_ON not implemented CHxP=1,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=1,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=1,CHxNP=1,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=0,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=0,CHxNP=0,ISOxN=0
CHx_O
CHx_ON
CHxEN=1, CHxP=0,CHxNEN=0,CHxNP=0,ISOx=0,ISOxN=0 or ISOx=0,ISOxN=1

### Single Pulse Mode

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 TIMER0_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. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

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 TIMER0_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxOFE bit in each TIMER0_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 CHxOFE 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 9-34. Single pulse mode**



### Quadrature Decoder

The Quadrature Decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMER0_CH0 and TIMER0_CH1 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either CI0 only, CI1 only or both CI0 and CI1, the selection made by setting the SMC field to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. 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-reload value. Therefore, users must configure the TIMER0_CAR register before the counter starts to count.

**Table 9-1. Counting direction versus encoder signals**

| Counting mode | Level | CI0FE0 | | CI1FE1 | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |

| | | | | | |
|---|---|---|---|---|---|
| CI0 only counting | CI1FE=High | Down | Up | - | - |
| | CI1FE=Low | Up | Down | - | - |
| CI1 only counting | CI0FE=High | - | - | Up | Down |
| | CI0FE=Low | - | - | Down | Up |
| CI0 and CI1 counting | CI1FE=High | Down | Up | X | X |
| | CI1FE=Low | Up | Down | X | X |
| | CI0FE=High | X | X | Up | Down |
| | CI0FE=Low | X | X | Down | Up |

**Note:** "-" means "no counting"; "X" means impossible.

**Figure 9-35. Example of counter operation in encoder interface mode**

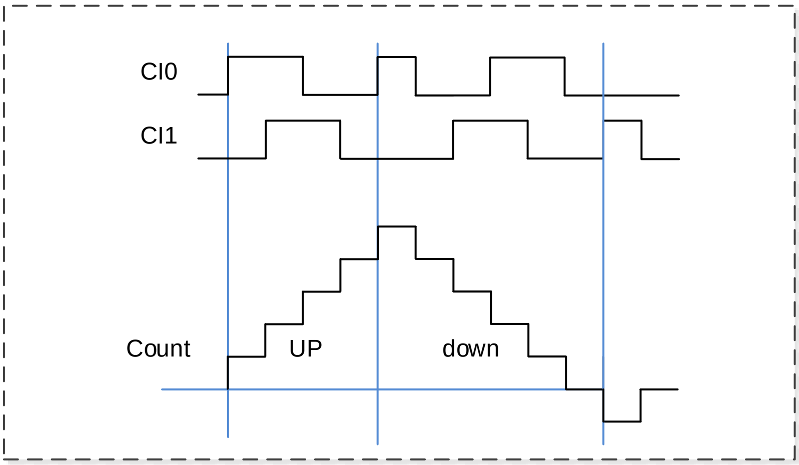
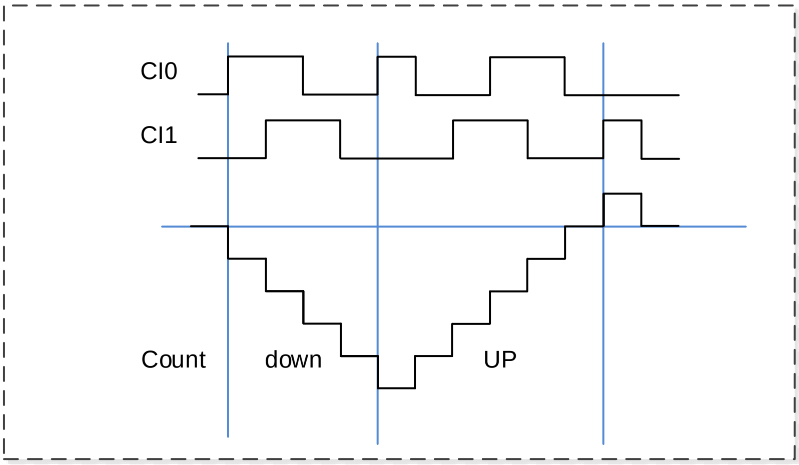


**Figure 9-36. Example of encoder interface mode with CI0FE0 polarity inverted**



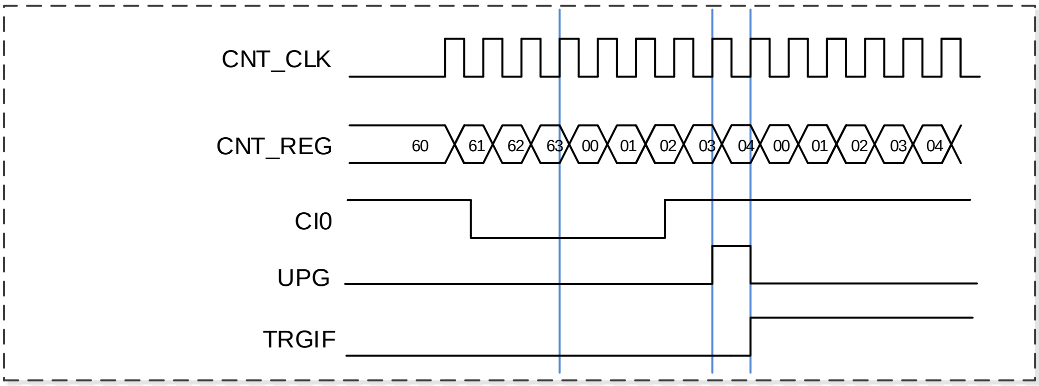
## Slave Controller

The TIMER0 can be synchronized with an external trigger in several modes including the Restart mode, the Pause mode and the event mode which is selected by the SMC field in the TIMER0_SMCFG register. The trigger input of these modes can be selected by the TRGS field in the TIMERx_SMCFG register, below to CI0 signal as an example. The operation modes in the Slave Controller are described in the accompanying sections.

- Restart mode

The counter and its prescaler can be reinitialized in response to a rising edge of the CI0 signal. When a CI0 rising edge occurs, the update event software generation bit named UPG will automatically be asserted by hardware and the trigger event flag will also be set. Then the counter and prescaler will be reinitialized. Although the UPG bit is set to 1 by hardware, the update event does not really occur. It depends upon whether the update event disable control bit UPDIS is set to 1 or not. If UPDIS is set to 1 to disable the update event to occur, there will no update event will be generated, however the counter and prescaler are still reinitialized when the CI0 rising edge occurs. If the UPDIS bit in the TIMER0_CTL0 register is cleared to enable the update event to occur, an update event will be generated together with the CI0 rising edge, then all the preloaded registers will be updated.

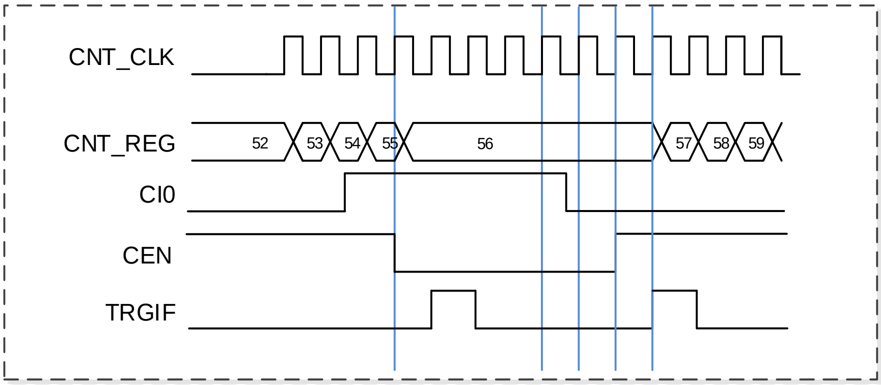
**Figure 9-37. Control circuit in restart mode**



- Pause mode

In the Pause Mode, the selected CI0 input signal level is used to control the counter start/stop operation. The counter starts to count when the selected CI0 signal is at a high level and stops counting when the CI0 signal is changed to a low level, here the counter will maintain its present value and will not be reset.

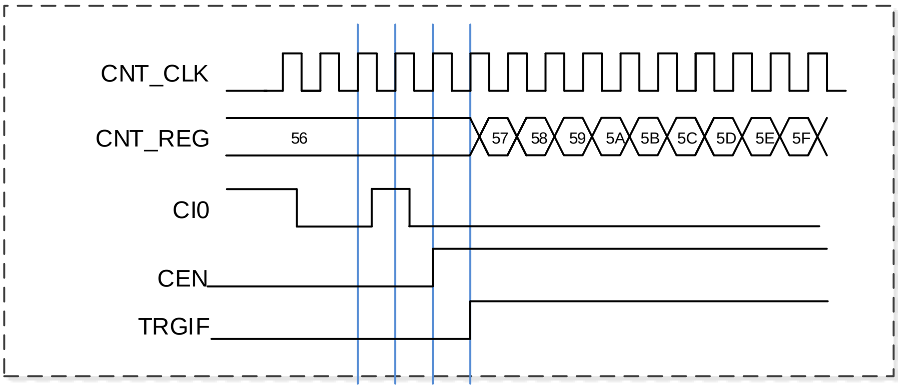
**Figure 9-38. Control circuit in pause mode**



- Event mode

After the counter is disabled to count, the counter can resume counting when a CI0 rising edge signal occurs. When a CI0 rising edge occurs, the counter will start to count from the current value in the counter. Note that the CI0 signal is only used to enable the counter to resume counting and has no effect on controlling the counter to stop counting.

**Figure 9-39. Control circuit in event mode**

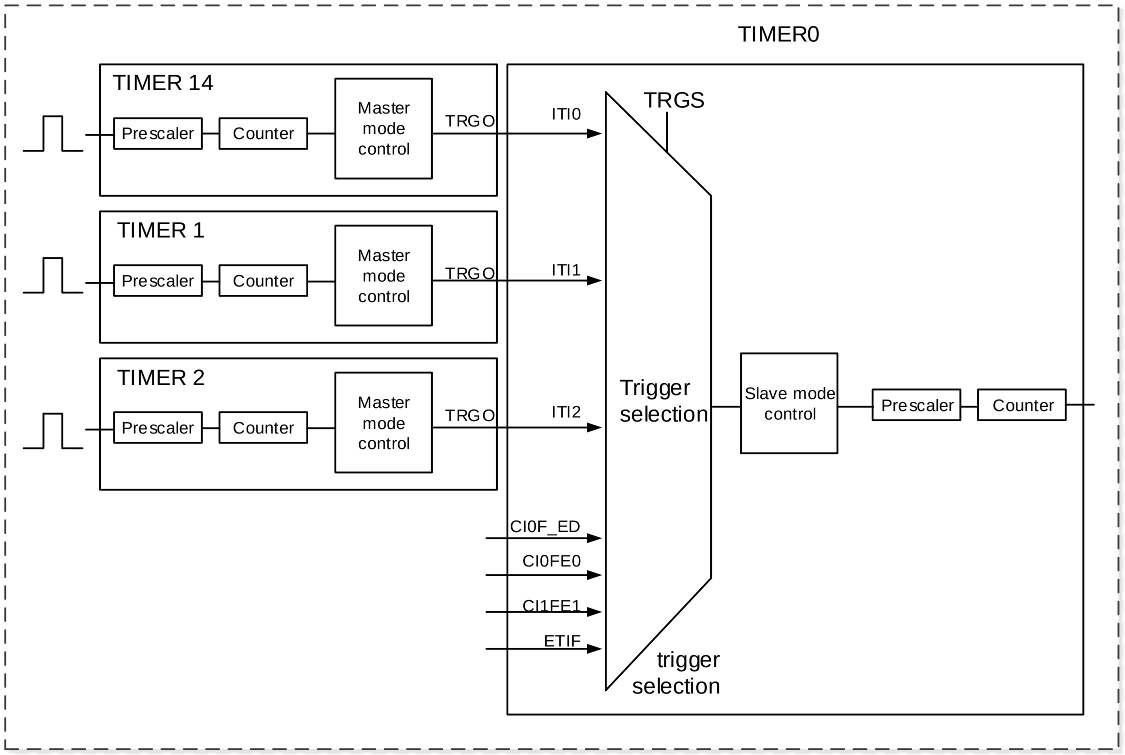


### Timer Interconnection

The timers can be internally connected together for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the Master mode while configuring another timer to be in the Slave mode. The following figures present several examples of trigger selection for the master and slave modes.

Figure below shows the TIMER0 trigger selection when it is configured in slave mode.

**Figure 9-40. TIMER0 Master/Slave mode timer example**



Other interconnection examples:

- TIMER1 as prescaler for TIMER0

We configure TIMER1 as a prescaler for TIMER0. Refer to Figure below for connections. Do as follow:

1. Configure TIMER1 in master mode and select its Update Event (UPE) as trigger output (MMC=010 in the TIMER1_CTL1 register). Then TIMER1 drives a periodic signal on each counter overflow.
2. Configure the TIMER1 period (TIMER1_CAR registers).
3. Select the TIMER0 input trigger source from TIMER1 (TRGS=001 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 TIMER1 by writing ‘1 in the CEN bit (TIMER1_CTL0 register).

- Start TIMER0 with TIMER1’s Enable/Update signal

First, we enable TIMER0 with the enable out of TIMER1. Refer to Figure below. TIMER0 starts counting from its current value on the divided internal clock after trigger by TIMER1 enable output.

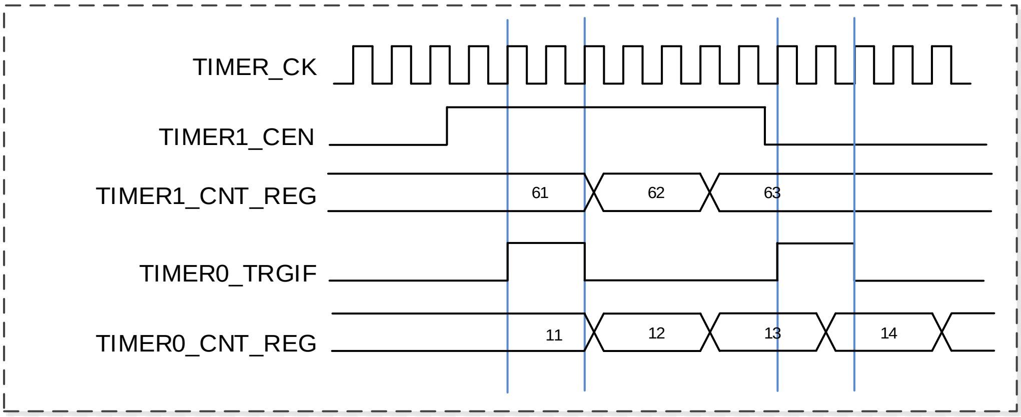
When TIMER0 receives the trigger signal its CEN bit is automatically set and the counter counts until we disable TIMER0. Both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT\_CLK} = f_{TIMER\_CK}$ /3). Do as follow:

1. Configure TIMER1 master mode to send its enable signal as trigger output(MMC=001 in

the TIMER1_CTL1 register)

2. Configure TIMER0 to select the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
3. Configure TIMER0 in event mode (SMC=110 in TIMER0_SMCFG register).
4. Start TIMER1 by writing 1 in the CEN bit (TIMER1_CTL0 register).

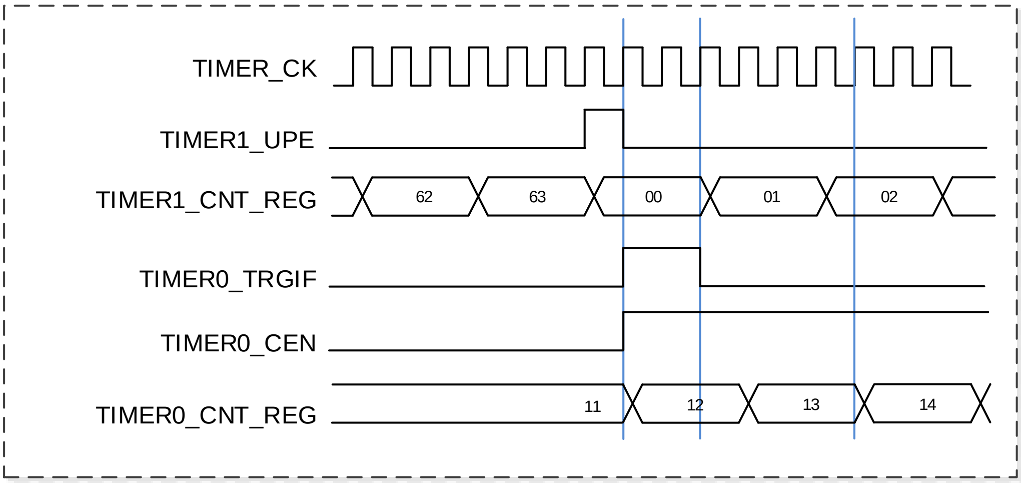
**Figure 9-41. Triggering TIMER0 with Enable of TIMER1**



In this example, we also can use update Event as trigger source instead of enable signal. Refer to figure below. Do as follow:

1. Configure TIMER1 in master mode and send its Update Event (UPE) as trigger output (MMC=010 in the TIMER1_CTL1 register).
2. Configure the TIMER1 period (TIMER1_CAR registers).
3. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
4. Configure TIMER0 in event mode (SMC=110 in TIMER0_SMCFG register).
5. Start TIMER1 by writing ‘1 in the CEN bit (TIMER1_CTL0 register).

**Figure 9-42. Triggering TIMER0 with update of TIMER1**



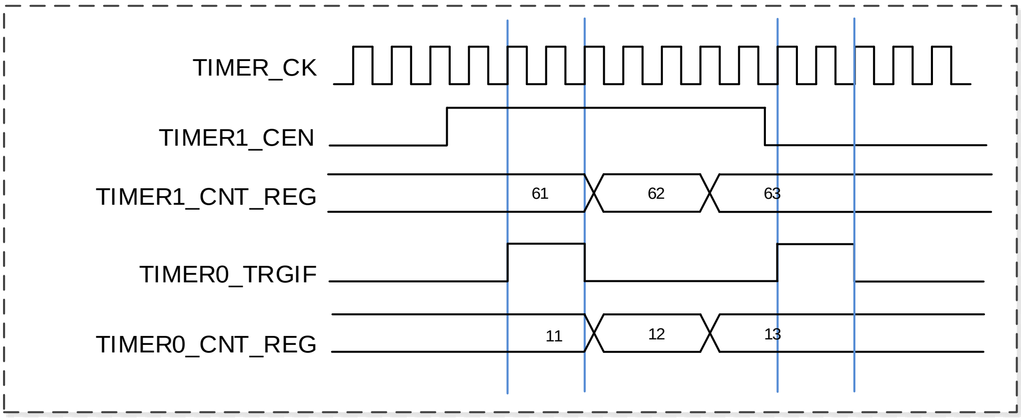
- Enable TIMER0 count with TIMER1’s enable/O0CPRE signal

In this example, we control the enable of TIMER0 with the enable output of TIMER1 .Refer to figure below. TIMER0 counts on the divided internal clock only when TIMER1 is enable. Both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT\_CLK}$

= f$_{TIMER\_CK}$ /3). Do as follow:

1. Configure TIMER1 input master mode and Output enable signal as trigger output (MMC=001 in the TIMER1_CTL1 register).
2. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
3. Configure TIMER0 in pause mode (SMC=101 in TIMER0_SMCFG register).
4. Enable TIMER0 by writing '1 in the CEN bit (TIMER0_CTL0 register)
5. Start TIMER1 by writing '1 in the CEN bit (TIMER1_CTL0 register).
6. Stop TIMER1 by writing '0 in the CEN bit (TIMER1_CTL0 register).

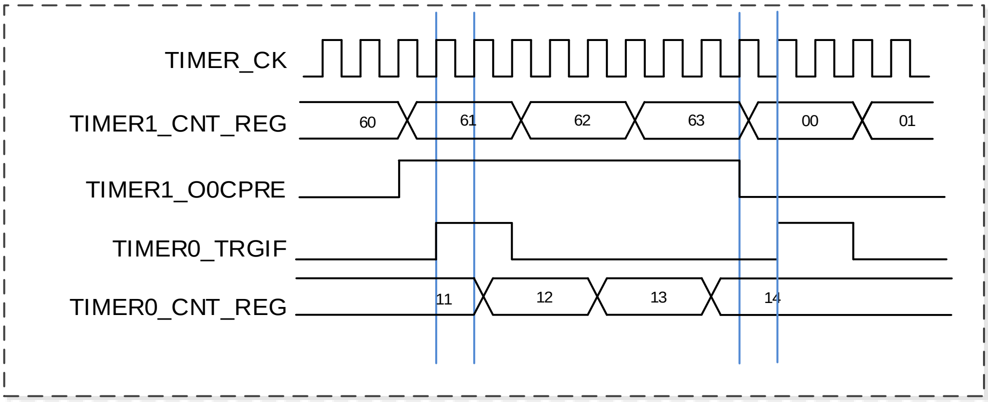
**Figure 9-43. Gating TIMER0 with enable of TIMER1**



In this example, we also can use OxCPRE as trigger source instead of enable signal output. Do as follow:

1. Configure TIMER1 in master mode and Output Compare 1 Reference (O0CPRE) signal as trigger output (MMS=100 in the TIMER1_CTL1 register).
2. Configure the TIMER1 O0CPRE waveform (TIMER1_ CHCTL0 register).
3. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
4. Configure TIMER0 in pause mode (SMC=101 in TIMER0_SMCFG register).
5. Enable TIMER0 by writing '1 in the CEN bit (TIMER0_CTL0 register).
6. Start TIMER1 by writing '1 in the CEN bit (TIMER1_CTL0 register).

**Figure 9-44. Gating TIMER0 with O0CPREof TIMER1**



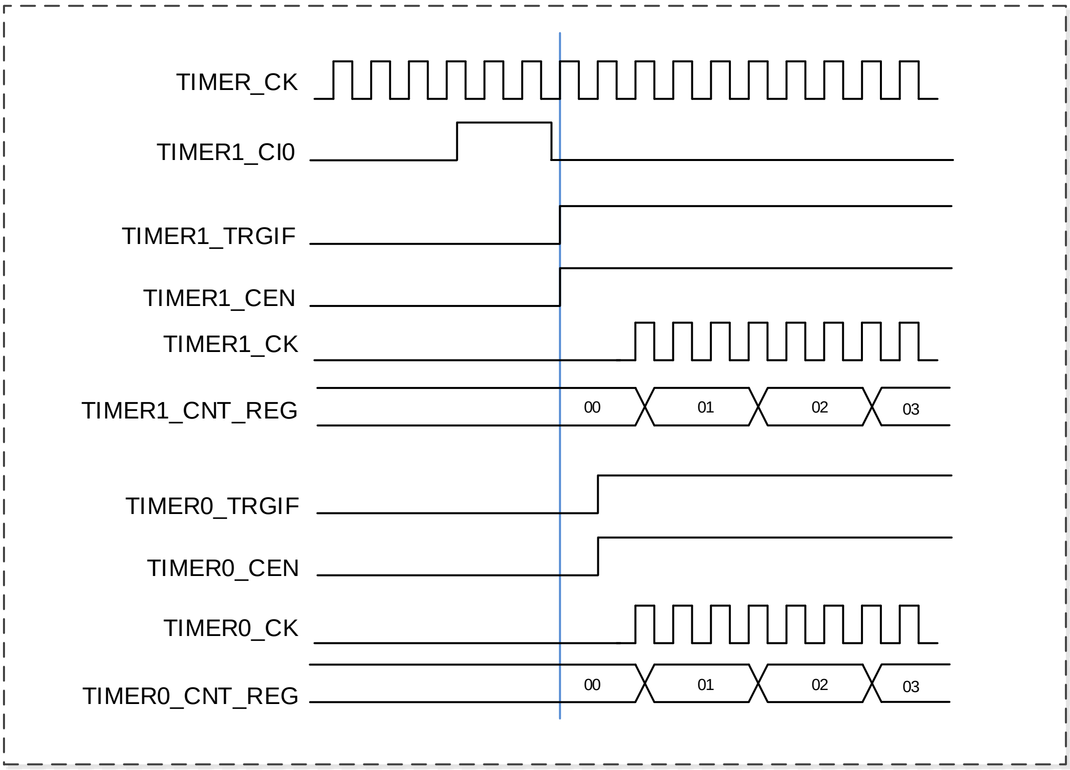
- Using an external trigger to start 2 timers synchronously

We configure the start of TIMER0 is triggered by the enable of TIMER1, and TIMER1 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, TIMER1 must be configured in Master/Slave mode. Do as follow:

1. Configure TIMER1 slave mode to get the input trigger from CI0 (TRGS=100 in the TIMER1_SMCFG register).
2. Configure TIMER1 in event mode (SMC=110 in the TIMER1_SMCFG register).
3. Configure the TIMER1 in Master/Slave mode by writing MSM=1 (TIMER1_SMCFG register).
4. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
5. Configure TIMER0 in event mode (SMC=110 in the TIMER1_SMCFG register).

When a rising edge occurs on TIMER1's CI0, two timer counters starts counting synchronously on the internal clock and both TRGIF flags are set.

**Figure 9-45. Triggering TIMER0 and TIMER1 with TIMER1's CI0 input**



### Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.1.4. TIMER0 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:10 | 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 ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.<br>00: $f_{DTS}= f_{TIMER\_CK}$<br>01: $f_{DTS}= f_{TIMER\_CK}$ /2<br>10: $f_{DTS}= f_{TIMER\_CK}$ /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 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 up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting down.<br>10: Center-aligned and counting up assert mode. The counter counts up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting up.<br>11: Center-aligned and counting up/down assert mode. The counter counts up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting both up and down.<br>After the counter is enabled, cannot be switched from 0x00 to non 0x00. |

4 DIR

Direction

0: Count up

1: Count down

This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

3 SPM

Single pulse mode.

0: Counter continues after update event.

1: The CEN is cleared by hardware and the counter stops at next update event.

2 UPS

Update source

This bit is used to select the update event sources by software.

0: Any of the following events generate an update interrupt or DMA request:

- The UPG bit is set
- The counter generates an overflow or underflow event
- The slave mode controller generates an update event.

1: Only counter overflow/underflow generates an update interrupt or DMA request.

1 UPDIS

Update disable.

This bit is used to enable or disable the update event generation.

0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:

- The UPG bit is set
- The counter generates an overflow or underflow event
- The slave mode controller generates an update event.

1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event.

0 CEN

Counter enable

0: Counter disable

1: Counter enable

The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | ISO3 | ISO2N | ISO2 | ISO1N | ISO1 | ISO0N | ISO0 | TI0S | MMC[2:0] | | | DMAS | CCUC | Res. | CCSE |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 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: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset.<br>001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.<br>010: Update. In this mode the master mode controller selects the update event as TRGO. |

| | | |
|---|---|---|
| | | 011: Capture/compare pulse.In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred.<br>100: Compare.In this mode the master mode controller selects the O0CPRE signal is used as TRGO<br>101: Compare.In this mode the master mode controller selects the O1CPRE signal is used as TRGO<br>110: Compare.In this mode the master mode controller selects the O2CPRE signal is used as TRGO<br>111: Compare.In this mode the master mode controller selects the O3CPRE signal is used as TRGO |
| 3 | DMAS | DMA request source selection<br>0: DMA request of channel x is sent when capture/compare event occurs.<br>1: DMA request of channel x is sent when update event occurs. |
| 2 | CCUC | Commutation control shadow register update control<br>When the commutation control shadow registers (for CHxEN, CHxNEN and CHxCOMCTL bits) are enabled (CCSE=1), this bit control when these shadow registers update.<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 register 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. 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

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

| 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 |
|---|---|---|
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal |

| | | |
|---|---|---|
| | | 0: ETI is active at high level or rising edge.<br>1: ETI is active at low level or falling edge. |
| 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>Setting the SMC1 bit has the same effect as selecting external clock mode 0 with TRGI connected to ETIF (SMC=111 and TRGS =111).<br>It is possible to simultaneously use external clock mode 1 with the reset mode, pause mode or event mode. But the TRGS bits must not be 111 in this case.<br>The external clock input will be ETIF if external clock mode 1 and external clock mode 1 are enabled at the same time. |
| 13:12 | ETPSC[1:0] | External trigger prescaler<br>The frequency of external trigger signal ETI must not be at higher than 1/4 of TIMERx_CK frequency. When the external trigger signal is a fast clocks, the prescaler can be enabled to reduce ETI frequency.<br>00: Prescaler disable<br>01: ETI frequency will be divided by 2<br>10: ETI frequency will be divided by 4<br>11: ETI frequency will be divided by 8 |
| 11:8 | ETFC[3:0] | External trigger filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETI signal and the length of the digital filter applied to ETI.<br>0000: Filter disalble. $f_{SAMP}$= $f_{DTS}$, N=1.<br>0001: $f_{SAMP}$= $f_{TIMER\_CK}$, N=2.<br>0010: $f_{SAMP}$= $f_{TIMER\_CK}$, N=4.<br>0011: $f_{SAMP}$= $f_{TIMER\_CK}$, N=8.<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6.<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8.<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6.<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8.<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6.<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8.<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5.<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6.<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8.<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5.<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6.<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8. |
| 7 | MSM | Master-slave mode |

| | | |
|---|---|---|
| | | The effect of an event on the trigger input is delayed in this mode to allow a perfect synchronization between the current timer and its slaves through TRGO. If we want to synchronize several timers on a single external event, this mode can be used.<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: Internal trigger input 0 (ITI0) TIMER14<br>001: Internal trigger input 1 (ITI1) TIMER1<br>010: Internal trigger input 2 (ITI2) TIMER2<br>011: Internal trigger input 3 (ITI3) Res.<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: External trigger input filter output (ETIF)<br>These bits must not be changed when slave mode is enabled. |
| 3 | OCRC | OCPRE clear source selection<br>0: OCPRE_CLR_INT is connected to the OCPRE_CLR input<br>1: OCPRE_CLR_INT is connected to ETIF |
| 2:0 | SMC[2:0] | Slave mode control<br>000: Disable mode.The prescaler is clocked directly by the internal clock 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 the shadow registers are updated 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 when it is low.<br>110: Event Mode. A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller.<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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | 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 |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Trigger DMA request disabled<br>1: Trigger DMA request enabled |
| 13 | CMTDEN | CMT DMA request enable<br>0: CMT DMA request disabled<br>1: CMT DMA request enabled |
| 12 | CH3DEN | Channel 3 Capture/Compare DMA request enable<br>0: Channel 3 Capture/Compare DMA request disabled<br>1: Channel 3 Capture/Compare DMA request enabled |
| 11 | CH2DEN | Channel 2 Capture/Compare DMA request enable<br>0: Channel 2 Capture/Compare DMA request disabled<br>1: Channel 2 Capture/Compare DMA request enabled |
| 10 | CH1DEN | Channel 1 Capture/Compare DMA request enable<br>0: Channel 1 Capture/Compare DMA request disabled<br>1: Channel 1 Capture/Compare DMA request enabled |
| 9 | CH0DEN | Channel 0 Capture/Compare DMA request enable<br>0: Channel 0 Capture/Compare DMA request disabled<br>1: Channel 0 Capture/Compare DMA request enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Update DMA request disabled<br>1: Update DMA request enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Break interrupt disabled<br>1: Break interrupt enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: Trigger interrupt disabled<br>1: Trigger interrupt enabled |

| | | |
|---|---|---|
| 5 | CMTIE | CMT interrupt enable<br>0: CMT interrupt disabled<br>1: CMT interrupt enabled |
| 4 | CH3IE | Channel 3 Capture/Compare interrupt enable<br>0: Channel 3 Capture/Compare interrupt disabled<br>1: Channel 3 Capture/Compare interrupt enabled |
| 3 | CH2IE | Channel 2 Capture/Compare interrupt enable<br>0: Channel 2 Capture/Compare interrupt disabled<br>1: Channel 2 Capture/Compare interrupt enabled |
| 2 | CH1IE | Channel 1 Capture/Compare interrupt enable<br>0: Channel 1 Capture/Compare interrupt disabled<br>1: Channel 1 Capture/Compare interrupt enabled |
| 1 | CH0IE | Channel 0 Capture/Compare interrupt enable<br>0: Channel 0 Capture/Compare interrupt disabled<br>1: Channel 0 Capture/Compare interrupt enabled |
| 0 | UPIE | Update interrupt enable<br>0: Update interrupt disabled<br>1: Update interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10
Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CH3OF | CH2OF | CH1OF | CH0OF | Res. | 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 |
|---|---|---|
| 15:13 | Reserved | Must be kept at reset value. |
| 12 | CH3OF | Channel 3 Capture overflow flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 Capture overflow flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 Capture overflow flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 Capture overflow 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 Capture overflow interrupt occurred<br>1: Capture overflow interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |
| 7 | BRKIF | Break interrupt flag<br>This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active.<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 by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on TRGI input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on TRGI input 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 | 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 enable<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

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

| 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 |
|---|---|---|
| 15:8 | 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 the channel control registers (CHxEN, CHxNEN and CHxCOMCTL bits) of the channels having a complementary output are updated.<br>0: no affect<br>1: generate channel’s capture/compare 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 0 is configured in input mode, the current value of the counter is captured in TIMER0_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 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 upcounting mode is selected, else (downcounting) 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

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

| 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 |
|---|---|---|
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 output compare mode<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 OFF (CH1EN bit in TIMER0_CHCTL2 register is reset).<br>00: channel 1 is configured as output<br>01: channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: channel 1 is configured as input, IC1 is connected to CI0FE1<br>11: channel 1 is configured as input, IC1 is connected to ITS. This mode is |

| | | |
|---|---|---|
| | | working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.<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 behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.<br>000: Frozen.The O0CPRE signal keeps stable, independent of the comparison between the register TIMER0_CH0CV and the counter.<br>001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced low level.<br>101: Force high. O0CPRE is forced high level.<br>110: PWM mode 0. When counting up, O0CPRE is high as long as the counter is smaller than TIMER0_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMER0_CH0CV else high.<br>111: PWM mode 1. When counting up, O0CPRE is low as long as the counter is smaller than TIMER0_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMER0_CH0CV else low.<br>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison 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. |
| 3 | CH0COMSEN | Channel 0 output compare shadow enable<br>When this bit is set, the shadow register of TIMER0_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 validating the shadow register only in one pulse mode (SPM bit set in TIMER0_CTL0 register is set).<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 |

| | | |
|---|---|---|
| | | 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 compare fast disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br>1: Channel 0 output compare fast enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O 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 OFF (CH0EN bit in TIMER0_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 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>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH1EN bit in TIMER0_CHCTL2 register is reset).<br>00: Channel 1 is configured as output<br>01: Channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: Cchannel 1 is configured as input, IC1 is connected to CI0FE1<br>11: Channel 1 is configured as input, IC1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disable, $f_{SAMP}$=$f_{DTS}$, N=1 |

0001: $f_{SAMP}$= $f_{TIMER\_CK}$, N=2

0010: $f_{SAMP}$= $f_{TIMER\_CK}$, N=4

0011: $f_{SAMP}$= $f_{TIMER\_CK}$, N=8

0100: $f_{SAMP}$=$f_{DTS}$/2, N=6

0101: $f_{SAMP}$=$f_{DTS}$/2, N=8

0110: $f_{SAMP}$=$f_{DTS}$/4, N=6

0111: $f_{SAMP}$=$f_{DTS}$/4, N=8

1000: $f_{SAMP}$=$f_{DTS}$/8, N=6

1001: $f_{SAMP}$=$f_{DTS}$/8, N=8

1010: $f_{SAMP}$=$f_{DTS}$/16, N=5

1011: $f_{SAMP}$=$f_{DTS}$/16, N=6

1100: $f_{SAMP}$=$f_{DTS}$/16, N=8

1101: $f_{SAMP}$=$f_{DTS}$/32, N=5

1110: $f_{SAMP}$=$f_{DTS}$/32, N=6

1111: $f_{SAMP}$=$f_{DTS}$/32, N=8

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

This bit-field specifies the ratio of the prescaler on channel 0 input.The prescaler is reset when CH0EN bit in TIMER0_CHCTL2 register is reset.

00: Prescaler disable, capture is done on each channel input edge

01: Capture is done every 2 channel input edges

10: Capture is done every 4 channel input edges

11: Capture is done every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH0EN bit in TIMER0_CHCTL2 register is reset).

00: Channel 0 is configured as output

01: Channel 0 is configured as input, IC0 is connected to CI0FE0

10: Channel 0 is configured as input, IC0 is connected to CI1FE0

11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C
Reset value: 0x0000

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

| 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 |
|---|---|---|
| 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 OFF (CH3EN bit in TIMER0_CHCTL2 register is reset).<br>00: Channel 3 is configured as output<br>01: Channel 3 is configured as input, IC3 is connected to CI3FE3<br>10: Channel 3 is configured as input, IC3 is connected to CI2FE3<br>11: Channel 3 is configured as input, IC3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable<br>Refer to CH0COMCEN description |
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>Refer to CH0COMCTL description |
| 3 | CH2COMSEN | Channel 2 output compare shadow enable<br>Refer to CH0COMSEN description |
| 2 | CH2COMFEN | Channel 2 output compare fast enable<br>Refer to CH0COMFEN description |
| 1:0 | CH2MS[1:0] | Channel 2 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 OFF (CH2EN bit in TIMER0_CHCTL2 register is reset).<br>00: Channel 2 is configured as output<br>01: Channel 2 is configured as input, IC2 is connected to CI2FE2<br>10: Channel 2 is configured as input, IC2 is connected to CI3FE2<br>11: Channel 2 is configured as input, IC2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 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>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH3EN bit in TIMER0_CHCTL2 register is reset).<br>00: channel 3 is configured as output<br>01: channel 3 is configured as input, IC3 is connected to CI3FE3<br>10: channel 3 is configured as input, IC3 is connected to CI2FE3<br>11: channel 3 is configured as input, IC3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>Refer to CH0CAPFLT description |
| 3:2 | CH2CAPPSC[1:0] | Channel 2 input capture prescaler<br>Refer to CH0CAPPSC description |
| 1:0 | CH2MS[1:0] | Channel 2 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 OFF (CH2EN bit in TIMER0_CHCTL2 register is reset).<br>00: channel 2 is configured as output<br>01: channel 2 is configured as input, IC2 is connected to CI2FE2<br>10: channel 2 is configured as input, IC2 is connected to CI3FE2<br>11: channel 2 is configured as input, IC2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

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

| 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 | CHt0EN |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value |
| 13 | CH3P | Channel 3 polarity<br>Refer to CH0P description |
| 12 | CH3EN | Channel 3 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 polarity<br>Refer to CH0P description |
| 8 | CH2EN | Channel 2 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 polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 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 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, In conjunction 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 channel 0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 polarity |

| | | |
|---|---|---|
| | | When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will selete 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]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 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 channel 0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CAR[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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CH0VAL[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel 0<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

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CH1VAL[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH1VAL[15:0] | Capture or compare value of channel 1<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

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

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

| |
|---|
| CH2VAL[15:0] |
| rw |

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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH3VAL[15:0] | Capture or compare value of channel 3<br>When channel 3 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. |

### Channel Complementary Protection register (TIMERx_CCHP)

Address offset: 0x44
Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15 | POEN | Primary output enable<br>This bit s set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware assoon as the break input is active. When |

| | | |
|---|---|---|
| | | a channel 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: Channel outputs are disabled or forced to idle state.<br>1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN can be not set by hardware.<br>1: POEN can be set by hardware automatically at the next update event,if the break input is notactive.<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 BRK input signal.<br>0: BRK input active low<br>1; BRK input active high |
| 12 | BRKEN | Break enable<br>This bit can be set to enable the BRK and CCS clock failure event inputs.<br>0: Break inputs disabled<br>1; Break inputs enabled<br>This bit can be modified onlywhen PROT[1:0] bit-filed in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode off-state configure<br>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.<br>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled.<br>1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMER0_CHCTL2 register.<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 configure<br>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode.<br>0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled.<br>1: When POEN bit is reset, he channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMER0_CHCTL2 register.<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 |

| Bits | Fields | Descriptions |
|---|---|---|
| | | This bit-filed specifies the write protection property of registers.<br>00: protect disable. No write protection.<br>01: PROT mode 0.The ISOx/ISOxN bits in TIMER0_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMER0_CCHP register are writing protected.<br>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 TIMER0_CCHP register are writing protected.<br>11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMER0_CHCTLx registers (if the related channel is configured in output) are writing protected.<br>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<br>This bit-field specifies the duration of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG[7:5]=0xx: duration =DTCFG[7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG[7:5]=10x:duration =(64+DTCFG[5:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*2.<br>DTCFG[7:5]=110: duration =(32+DTCFG[4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG[7:5]=111:duration =(32+DTCFG[4:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*16.<br>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

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

| 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 |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value. |
| 12:8 | DMATC[4:0] | DMA transfer count<br>When register access are done through the TIMERx_DMATB address, this 5-bit bit-field specifies the number of transfers. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | DMATA[4:0] | DMA transfer access start address<br>When register access are done through the TIMERx_DMATB address, this bit-field |

specifies the offset of the starting address from the TIMER0_CTL0 register.

### DMA Transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or 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 |
|---|---|---|
| 15:0 | DMATB[15:0] | DMA transfer<br>When a read or write operation is assigned to this register, the register located at the address range (DMATA + burst counter) x 4 from TIMERx_CTL0 will be accessed.<br>The burst counter is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value |
| 1 | CCSEL | Write Capture/Compare register selection<br>This bit-field set and reset by software.<br>1: If write the Capture/Compare register, the write value is same as the Capture/Compare 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 |

## 9.2. General timers (TIMER1/TIMER2)

### 9.2.1. Introduction

The general timers (TIMER1 and TIMER2) consist of one 16-bit or 32-bit up/down-counter; four capture/compare registers (TIMERx_CHxCV), one counter auto reload register (TIMERx_CAR) and several control registers. They can be used for a variety of purposes including general timer, input signal pulse width measurement or output waveform generation such as single pulse generation or PWM output. The TIMERx supports an Encoder Interface using a decoder with two inputs.

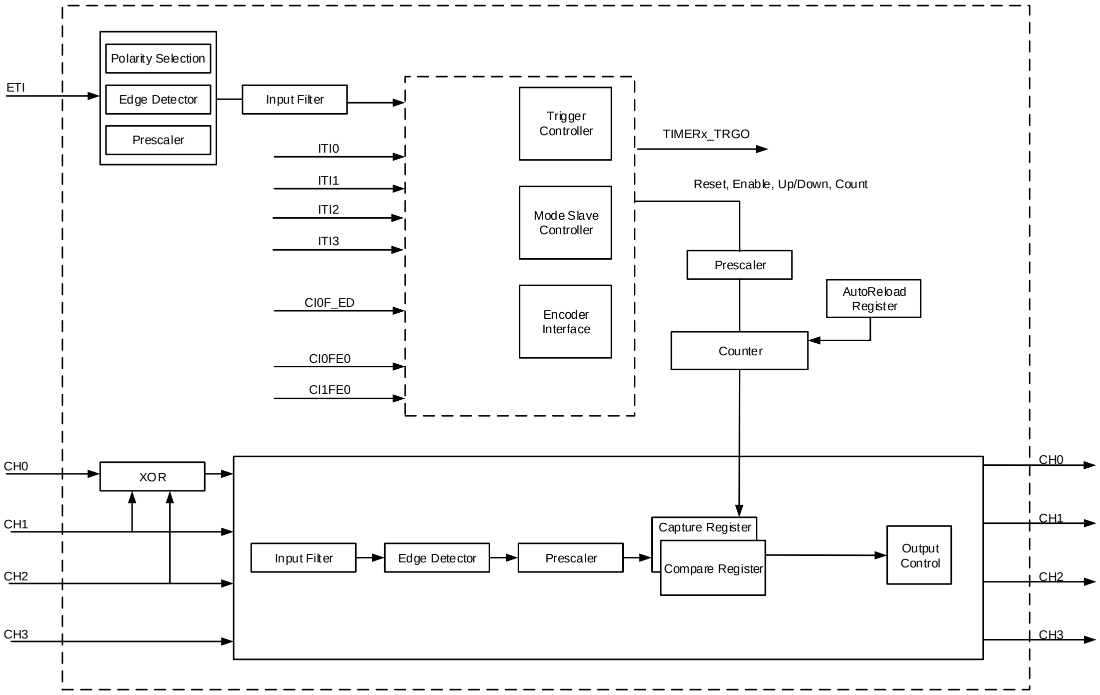
### 9.2.2. Main features

- 16-bit (TIMER2) or 32-bit (TIMER1) down, up, down/up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels support functions including input capture, compare match output, generation of PWM waveform (edge and center-aligned Mode), and single pulse mode output.
- Interrupt/DMA generation by update, trigger event, input capture event, output compare match event
- Synchronization circuit to control TIMERx with external signals or to interconnect several timers together.
- Encoder interface controller with two inputs using quadrature decoder
- TIMERx Master/Slave mode controller

### 9.2.3. Function description

Figure below provides details on the internal configuration of the general timer.

**Figure 9-46. General timer block diagram (TIMER1 and TIMER2)**



## Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx_PSC) which can be changed on the fly but be taken into account at the next update event.

**Figure 9-47. Counter timing diagram with prescaler division change from 1 to 2**

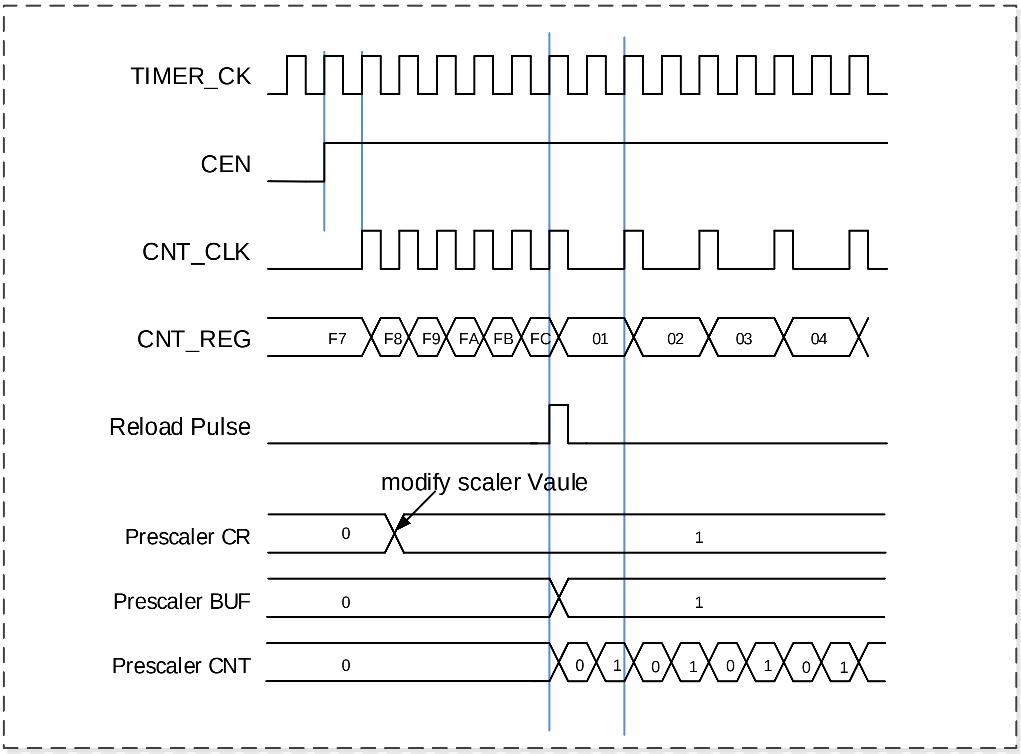
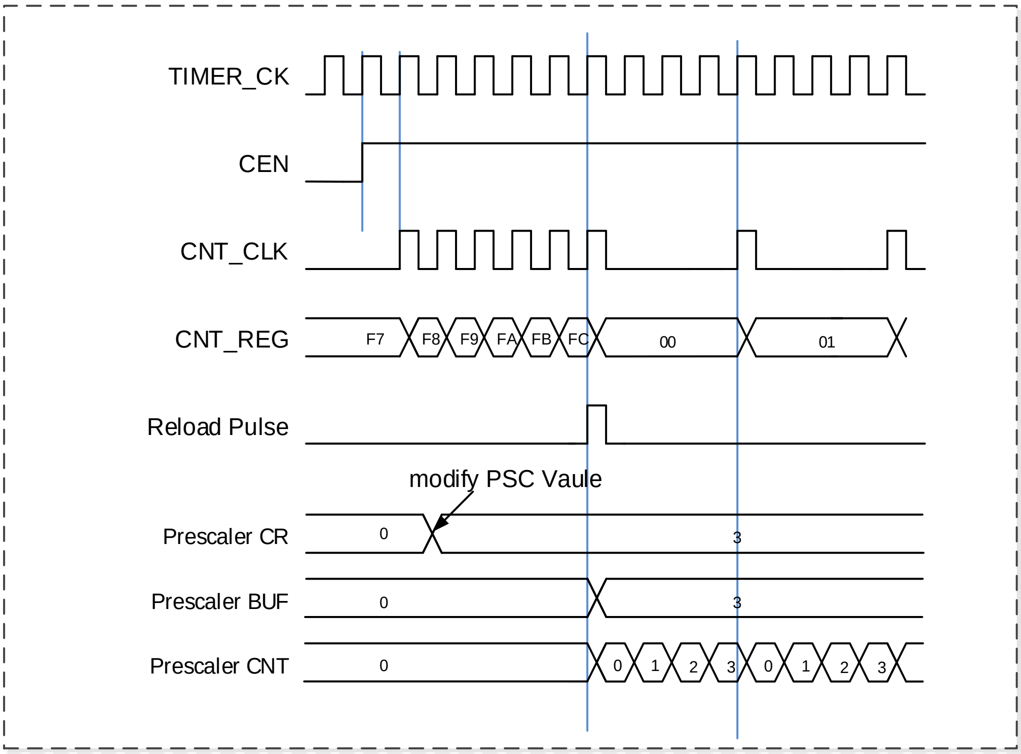


**Figure 9-48. Counter timing diagram with prescaler division change from 1 to 4**



## Upcounting mode

In this mode the counter counts 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 to count once again from 0. The update event is generated at each counter overflow.The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the upcounting 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 set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

**Figure 9-49. Counter timing diagram, internal clock divided by 1**

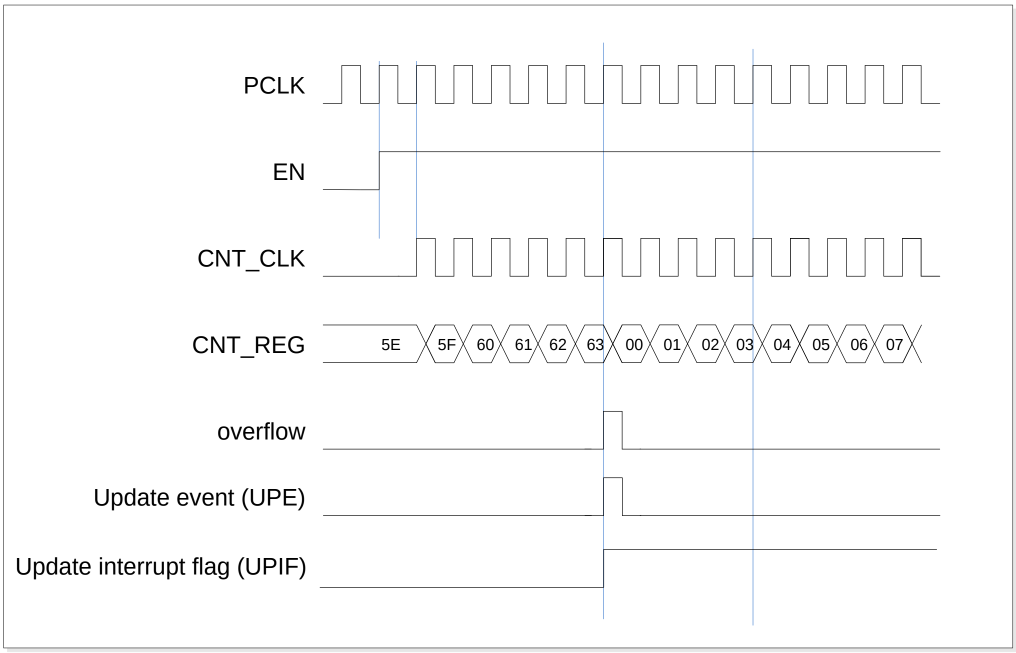


Figure 9-50. Counter timing diagram, internal clock divided by 2

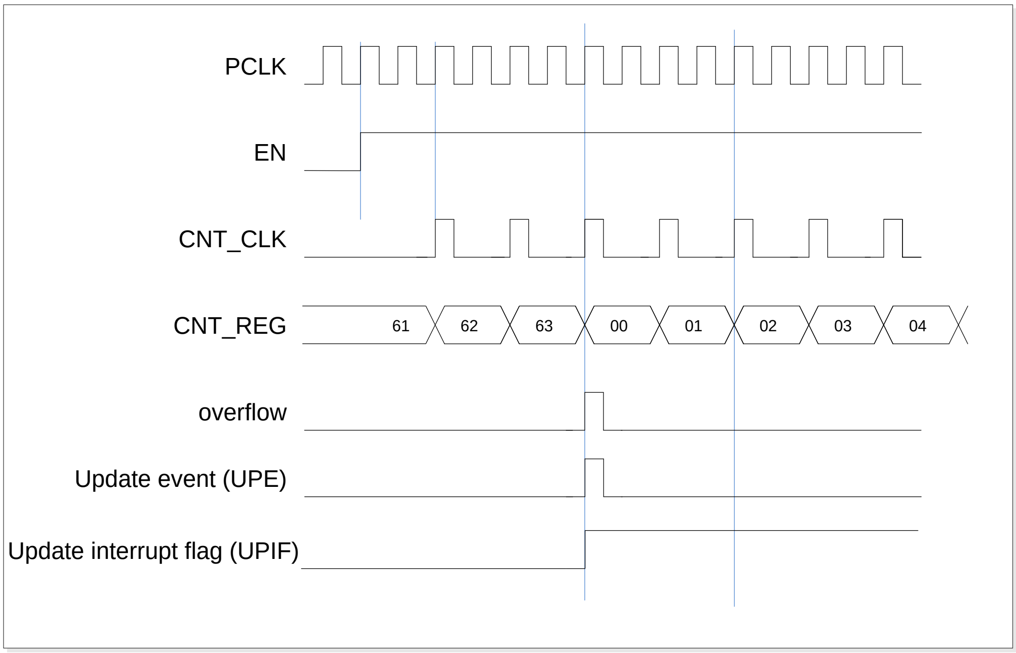


Figure 9-51. Counter timing diagram, internal clock divided by 4

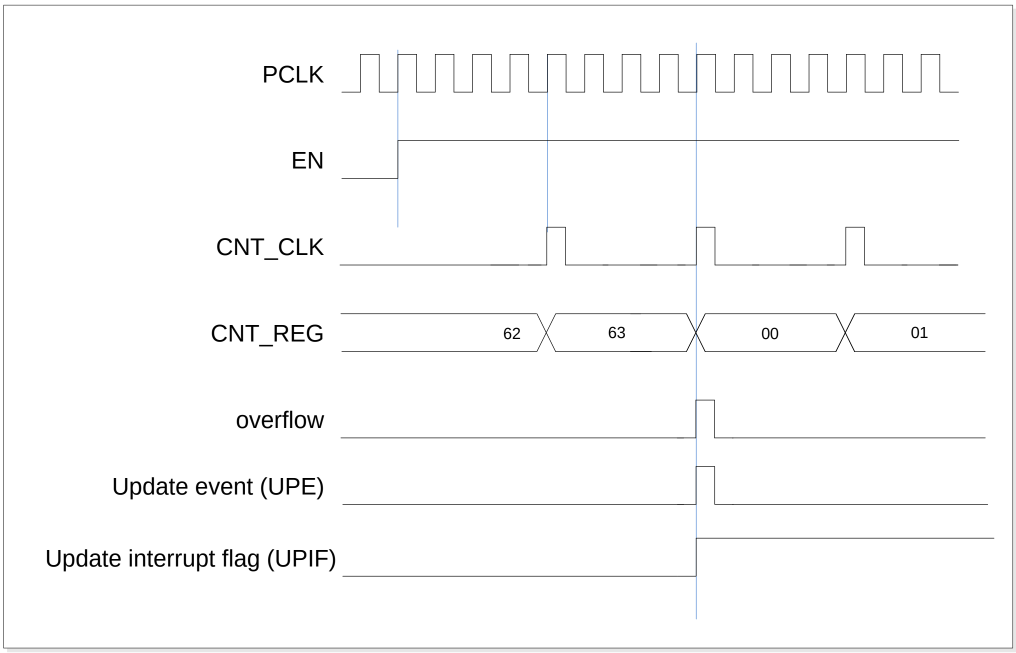


Figure 9-52. Counter timing diagram, internal clock divided by N

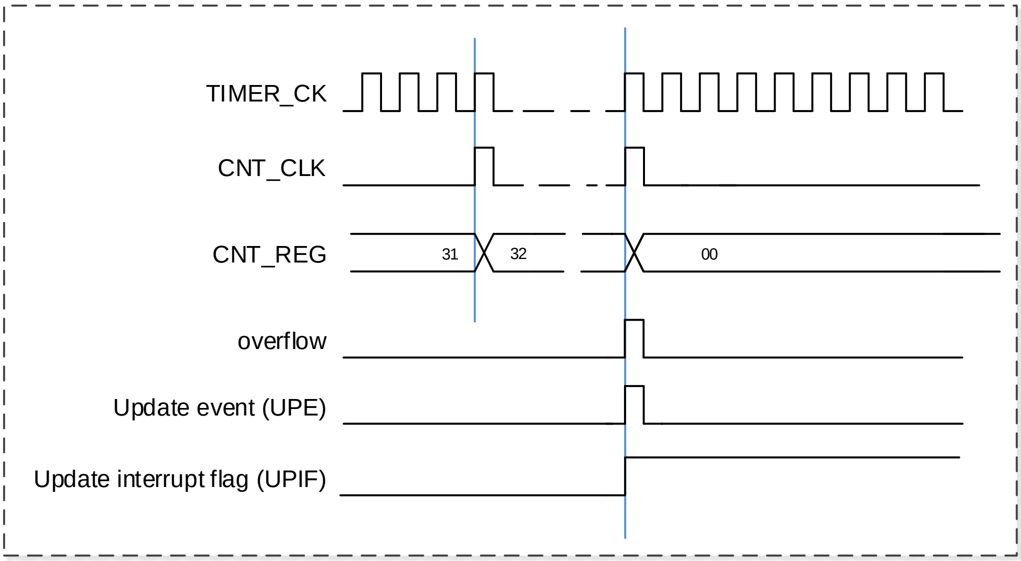


Figure 9-53. Counter timing diagram, update event when ARSE=0

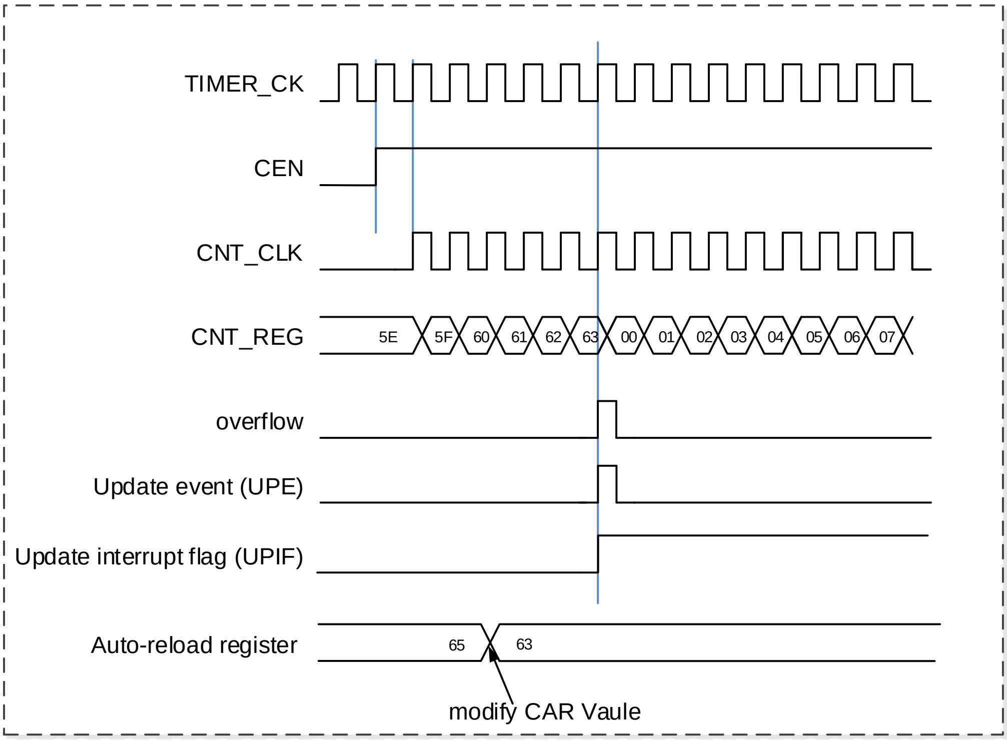
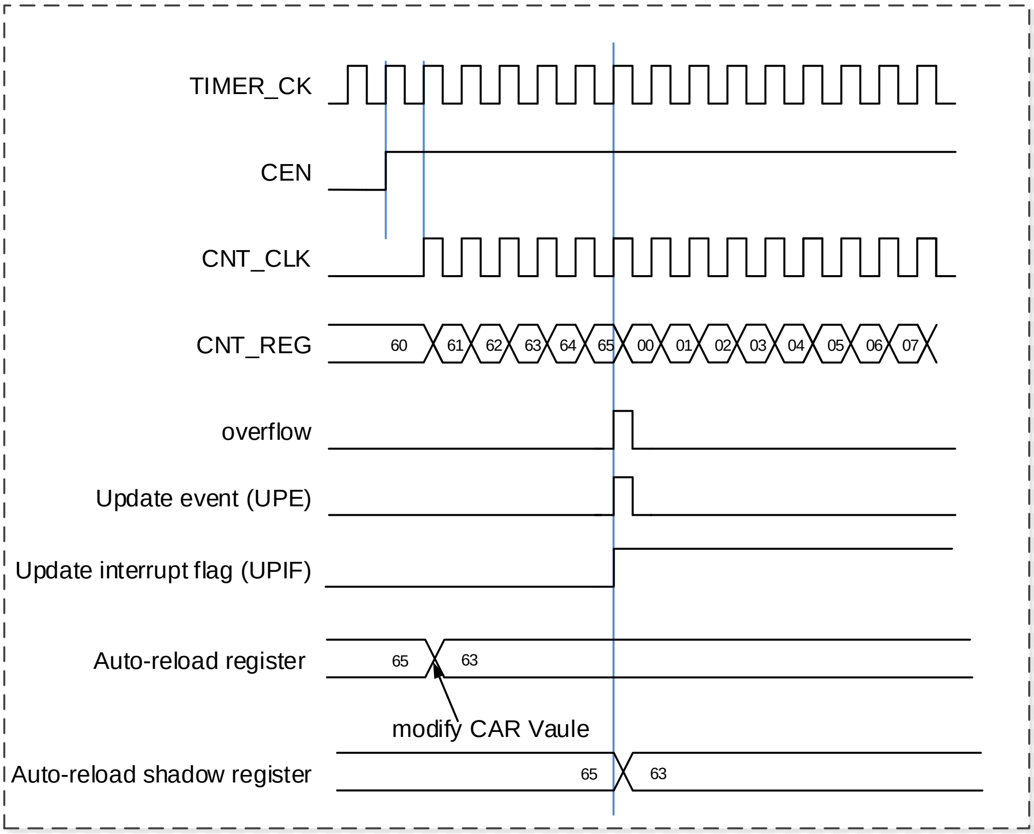


**Figure 9-54. Counter timing diagram, update event when ARSE=1**



### Downcounting mode

In this mode the counter counts 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 0, the counter restarts to count once again from the counter-reload value. If the repetition counter is set, the update event generated after the number of underflow. Else the update event is generated at each counter 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 registers (repeat counter, reload register, prescaler register) are updated.

**Figure 9-55. Counter timing diagram, internal clock divided by 1**

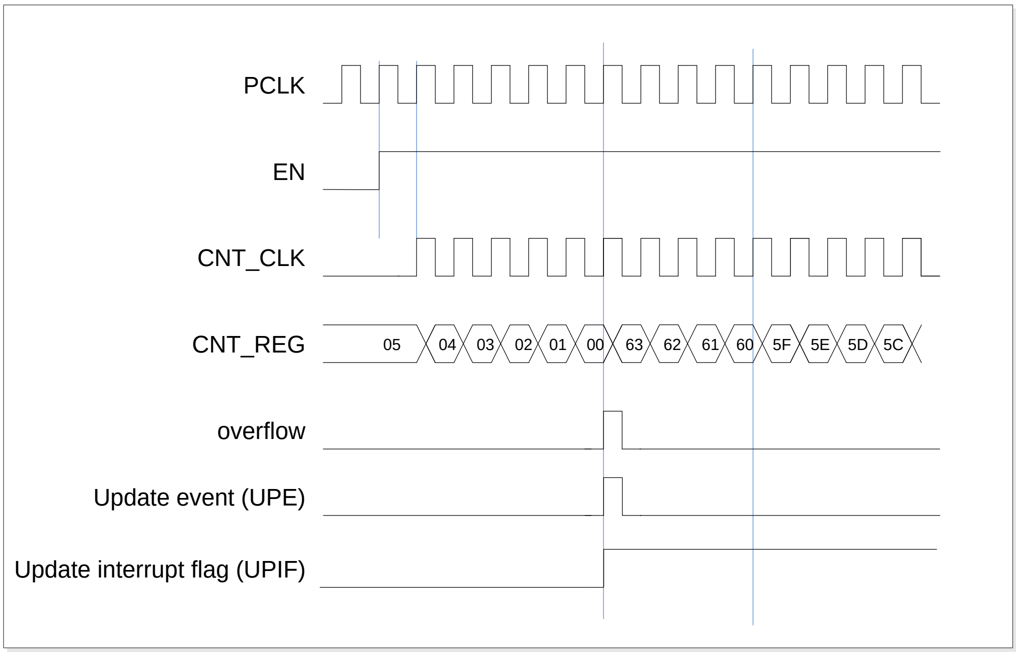


**Figure 9-56. Counter timing diagram, internal clock divided by 2**

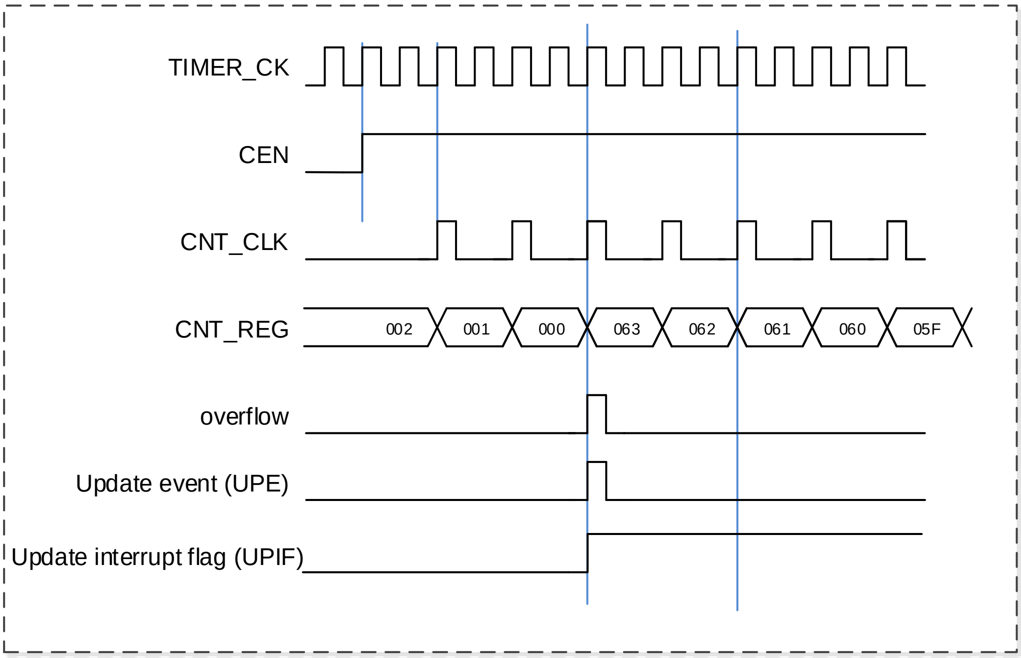


Figure 9-57. Counter timing diagram, internal clock divided by 4

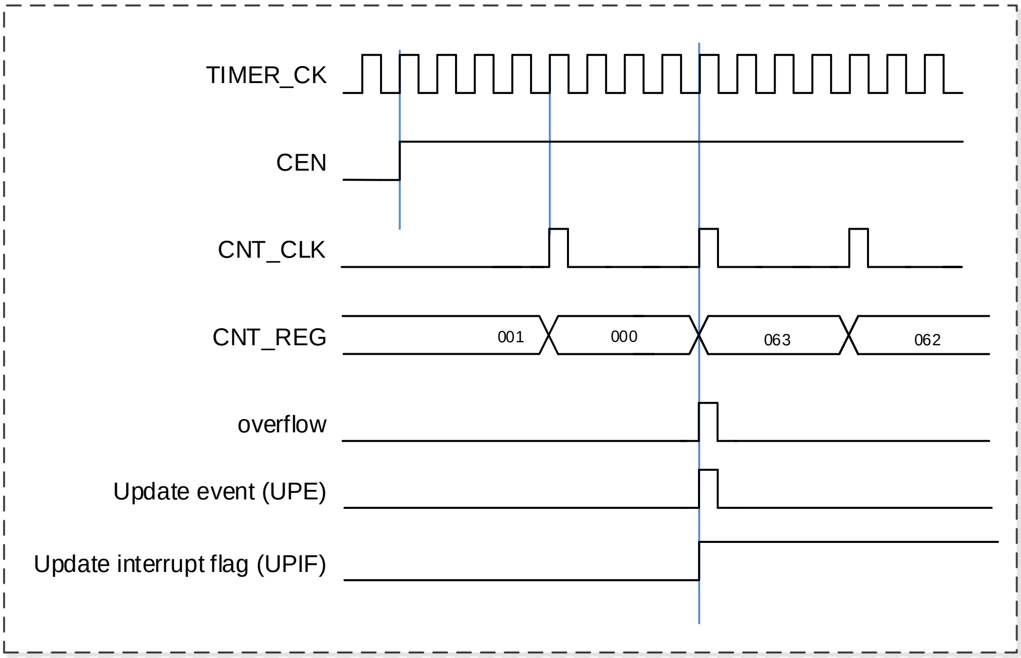


Figure 9-58. Counter timing diagram, internal clock divided by N

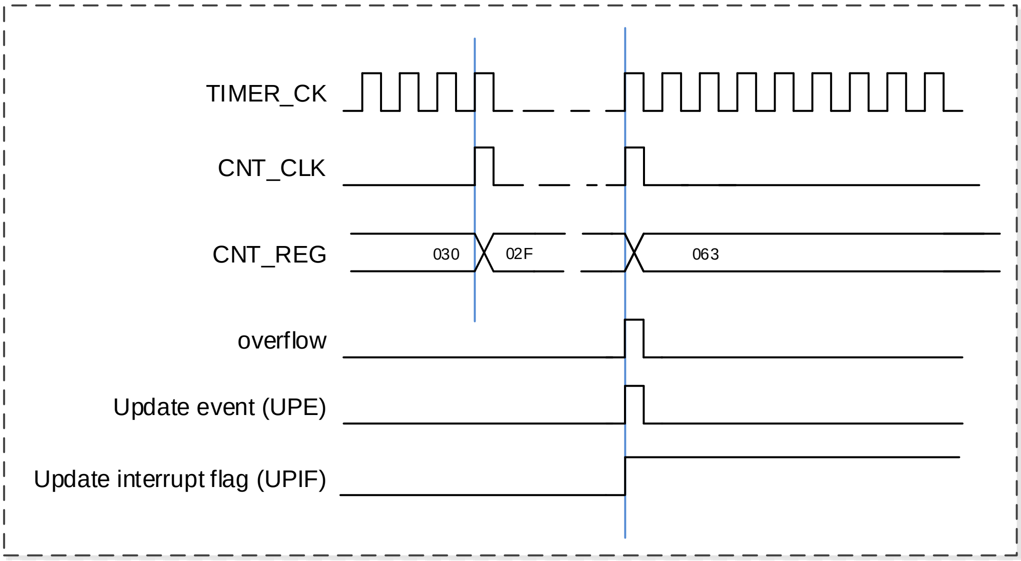
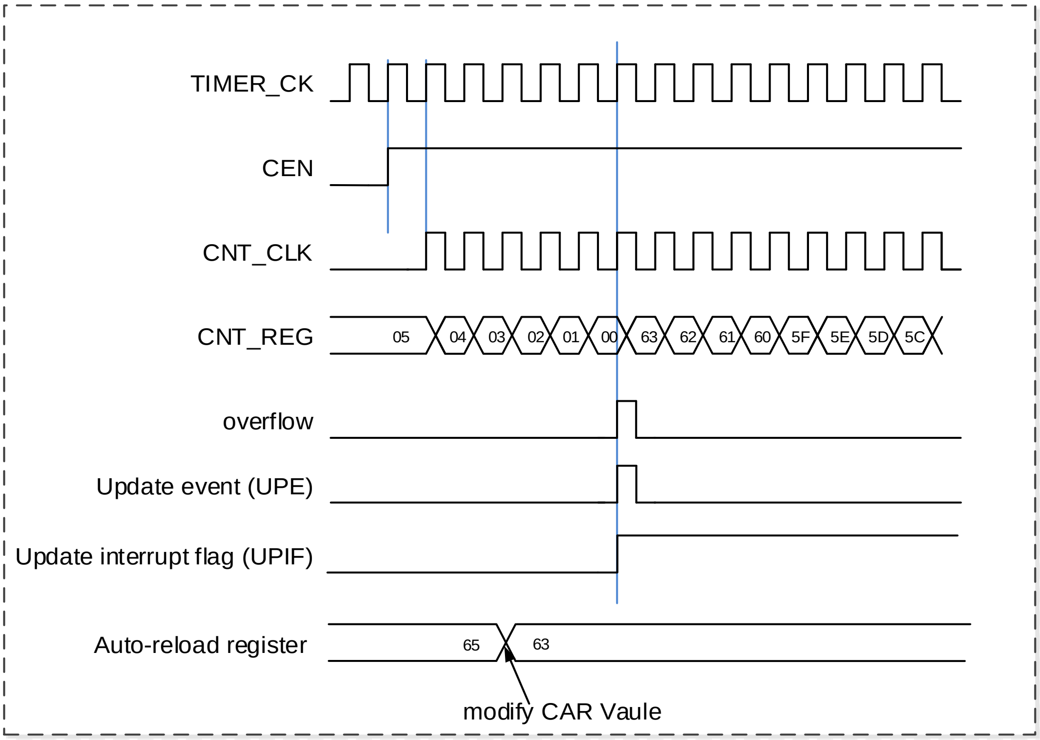


**Figure 9-59. Counter timing diagram, update event when counter is not used**



### Center-aligned counting mode

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. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 irrespective of whether the counter is counting up or down in the center-align counting mode and generates an update event.

The UPIF bit in the TIMER0_INTF register can be set to 1 when an underflow event at count-down (CAM in TIMERx_CTL0 is “01”), an overflow event at count-up (CAM in TIMERx_CTL0 is “10”) or both of them occur (CAM in TIMERx_CTL0 is “11”).

If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x5.

**Figure 9-60. Counter timing diagram, internal clock divided by 1, TIMERx_CAR = 0x5**

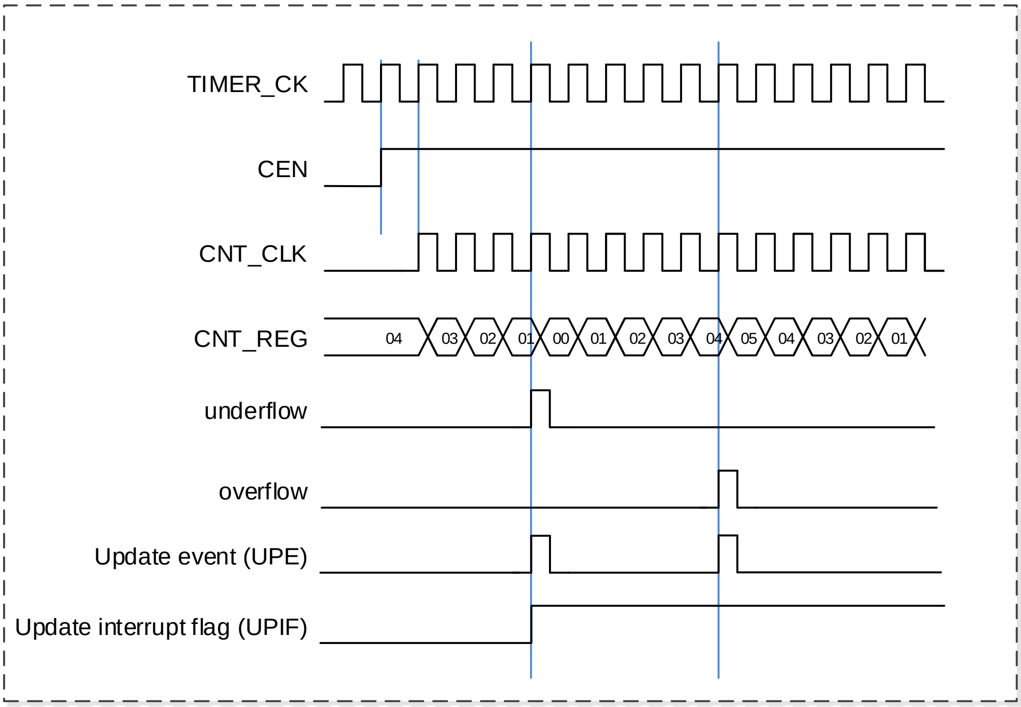


**Figure 9-61. Counter timing diagram, internal clock divided by 2**

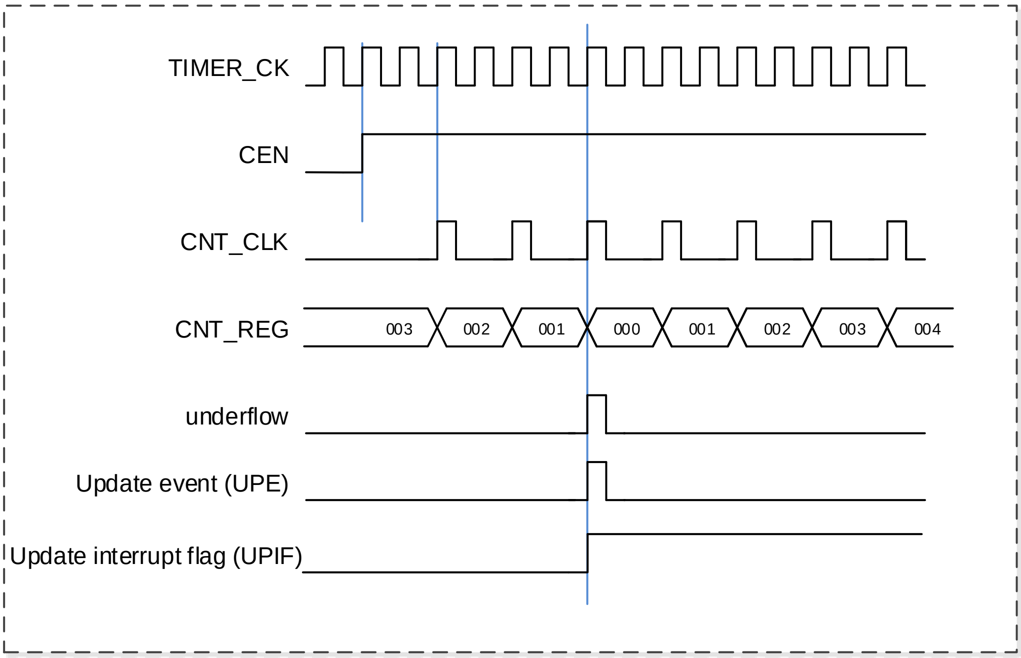


Figure 9-62. Counter timing diagram, internal clock divided by 4, TIMERx_CAR=0x63

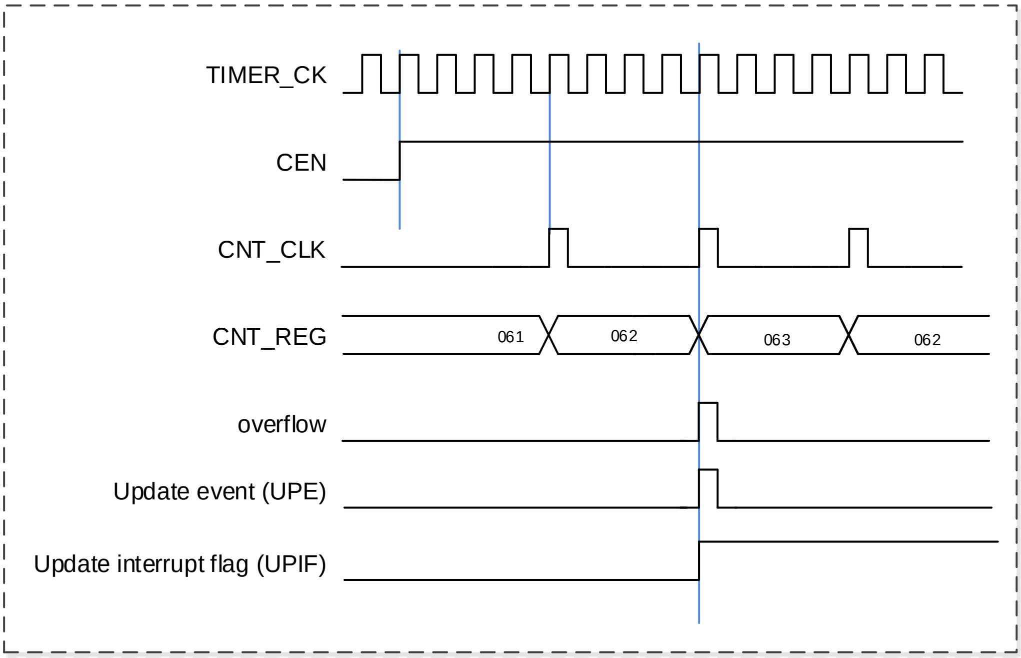


Figure 9-63. Counter timing diagram, internal clock divided by N

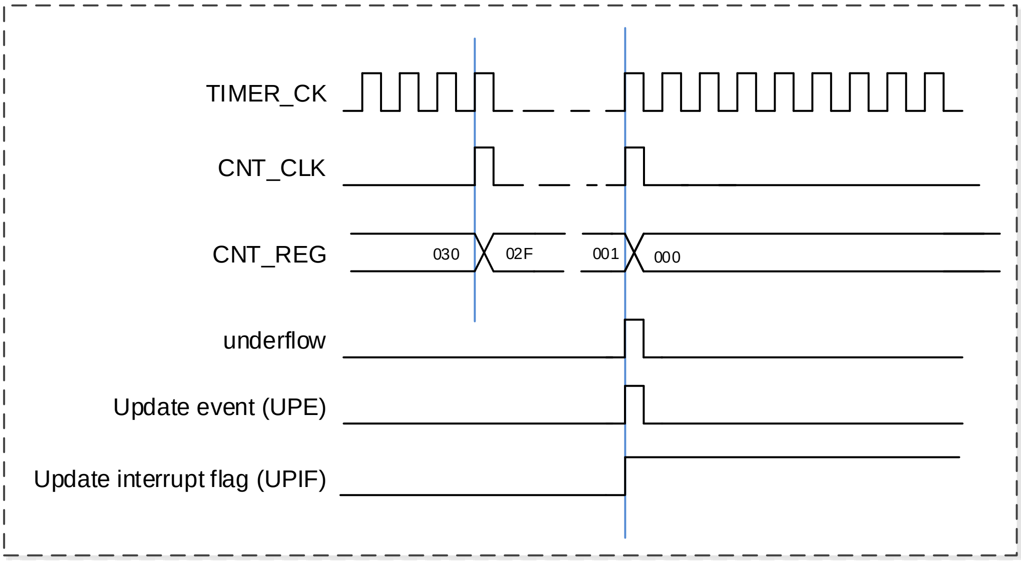


**Figure 9-64. Counter timing diagram, update event with ARSE=1(counter underflow)**

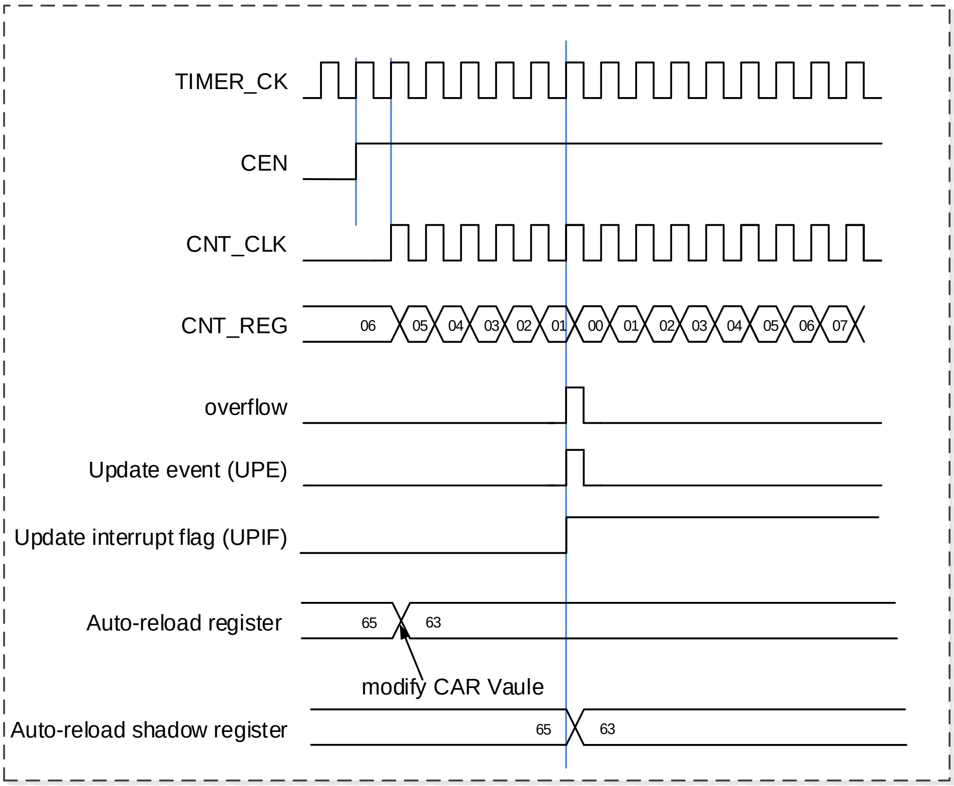
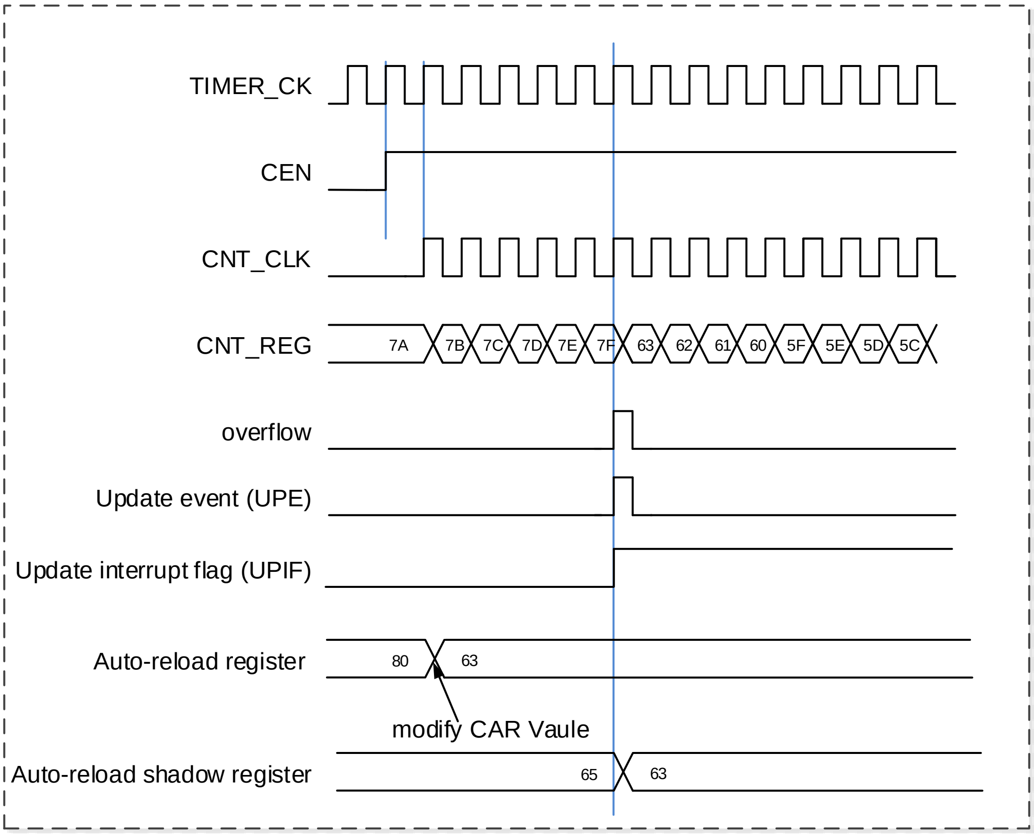


**Figure 9-65. Counter timing diagram, Update event with ARSE=1 (counter overflow)**



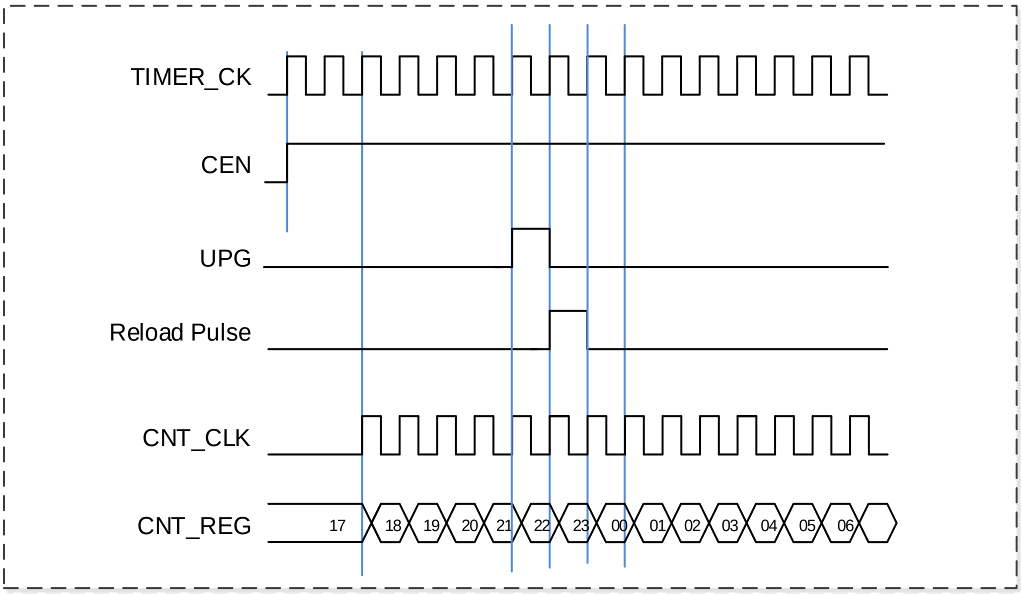
## Clock selection

The following describes the Timer Module clock controller which determines the clock source of the internal prescaler counter.

- Internal timer clock TIMER_CK

The default internal clock source is the APB2 clock CK_APB2 used to drive the counter prescaler when the slave mode is disabled. If the slave mode controller is enabled by setting SMC field in the TIMERx_SMCFG register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS field in the TIMERx_SMCFG register and described as follows. When the slave mode selection bits SMC are set to 0x4 , 0x5 or 0x6, the internal clock TIMER_CK is the counter prescaler driving clock source.

**Figure 9-66. Control circuit in normal mode, internal clock divided by 1**



- Quadrature decoder

To select Quadrature Decoder mode the SMC field should be set to 0x1, 0x2 or 0x3 in the TIMERx_SMCFG register. The Quadrature Decoder function uses two input states of the TIMERx_CH0 and TIMERx_CH1 pins to generate the clock pulse to drive the counter prescaler. The counting direction bit DIR is modified by hardware automatically at each transition on the input source signal. The input source signal can be derived from the TIMERx_CH0 pin only, the TIMERx_CH1 pin only or both TIMERx_CH0 and TIMERx_CH1 pins.

- Internal trigger inputs (ITI)

The counter prescaler can count during each rising or falling edge of the ITI signal. This mode can be selected by setting the SMC field to 0x6 in the TIMERx_SMCFG Rregister; here the counter will act as an event counter. The input event, known as ITI here, can be selected by setting the TRGS field. When the ITI signal is selected as the clock source, the internal edge detection circuitry will generate a clock pulse during each ITI signal rising or falling edge to drive the counter prescaler.

- Channel input pin (CI)

The counter prescaler can be driven to count during each rising or falling edge on the external pin TIMERx_CIx. This mode can be selected by setting SMC field to 0x7 and the TRGS field to 0x4, 0x5 or 0x6. Note that the CIx is derived from the TIMERx_CHx sampled by a digital filter.

- External trigger input (ETIF)

The counter prescaler can be driven to count during each rising or falling edge on the external pin TIMERx_ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETIF signal as the clock source is to set the SMC field to 0x6 and the TRGS field to 0x7 respectively. Note that the ETIF signal is derived from the TIMERx_ETI pin sampled by a digital filter. When the clock source is selected to come from the ETIF signal, the Trigger Controller including the edge detection circuitry will generate a clock pulse during each ETIF signal rising edge to clock the counter prescaler.

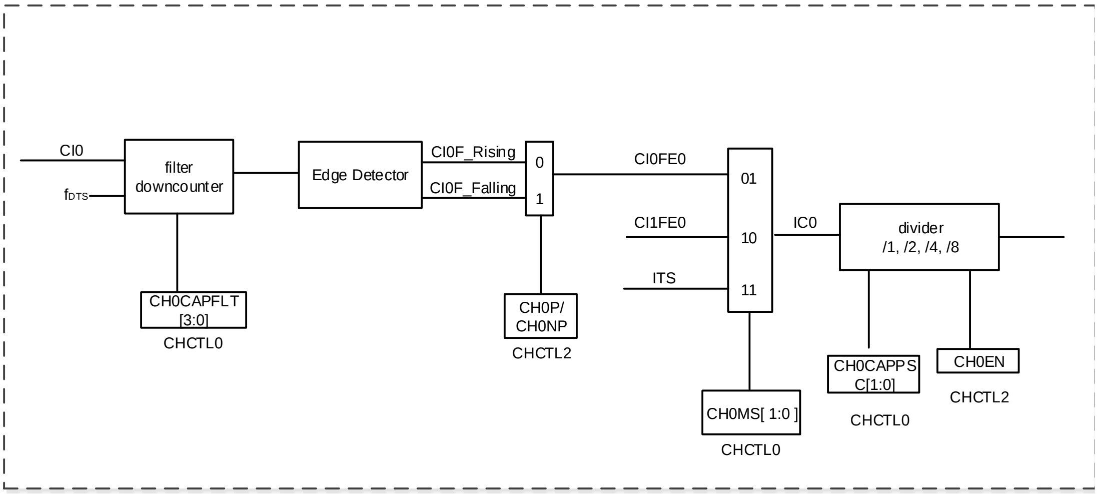
### Capture/compare channels

The TIMER1/2 has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture/compare value register including an input stage, channel controller and an output stage.

- Input capture stage

The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. The channel 0 input signal (CI0) can be chosen to come from the TIMERx_CH0 signal or the Excusive-OR function of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. The channel input signal (CIx) is sampled by a digital filter to generate a filtered input signal CIxF. Then the channel polarity and the edge detection block can generate a CIxFEx signal for the input capture function. The effective input event number can be set by the channel input prescaler (CHxCAPPSC).

Figure 9-67. Capture/compare channel (example: channel 0 input stage)



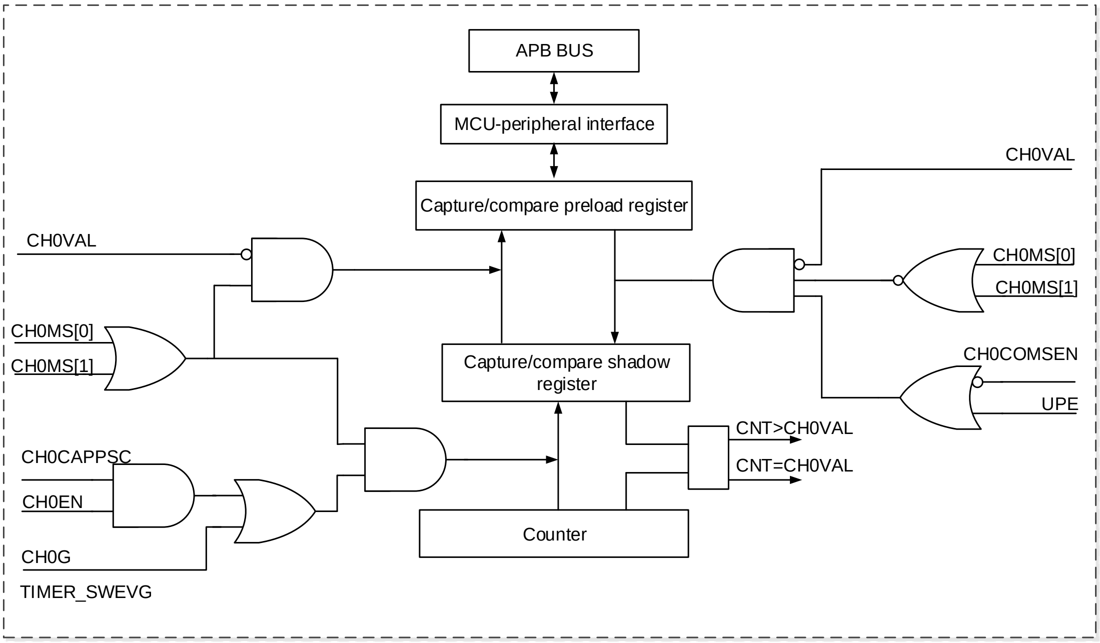
- Channel controller

The GPTM has four independent channels which can be used as capture inputs or compare match outputs.

When used in the input capture mode, the counter value is captured into the TIMERx_CHxCV shadow register first and then transferred into the TIMERx_CHxCV preload register when the capture event occurs.

When used in the compare match output mode, the contents of the TIMERx_CHxCV preload register is copied into the associated shadow register; the counter value is then compared with the register value.

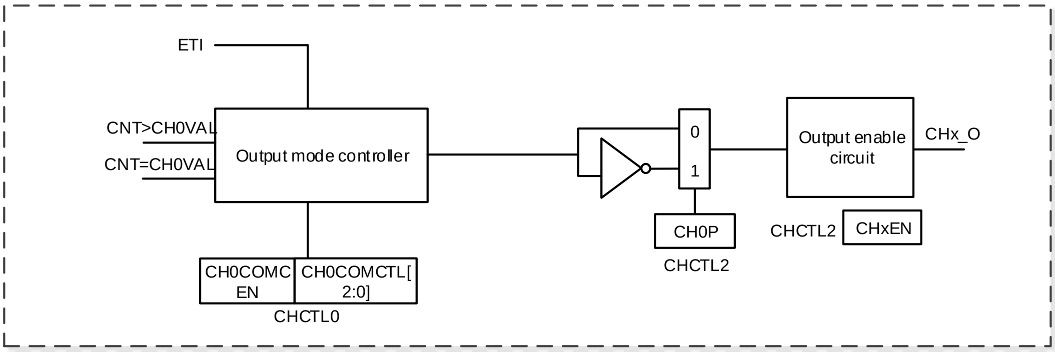
**Figure 9-68. Capture/compare channel 0 main circuit**



- Output stage

The TIMERx has four channels for compare match, single pulse or PWM output function.

**Figure 9-69. Output stage of capture/compare channel (channel 0)**



## Input capture mode

When the channel is used as a capture input, the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV) when an effective input signal transition occurs. Once the capture event occurs, the CHxIF flag in the TIMERx_INTF register is set. If the CHxIF bit is already set, i.e., the flag has not yet been cleared by software, and another capture event on this channel occurs, the corresponding channel Over-Capture flag, named CHxOF, will be set.Once the capture event occurs, a DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit. The fllowing step maybe used to implement input capture mode:

- Select input signal by set CHxMS bits in the channel control register (TIMERx_CHCTLx).
- Add input filter to input signal by set CHxCAPFLT bitst in the the channel control register

(TIMERx_CHCTLx).

- Select input capture edge (rising or falling) by set CHxP/CHxNP bit in the channel control register 2 (TIMERx_CHCTL2).
- If input signal need prescaler, set CHxCAPPSC bits in the the channel control register (TIMERx_CHCTLx).
- If need interrupt, set CHxIE bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- If need DMA request, set CHxDEN bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- Enable the channel by set CHxEN bit in the channel control register 2 (TIMERx_CHCTL2) to 1.

After configure this registers, the input capture event on this channel may occurs at the corresponding edge. And the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV). An interrupt generated if CHxIE bit set. A DMA request generated if CHxDEN set.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins (CIx). For example, PWM signal connect to CI0 input. Select channel 0 capture signal to CI0 by setting CH0MS to 01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 10 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.

### Output compare mode/PWM mode

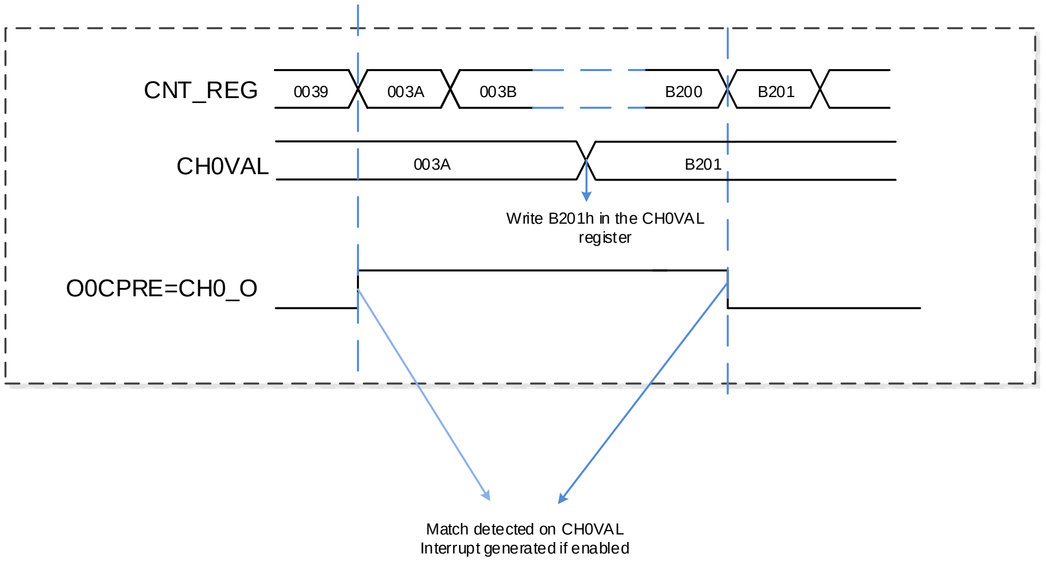
When the channel is used as a output mode by setting the CHxMS in the the channel control register (TIMERx_CHCTLx) to 00, the channel outputs waveform according to the CHxCOMCTL bits in the the the channel control register (TIMERx_CHCTLx), and the comparision between the counter and TIMERx_CHxCV registers. When the comparision equals, the CHxIF flag in the Interrupt flag registers (TIMERx_INTF) set to 1. A DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit.

In output mode, the channel can outputs active level by set the CHxCOMCTL to 101, and outputs inactive level by set the CHxCOMCTL bits to 100.

In ouput mode, the channel output set to active level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 001. The channel output set to inactive level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 010.

In output compare toggle mode (by set the CHxCOMCTL to 011), the channel output toggles when the comparision between the counter and TIMERx_CHxCV registers match.

**Figure 9-70. Output compare toggle mode, toggle on CH0_O**



In the output PWM mode (by setting the CHxCOMCTL bits to 110 (PWM mode 0) or to 111 (PWM mode 1)), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

**Figure 9-71. Output compare PWM mode 0 on CH0_O, upcounting mode**

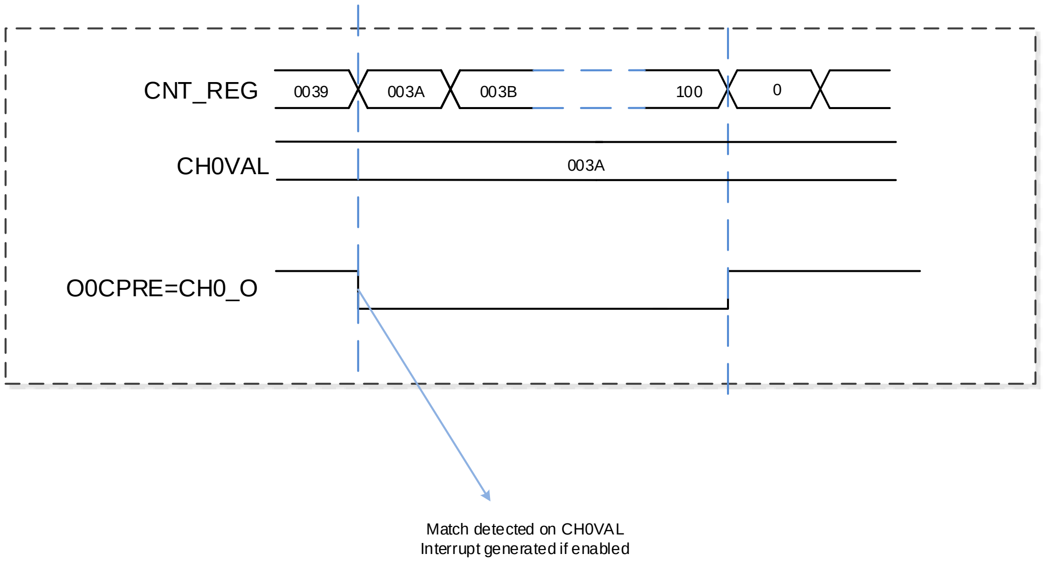
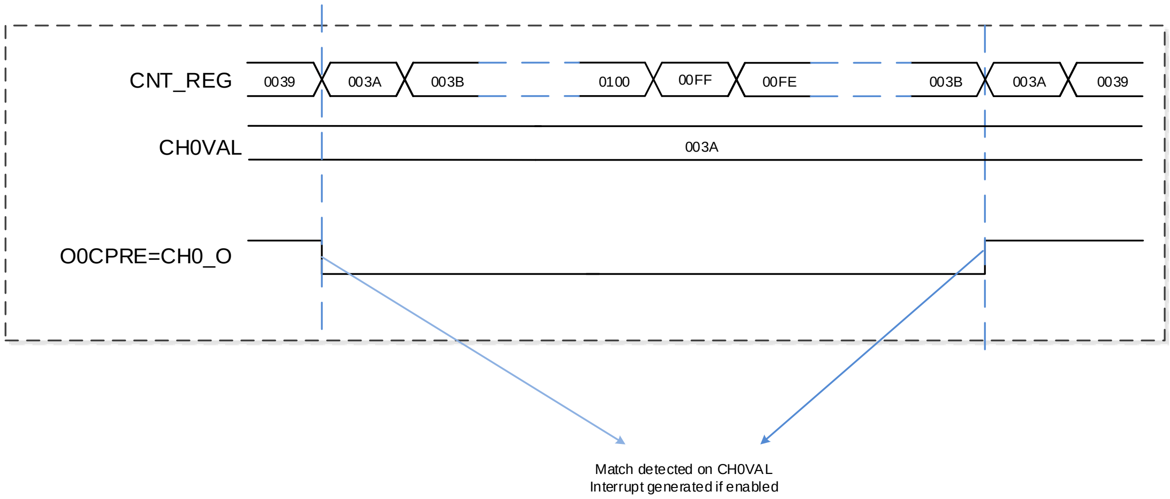


Figure 9-72. Output compare PWM mode 0 on CH0_O, center-aligned counting mode

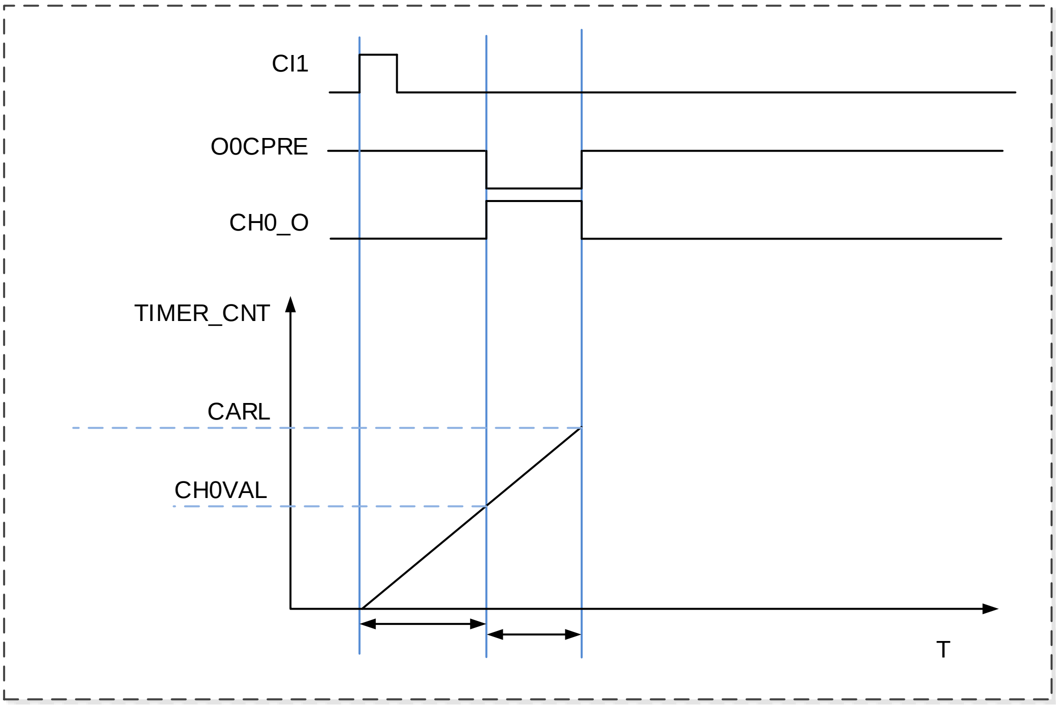


## Single pulse mode

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. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

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 CHxOFE bit in each TIMERx_CHCTL0 register. After a trigger rising edge 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 CHxOFE 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 9-73. Single pulse mode**



### Channel output reference signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output Reference signal) is defined by setting the CHxCOMCTL bits. 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 detailed 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.

The OxCPRE signal can be forced to 0 when the ETIF signal is derived from the external TIMERx_ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Quadrature decoder

The Quadrature Decoder function uses two quadrature inputs CI0 and CI1 derived from the

TIMERx_CH0 and TIMERx_CH1 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either CI0 only, CI1 only or both CI0 and CI1, the selection made by setting the SMC field to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. 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-reload value. Therefore, users must configure the TIMERx_CAR register before the counter starts to count.

**Table 9-2. Counting direction versus encoder signals**

| Counting mode | Level | CI0FE0 | | CI1FE1 | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |
| CI0 only counting | CI1FE=High | Down | Up | - | - |
| | CI1FE=Low | Up | Down | - | - |
| CI1 only counting | CI0FE=High | - | - | Up | Down |
| | CI0FE=Low | - | - | Down | Up |
| CI0 and CI1 counting | CI1FE=High | Down | Up | X | X |
| | CI1FE=Low | Up | Down | X | X |
| | CI0FE=High | X | X | Up | Down |
| | CI0FE=Low | X | X | Down | Up |

**Note:** "-" means "no counting"; "X" means impossible.

**Figure 9-74. Example of counter operation in encoder interface mode**

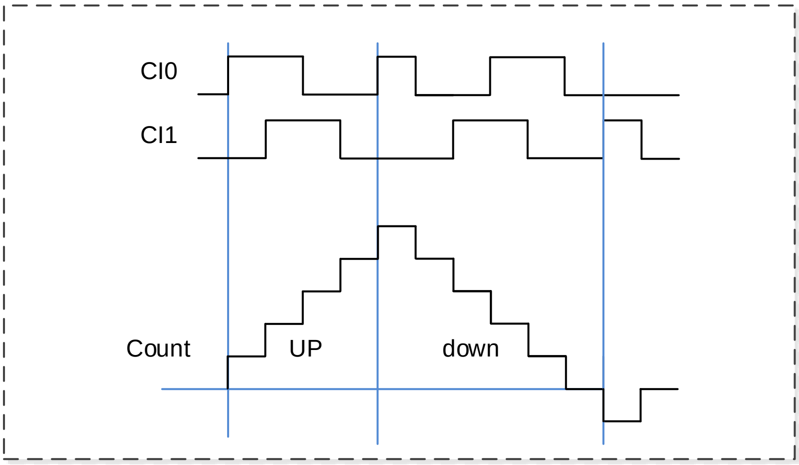
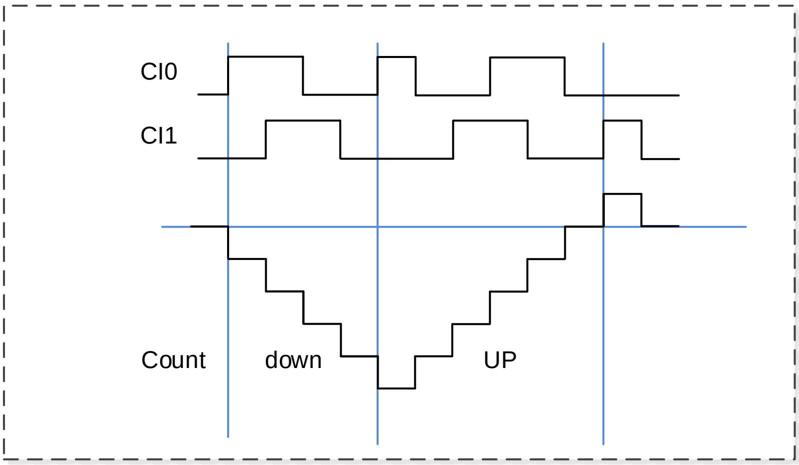


**Figure 9-75. Example of encoder interface mode with CI0FE0 polarity inverted**



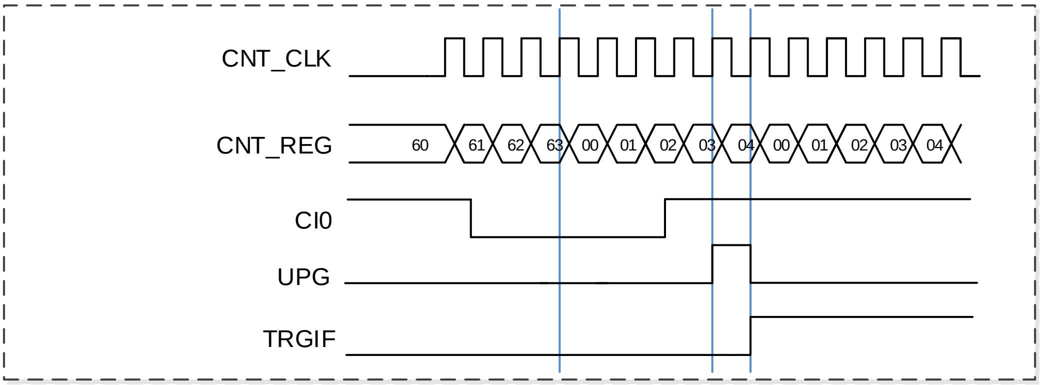
### Slave controller

The TIMERx can be synchronized with an external trigger in several modes including the Restart mode, the Pause mode and the event mode which is selected by the SMC field in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS field in the TIMERx_SMCFG register, below to CI0 signal as an example.The operation modes in the Slave Controller are described in the accompanying sections.

- Restart mode

The counter and its prescaler can be reinitialized in response to a rising edge of the CI0 signal. When a CI0 rising edge occurs, the update event software generation bit named UPG will automatically be asserted by hardware and the trigger event flag will also be set. Then the counter and prescaler will be reinitialized. Although the UPG bit is set to 1 by hardware, the update event does not really occur. It depends upon whether the update event disable control bit UPDIS is set to 1 or not. If UPDIS is set to 1 to disable the update event to occur, there will no update event will be generated, however the counter and prescaler are still reinitialized when the CI0 rising edge occurs. If the UPDIS bit in the TIMERx_CTL0 register is cleared to enable the update event to occur, an update event will be generated together with the CI0 rising edge, then all the preloaded registers will be updated.

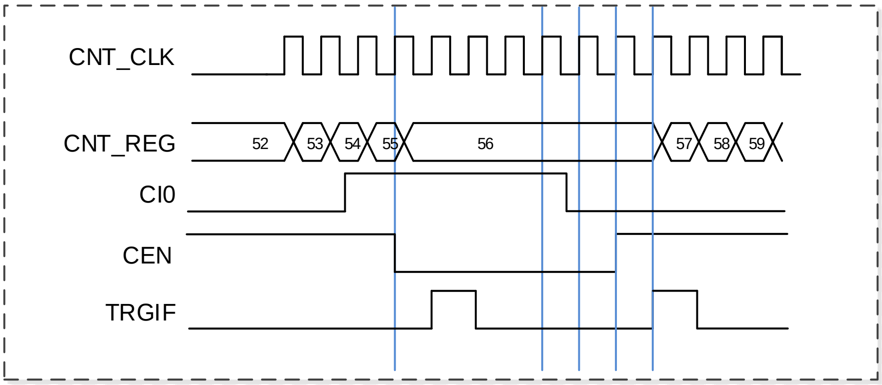
**Figure 9-76. Control circuit in restart mode**



- Pause mode

In the Pause Mode, the selected CI0 input signal level is used to control the counter start/stop operation. The counter starts to count when the selected CI0 signal is at a high level and stops counting when the CI0 signal is changed to a low level, here the counter will maintain its present value and will not be reset.

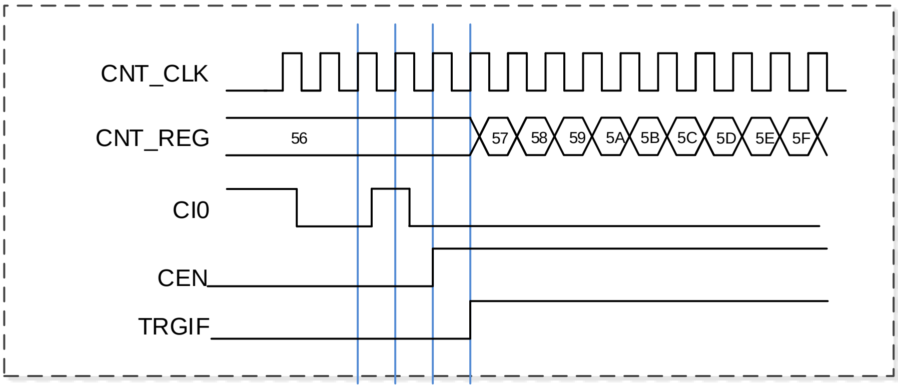
**Figure 9-77. Control circuit in pause mode**



- Event mode

After the counter is disabled to count, the counter can resume counting when a CI0 rising edge signal occurs. When an CI0 rising edge occurs, the counter will start to count from the current value in the counter. Note that the CI0 signal is only used to enable the counter to resume counting and has no effect on controlling the counter to stop counting.

**Figure 9-78. Control circuit in event mode**

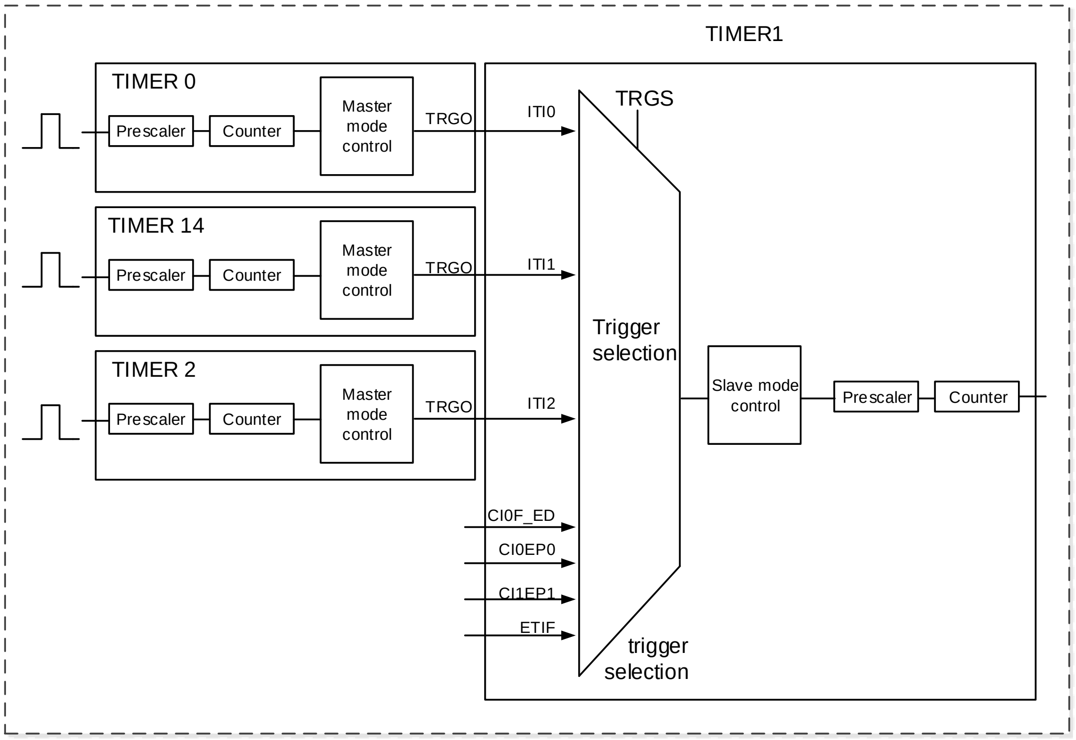


### Timer interconnection

The timers can be internally connected together for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the Master mode while configuring another timer to be in the Slave mode. The following figures present several examples of trigger selection for the master and slave modes.

Figure below shows the timer x trigger selection when it is configured in slave mode.

**Figure 9-79. Master/Slave mode timer example**



### Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.2.4. TIMER1/TIMER2 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:10 | 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 ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. |

| | | |
|---|---|---|
| | | 00: $f_{DTS}=f_{TIMER\_CK}$<br>01: $f_{DTS}=f_{TIMER\_CK}$ /2<br>10: $f_{DTS}= f_{TIMER\_CK}$ /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 up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting down.<br>10: Center-aligned and counting up assert mode. The counter counts up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting up.<br>11: Center-aligned and counting up/down assert mode. The counter counts up and down alternatively. Output compare interrupt flags of channels, which are configured in output mode (CHxMS=00 in TIMERx_CHCTLx register), are set only when the counter is counting both up and down.<br>The counter should be disabled when switches from edge-aligned mode to center-aligned mode. |
| 4 | DIR | Direction<br>0: Count up<br>1: Count down<br>This bit is read only when the timer is configured in Center-aligned mode or Encoder mode. |
| 3 | SPM | Single pulse mode.<br>0: Counter continues after update event.<br>1: The CEN is cleared by hardware and the counter stops at update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources.<br>0: When enabled, any of the following events generate an update interrupt or DMA request:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request. |

| | | |
|---|---|---|
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event. |
| 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 encoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000

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

| 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 | | | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset.<br>001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal TIMERx_EN as TRGO. The counter |

| | | |
|---|---|---|
| | | enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.<br>010: Update. In this mode the master mode controller selects the update event as TRGO.<br>011: Capture/compare pulse.In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred.<br>100: Compare.In this mode the master mode controller selects the O0CPRE signal is used as TRGO<br>101: Compare.In this mode the master mode controller selects the O1CPRE signal is used as TRGO<br>110: Compare.In this mode the master mode controller selects the O2CPRE signal is used as TRGO<br>111: Compare.In this mode the master mode controller selects the O3CPRE signal is used as TRGO |
| 3 | DMAS | DMA request source selection<br>0: DMA request of channel x is sent when capture/compare event occurs.<br>1: DMA request of channel x is sent when update event occurs. |
| 2:1 | Reserved | Must be kept at reset value. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal<br>0: ETI is active at high level or rising edge.<br>1: ETI is active at low level or falling edge. |
| 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>Setting the SMC1 bit has the same effect as selecting external clock mode 0 with TRGI connected to ETIF (SMC=111 and TRGS =111). |

| | | |
|---|---|---|
| | | It is possible to simultaneously use external clock mode 1 with the reset mode, pause mode or event mode. But the TRGS bits must not be 111 in this case.<br>The external clock input will be ETIF if external clock mode 0 and external clock mode 1 are enabled at the same time. |
| 13:12 | ETPSC[1:0] | External trigger prescaler<br>The frequency of external trigger signal ETI must not be at higher than 1/4 of TIMERx_CK frequency. When the external trigger signal is a fast clocks, the prescaler can be enabled to reduce ETI frequency.<br>00: Prescaler disable<br>01: ETI frequency will be divided by 2<br>10: ETI frequency will be divided by 4<br>11: ETI frequency will be divided by 8 |
| 11:8 | ETFC[3:0] | External trigger filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETI signal and the length of the digital filter applied to ETI.<br>0000: Filter disalble. $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}= f_{TIMER\_CK}$, N=2.<br>0010: $f_{SAMP}= f_{TIMER\_CK}$, N=4.<br>0011: $f_{SAMP}= f_{TIMER\_CK}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 7 | MSM | Master-slave mode<br>The effect of an event on the trigger input is delayed in this mode to allow a perfect synchronization between the current timer and its slaves through TRGO. If we want to synchronize several timers on a single external event, this mode can be used.<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: Internal trigger input 0 (ITI0) |

001: Internal trigger input 1 (ITI1)

010: Internal trigger input 2 (ITI2)

011: Internal trigger input 3 (ITI3)

100: CI0 edge flag (CI0F_ED)

101: Filtered channel 0 input (CI0FE0)

110: Filtered channel 1 Input (CI1FE1)

111: External trigger input (ETIF)

These bits must not be changed when slave mode is enabled.

| Slave TIMER | ITI0 (TRGS = 000) | ITI1 (TRGS = 001) | ITI2 (TRGS = 010) | ITI3 (TRGS = 011) |
|---|---|---|---|---|
| **TIMER1** | TIMER0 | TIMER14 | TIMER2 | Reserved |
| **TIMER2** | TIMER0 | TIMER1 | TIMER14 | Reserved |

3 OCRC OCPRE clear source selection

0: OCPRE_CLR_INT is connected to the OCPRE_CLR input

1: OCPRE_CLR_INT is connected to ETIF

2:0 SMC[2:0] Slave mode control

000: Disable mode.The prescaler is clocked directly by the internal clock when CEN bit is set high.

001: Quadrature decodermode 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: RestartMode.The counter is reinitialized and the shadow registers are updated 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 when it is low.

110: Event Mode.A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller.

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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | TRGDEN | Res | CH3DEN | CH2DEN | CH1DEN | CH0DEN | UPDEN | Res | TRGIE | Res | CH3IE | CH2IE | CH1IE | CH0IE | UPIE |
| | rw | | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Trigger DMA request disabled<br>1: Trigger DMA request enabled |
| 13 | Reserved | Must be kept at reset value |
| 12 | CH3DEN | Channel 3 Capture/Compare DMA request enable<br>0: Channel 3 Capture/Compare DMA request disabled<br>1: Channel 3 Capture/Compare DMA request enabled |
| 11 | CH2DEN | Channel 2 Capture/Compare DMA request enable<br>0: Channel 2Capture/Compare DMA request disabled<br>1: Channel 2 Capture/Compare DMA request enabled |
| 10 | CH1DEN | Channel 1 Capture/Compare DMA request enable<br>0: Channel 1Capture/Compare DMA request disabled<br>1: Channel 1 Capture/Compare DMA request enabled |
| 9 | CH0DEN | Channel 0 Capture/Compare DMA request enable<br>0: Channel 0 Capture/Compare DMA request disabled<br>1: Channel 0 Capture/Compare DMA request enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Update DMA request disabled<br>1: Update DMA request enabled |
| 7 | Reserved | Must be kept at reset value |
| 6 | TRGIE | Trigger interrupt enable<br>0: Trigger interrupt disabled<br>1: Trigger interrupt enabled |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3IE | Channel 3 Capture/Compare interrupt enable<br>0: Channel 3 Capture/Compare interrupt disabled<br>1: Channel 3 Capture/Compare interrupt enabled |
| 3 | CH2IE | Channel 2 Capture/Compare interrupt enable<br>0: Channel 2 Capture/Compare interrupt disabled<br>1: Channel 2 Capture/Compare interrupt enabled |
| 2 | CH1IE | Channel 1 Capture/Compare interrupt enable |

| | | |
|---|---|---|
| | | 0: Channel 1 Capture/Compare interrupt disabled<br>1: Channel 1 Capture/Compare interrupt enabled |
| 1 | CH0IE | Channel 0 Capture/Compare interrupt enable<br>0: Channel 0 Capture/Compare interrupt disabled<br>1: Channel 0 Capture/Compare interrupt enabled |
| 0 | UPIE | Update Capture/Compare interrupt enable<br>0: Update Capture/Compare interrupt disabled<br>1: Update Capture/Compare interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CH3OF | CH2OF | CH1OF | CH0OF | Reserved. | | TRGIF | Res. | 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 |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:13 | Reserved | Must be kept at reset value |
| 12 | CH3OF | Channel 3 Capture overflow flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 Capture overflow flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 Capture overflow flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 Capture overflow 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 Capture overflow interrupt occurred<br>1: Capture overflow interrupt occurred |
| 8:7 | Reserved | Must be kept at reset value |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on TRGI input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on TRGI input generates a trigger event.<br>0: No trigger event occurred |

| | | |
|---|---|---|
| | | 1: Trigger interrupt occurred |
| 5 | Reserved | Must be kept at reset value |
| 4 | CH3IF | Channel 3 Capture/Compare interrupt enable<br>Refer to CH0IF description |
| 3 | CH2IF | Channel 2 Capture/Compare interrupt enable<br>Refer to CH0IF description |
| 2 | CH1IF | Channel 1 Capture/Compare interrupt enable<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BRKG | TRGG | Res. | CH3G | CH2G | CH1G | CH0G | UPG |
| | | | | | | | | w | w | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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 | 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<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 0 is configured in input mode, the current value of the counter is captured in TIMER0_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 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 upcounting mode is selected, else (downcounting)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

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

| 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**

| | | |
|---|---|---|
| 15 | CH1COMCEN | Channel 1 output compare clear enable.<br>Refer to CH0COMCEN description. |
| 14:12 | CH1COMCTL[2:0] | Channel 1 output compare mode<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 OFF (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is configured as output<br>01: Channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: Channel 1 is configured as input, IC1 is connected to CI0FE1<br>11: Channel 1 is configured as input, IC1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.<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 behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.<br>000: Frozen.The O0CPRE signal keep stable, independent of the comparison between the output compare register TIMERx_CH0CV and the counter.<br>001: Set high on match.O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>010: Set low on match. O0CPRE signal is forced low when the counter matches the c output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter matches the c output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced low level.<br>101: Force high. O0CPRE is forced high level.<br>110: PWM mode 0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high.<br>111: PWM mode 1. When counting up, O0CPRE is low as long as the counter is |

| | | |
|---|---|---|
| | | smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low.<br>When configured in PWM mode, the OCPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison 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. |
| 3 | CH0COMSEN | Channel 0 output compare 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 validating the shadow register only in one pulse mode (SPM bit set in TIMERx_CTL0 register is set).<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 compare fast disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br>1: Channel 0 output compare fast enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles. |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 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>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is configured as output<br>01: Channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: Channel 1 is configured as input, IC1 is connected to CI0FE1<br>11: Channel 1 is configured as input, IC1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disable, $f_{SAMP}=f_{DTS}$, N=1<br>0001: $f_{SAMP}= f_{TIMER\_CK}$, N=2<br>0010: $f_{SAMP}= f_{TIMER\_CK}$, N=4<br>0011: $f_{SAMP}= f_{TIMER\_CK}$, N=8<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8 |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the ratio of the prescaler on channel 0 input.The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is reset.<br>00: prescaler disable, capture is done on each channel input edge<br>01: Capture is done every 2 channel input edges<br>10: Capture is done every 4 channel input edges<br>11: Capture is done every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in |

TIMERx_CHCTL2 register is reset).

00: Channel 0 is configured as output

01: Channel 0 is configured as input, IC0 is connected to CI0FE0

10: Channel 0 is configured as input, IC0 is connected to CI1FE0

11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 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 OFF (CH3EN bit in TIMERx_CHCTL2 register is reset).<br>00: channel 3 is configured as output<br>01: Channel 3 is configured as input, IC3 is connected to CI3FE3<br>10: Channel 3 is configured as input, IC3 is connected to CI2FE3<br>11: channel 3 is configured as input, IC3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable<br>Refer to CH0COMCEN description |

| | | |
|---|---|---|
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>Refer to CH0COMCTL description |
| 3 | CH2COMSEN | Channel 2 output compare shadow enable<br>Refer to CH0COMSEN description |
| 2 | CH2COMFEN | Channel 2 output compare fast enable<br>Refer to CH0COMFEN description |
| 1:0 | CH2MS[1:0] | Channel 2 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 OFF (CH2EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 2 is configured as output<br>01: Channel 2 is configured as input, IC2 is connected to CI2FE2<br>10: Channel 2 is configured as input, IC2 is connected to CI3FE2<br>11: Channel 2 is configured as input, IC2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 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>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH3EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 3 is configured as output<br>01: Channel 3 is configured as input, IC3 is connected to CI3FE3<br>10: Channel 3 is configured as input, IC3 is connected to CI2FE3<br>11: Channel 3 is configured as input, IC3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>Refer to CH0CAPFLT description |
| 3:2 | CH2CAPPSC[1:0] | Channel 2 input capture prescaler<br>Refer to CH0CAPPSC description |
| 1:0 | CH2MS[1:0] | Channel 2 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 OFF (CH2EN bit in TIMERx_CHCTL2 register is reset).

00: Channel 2 is configured as output

01: Channel 2 is configured as input, IC2 is connected to CI2FE2

10: Channel 2 is configured as input, IC2 is connected to CI3FE2

11: Channel 2 is configured as input, IC2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20
Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3NP | Res. | CH3P | CH3EN | CH2NP | Res. | CH2P | CH2EN | CH1NP | Res. | CH1P | CH1EN | CH0NP | Res. | CH0P | CH0EN |
| rw |  | rw | rw | rw |  | rw | rw | rw |  | rw | rw | rw |  | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | CH3NP | Channel 3 complementary output polarity<br>Refer to CH0NP description |
| 14 | Reserved | Must be kept at reset value |
| 13 | CH3P | Channel 3 polarity<br>Refer to CH0P description |
| 12 | CH3EN | Channel 3 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 polarity<br>Refer to CH0P description |
| 8 | CH2EN | Channel 2 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 polarity |

| | | |
|---|---|---|
| | | Refer to CH0P description |
| 4 | CH1EN | Channel 1 enable<br>Refer to CH0EN description |
| 3 | CH0NP | Channel 0 complementary output polarity<br>**Channel 0 configured as output**<br>CH0NP need to be set to 0.<br>**Channel 0 configured as input**<br>In conjunction with CH0P, this bit is used to define the polarity of CI0. |
| 2 | Reserved | Must be kept at reset value |
| 1 | CH0P | Channel 0 polarity<br>**When channel 0 is configured in output mode, this bit specifies the output signal polarity.**<br>0: Channel 0 active high<br>1: Channel 0 active low<br>**When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.**<br>[CH0NP, CH0P] will selete 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 CIxFE1 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 CIxFE1 will be inverted.<br>[CH0NP==1, CH0P==0]: Reserved.<br>[CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted.<br>This bit cannot be modified when PROT[1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 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 channel 0.<br>0: Channel 0 disabled.<br>1: Channel 0 enabled. |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| CNT[31:16] TIMER1 ONLY |
|---|
| rw |

| 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 | CNT[31:16] | This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter. Only TIMER1 has this high 16bit counter 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAR [31:16] TIMER1 ONLY | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CAR [31:16] | Counter auto reload value. Only TIMER1 has this high 16bit counter value. |

| | | |
|---|---|---|
| 15:0 | CAR[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL [31:16] TIMER1 ONLY | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 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 | CH0VAL<br>[31:16] | Capture or compare value of channel 0. Only TIMER1 has this high 16bit counter value. |
| 15:0 | CH0VAL<br>[15:0] | Capture or compare value of channel 0<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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2VAL [31:16] TIMER1 ONLY | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 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 | CH1VAL<br>[31:16] | Capture or compare value of channel 1. Only TIMER1 has this high 16bit counter value. |
| 15:0 | CH1VAL<br>[15:0] | Capture or compare value of channel 1<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.
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

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">CH2VAL [31:16] TIMER1 ONLY</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CH2VAL [15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH2VAL[31:16] | Capture or compare value of channel 2. Only TIMER1 has this high 16bit counter 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

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">CH3VAL [31:16] TIMER1 ONLY</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CH3VAL [15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH3VAL [31:16] | Capture or compare value of channel 3. Only TIMER1 has this high 16bit counter value. |

| | | |
|---|---|---|
| 15:0 | CH3VAL<br>[15:0] | Capture or compare value of channel 3<br>When channel 3 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

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

| 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 |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value |
| 12:8 | DMATC[4:0] | DMA transfer count<br>When register access are done through the TIMERx_DMATB address, this 5-bit bit-field specifies the number of transfers. |
| 7:5 | Reserved | Must be kept at reset value |
| 4:0 | DMATA[4:0] | DMA transfer access start address<br>When register access are done through the TIMERx_DMATB address, this bit-field specifies the offset of the starting address from the TIMERx_CTL0 register. |

### DMA Transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C
Reset value: 0x0000

This register can be accessed by half-word(16-bit) or 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 |
|---|---|---|
| 15:0 | DMATB[15:0] | DMA transfer<br>When a read or write operation is assigned to this register, the register located at the address range (DMATA + burst counter) x 4 from TIMERx_CTL0 will be accessed. |

The burst counter is calculated by hardware, and ranges from 0 to DMATC.

### Configuration register (TIMERx_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value |
| 1 | CCSEL | Write Capture/Compare register selection<br>This bit-field set and reset by software.<br>1: If write the Capture/Compare register, the write value is same as the Capture/Compare value, the write access ignored<br>0: No effect |
| 0 | Reserved | Must be kept at reset value. |

## 9.3. Basic timer (TIMER5)

### 9.3.1. Introduction

The general timers (TIMER5) consist of one 16-bit counter auto reload register (TIMERx_CAR) and several control registers. It can be used for general timer, and it is also used by DAC(Digital to analog converter). TIMER5's TRGO is connected to DAC which can be drived by this trigger.

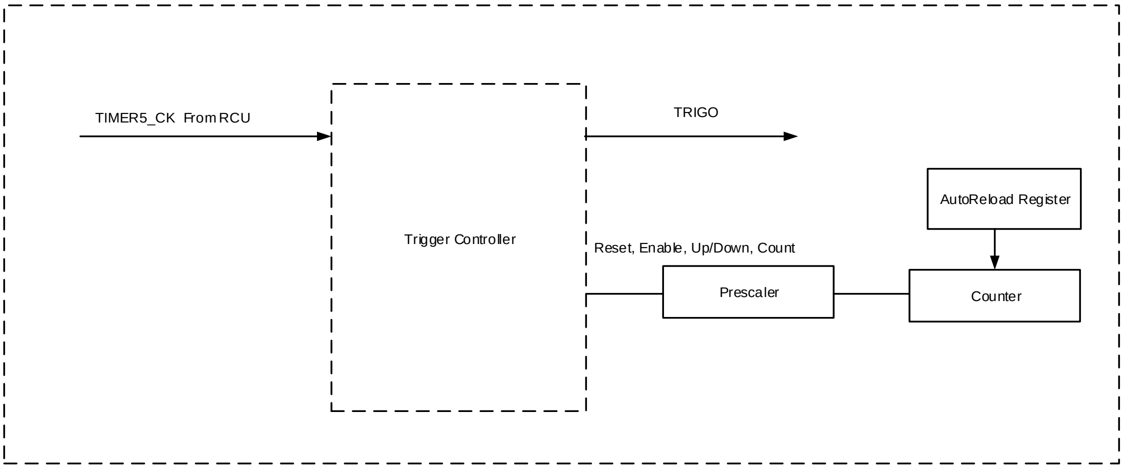
### 9.3.2. Main features

- 16-bit up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generated by counter overflow.

### 9.3.3. Function description

Figure below provides details on the internal configuration of the Basic timer.

**Figure 9-80. General timer block diagram (TIMER5)**



#### Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx_PSC) which can be changed on the fly but be taken into account at the next update event.

**Figure 9-81. Counter timing diagram with prescaler division change from 1 to 2**

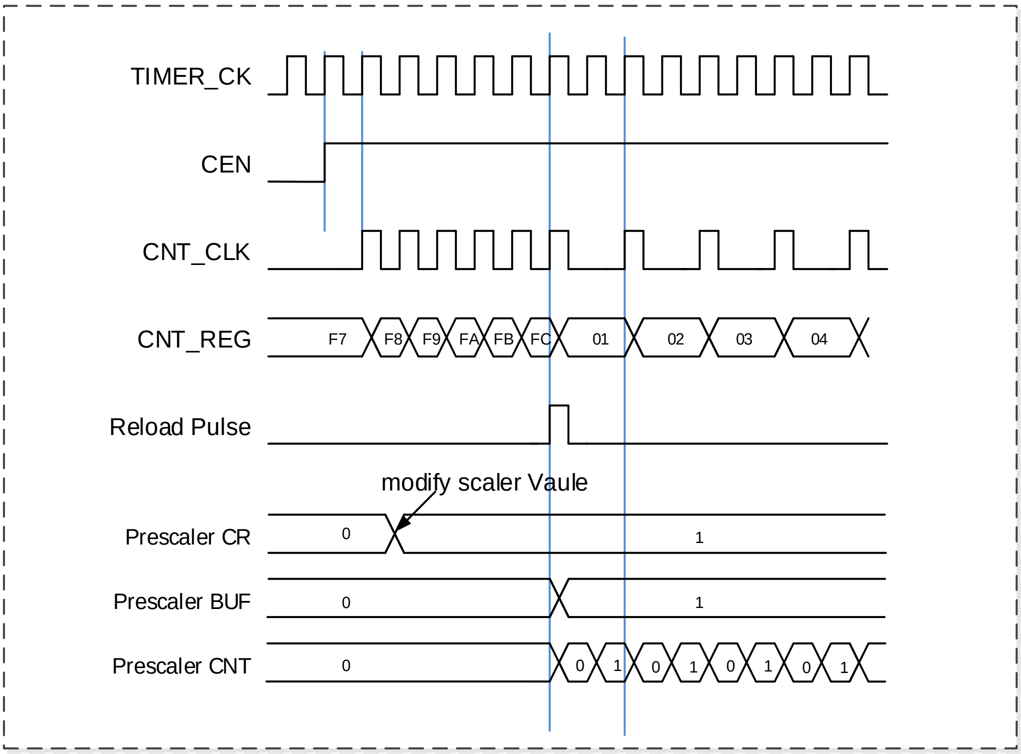
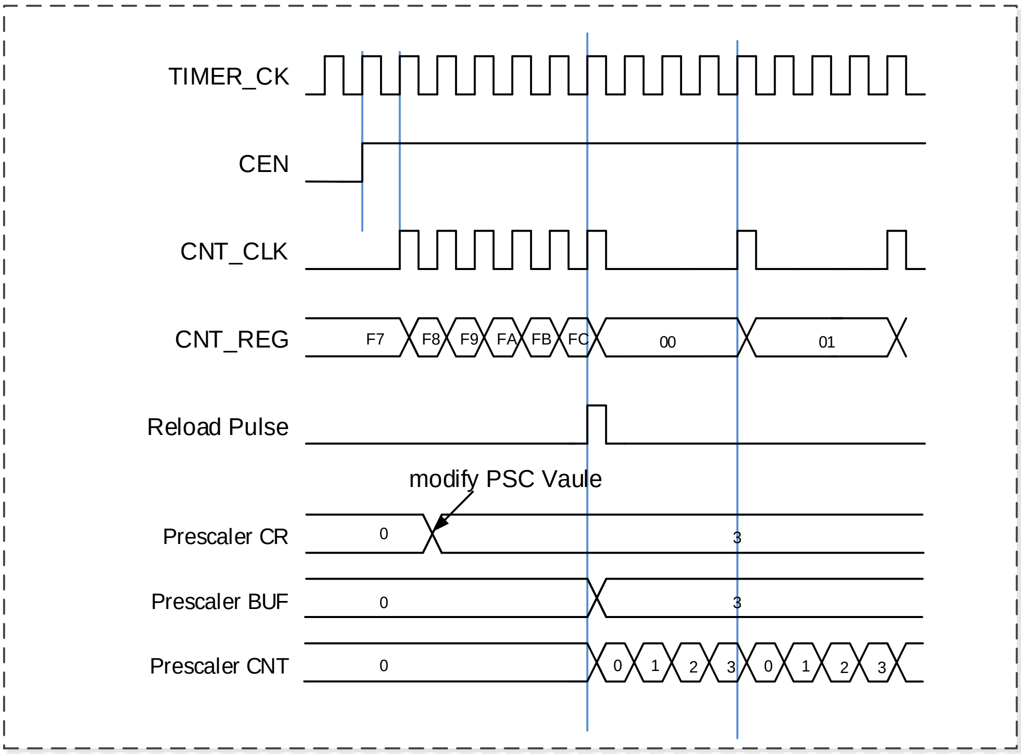


**Figure 9-82. Counter timing diagram with prescaler division change from 1 to 4**



## Upcounting mode

In this mode the counter counts 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 to count once again from 0. The update event is generated at each counter overflow.

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.

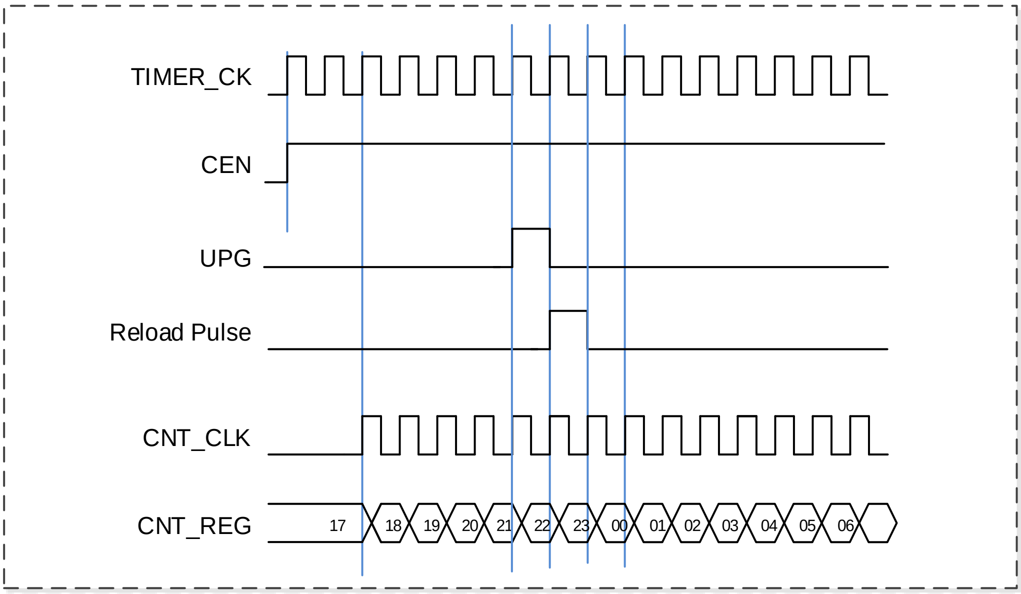
If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (autoreload register, prescaler register) are updated.

### Clock selection

Basic timer has the unique clock source which is controlled by RCU. Counter and prescaler counter are clocked by this internal clock TIMER_CK, except UPG is asserted. UPG's assert will initial counter and prescaler counter.

**Figure 9-83. Control circuit in normal mode, internal clock divided by 1**



### Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.3.4. TIMER5 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | ARSE | Reserved | | | SPM | UPS | UPDIS | CEN |
| | | | | | | | | rw | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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 TIMERx_ CAR register is enabled |
| 3 | SPM | Single pulse mode.<br>0: Counter continues after update event.<br>1: The CEN is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: When enabled, any of the following events generate an update interrupt or DMA request:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request. |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event. |
| 0 | CEN | Counter enable |

0: Counter disable

1: Counter enable

The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | 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: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset.<br>001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal TIMERx_EN as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.<br>010: Update. In this mode the master mode controller selects the update event as TRGO.<br>011: Capture/compare pulse.In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred.<br>100: Compare.In this mode the master mode controller selects the O0CPRE signal is used as TRGO<br>101: Compare.In this mode the master mode controller selects the O1CPRE signal is used as TRGO<br>110: Compare.In this mode the master mode controller selects the O2CPRE signal is used as TRGO<br>111: Compare.In this mode the master mode controller selects the O3CPRE signal is used as TRGO |

| | | |
|---|---|---|
| 3:0 | Reserved | Must be kept at reset value |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:9 | Reserved | Must be kept at reset value |
| 8 | UPDEN | Update DMA request enable<br>0: Update DMA request disabled<br>1: Update DMA request enabled |
| 7:1 | Reserved | Must be kept at reset value |
| 0 | UPIE | Update interrupt enable<br>0: Update interrupt disabled<br>1: Update interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | UPIF |
| | | | | | | | | | | | | | | | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:1 | 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:1 | 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 if the center-aligned or upcounting mode is selected, else (downcounting) 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 |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CAR[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter. |

# 9.4. General timer (TIMER13)

## 9.4.1. Introduction

The general timer (TIMER13) consists of one 16-bit up -counter; one capture/compare register (TIMERx_CHxCV), one counter auto reload register (TIMERx_CAR) and several control registers. They can be used for a variety of purposes including general timer, input signal pulse width measurement or output waveform generation such as output compare or PWM output.

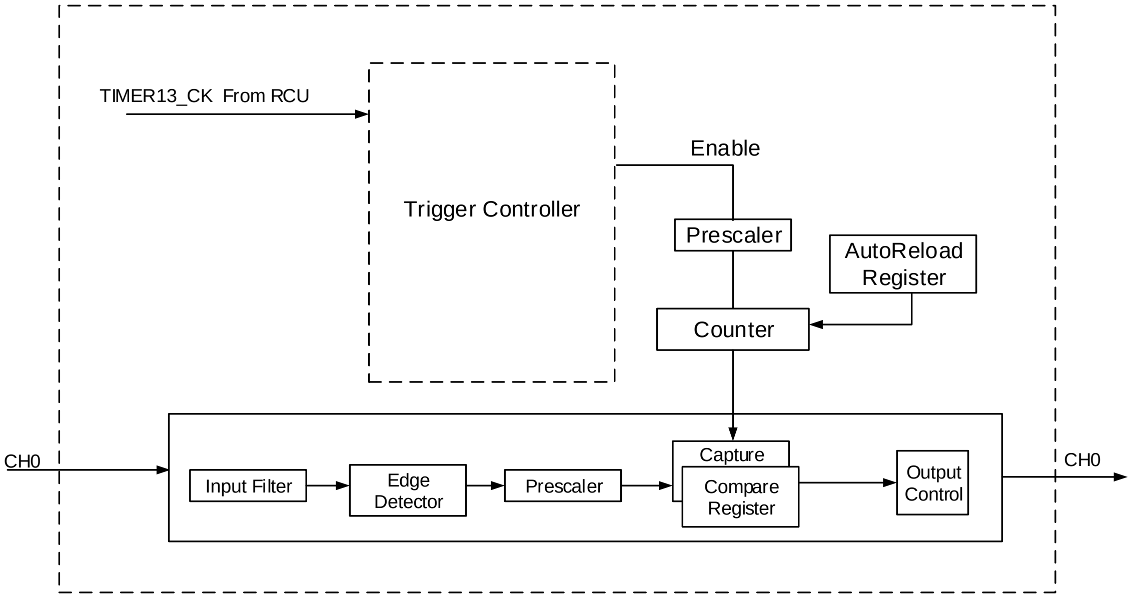
## 9.4.2. Main features

- 16-bit up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- One independent channel supports functions including input capture, compare match output, and generation of PWM waveform (edge and center-aligned Mode).
- Interrupt/DMA generation by update, input capture event, or output compare match event.

## 9.4.3. Function description

Figure below provides details on the internal configuration of the generic timer.

**Figure 9-84. General timer block diagram (TIMER13)**



### Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx_PSC) which can be changed on the fly but be taken into account at the next update event.

**Figure 9-85. Counter timing diagram with prescaler division change from 1 to 2**

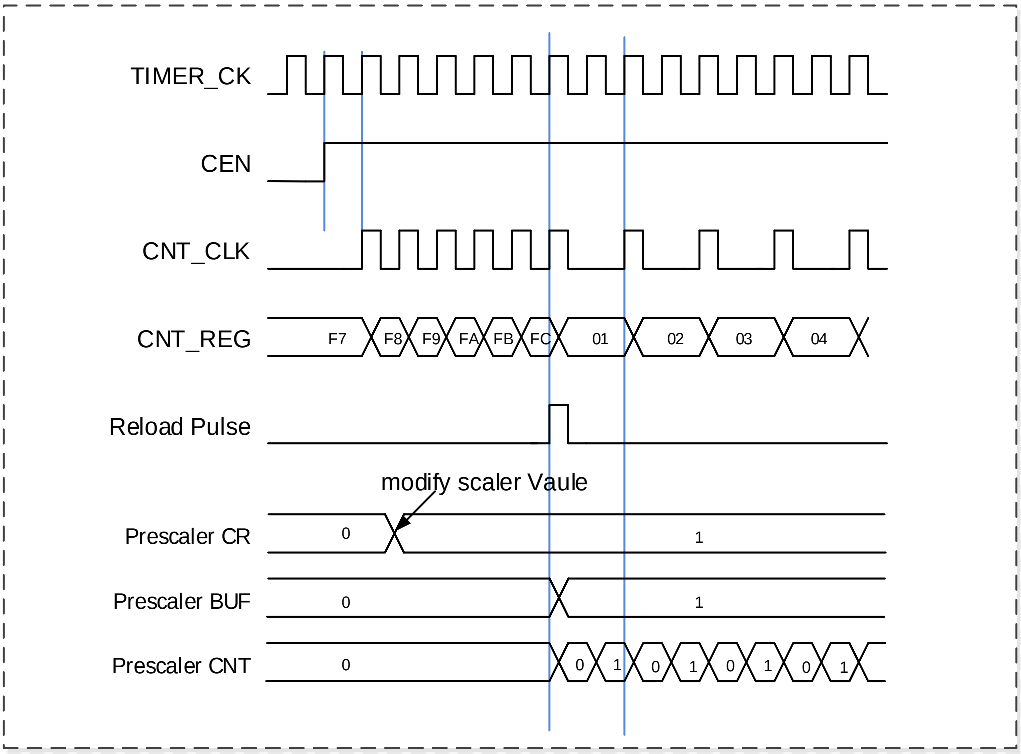
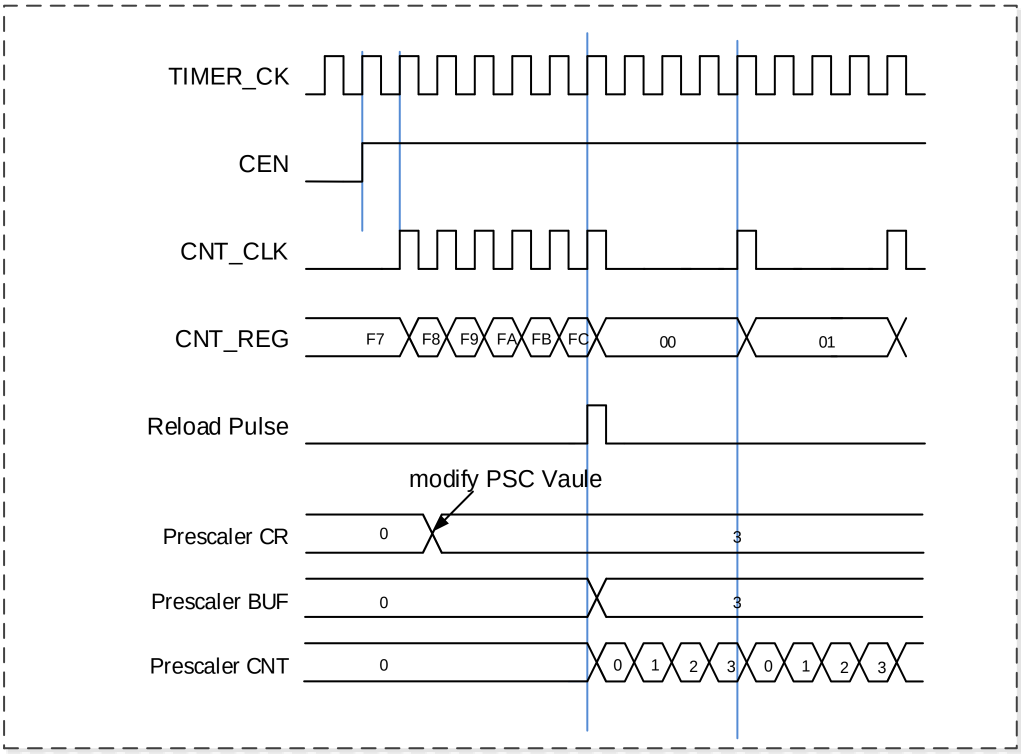


**Figure 9-86. Counter timing diagram with prescaler division change from 1 to 4**



### Upcounting mode

In this mode the counter counts continuously from 0 to the counter-reload value. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow.

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 set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

**Figure 9-87. Counter timing diagram, internal clock divided by 1**

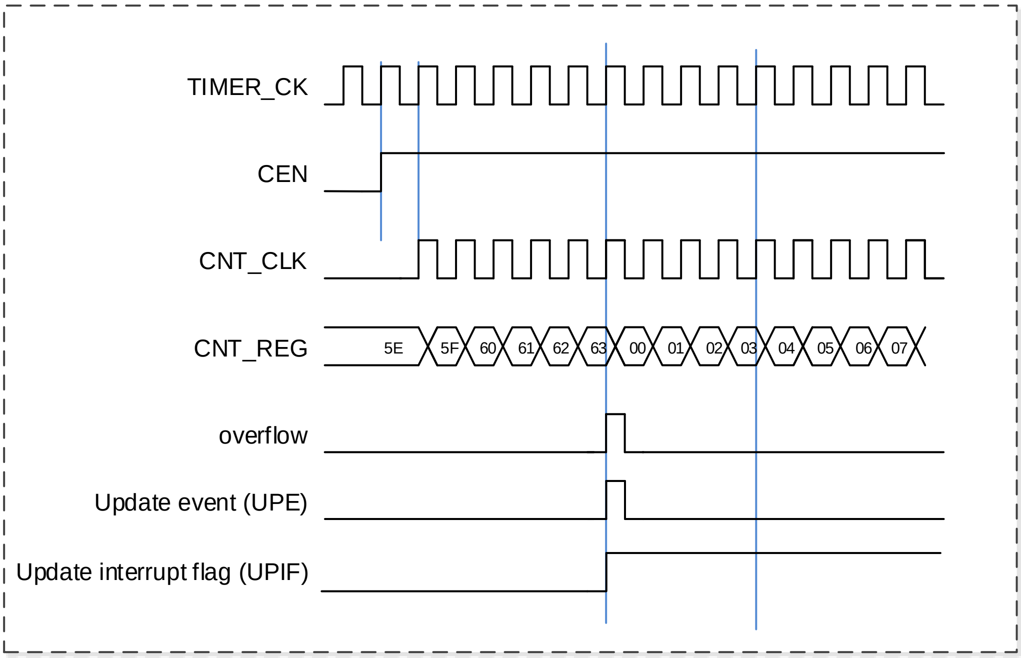


Figure 9-88. Counter timing diagram, internal clock divided by 2

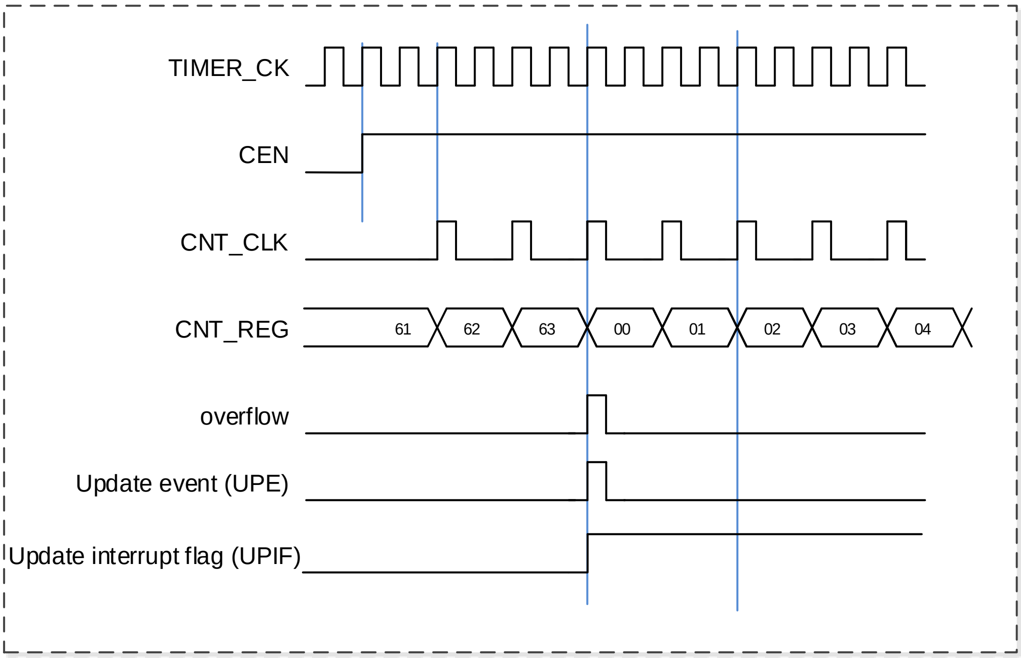


Figure 9-89. Counter timing diagram, internal clock divided by 4

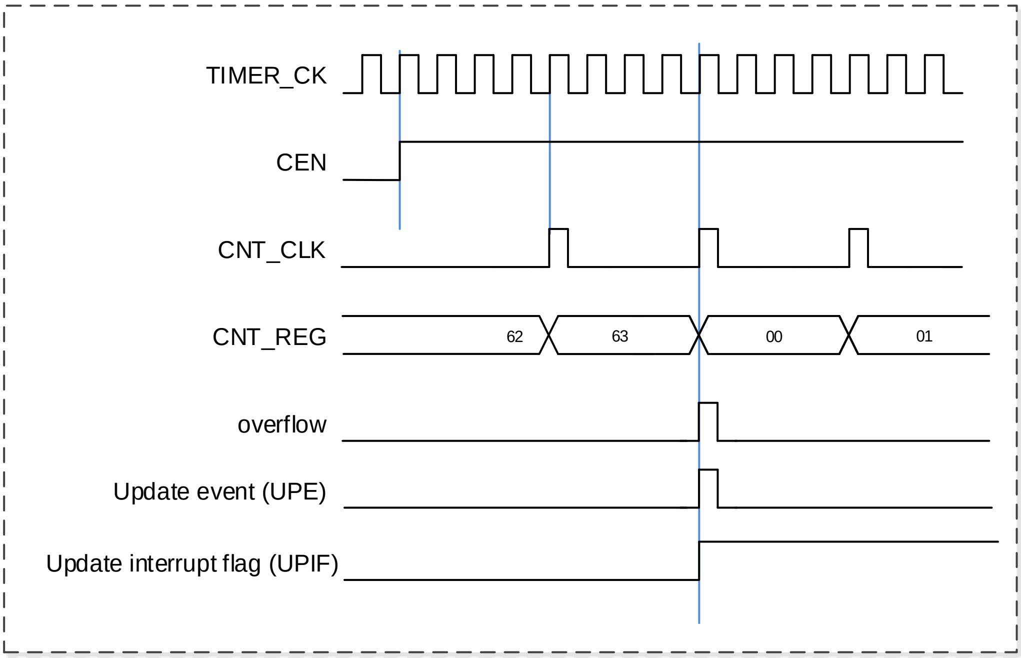


**Figure 9-90. Counter timing diagram, internal clock divided by N**

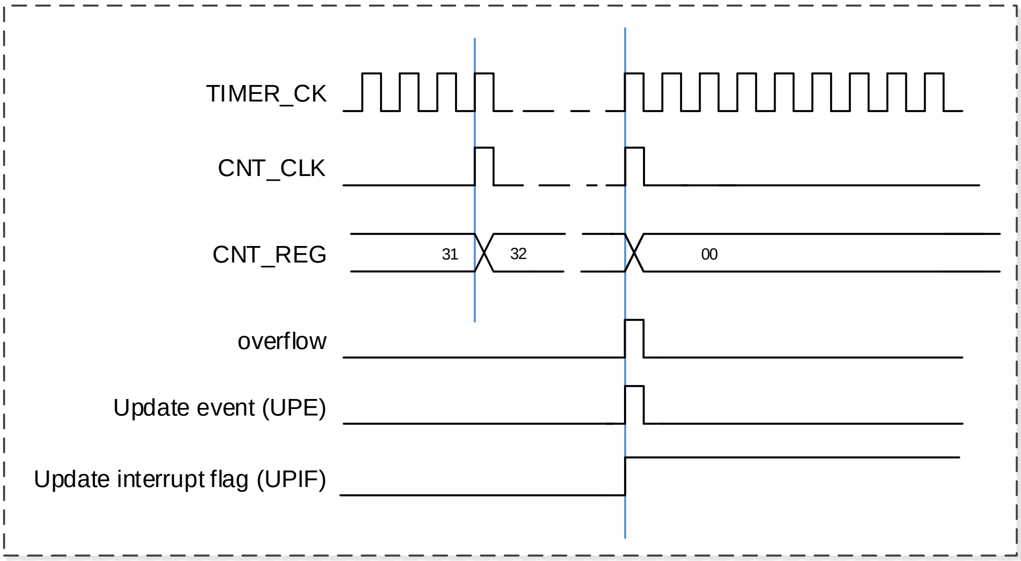


**Figure 9-91. Counter timing diagram, update event when ARSE=0**

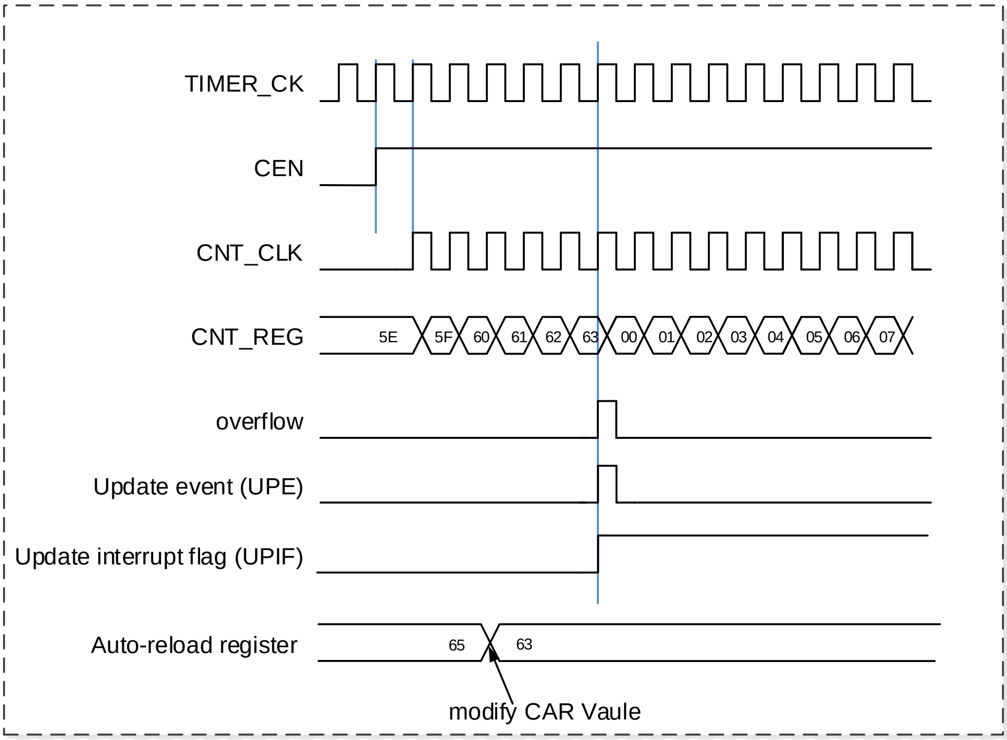
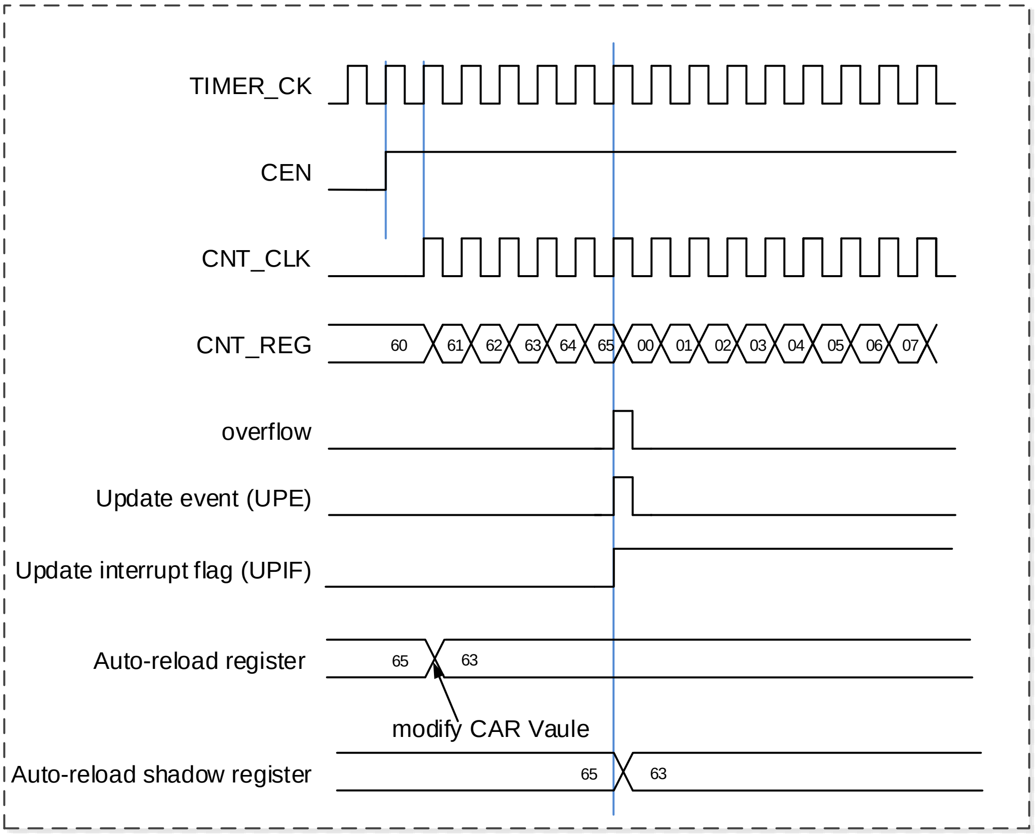


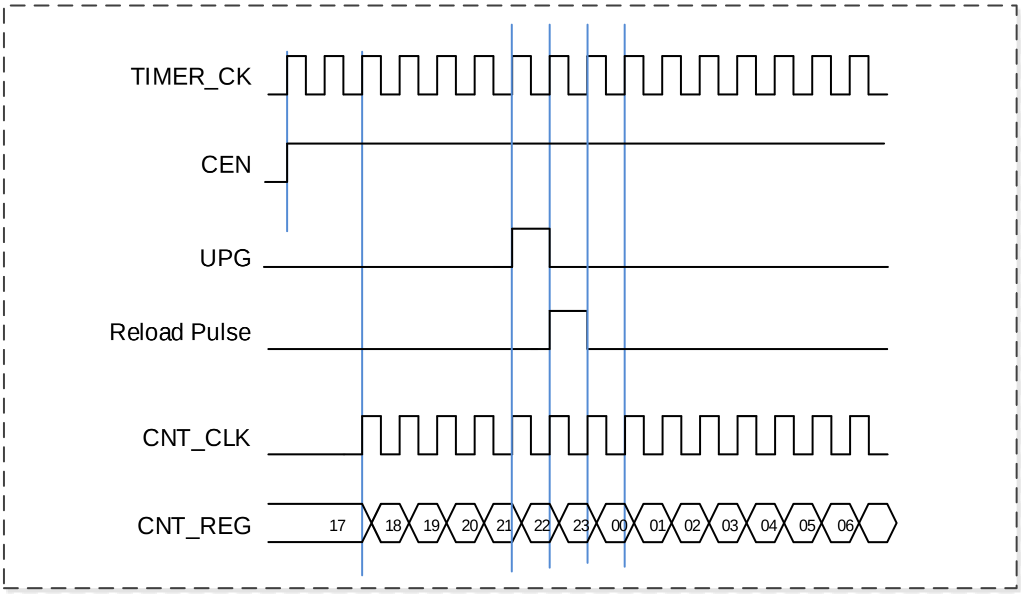
**Figure 9-92. Counter timing diagram, update event when ARSE=1**



### Clock selection

Basic timer has the unique clock source which is controlled by RCU. Counter and prescaler counter are clocked by this internal clock, except UPG is asserted. UPG's assert will initial counter and prescaler counter.

Figure 9-93. Control circuit in normal mode, internal clock divided by 1



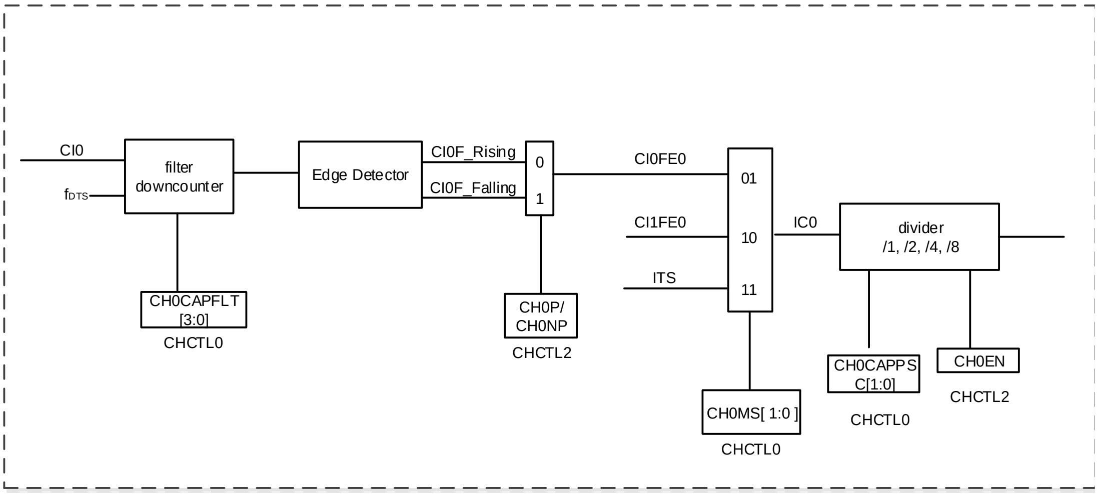
## Capture/compare channels

The TIMER13 has one independent channel which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture/compare value register including an input stage, channel controller and an output stage.

- Input capture stage

The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. The channel input signal (CIx) can be chosen to come from the TIMERx_CH0 signal. The channel input signal (CIx) is sampled by a digital filter to generate a filtered input signal CIxF. Then the channel polarity and the edge detection block can generate a CIxFEx signal for the input capture function. The effective input event number can be set by the channel input prescaler (CHxCAPPSC).

Figure 9-94. Capture/compare channel (example: input stage)



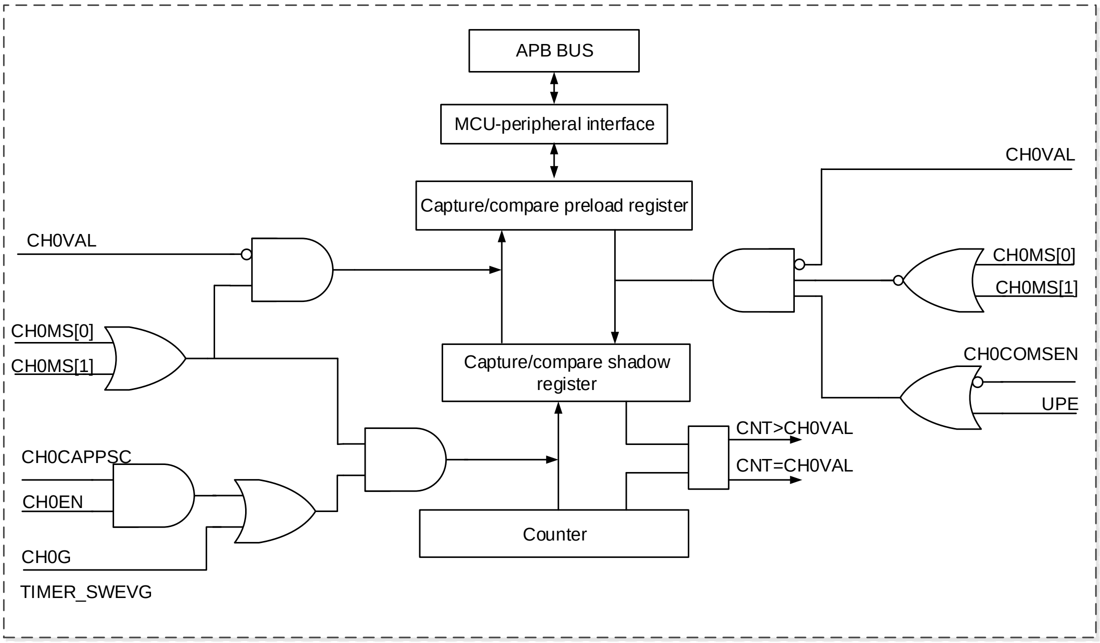
- Channel controller

The GPTM has one independent channels which can be used as capture inputs or compare match outputs.

When used in the input capture mode, the counter value is captured into the TIMERx_CHxCV shadow register first and then transferred into the TIMERx_CHxCV preload register when the capture event occurs.

When used in the compare match output mode, the contents of the TIMERx_CHxCV preload register is copied into the associated shadow register; the counter value is then compared with the register value.

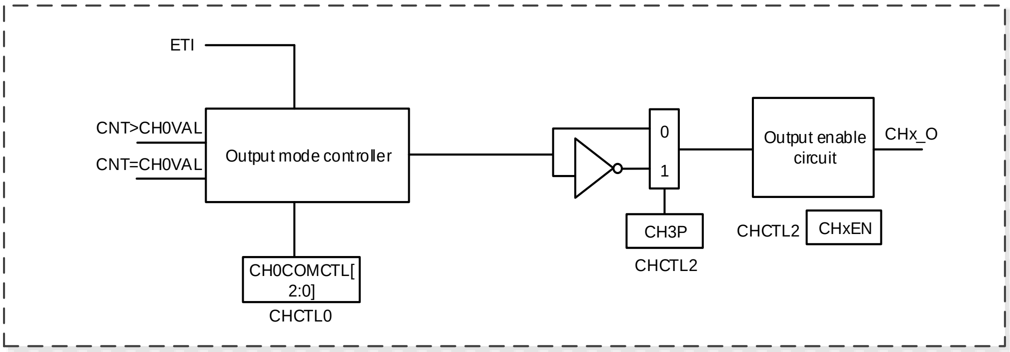
**Figure 9-95. Capture/compare channel 0 main circuit**



- Output stage

The TIMER13 has one channel for compare match or PWM output function.

**Figure 9-96. Output stage of capture/compare channel**



## Input Capture Mode

When the channel is used as a capture input, the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV) when an effective input signal transition occurs. Once the capture event occurs, the CHxIF flag in the TIMERx_INTF register is set. If the CHxIF bit is already set, i.e., the flag has not yet been cleared by software, and another capture event on this channel occurs, the corresponding channel Over-Capture flag, named CHxOF, will be set.Once the capture event occurs, an interrupt is generated depending on the CHxIE bit. The fllowing step maybe used to implement input capture mode:

- Select input signal by set CHxMS bits in the channel control register (TIMERx_CHCTLx).
- Add input filter to input signal by set CHxCAPFLT bitst in the the channel control register

(TIMERx_CHCTLx).

- Select input capture edge (rising or falling) by set CHxP/CHxNP bit in the channel control register 2 (TIMERx_CHCTL2).
- If input signal need prescaler, set CHxCAPPSC bits in the the channel control register (TIMERx_CHCTLx).
- If need interrupt, set CHxIE bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- Enable the channel by set CHxEN bit in the channel control register 2 (TIMERx_CHCTL2) to 1.

After configure this registers, the input capture event on this channel may occurs at the corresponding edge. And the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV). An interrupt generated if CHxIE bit set.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins (CIx). For example, PWM signal connect to CI0 input. Select channel 0 capture signal to CI0 by setting CH0MS to 01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 10 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.

### Output Compare Mode/PWM Mode

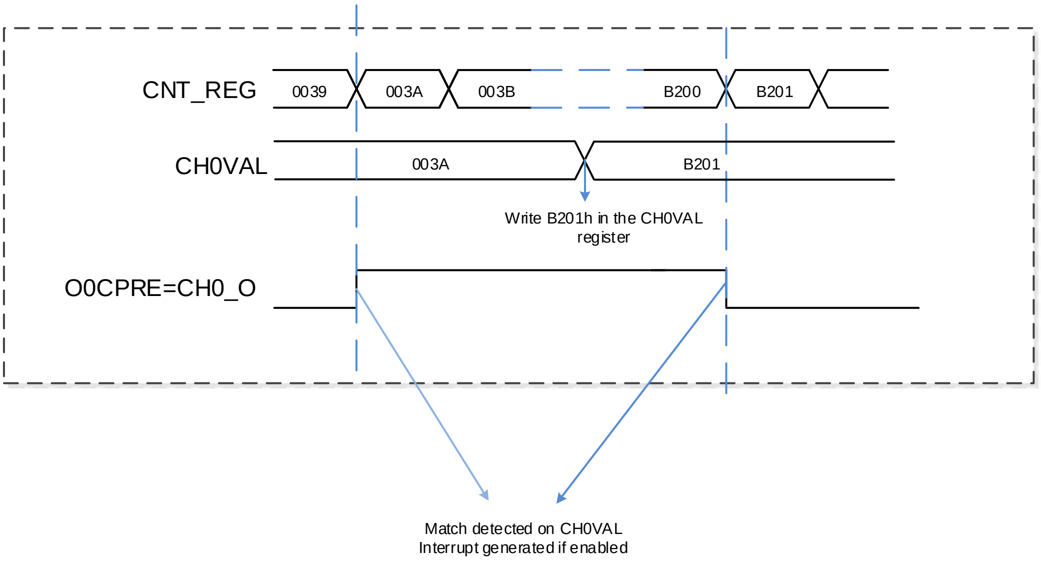
When the channel is used as an output mode by setting the CHxMS in the the channel control register (TIMERx_CHCTLx) to 00, the channel outputs waveform according to the CHxCOMCTL bits in the the the channel control register (TIMERx_CHCTLx), and the comparision between the counter and TIMERx_CHxCV registers. When the comparision equals, the CHxIF flag in the Interrupt flag registers (TIMERx_INTF) set to 1. An interrupt is generated depending on the CHxIE bit.

In output mode, the channel can outputs active level by set the CHxCOMCTL to 101, and outputs inactive level by set the CHxCOMCTL bits to 100.

In ouput mode, the channel output set to active level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 001. The channel output set to inactive level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 010.

In output compare toggle mode (by set the CHxCOMCTL to 011), the channel output toggles when the comparision between the counter and TIMERx_CHxCV registers match.

**Figure 9-97. Output compare toggle mode, toggle on CH0_O**



In the output PWM mode (by setting the CHxCOMCTL bits to 110 (PWM mode 0) or to 111 (PWM mode 1)), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

**Figure 9-98. Output compare PWM mode 0 on CH0_O, upcounting mode**

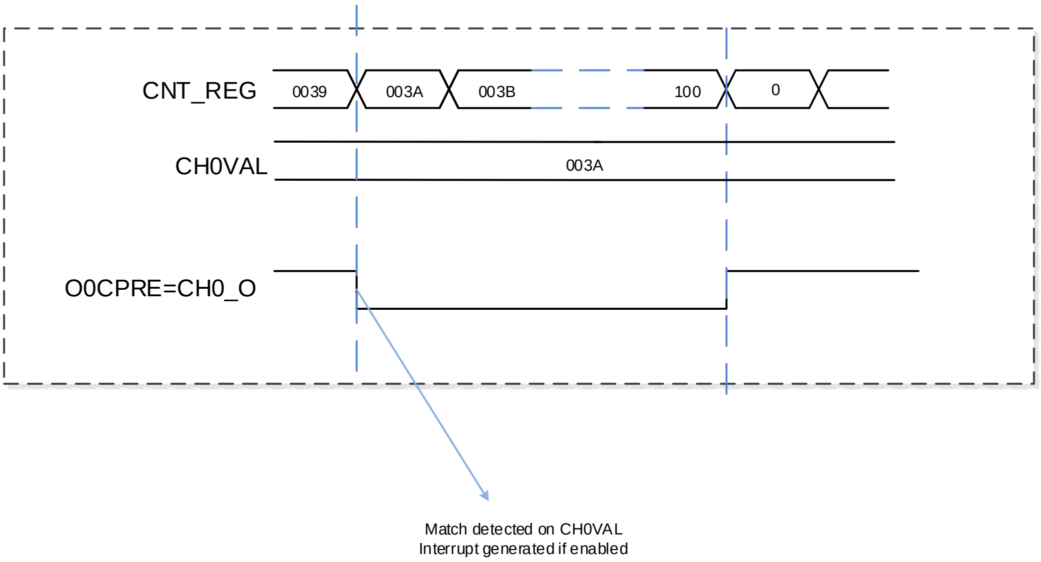
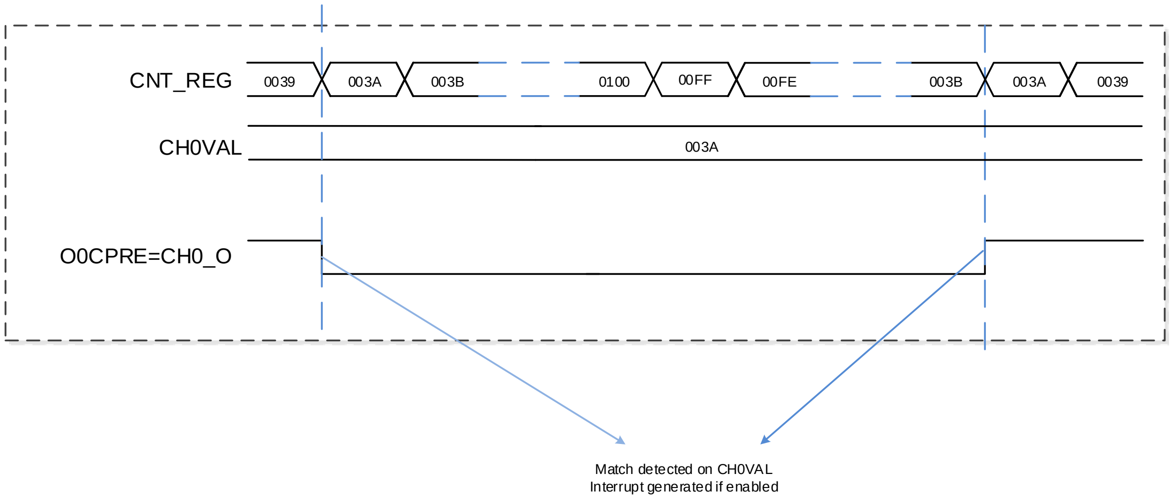


**Figure 9-99. Output compare PWM mode 0 on CH0_O, center-aligned counting mode**



### Channel Output Reference Signal

When the TIMER13 is used in the compare match output mode, the OxCPRE signal (Channel x Output Reference signal) is defined by setting the CHxCOMCTL bits. 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 detailed 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.

The OxCPRE signal can be forced to 0 when the ETIF signal is derived from the external TIMERx_ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.4.4. TIMER13 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CKDIV[1:0] | | ARSE | Reserved | | | | UPS | UPDIS | CEN |
| | | | | | | rw | | rw | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | 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 ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.<br>00: $f_{DTS}=f_{TIMER\_CK}$<br>01: $f_{DTS}= f_{TIMER\_CK}$ /2<br>10: $f_{DTS}= f_{TIMER\_CK}$ /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: When enabled, any of the following events generate an update interrupt:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: When enabled, only counter overflow/underflow generates an update interrupt. |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation.<br>0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event. |

| | | |
|---|---|---|
| | | 1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event. |
| 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 encoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | CH0IE | UPIE |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IE | Channel 0 interrupt enable<br>0: Channel 0 interrupt disabled<br>1: Channel 0 interrupt enabled |
| 0 | UPIE | Update interrupt enable<br>0: Update interrupt disabled<br>1: Update interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:10 | Reserved | Must be kept at reset value. |
| 9 | CH0OF | Channel 0 Capture overflow flag |

| | | |
|---|---|---|
| | | 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 Capture overflow interrupt occurred<br>1: Capture overflow interrupt occurred |
| 8:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IF | Channel 0 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | CH0G | UPG |
| | | | | | | | | | | | | | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | 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 0 is configured in input mode, the current value of the counter is captured in TIMER0_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 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 upcounting mode is selected, else (downcounting) it takes the auto-reload value. The prescaler counter |

is cleared at the same time.

0: No generate an update event

1: Generate an update event

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| | | | | | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.<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 behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.<br>000: Frozen.The O0CPRE signal keep stable, independent of the comparison between the output compare register TIMERx_CH0CV and the counter.<br>001: Set high on match.O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>010: Set low on match. O0CPRE signal is forced low when the counter matches the c output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter matches the c output compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced low level.<br>101: Force high. O0CPRE is forced high level.<br>110: PWM mode 0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high.<br>111: PWM mode 1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low. |

| | | |
|---|---|---|
| | | When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes. |
| 3 | CH0COMSEN | Channel 0 output compare 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 validating the shadow register only in one pulse mode (SPM bit set in TIMERx_CTL0 register is set). |
| 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 compare fast disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br>1: Channel 0 output compare fast enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles. |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disable, $f_{SAMP}$=$f_{DTS}$, N=1<br>0001: $f_{SAMP}$=$f_{TIMER\_CK}$, N=2<br>0010: $f_{SAMP}$= $f_{TIMER\_CK}$, N=4 |

| | | |
|---|---|---|
| | | 0011: $f_{SAMP}$= $f_{TIMER\_CK}$, N=8<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8 |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the ratio of the prescaler on channel 0 input.The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is reset.<br>00: Prescaler disable, capture is done on each channel input edge<br>01: Capture is done every 2 channel input edges<br>10: Capture is done every 4channel input edges<br>11: Capture is done every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CH0NP | Res. | CH0P | CH0EN |
| | | | | | | | | | | | | rw | | rw | rw |

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

| | | |
|---|---|---|
| 15:4 | 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 |
| 2 | Reserved | Must be kept at reset value |
| 1 | CH0P | Channel 0 polarity<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>This bit cannot be modified when PROT[1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 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 channel 0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CAR[15:0] | Counter auto reload value<br>This bit-filed specifies the auto reload value of the counter. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel 0<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

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

| 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 |
|---|---|---|
| 15:2 | 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(in GD32F130xx and GD32F150xx devices )/CKOUT0SEL(in GD32F170xx and GD32F190xx devices), which is controlled by RCU_CFG0. |

### Configuration register (TIMERx_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC
Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value |
| 1 | CCSEL | Write Capture/Compare register selection<br>This bit-field set and reset by software.<br>1: If write the Capture/Compare register, the write value is same as the Capture/Compare value, the write access ignored<br>0: No effect |
| 0 | Reserved | Must be kept at reset value |

# 9.5. General timer (TIMER14)

## 9.5.1. Introduction

The general timer, known as TIMER14, may be used for a variety of purposes. It consists of one 16-bit up-counter; two 16-bit capture/compare registers (TIMERx_CHxCV), one 16-bit counter auto reload register (TIMERx_CAR) and several control registers. They can be used for a variety of purposes including general timer, input signal pulse width measurement or output waveform generation such as single pulse generation or PWM output.

The general (TIMER14) timers are completely independent. They do not share any resources but can be synchronized together.

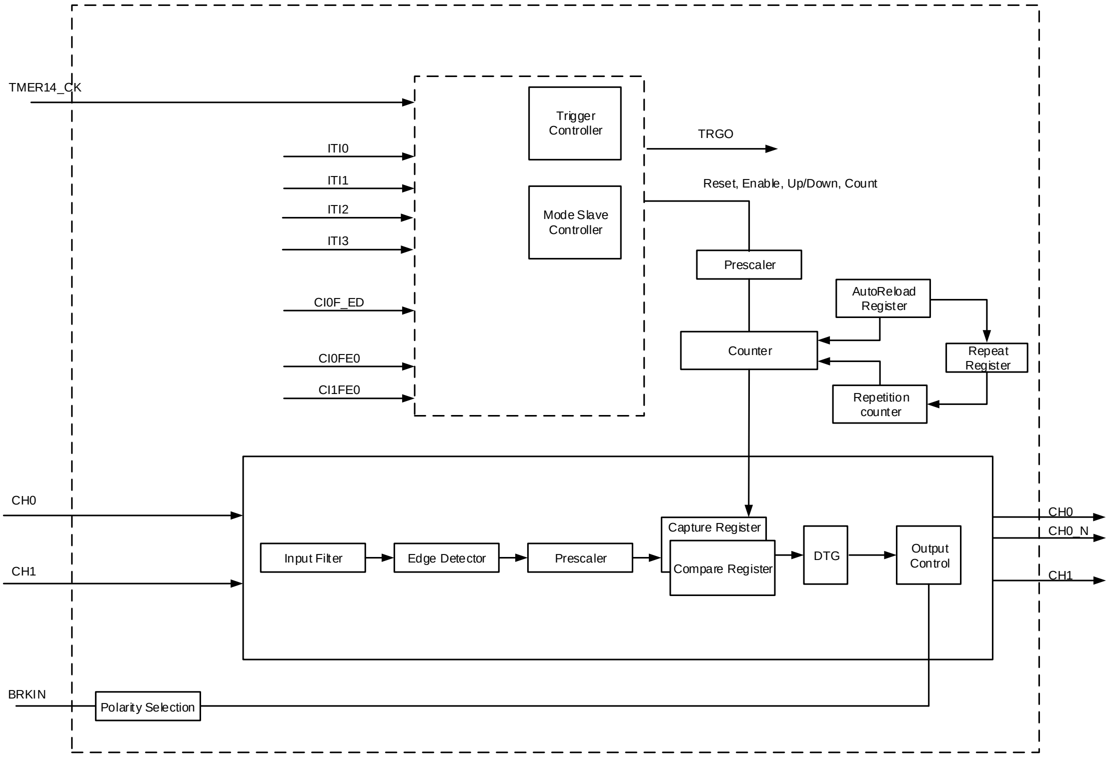
## 9.5.2. Main features

- 16-bit up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- Up to 2 independent channels support functions including input capture, compare match output, generation of PWM waveform (edge and center-aligned Mode), and single pulse mode output.
- Counter repetition to update the timer registers only after a given number of cycles of the counter.
- Break input to trigger the timer's output signals to a known state.
- Interrupt/DMA generation by update, trigger event, input capture event, output compare match event or break input
- Programmable dead-time for output match.
- Synchronization circuit to control TIMER14 with external signals or to interconnect several timers together.
- TIMER14 Master/Slave mode controller

## 9.5.3. Function description

Figure below provides details on the internal configuration of the advanced timer.

**Figure 9-100. Gereral timer block diagram (TIMER14)**



## Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx_PSC) which can be changed on the fly but be taken into account at the next update event.

Figure 9-101. Counter timing diagram with prescaler division change from 1 to 2

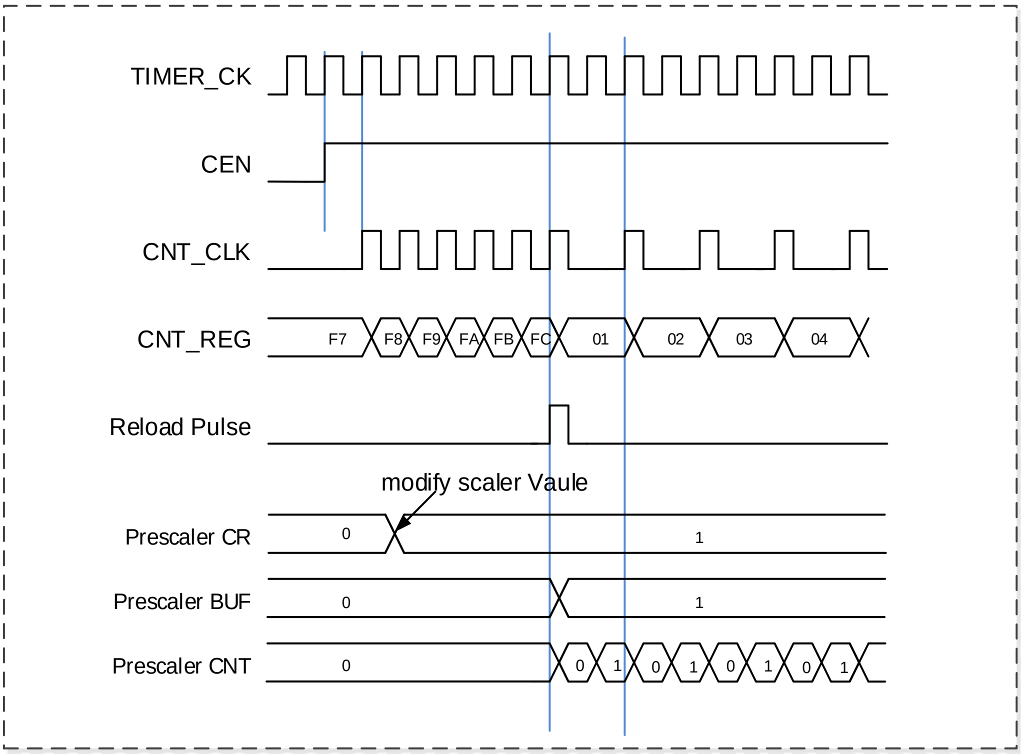
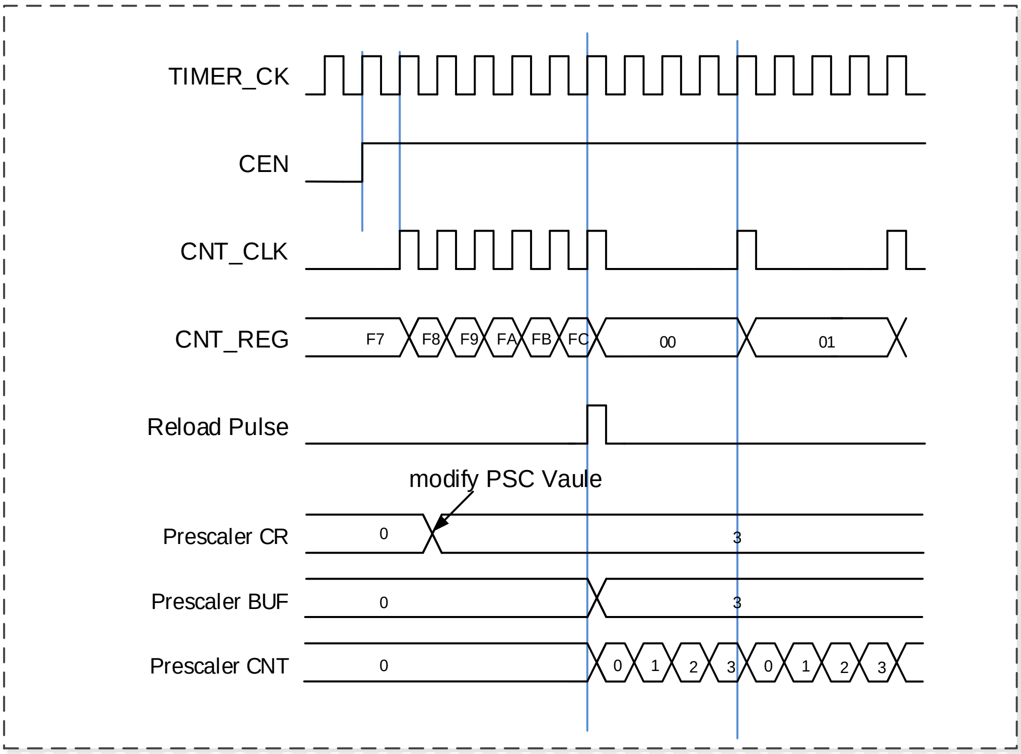


Figure 9-102. Counter timing diagram with prescaler division change from 1 to 4



## Upcounting mode

In this mode the counter counts 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 to count once again from 0. If the repetition counter is set, the update eventis generated after the number of overflow. Else the update event is generated at each counter overflow.

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 set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repeatition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

**Figure 9-103. Counter timing diagram, internal clock divided by 1**

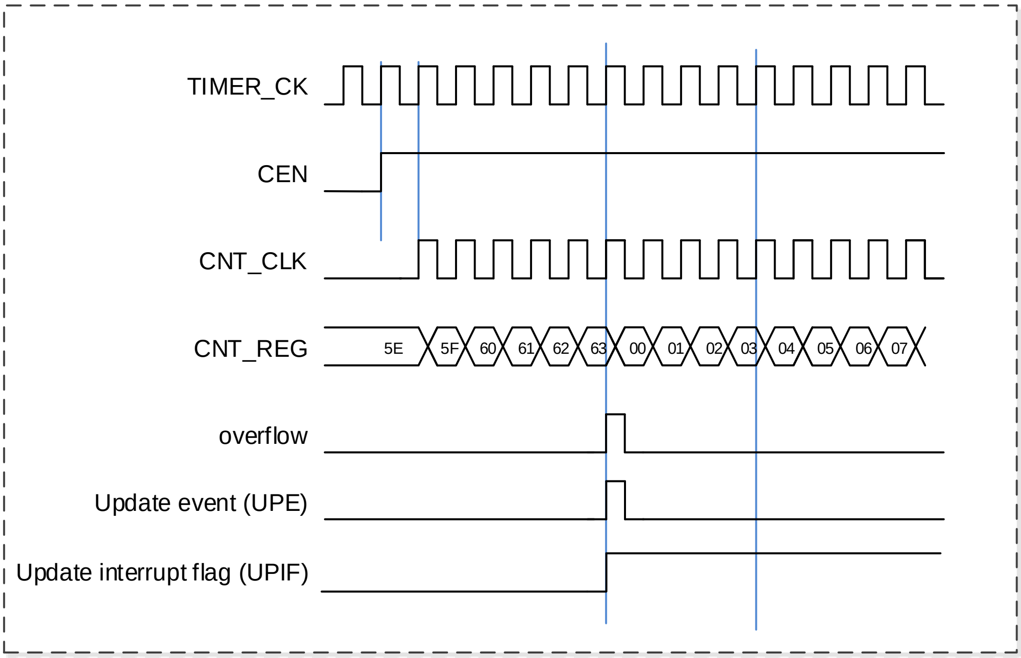


Figure 9-104. Counter timing diagram, internal clock divided by 2

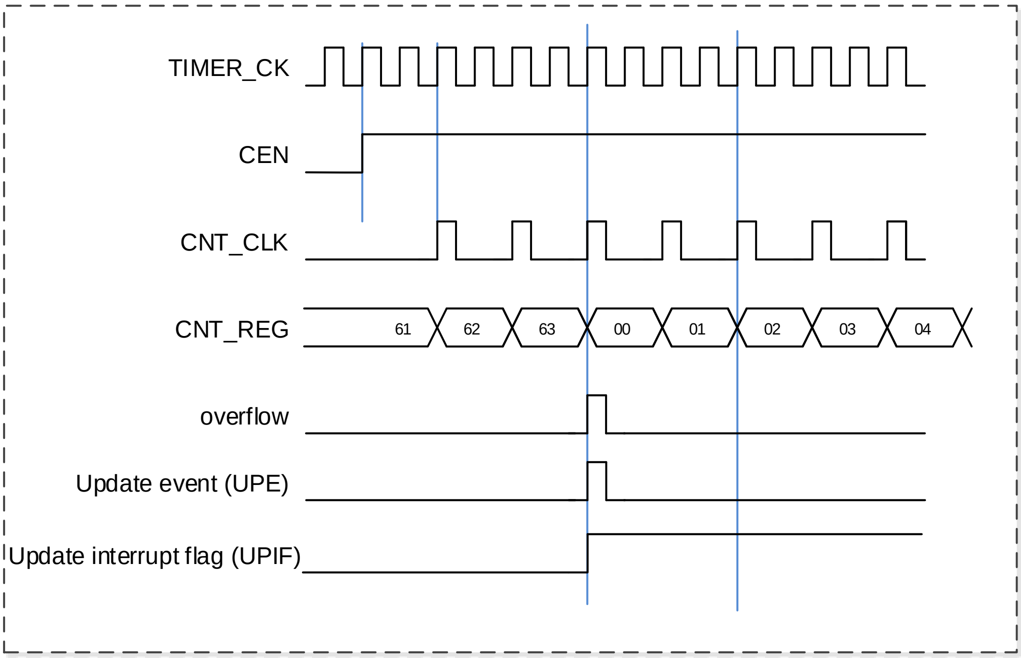


Figure 9-105. Counter timing diagram, internal clock divided by 4

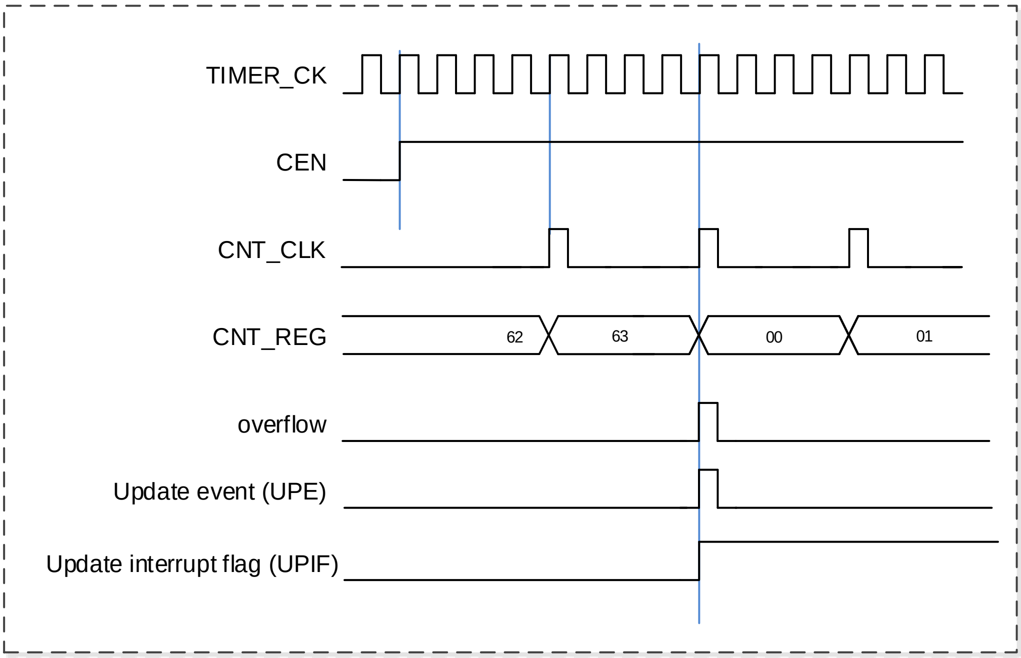


**Figure 9-106. Counter timing diagram, internal clock divided by N**

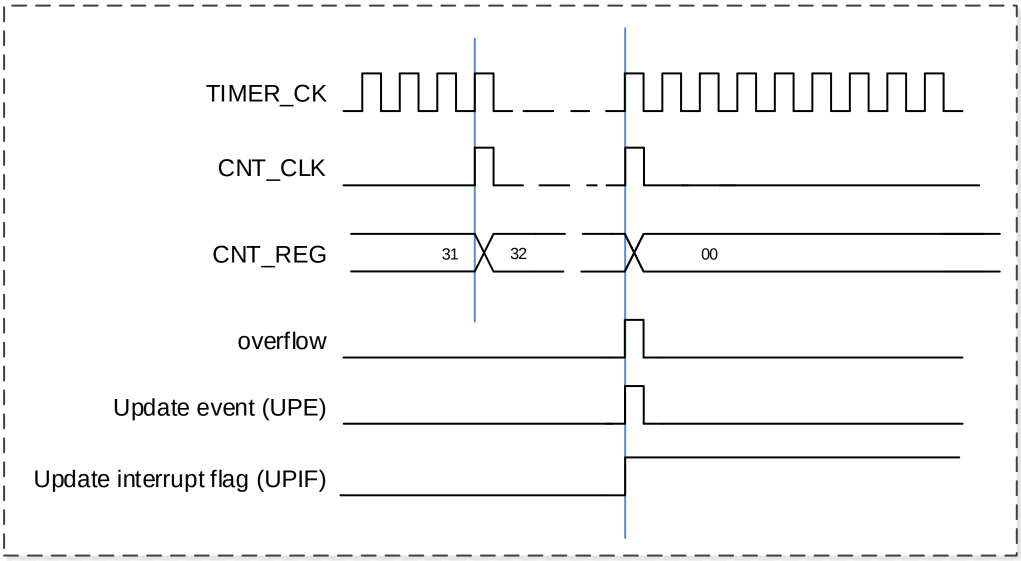


**Figure 9-107. Counter timing diagram, update event when ARSE=0**

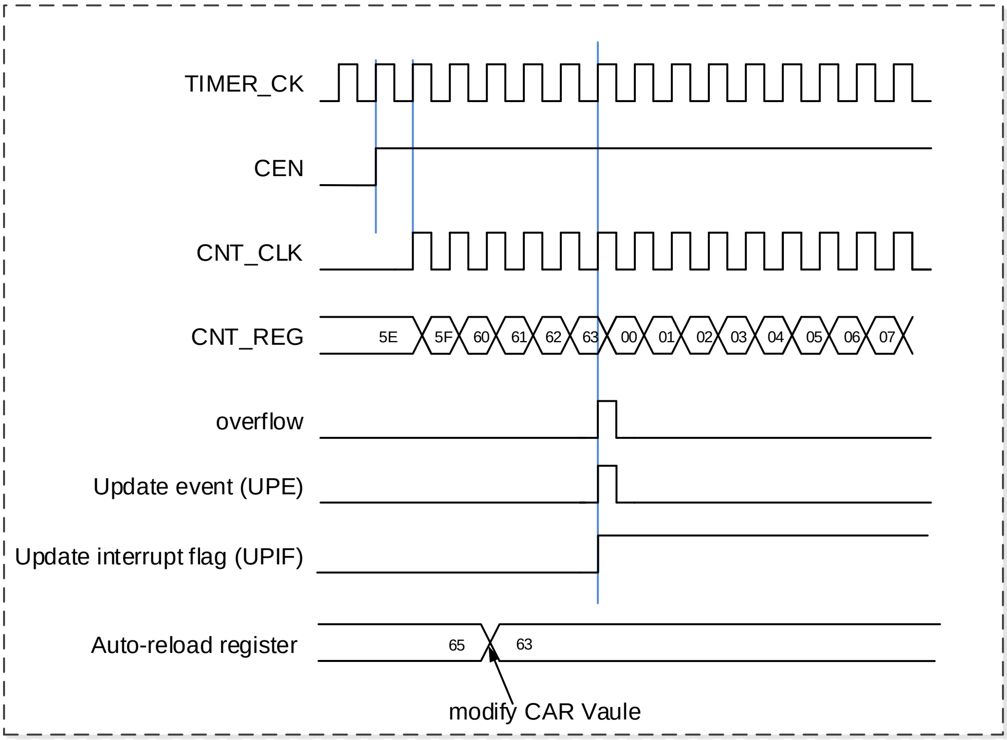
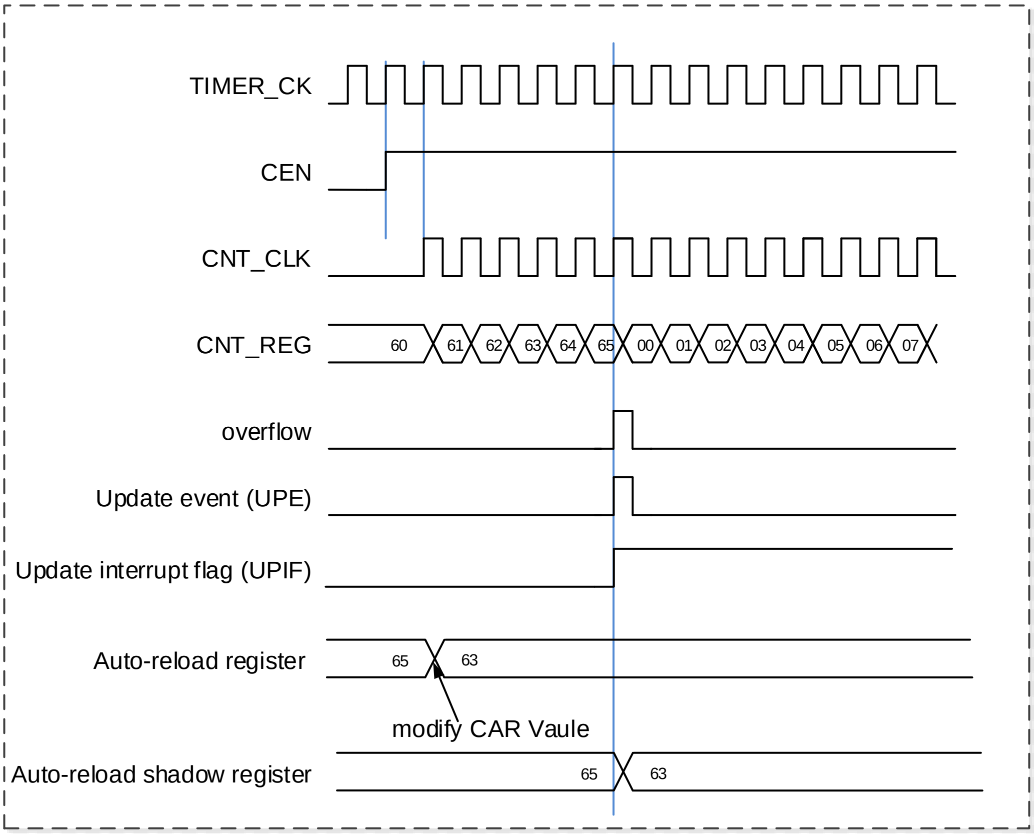


**Figure 9-108. Counter timing diagram, update event when ARSE=1**



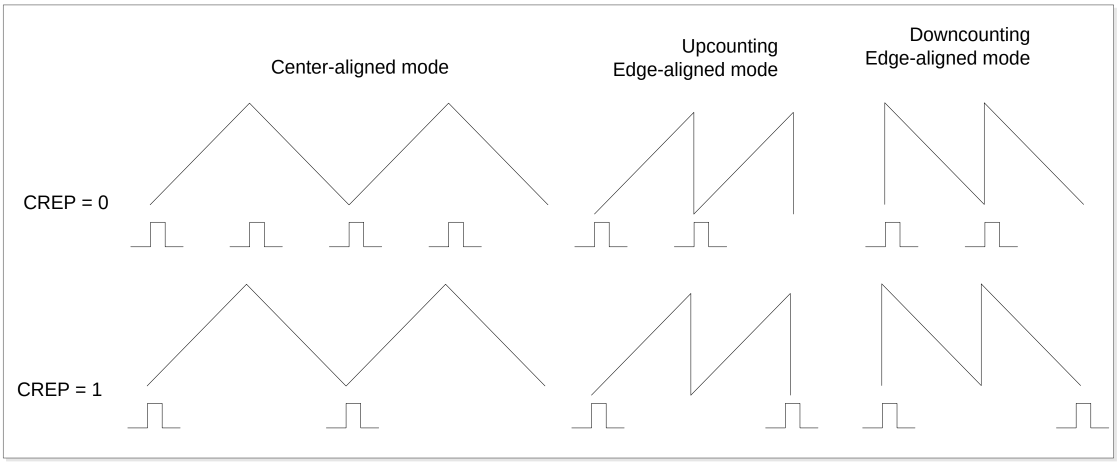
### Counter Repetition

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, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned 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 9-109. Update rate examples depending on mode and TIMERx_CREP**

TIMER14 has only upcounting edge-aligned mode.



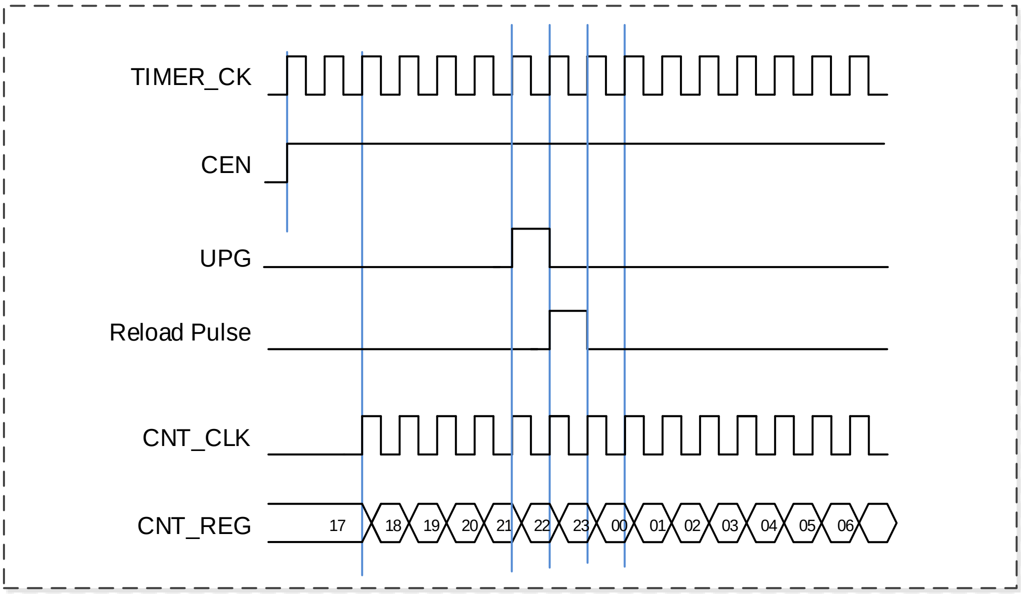
## Clock selection

The following describes the Timer Module clock controller which determines the clock source of the internal prescaler counter.

- Internal timer clock TIMER_CK

The default internal clock source is the APB2 clock CK_APB2 used to drive the counter prescaler when the slave mode is disabled. If the slave mode controller is enabled by setting SMC field in the TIMERx_SMCFG register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS field in the TIMERx_SMCFG register and described as follows. When the slave mode selection bits SMC are set to 0x4, 0x5 or 0x6, the internal clock TIMER_CK is the counter prescaler driving clock source.

**Figure 9-110. Control circuit in normal mode, internal clock divided by 1**



- Internal trigger inputs (ITI)

The counter prescaler can count during each rising or falling edge of the ITI signal. This mode can be selected by setting the SMC field to 0x6 in the TIMERx_SMCFG register;here the counter will act as an event counter. The input event, known as ITI here, can be selected by setting the TRGS field. When the ITI signal is selected as the clock source, the internal edge detection circuitry will generate a clock pulse during each ITI signal rising or falling edge to drive the counter prescaler.

- Channel input pin (CI)

The counter prescaler can be driven to count during each rising or falling edge on the external pin TIMERx_CIx. This mode can be selected by setting SMC field to 0x7 and the TRGS field to 0x4, 0x5 or 0x6. Note that the CIx is derived from the TIMERx_CHx sampled by a digital filter.

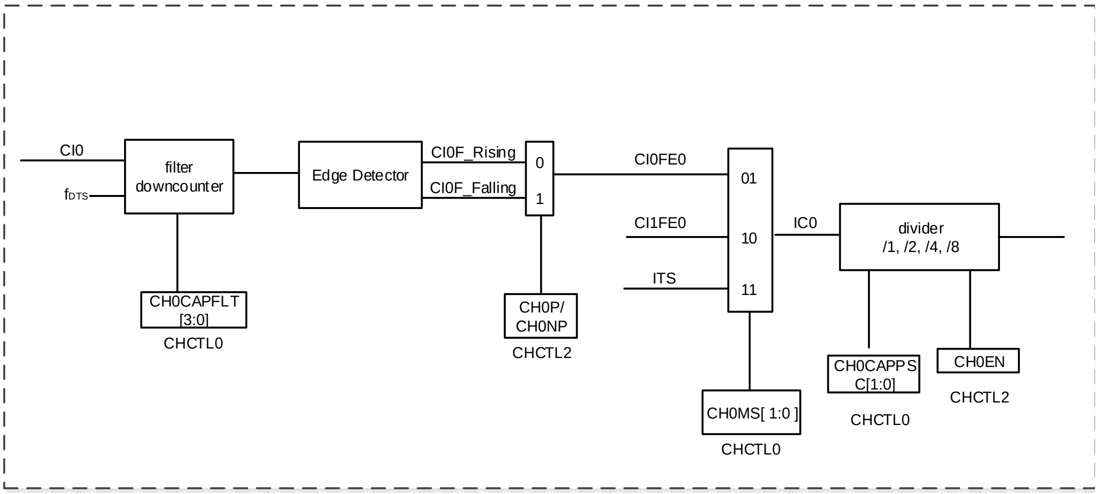
### Capture/compare channels

The TIMER14 has two independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture/compare value register including an input stage, channel controller and an output stage.

- Input capture stage

The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. The channel input signal (CIx) is sampled by a digital filter to generate a filtered input signal CIxF. Then the channel polarity and the edge detection block can generate a CIxFEx signal for the input capture function. The effective input event number can be set by the channel input prescaler (CHxCAPPSC).

**Figure 9-111. Capture/compare channel (example: channel 0 input stage)**



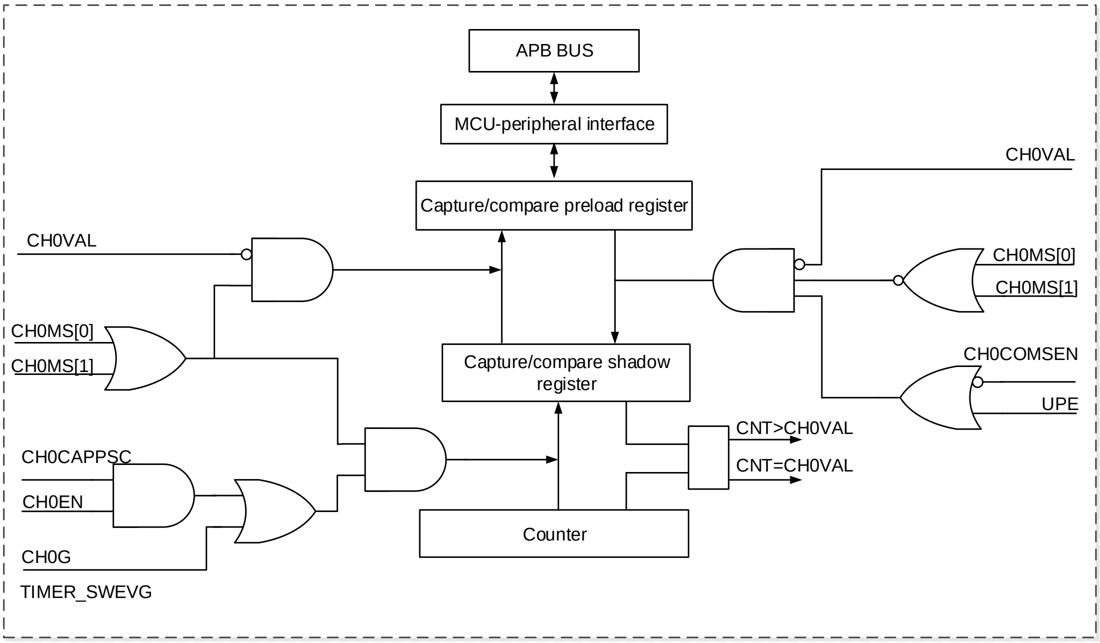
- Channel controller

The GPTM has two independent channels which can be used as capture inputs or compare match outputs.

When used in the input capture mode, the counter value is captured into the TIMERx_CHxCV shadow register first and then transferred into the TIMERx_CHxCV preload register when the capture event occurs.

When used in the compare match output mode, the contents of the TIMERx_CHxCV preload register is copied into the associated shadow register; the counter value is then compared with the register value.

**Figure 9-112. Capture/compare channel 0 main circuit**



- Output stage

The TIMER14 has two channels for compare match, single pulse or PWM output function.

**Figure 9-113. Output stage of capture/compare channel (channel 0)**

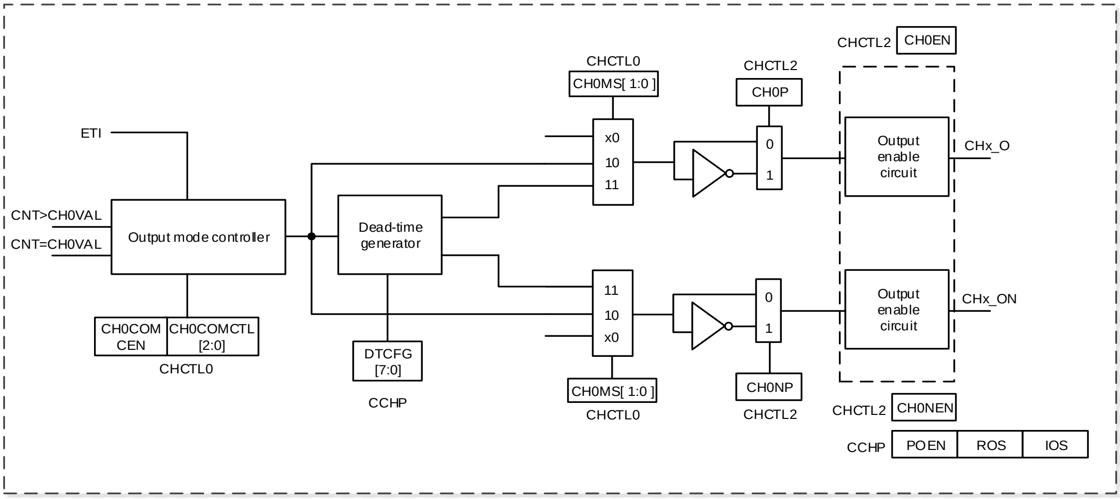
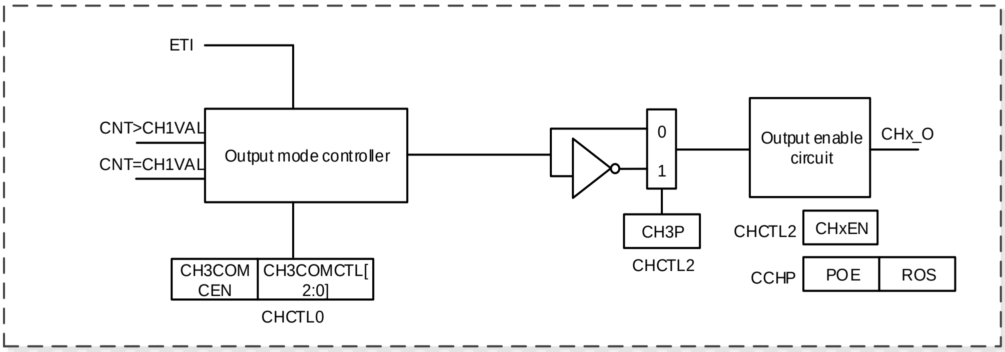


**Figure 9-114. Output stage of capture/compare channel (channel 1)**



## Input Capture Mode

When the channel is used as a capture input, the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV) when an effective input signal transition occurs. Once the capture event occurs, the CHxIF flag in the TIMERx_INTF register is set. If the CHxIF bit is already set, i.e., the flag has not yet been cleared by software, and another capture event on this channel occurs, the corresponding channel Over-Capture flag, named CHxOF, will be set.Once the capture event occurs, a DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit. The fllowing step maybe used to implement input capture mode:

- Select input signal by set CHxMS bits in the channel control register (TIMERx_CHCTLx).
- Add input filter to input signal by set CHxCAPFLT bitst in the the channel control register (TIMERx_CHCTLx).
- Select input capture edge (rising or falling) by set CHxP/CHxNP bit in the channel control register 2 (TIMERx_CHCTL2).
- If input signal need prescaler, set CHxCAPPSC bits in the the channel control register (TIMERx_CHCTLx).
- If need interrupt, set CHxIE bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- If need DMA request, set CHxDEN bit in DMA and interrupt enable register

(TIMERx_DMAINTEN) to 1.

- Enable the channel by set CHxEN bit in the channel control register 2 (TIMERx_CHCTL2) to 1.

After configure this registers, the input capture event on this channel may occurs at the corresponding edge. And the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV). An interrupt generated if CHxIE bit set. A DMA request generated if CHxDEN set.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins (CIx). For example, PWM signal connect to CI0 input. Select channel 0 capture signal to CI0 by setting CH0MS to 01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 10 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.

### Output Compare Mode/PWM Mode

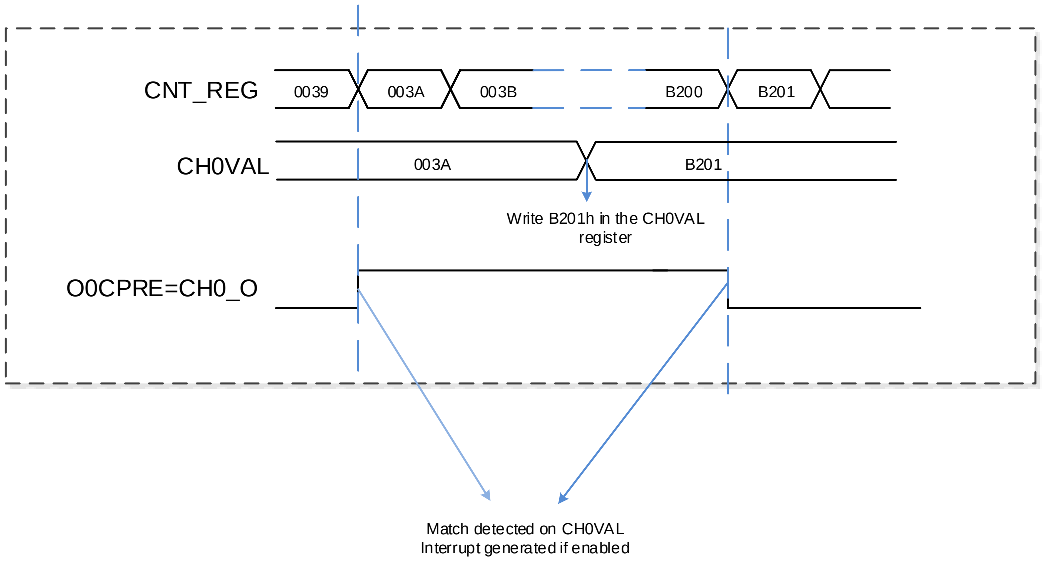
When the channel is used as an output mode by setting the CHxMS in the the channel control register (TIMERx_CHCTLx) to 00, the channel outputs waveform according to the CHxCOMCTL bits in the the the channel control register (TIMERx_CHCTLx), and the comparision between the counter and TIMERx_CHxCV registers. When the comparision equals, the CHxIF flag in the Interrupt flag registers (TIMERx_INTF) set to 1. A DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit.

In output mode, the channel can outputs active level by set the CHxCOMCTL to 101, and outputs inactive level by set the CHxCOMCTL bits to 100.

In ouput mode, the channel output set to active level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 001. The channel output set to inactive level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 010.

In output compare toggle mode (by set the CHxCOMCTL to 011), the channel output toggles when the comparision between the counter and TIMERx_CHxCV registers match.

**Figure 9-115. Output compare toggle mode, toggle onCH0_O**



In the output PWM mode (by setting the CHxCOMCTL bits to 110 (PWM mode 0) or to 111 (PWM mode 1)), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

**Figure 9-116. Output compare PWM mode 0 on CH0_O, upcounting mode**

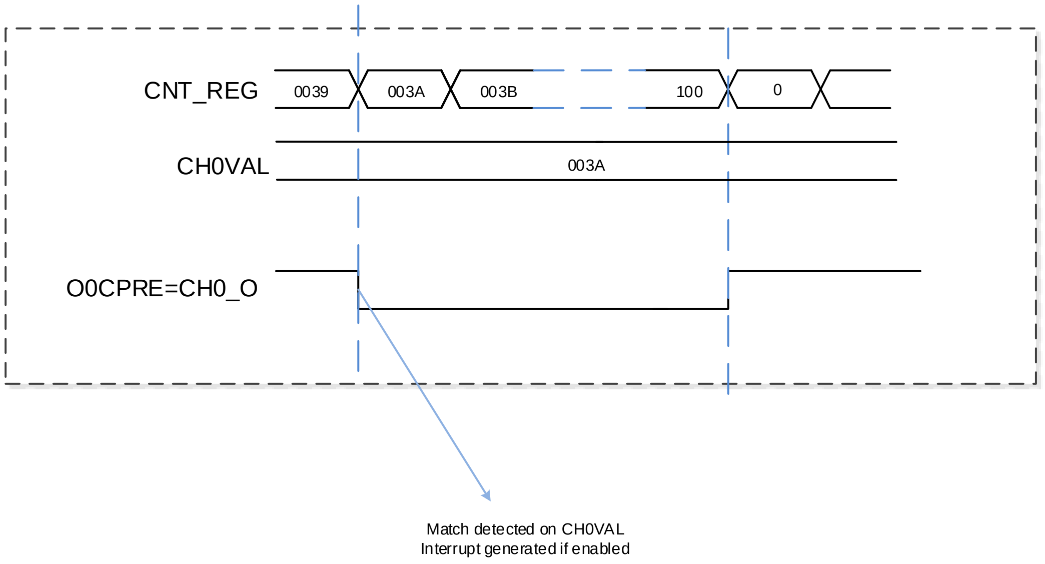
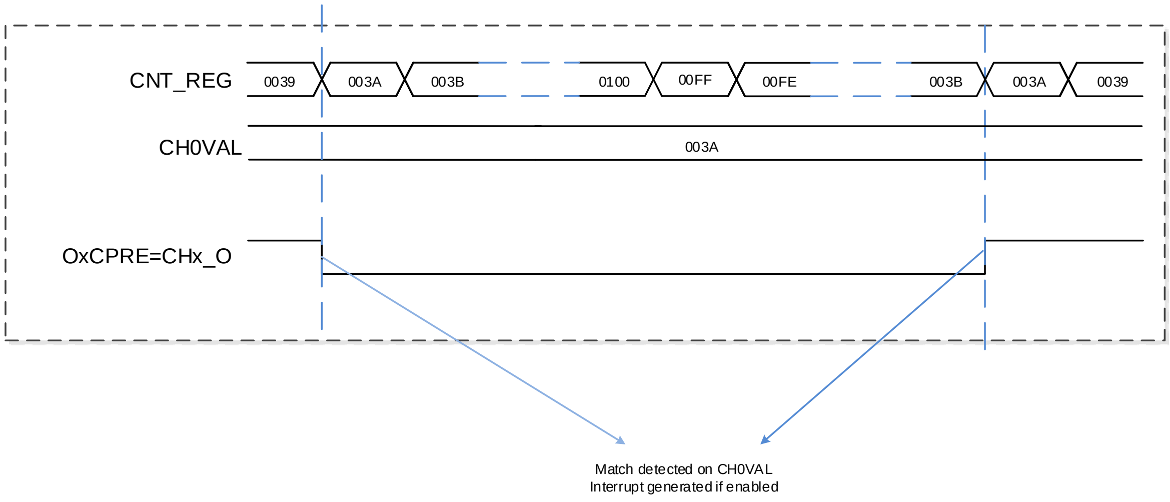


Figure 9-117. Output compare PWM mode 0 on CH0_O, center-aligned counting mode



### Channel Output Reference Signal

When the TIMER14 is used in the compare match output mode, the OxCPRE signal (Channel x Output Reference signal) is defined by setting the CHxCOMCTL bits. 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 detailed 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.

The OxCPRE signal can be forced to 0 when the ETIF signal is derived from the external TIMERx_ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs Complementary and Dead-time

The TIMER14 can output two complementary signals. The complementary signals CHx_O and CHx_ON are activated by a combination of several control bits: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register and the POEN, ISOx, ISOxN, IOS and ROS bits in the TIMERx_CCHP and TIMERx_CTL1 registers.The outputs polarity are determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register

If CHxEN, CHxNEN and POEN bits are 1, dead-time should insertion. The rising edge of CHx_O output is delayed relative to the rising edge of OxCPRE and the rising edge of CHx_ON output is delayed relative to the falling edge of OxCPRE.The delay value is a 8-bit dead-time counter determined by DTCFG field in TIMERx_CCHP register.If the delay value is greater than the width of the active output (CHx_O or CHx_ON) then the corresponding pulse is not generated.

**Figure 9-118. Complementary output with dead-time insertion.**

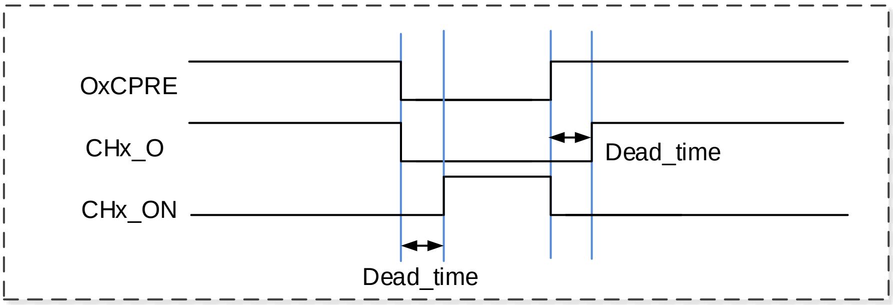


**Figure 9-119. Dead-time waveforms with delay greater than the negative pulse**

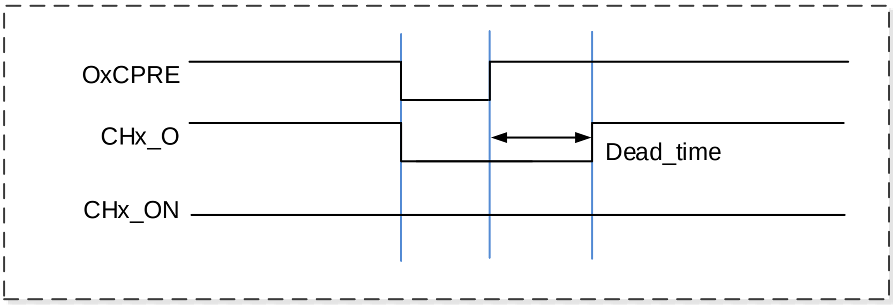
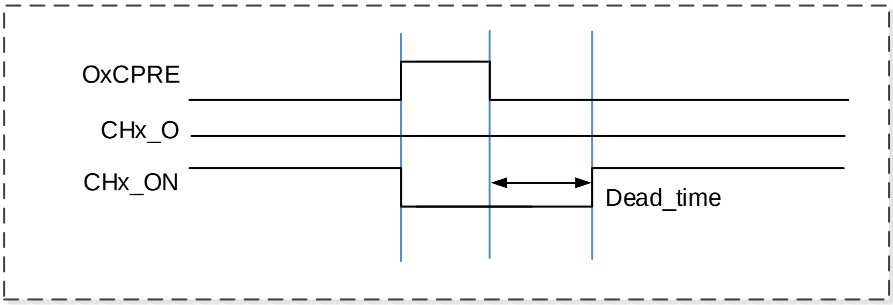


**Figure 9-120. Dead-time waveforms with delay greater than the positive pulse**



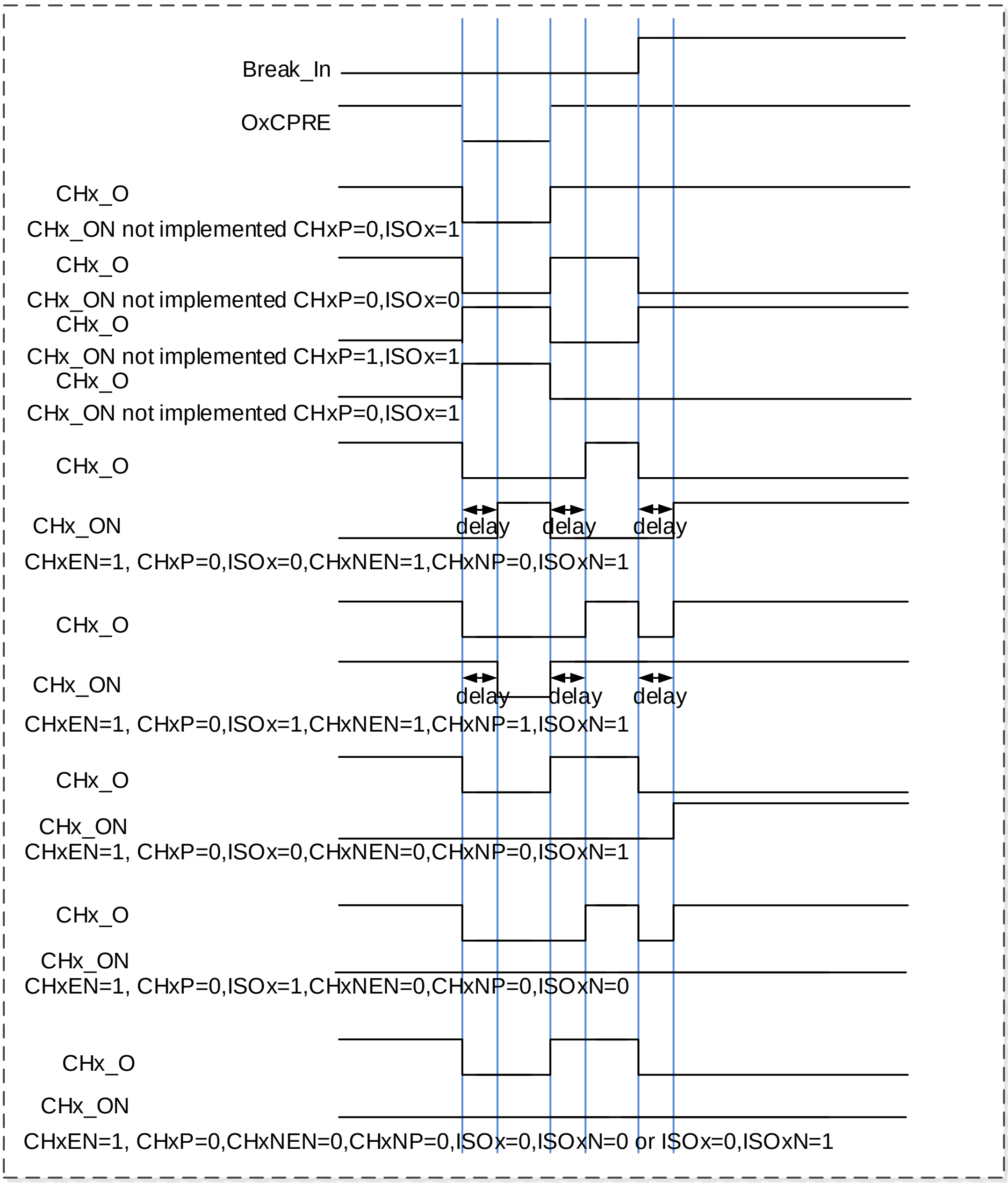
## Break function

In this function, 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 or HXTAL stack 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 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.

**Figure 9-121. Output behavior in response to a break（The break input is acting on high level）**

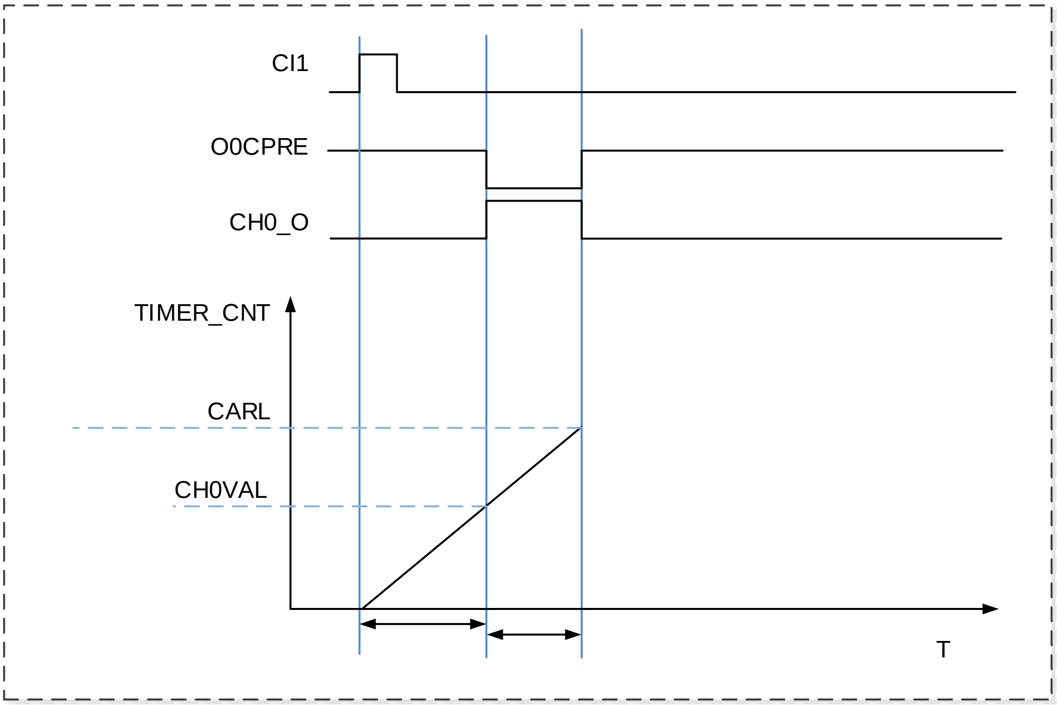

Break_In
OxCPRE
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=0
CHx_O
CHx_ON not implemented CHxP=1,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=1,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=1,CHxNP=1,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=0,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=0,CHxNP=0,ISOxN=0
CHx_O
CHx_ON
CHxEN=1, CHxP=0,CHxNEN=0,CHxNP=0,ISOx=0,ISOxN=0 or ISOx=0,ISOxN=1

### Single Pulse Mode

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. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

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 CHxOFE bit in each TIMERx_CHCTL0 register. After a trigger rising edge 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 CHxOFE 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 9-122. Single pulse mode**



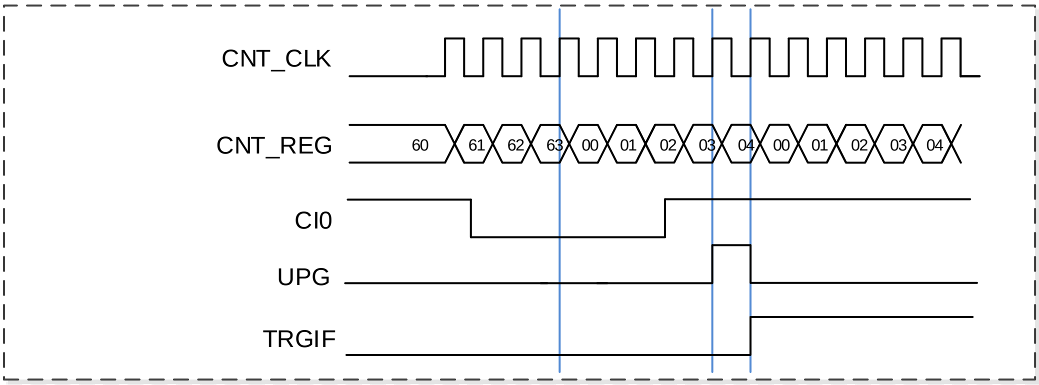
### Slave Controller

The TIMER14 can be synchronized with an external trigger in several modes including the Restart mode, the Pause mode and the event mode which is selected by the SMC field in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS field in the TIMERx_SMCFG register, below to CI0 signal as an example.The operation modes in the Slave Controller are described in the accompanying sections.

- Restart mode

The counter and its prescaler can be reinitialized in response to a rising edge of the CI0 signal. When a CI0 rising edge occurs, the update event software generation bit named UPG will automatically be asserted by hardware and the trigger event flag will also be set. Then the counter and prescaler will be reinitialized. Although the UPG bit is set to 1 by hardware, the update event does not really occur. It depends upon whether the update event disable control bit UPDIS is set to 1 or not. If UPDIS is set to 1 to disable the update event to occur, there will no update event will be generated, however the counter and prescaler are still reinitialized when the CI0 rising edge occurs. If the UPDIS bit in the TIMERx_CTL0 register is cleared to enable the update event to occur, an update event will be generated together with the CI0 rising edge, then all the preloaded registers will be updated.

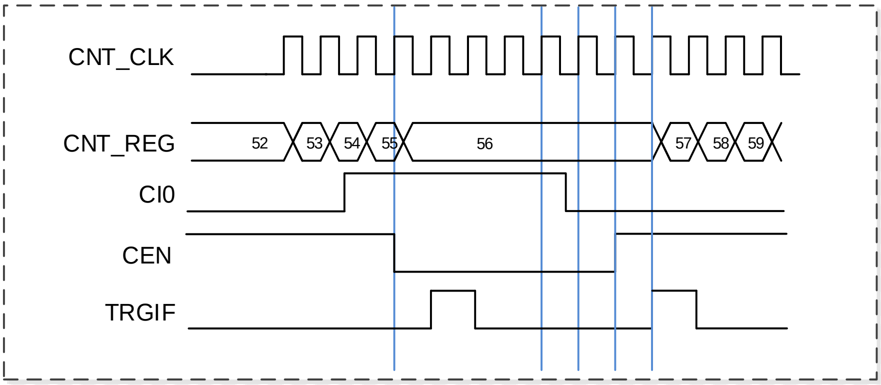
**Figure 9-123. Control circuit in restart mode**



- Pause mode

In the Pause Mode, the selected CI0 input signal level is used to control the counter start/stop operation. The counter starts to count when the selected CI0 signal is at a high level and stops counting when the CI0 signal is changed to a low level, here the counter will maintain its present value and will not be reset.

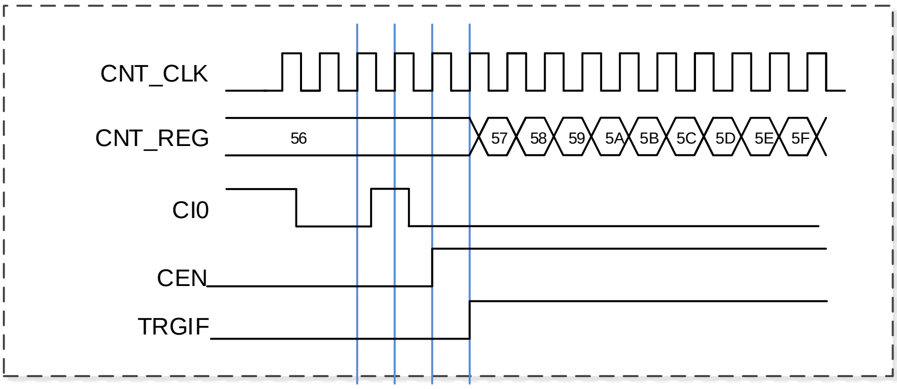
**Figure 9-124. Control circuit in pause mode**



- Event mode

After the counter is disabled to count, the counter can resume counting when a CI0 rising edge signal occurs. When a CI0 rising edge occurs, the counter will start to count from the current value in the counter. Note that the CI0 signal is only used to enable the counter to resume counting and has no effect on controlling the counter to stop counting.

**Figure 9-125. Control circuit in event mode**

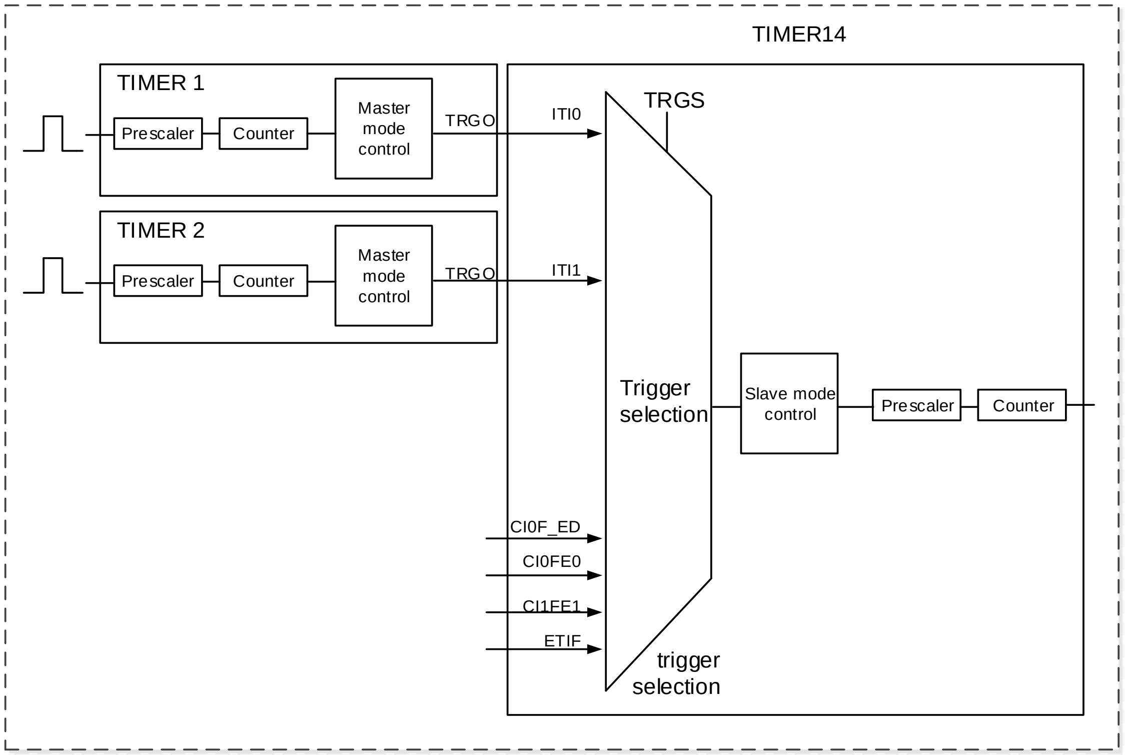


### Timer Interconnection

The timers can be internally connected together for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the Master mode while configuring another timer to be in the Slave mode. The following figures present several examples of trigger selection for the master and slave modes.

Figure below shows the TIMER14 trigger selection when it is configured in slave mode.

**Figure 9-126. TIMER14 Master/Slave mode timer example**



Other interconnection examples:

- TIMER1 as prescaler for TIMER14

We configure TIMER1 as a prescaler for TIMER14. Refer to Figure below for connections. Do as follow:

1. Configure TIMER1 in master mode and select its Update Event (UPE) as trigger output (MMC=010 in the TIMER1_CTL1 register). Then TIMER1 driver a periodic signal on each counter overflow.
2. Configure the TIMER1 period (TIMER1_CAR registers).
3. Select the TIMER14 input trigger source from TIMER1 (TRGS=001 in the TIMER14_SMCFG register).
4. Configure TIMER14 in external clock mode 1 (SMC=111 in TIMER14_SMCFG register).
5. Start TIMER14 by writing ‘1 in the CEN bit (TIMER14_CTL0 register).
6. Start TIMER1 by writing ‘1 in the CEN bit (TIMER1_CTL0 register).

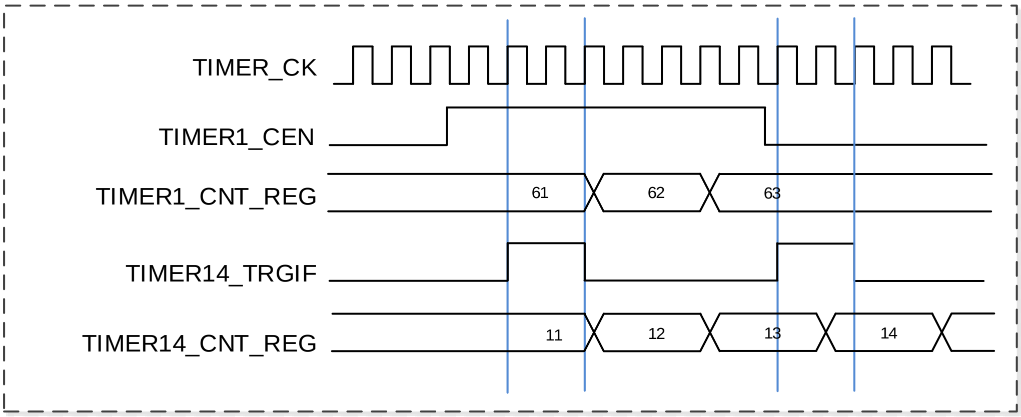
- Start TIMER14 with TIMER1’s Enable/Update signal

First, we enable TIMER14 with the enable out of TIMER1. Refer to Figure below. TIMER14 starts counting from its current value on the divided internal clock after trigger by TIMER1 enable output.

When TIMER14 receives the trigger signal its CEN bit is automatically set and the counter counts until we disable TIMER14. Both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT\_CLK} = f_{TIMER\_CK}$ /3). Do as follow:

1. Configure TIMER1 master mode to send its enable signal as trigger output(MMC=001 in the TIMER1_CTL1 register)
2. Configure TIMER14 to select the input trigger from TIMER1 (TRGS=000 in the TIMER14_SMCFG register).
3. Configure TIMER14 in event mode (SMC=110 in TIMER14_SMCFG register).
4. Start TIMER1 by writing 1 in the CEN bit (TIMER1_CTL0 register).

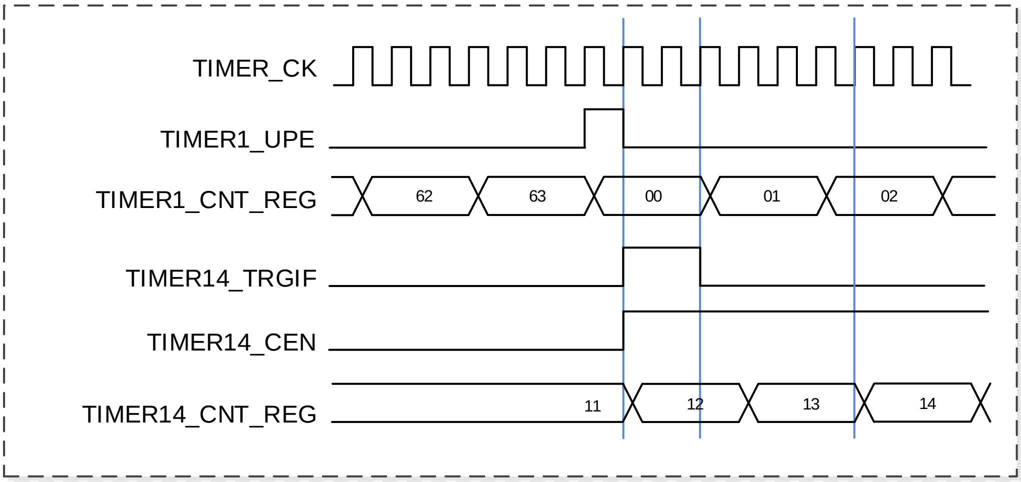
**Figure 9-127. Triggering TIMER14 with Enable of TIMER1**



In this example, we also can use update Event as trigger source instead of enable signal. Refer to figure below. Do as follow:

1. Configure TIMER1 in master mode and send its Update Event (UPE) as trigger output (MMC=010 in the TIMER1_CTL1 register).
2. Configure the TIMER1 period (TIMER1_CAR registers).
3. Configure TIMER14 to get the input trigger from TIMER1 (TRGS=001 in the TIMER14_SMCFG register).
4. Configure TIMER14 in event mode (SMC=110 in TIMER14_SMCFG register).
5. Start TIMER1 by writing ‘1 in the CEN bit (TIMER1_CTL0 register).

**Figure 9-128. Triggering TIMER14 with update of TIMER1**

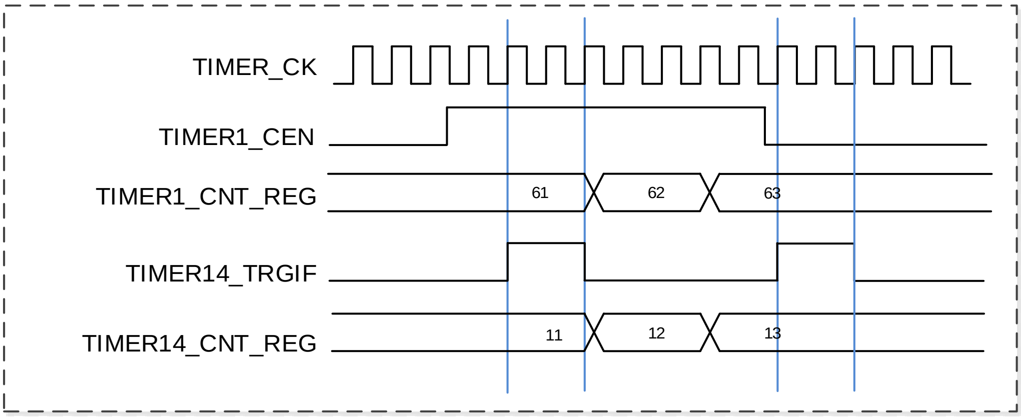


- Enable TIMER14 count with TIMER1’s enable/O0CPRE signal

In this example, we control the enable of TIMER14 with the enable output of TIMER1 .Refer to figure below. TIMER14 counts on the divided internal clock only when TIMER1 is enable. Both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT\_CLK}$ = $f_{TIMER\_CK}$). Do as follow:

1. Configure TIMER1 in master mode and Output enable signal as trigger output (MMC=001 in the TIMER1_CTL1 register).
2. Configure TIMER14 to get the input trigger from TIMER1 (TRGS=001 in the TIMER14_SMCFG register).
3. Configure TIMER14 in pause mode (SMC=101 in TIMER14_SMCFG register).
4. Enable TIMER14 by writing '1 in the CEN bit (TIMER14_CTL0 register)
5. Start TIMER14 by writing '1 in the CEN bit (TIMER14_CTL0 register).
6. Stop TIMER1 by writing '0 in the CEN bit (TIMER1_CTL0 register).

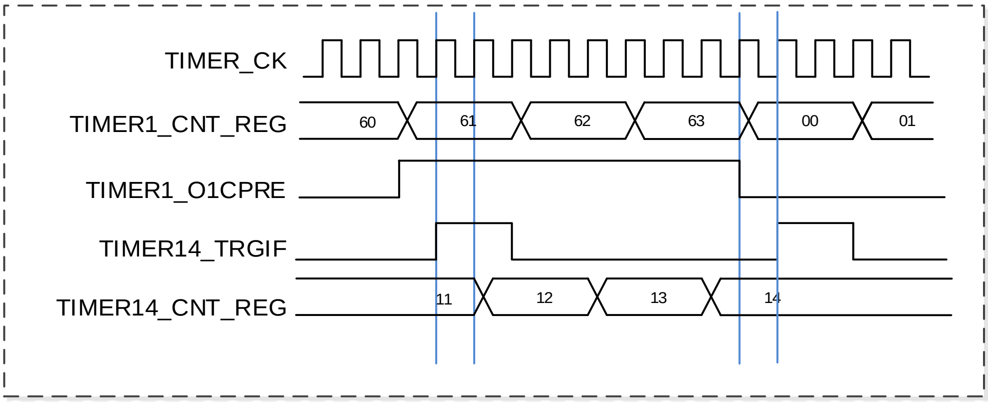
**Figure 9-129. Gating TIMER14 with enable of TIMER1**



In this example, we also can use OxCPRE as trigger source instead of enable signal output. Do as follow:

1. Configure TIMER1 in master mode and Output Compare 1 Reference (O0CPRE) signal as trigger output (MMC=100 in the TIMER1_CTL1 register).
2. Configure the TIMER1 O0CPRE waveform (TIMER1_CHCTL0 register).
3. Configure TIMER14 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0_SMCFG register).
4. Configure TIMER14 in pause mode (SMC=101 in TIMER14_SMCFG register).
5. Enable TIMER14 by writing '1 in the CEN bit (TIMER14_CTL0 register).
6. Start TIMER1 by writing '1 in the CEN bit (TIMER1_CTL0 register).

**Figure 9-130. Gating TIMER14 with O0CPREof TIMER1**



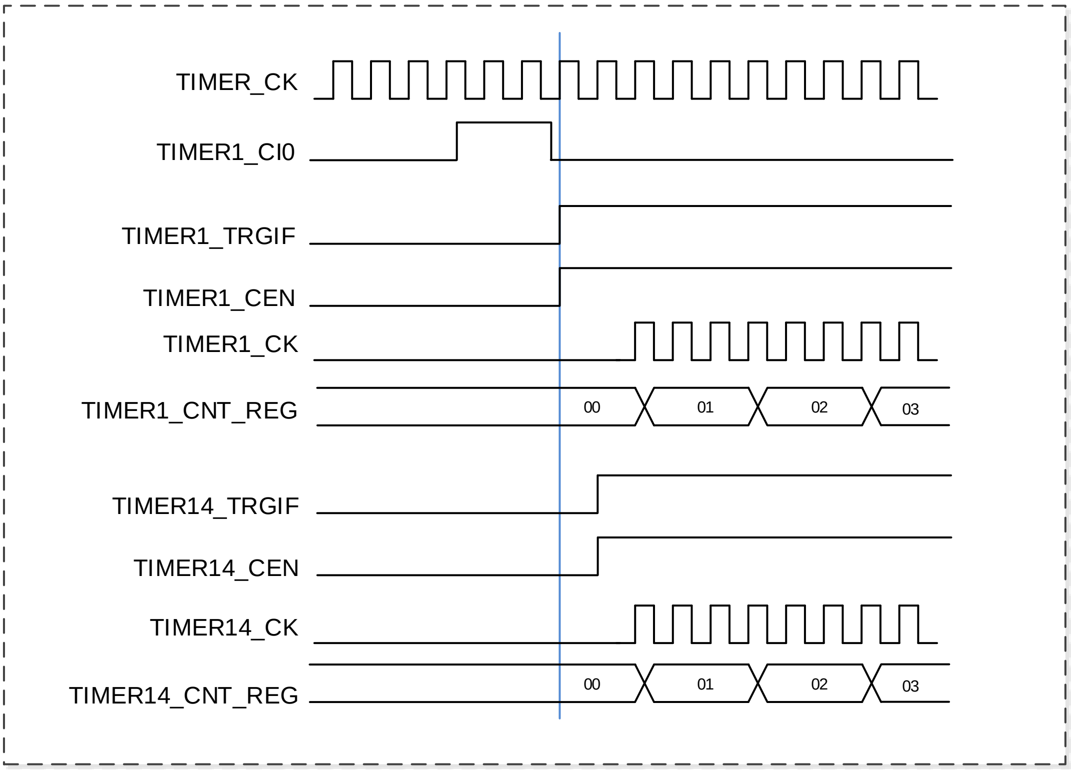
- Using an external trigger to start 2 timers synchronously

We configure the start of TIMER14 is triggered by the enable of TIMER1, and TIMER1 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, TIMER1 must be configured in Master/Slave mode. Do as follow:

1. Configure TIMER1 slave mode to get the input trigger from CI0 (TRGS=100 in the TIMER1_SMCFG register).
2. Configure TIMER1 in event mode (SMC=110 in the TIMER1_SMCFG register).
3. Configure the TIMER1 in Master/Slave mode by writing MSC=1 (TIMER1_SMCFG register).
4. Configure TIMER14 to get the input trigger from TIMER1 (TRGS=001 in the TIMER14_SMCFG register).
5. Configure TIMER14 in event mode (SMC=110 in the TIMER14_SMCFG register).

When a rising edge occurs on TIMER1's CI0, two timer counters starts counting synchronously on the internal clock and both TRGIF flags are set.

**Figure 9-131. Triggering TIMER14 and TIMER1 with TIMER1's CI0 input**



### Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.5.4. TIMER14 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:10 | 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 ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.<br>00: $f_{DTS}=f_{TIMER\_CK}$<br>01: $f_{DTS}= f_{TIMER\_CK} /2$<br>10: $f_{DTS}= f_{TIMER\_CK} /4$<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 TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value |
| 3 | SPM | Single pulse mode.<br>0: Counter continues after update event.<br>1: The CEN is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: When enabled, any of the following events generate an update interrupt or DMA request:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request. |
| 1 | UPDIS | Update disable<br>This bit is used to enable or disable the update event generation. |

| | | |
|---|---|---|
| | | 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event. |
| 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 encoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | ISO1 | ISO0N | ISO0 | Res. | MMC[2:0] | | | DMAS | CCUC | Res. | CCSE |
| | | | | | rw | rw | rw | | rw | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:11 | 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<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 | 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: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset.<br>001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal TIMERx_EN as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.<br>010: Update. In this mode the master mode controller selects the update event as TRGO.<br>011: Capture/compare pulse.In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred.<br>100: Compare.In this mode the master mode controller selects the O0CPRE signal is used as TRGO<br>101: Compare.In this mode the master mode controller selects the O1CPRE signal is used as TRGO<br>110: Compare.In this mode the master mode controller selects the O2CPRE signal is used as TRGO<br>111: Compare.In this mode the master mode controller selects the O3CPRE signal is used as TRGO |
| 3 | DMAS | DMA request source selection<br>0: DMA request of channel x is sent when capture/compare event occurs<br>1: DMA request of channel x is sent when update event occurs |
| 2 | CCUC | Commutation control shadow register update control<br>When the Commutation control shadow registers (for CHxEN, CHxNEN and CHxCOMCTL bits) are enabled (CCSE=1), this bit control when these shadow registers update.<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 register 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.After these bits have been written, they are updated only when CMTG bit is set<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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  | MSM | TRGS[2:0] |  |  | Res. | SMC[2:0] |  |  |
|  |  |  |  |  |  |  |  | rw | rw |  |  |  | rw |  |  |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value |
| 7 | MSM | Master-slave mode<br>The effect of an event on the trigger input is delayed in this mode to allow a perfect synchronization between the current timer and its slaves through TRGO. If we want to synchronize several timers on a single external event, this mode can be used.<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: Internal trigger input 0 (ITI0) TIMER1<br>001: Internal trigger input 1 (ITI1) TIMER2<br>010: Internal trigger input 2 (ITI2) Res<br>011: Internal trigger input 3 (ITI3) Res<br>100: CI0 edge flag (CI0F_ED)<br>101: Filtered channel 0 input (CI0FE0)<br>110: Filtered channel 1 Input (CI1FE1)<br>111: External trigger input (ETIF)<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<br>000: Disable mode.The prescaler is clocked directly by the internal clock when CEN bit is set high<br>001: Reserved<br>010: Reserved<br>011: Reserved<br>100: Restart Mode. The counter is reinitialized and the shadow registers are updated 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 when it is low.<br>110: Event Mode. A rising edge of the trigger input enables the counter. The counter |

cannot be disabled by the slave mode controller.

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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | TRGDEN | Reserved | | | CH1DEN | CH0DEN | UPDEN | BRKIE | TRGIE | CMTIE | Reserved | | CH1IE | CH0IE | UPIE |
| | rw | | | | rw | rw | rw | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Trigger DMA request disabled<br>1: Trigger DMA request enabled |
| 13:11 | Reserved | Must be kept at reset value. |
| 10 | CH1DEN | Channel 1 Capture/Compare DMA request enable<br>0: Channel 1 Capture/Compare DMA request disabled<br>1: Channel 1 Capture/Compare DMA request enabled |
| 9 | CH0DEN | Channel 0 Capture/Compare DMA request enable<br>0: Channel 0 Capture/Compare DMA request disabled<br>1: Channel 0 Capture/Compare DMA request enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Update DMA request disabled<br>1: Update DMA request enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Break interrupt disabled<br>1: Break interrupt enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: Trigger interrupt disabled<br>1: Trigger interrupt enabled |
| 5 | CMTIE | CMT interrupt enable<br>0: CMT interrupt disabled |

| | | |
|---|---|---|
| | | 1: CMT interrupt enabled |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | CH1IE | Channel 1 Capture/Compare interrupt enable<br>0: Channel 1 Capture/Compare interrupt disabled<br>1: Channel 1 Capture/Compare interrupt enabled |
| 1 | CH0IE | Channel 0 Capture/Compare interrupt enable<br>0: Channel 0 Capture/Compare interrupt disabled<br>1: Channel 0 Capture/Compare interrupt enabled |
| 0 | UPIE | Update interrupt enable<br>0: Update interrupt disabled<br>1: Update interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | CH1OF | CH0OF | Res. | BRKIF | TRGIF | CMTIF | Res. | | CH1IF | CH0IF | UPIF |
| | | | | | rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:11 | Reserved | Must be kept at reset value |
| 10 | CH1OF | Channel 1 Capture overflow flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 Capture overflow 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 Capture overflow interrupt occurred<br>1: Capture overflow interrupt occurred |
| 8 | Reserved | Must be kept at reset value |
| 7 | BRKIF | Break interrupt flag<br>This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active.<br>0: No active level break has been detected<br>1: An active level has been detected |
| 6 | TRGIF | Trigger interrupt flag |

| | | |
|---|---|---|
| | | This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on TRGI input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on TRGI input 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 CMT event occurs, and cleared by software<br>0: No CMT interrupt occurred<br>1: CMT interrupt occurred |
| 4:3 | Reserved | Must be kept at reset value |
| 2 | CH1IF | Channel 1 interrupt enable<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BRKG | TRGG | CMTG | Reserved | | CH1G | CH0G | UPG |
| | | | | | | | | w | w | w | | | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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. |

| | | |
|---|---|---|
| | | 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 the channel control registers (CHxEN, CHxNEN and CHxCOMCTL bits) of the channels having a complementary output are updated.<br>0: no generate CMT event<br>1: generate CMT 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 0 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 0 capture or compare event<br>1: Generate a channel 0 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 upcounting mode is selected, else (downcounting) 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

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

| 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 |
|---|---|---|
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 output compare mode<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 OFF (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is configured as output<br>01: Channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: Channel 1 is configured as input, IC1 is connected to CI0FE1<br>11: Channel 1 is configured as input, IC1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.<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 behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.<br>000: Frozen.The O0CPRE signal keep stable, independent of the comparison between the output compare register TIMERx_CH0CV and the counter.<br>001: Set high on match.O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>010: Set low on match. O0CPRE signal is forced low when the counter matches the c output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter matches the c output |

| | | |
|---|---|---|
| | | compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced low level.<br>101: Force high. O0CPRE is forced high level.<br>110: PWM mode 0. When counting up, O0CPRE is high as long as the counter is smaller than TIMER0_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high.<br>111: PWM mode 1. When counting up, O0CPRE is low as long as the counter is smaller than TIMER0_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low.<br>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from "frozen" mode to "PWM" mode or when the result of the comparison 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. |
| 3 | CH0COMSEN | Channel 0 output compare shadow enable<br>When this bit is set, the shadow register of TIMER0_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 validating the shadow register only in one pulse mode (SPM bit set in TIMERx_CTL0 register is set).<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 compare fast disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br>1: Channel 0 output compare fast enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles. |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMER0_CHCTL2 register is reset).<br>00: Channel 0 is configured as output.<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0.<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0.<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG |

register.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 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>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is OFF (CH1EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 1 is configured as output<br>01: Channel 1 is configured as input, IC1 is connected to CI1FE1<br>10: Channel 1 is configured as input, IC1 is connected to CI0FE1<br>11: Channel 1 is configured as input, IC1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disable, $f_{SAMP}$=$f_{DTS}$, N=1<br>0001: $f_{SAMP}$=$f_{TIMER\_CK}$, N=2<br>0010: $f_{SAMP}$= $f_{TIMER\_CK}$, N=4<br>0011: $f_{SAMP}$= $f_{TIMER\_CK}$, N=8<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8 |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the ratio of the prescaler on channel 0 input.The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is reset. |

| | | |
|---|---|---|
| | | 00: Prescaler disable, capture is done on each channel input edge<br>01: Capture is done every 2 channel input edges<br>10: Capture is done every 4channel input edges<br>11: Capture is done every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20
Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CH1NP | Res. | CH1P | CH1EN | CH0NP | CH0NEN | CH0P | CH0EN |
| | | | | | | | | rw | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value |
| 7 | CH1NP | Channel 1 complementary output polarity<br>Refer to CH0NP description |
| 6 | Reserved | Must be kept at reset value |
| 5 | CH1P | Channel 1 polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 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 active high<br>1: Channel 0 active low<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 channel 0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 polarity<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>This bit cannot be modified when PROT[1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 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 channel 0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000

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

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

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

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

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

| |
|---|
| PSC[15:0] |
| rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CAR[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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel 0<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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH1VAL[15:0] | Capture or compare value of channel 1<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

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

| 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 |
|---|---|---|
| 15 | POEN | Primary output enable<br>This bit s set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware assoon as the break input is active. When a channel 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: Channel outputs are disabled or forced to idle state.<br>1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN can be not set by hardware.<br>1: POEN can be set by hardware automatically at the next update event,if the break input is notactive.<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 BRK input signal.<br>0: BRK input active low<br>1: BRK input active high |
| 12 | BRKEN | Break enable<br>This bit can be set to enable the BRK and CCS clock failure event inputs.<br>0: Break inputs disabled<br>1: Break inputs enabled<br>This bit can be modified onlywhen PROT[1:0] bit-filed in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode off-state configure<br>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.<br>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled.<br>1: When POEN bit is set, he channel output signals (CHx_O/CHx_ON)are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.<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 configure<br>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode. |

| | | |
|---|---|---|
| | | 0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled.<br>1: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.<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.<br>00: protect disable. No write protection.<br>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.<br>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.<br>11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTLx registers (if the related channel is configured in output) are writing protected.<br>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<br>This bit-field specifies the duration of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG [7:5]=0xx: duration =DTCFG[7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG [7:5]=10x:duration =(64+DTCFG[5:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*2.<br>DTCFG [7:5]=110: duration =(32+DTCFG[4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG [7:5]=111:duration =(32+DTCFG[4:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*16.<br>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

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

| 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 |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 12:8 | DMATC[4:0] | DMA transfer count<br>When register access are done through the TIMERx_DMATB address, this 5-bit bit-field specifies the number of transfers. |
| 7:5 | Reserved | Must be kept at reset value |
| 4:0 | DMATA[4:0] | DMA transfer access start address<br>When register access are done through the TIMERx_DMATB address, this bit-field specifies the offset of the starting address from the TIMERx_CTL0 register. |

### DMA Transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or 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 |
|---|---|---|
| 15:0 | DMATB[15:0] | DMA transfer<br>When a read or write operation is assigned to this register, the register located at the address range (DMATA + burst counter) x 4 from TIMERx_CTL0 will be accessed.<br>The burst counter is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value |
| 1 | CCSEL | Write Capture/Compare register selection<br>This bit-field set and reset by software.<br>1: If write the Capture/Compare register, the write value is same as the |

| | | |
|---|---|---|
| | | Capture/Compare 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 |

## 9.6. General timer (TIMER15/TIMER16)

### 9.6.1. Introduction

The general timer, known as TIMER15/TIMER16, may be used for a variety of purposes. It consists of one 16-bit up-counter; one 16-bit capture/compare register (TIMERx_CH0CV), one 16-bit counter auto reload register (TIMERx_CAR) and several control registers. They can be used for a variety of purposes including general timer, input signal pulse width measurement or output waveform generation such as single pulse generation or PWM output.

The general (TIMER15/TIMER16) timers are completely independent. They do not share any resources but can be synchronized together.

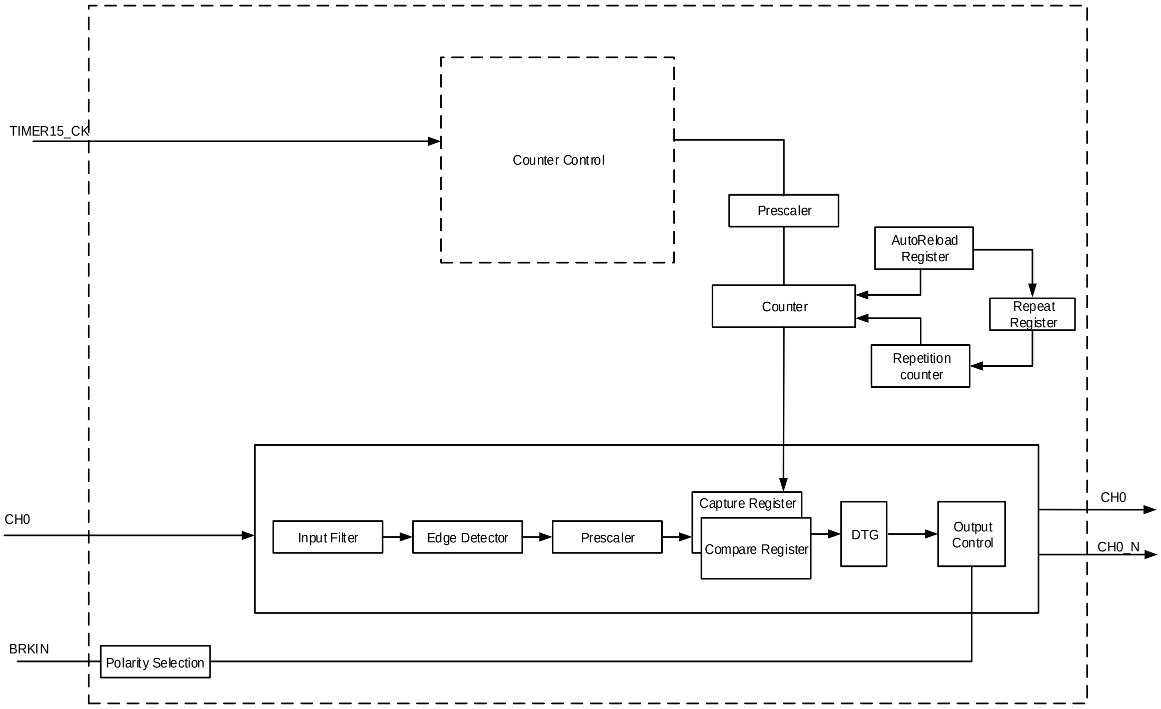
### 9.6.2. Main features

- 16-bit up auto-reload counter.
- 16-bit programmable prescaler that allows division of the counter clock frequency by any factor between 1 and 65536.
- 1 independent channel supports functions including input capture, compare match output, generation of PWM waveform (edge and center-aligned Mode), and single pulse mode output.
- Counter repetition to update the timer registers only after a given number of cycles of the counter.
- Break input to trigger the timer’s output signals to a known state.
- Interrupt/DMA generation by update, trigger event, input capture event, output compare match event or break input
- Programmable dead-time for output match.

### 9.6.3. Function description

Figure below provides details on the internal configuration of the advanced timer.

**Figure 9-132. Gereral timer block diagram (TIMER15/16)**



### Prescaler counter

The prescaler can divide the timer clock (TIMER_CK) to the counter clock (CNT_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx_PSC) which can be changed on the fly but be taken into account at the next update event.

**Figure 9-133. Counter timing diagram with prescaler division change from 1 to 2**

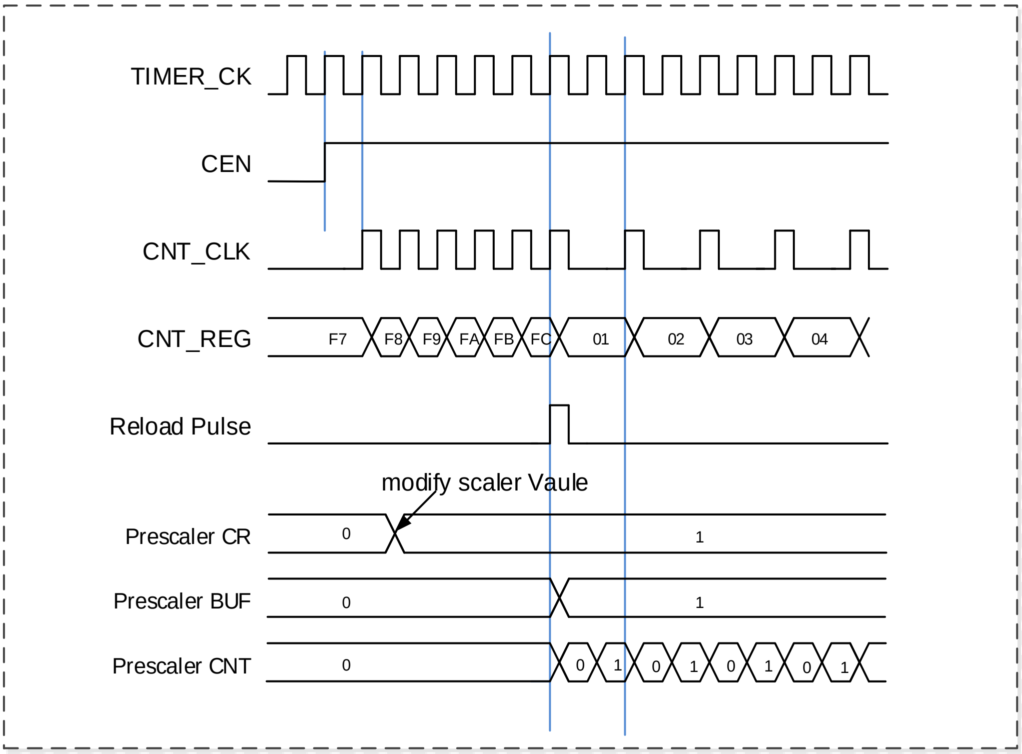
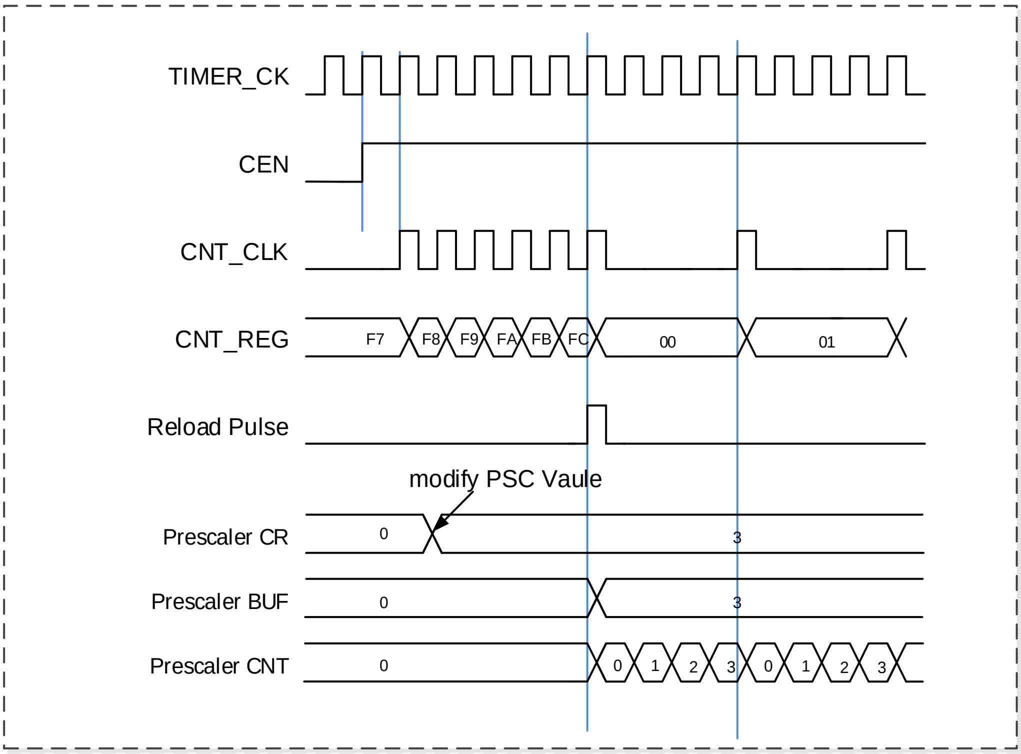


**Figure 9-134. Counter timing diagram with prescaler division change from 1 to 4**



## Upcounting mode

In this mode the counter counts 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 to count once again from 0. If the repetition counter is set, the update eventis generated after the number of overflow. Else the update event is generated at each counter overflow.

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 set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63.

Figure 9-135. Counter timing diagram, internal clock divided by 1

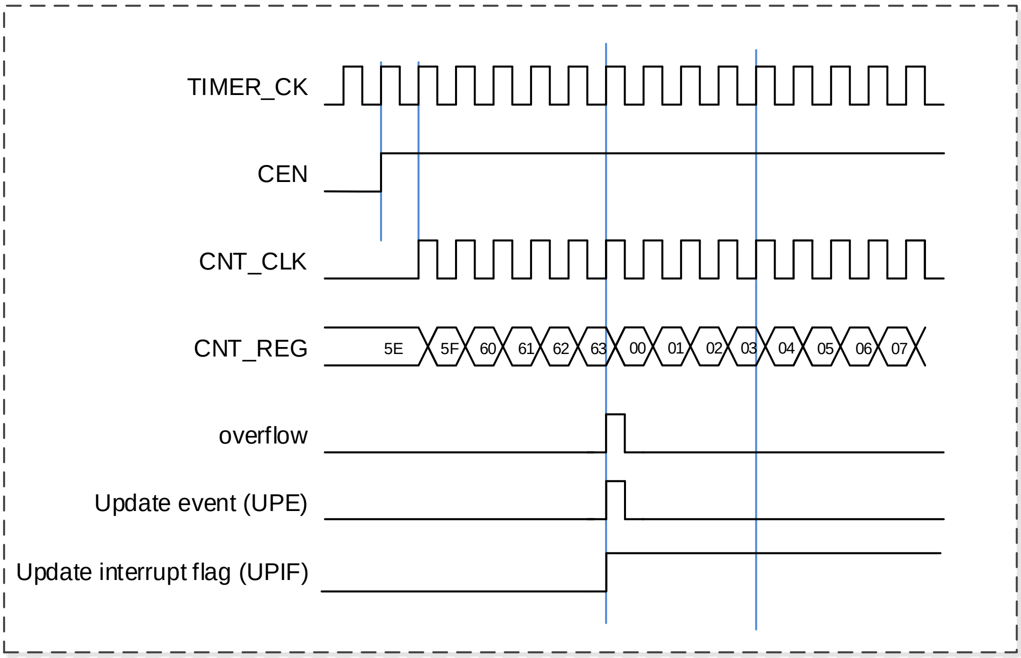


Figure 9-136. Counter timing diagram, internal clock divided by 2

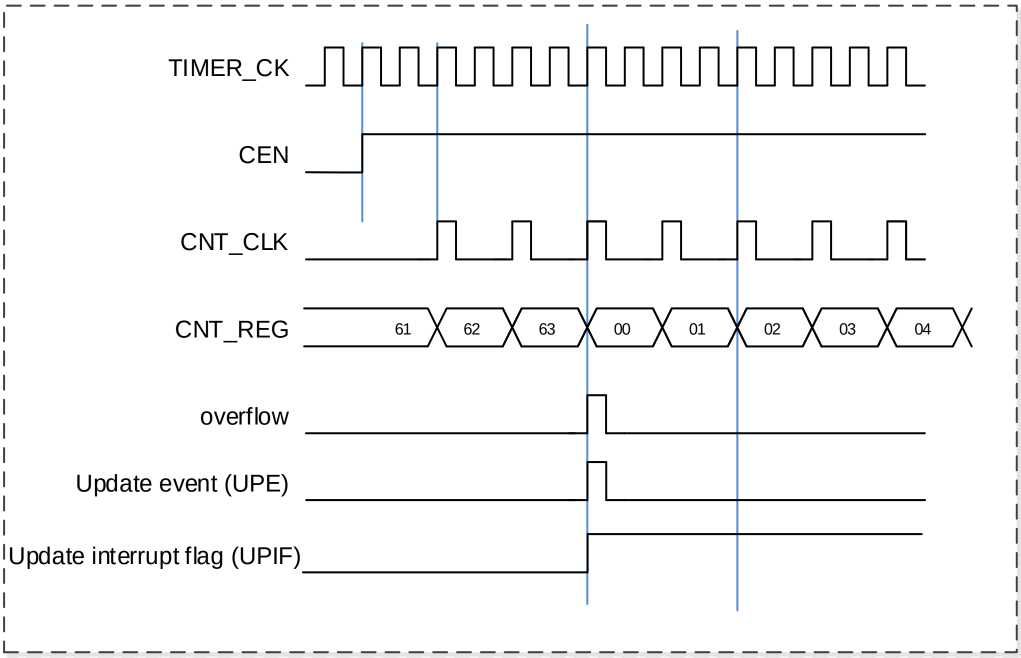


Figure 9-137. Counter timing diagram, internal clock divided by 4

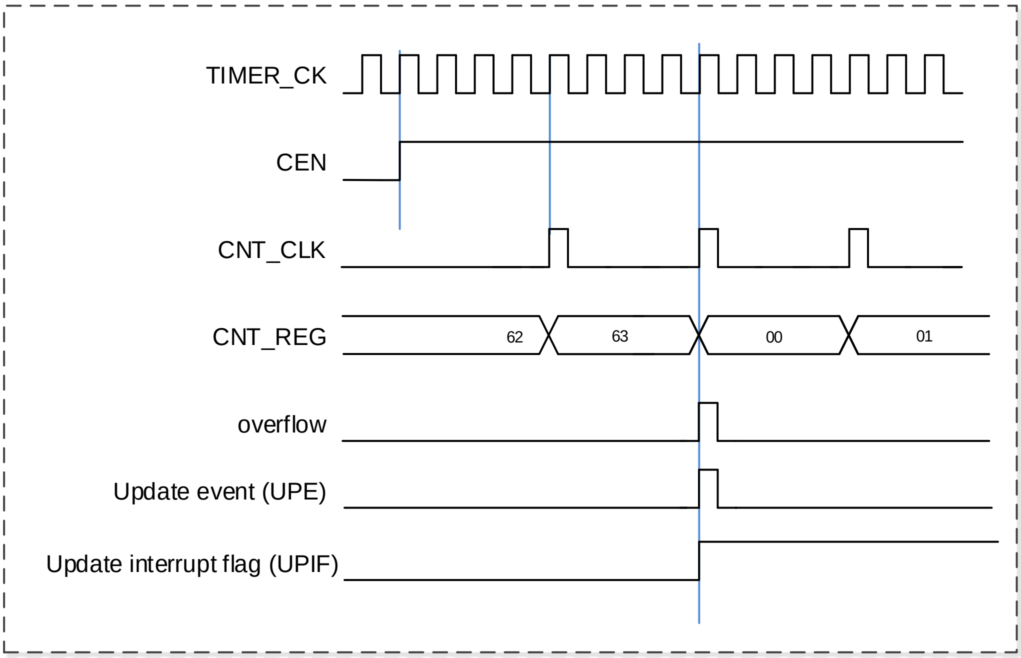


Figure 9-138. Counter timing diagram, internal clock divided by N

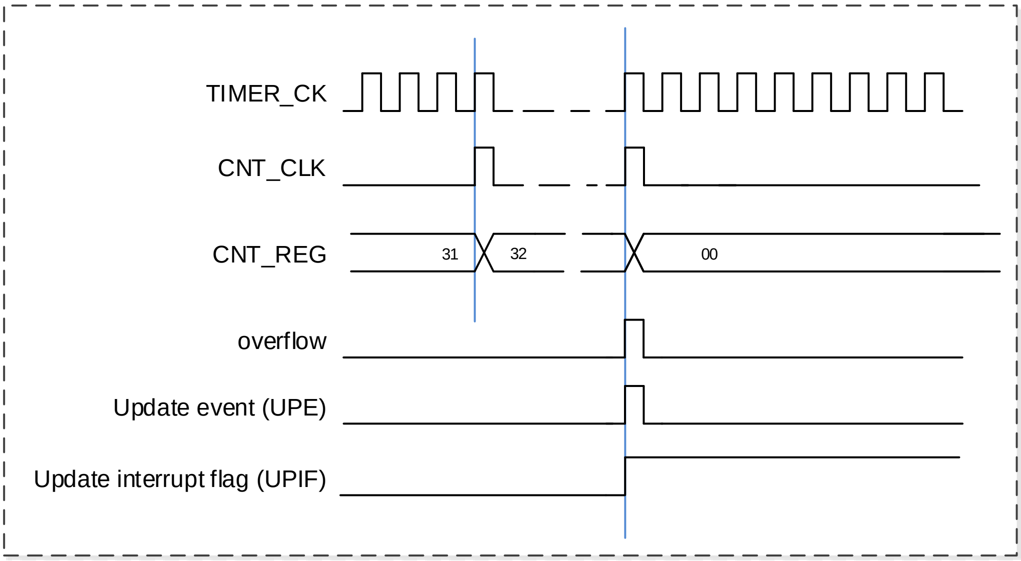


**Figure 9-139. Counter timing diagram, update event when ARSE=0**

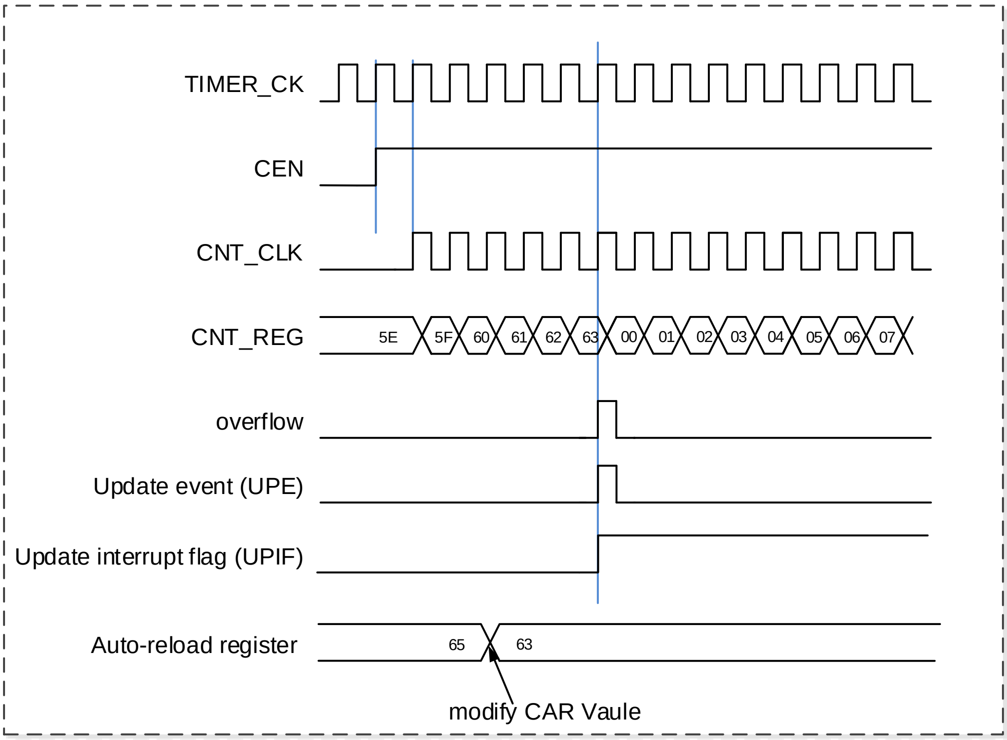
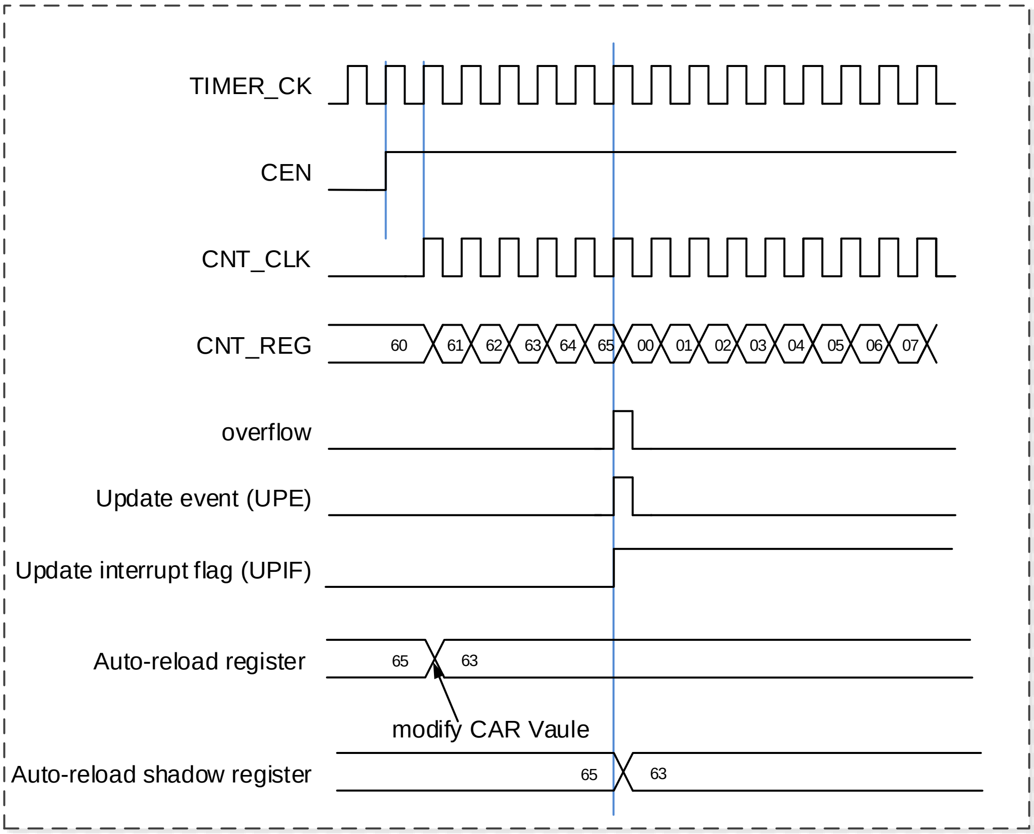


**Figure 9-140. Counter timing diagram, update event when ARSE=1**



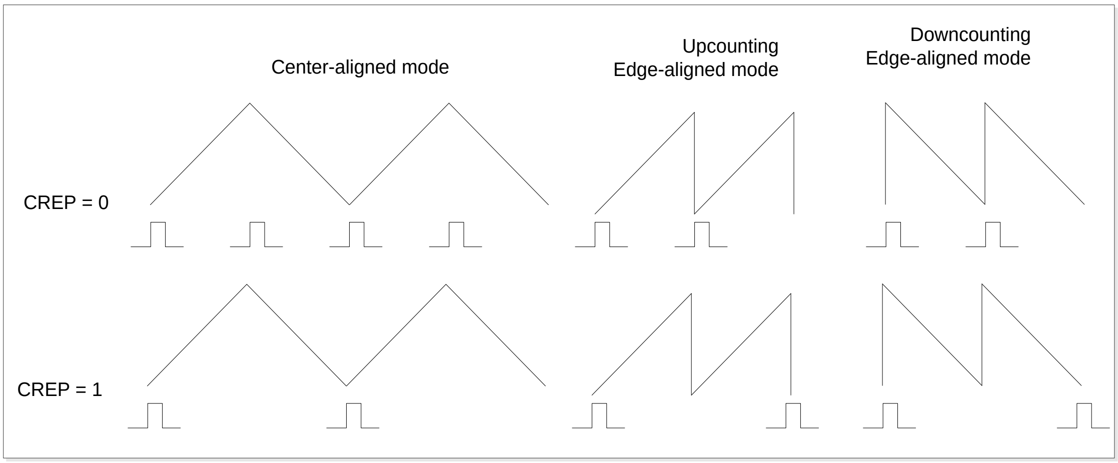
### Counter Repetition

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, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned 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 9-141. Update rate examples depending on mode and TIMERx_CREP**

TIMER15/TIMER16 has only upcounting edge-aligned mode.



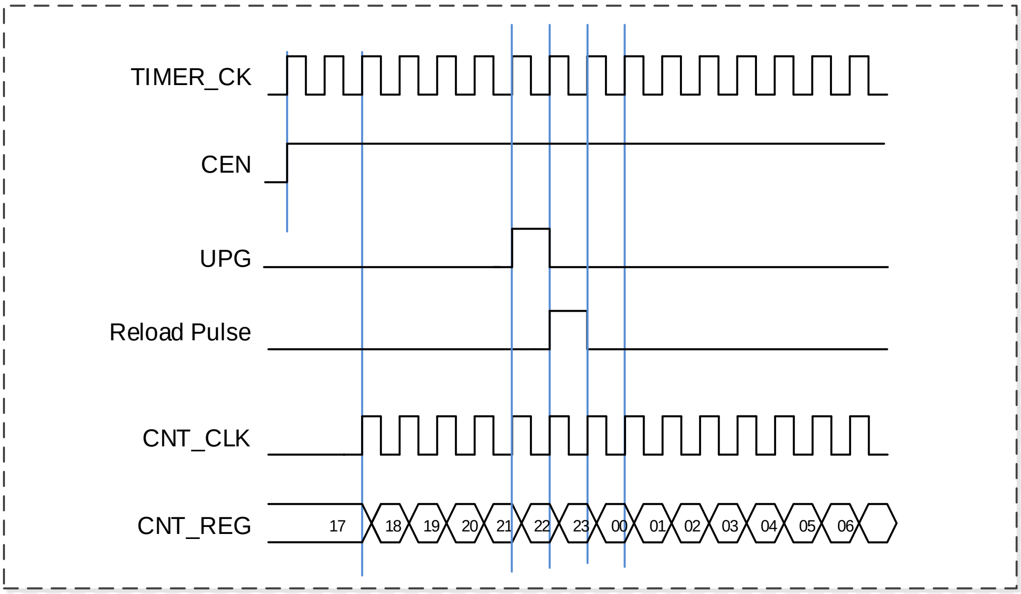
### Clock selection

The following describes the Timer Module clock controller which determines the clock source of the internal prescaler counter.

- Internal timer clock TIMER_CK

The only clock source is the TIMER_CK clock.

**Figure 9-142. Control circuit in normal mode, internal clock divided by 1**



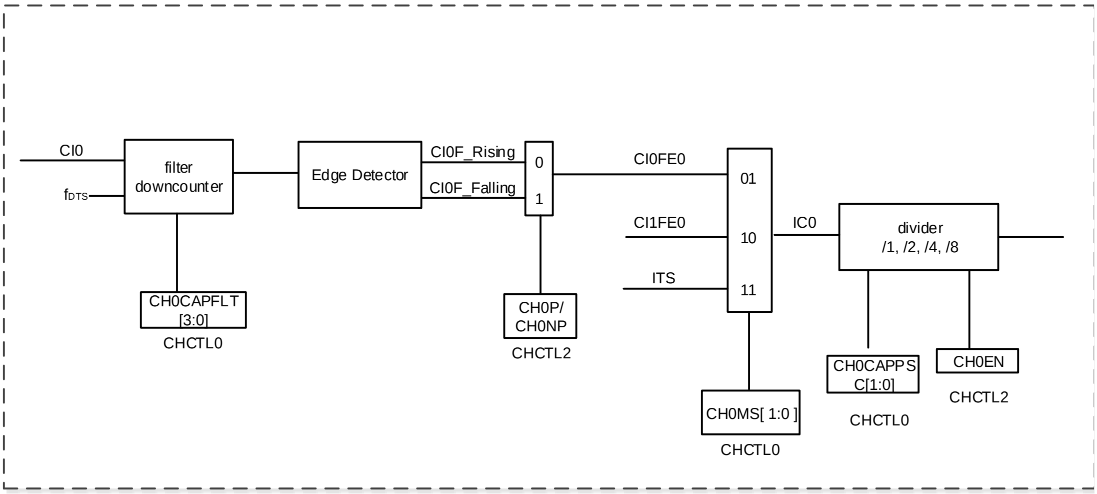
## Capture/compare channels

The TIMER15/TIMER16 has one independent channel which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture/compare value register including an input stage, channel controller and an output stage.

- Input capture stage

The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. The channel input signal (CIx) is sampled by a digital filter to generate a filtered input signal CIxF. Then the channel polarity and the edge detection block can generate a CIxFEx signal for the input capture function. The effective input event number can be set by the channel input prescaler (CHxCAPPSC).

**Figure 9-143. Capture/compare channel (example: channel 0 input stage)**



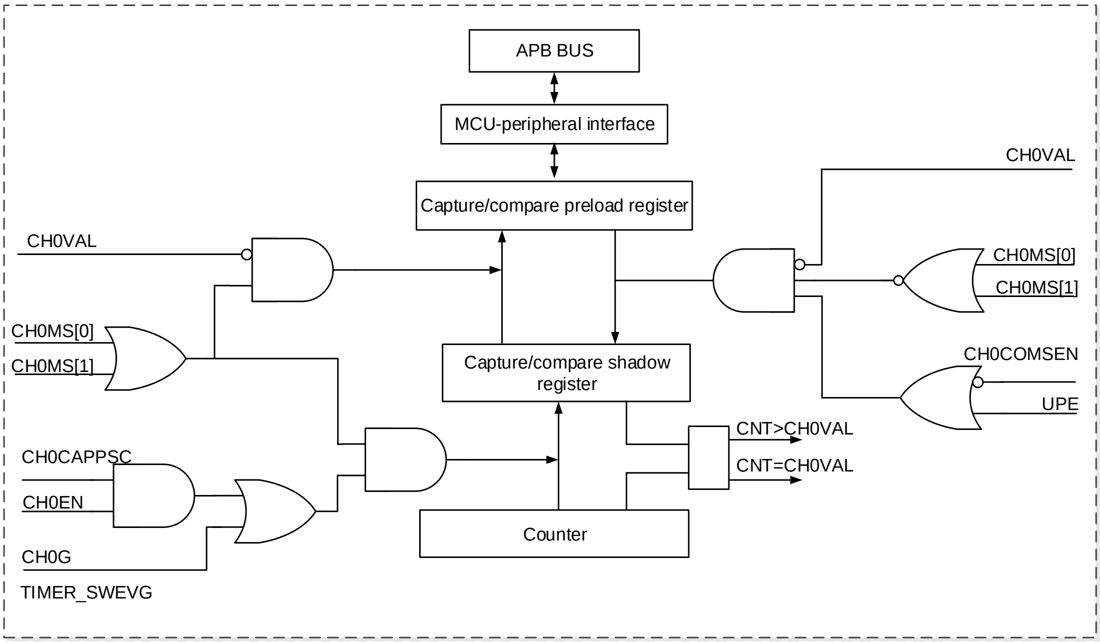
■ Channel controller

The GPTM has one independent channels which can be used as capture inputs or compare match outputs.

When used in the input capture mode, the counter value is captured into the TIMERx_CHxCV shadow register first and then transferred into the TIMERx_CHxCV preload register when the capture event occurs.

When used in the compare match output mode, the contents of the TIMERx_CHxCV preload register is copied into the associated shadow register; the counter value is then compared with the register value.

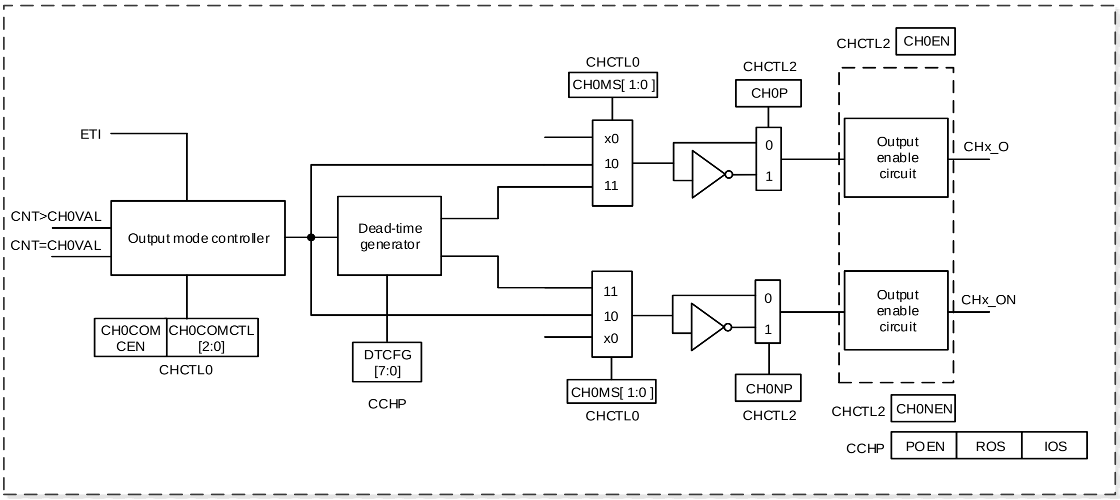
**Figure 9-144. Capture/compare channel 0 main circuit**



- Output stage

The TIMER15/TIMER16 has one channels for compare match, single pulse or PWM output function.

**Figure 9-145. Output stage of capture/compare channel (channel 0)**



## Input Capture Mode

When the channel is used as a capture input, the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV) when an effective input signal transition occurs. Once the capture event occurs, the CHxIF flag in the TIMERx_INTF register is set. If the CHxIF bit is already set, i.e., the flag has not yet been cleared by software, and another capture event on this channel occurs, the corresponding channel Over-Capture flag, named CHxOF, will be set.Once the capture event occurs, a DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit. The fllowing step maybe used to implement input capture mode:

- Select input signal by set CHxMS bits in the channel control register (TIMERx_CHCTLx).
- Add input filter to input signal by set CHxCAPFLT bitst in the the channel control register (TIMERx_CHCTLx).
- Select input capture edge (rising or falling) by set CHxP/CHxNP bit in the channel control register 2 (TIMERx_CHCTL2).
- If input signal need prescaler, set CHxCAPPSC bits in the the channel control register (TIMERx_CHCTLx).
- If need interrupt, set CHxIE bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- If need DMA request, set CHxDEN bit in DMA and interrupt enable register (TIMERx_DMAINTEN) to 1.
- Enable the channel by set CHxEN bit in the channel control register 2 (TIMERx_CHCTL2) to 1.

After configure this registers, the input capture event on this channel may occurs at the corresponding edge. And the counter value is captured into the Channel Capture/Compare Register (TIMERx_CHxCV). An interrupt generated if CHxIE bit set. A DMA request generated if CHxDEN set.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins (CIx). For example, PWM signal connect to CI0 input. Select channel 0 capture signal to CI0 by setting CH0MS to 01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 10 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.

### Output Compare Mode/PWM Mode

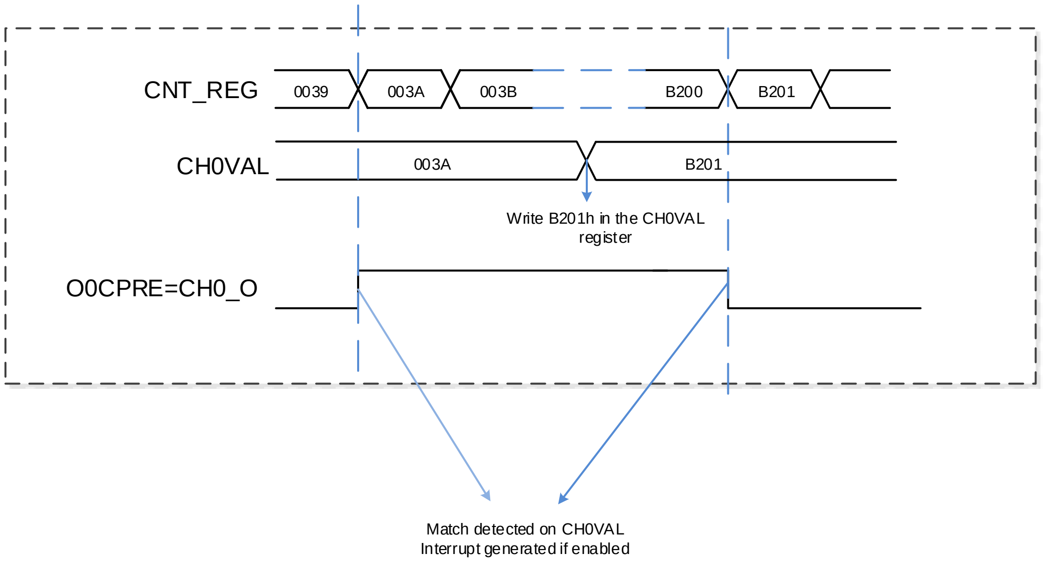
When the channel is used as an output mode by setting the CHxMS in the the channel control register (TIMERx_CHCTLx) to 00, the channel outputs waveform according to the CHxCOMCTL bits in the the the channel control register (TIMERx_CHCTLx), and the comparision between the counter and TIMERx_CHxCV registers. When the comparision equals, the CHxIF flag in the Interrupt flag registers (TIMERx_INTF) set to 1. A DMA request is generated depending on the CHxDEN bit and an interrupt is generated depending on the CHxIE bit.

In output mode, the channel can outputs active level by set the CHxCOMCTL to 101, and outputs inactive level by set the CHxCOMCTL bits to 100.

In ouput mode, the channel output set to active level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 001. The channel output set to inactive level when the comparision between the counter and TIMERx_CHxCV registers match, by set the CHxCOMCTL to 010.

In output compare toggle mode (by set the CHxCOMCTL to 011), the channel output toggles when the comparision between the counter and TIMERx_CHxCV registers match.

**Figure 9-146. Output compare toggle mode, toggle on CH0_O**



In the output PWM mode (by setting the CHxCOMCTL bits to 110 (PWM mode 0) or to 111 (PWM mode 1)), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

**Figure 9-147. Output compare PWM mode 0 on CH0_O, upcounting mode**

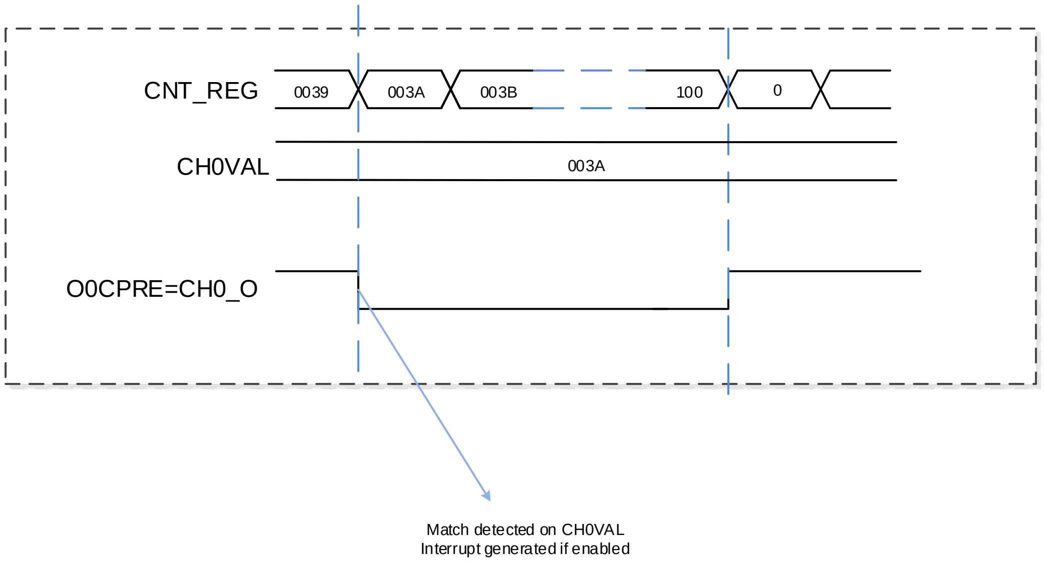
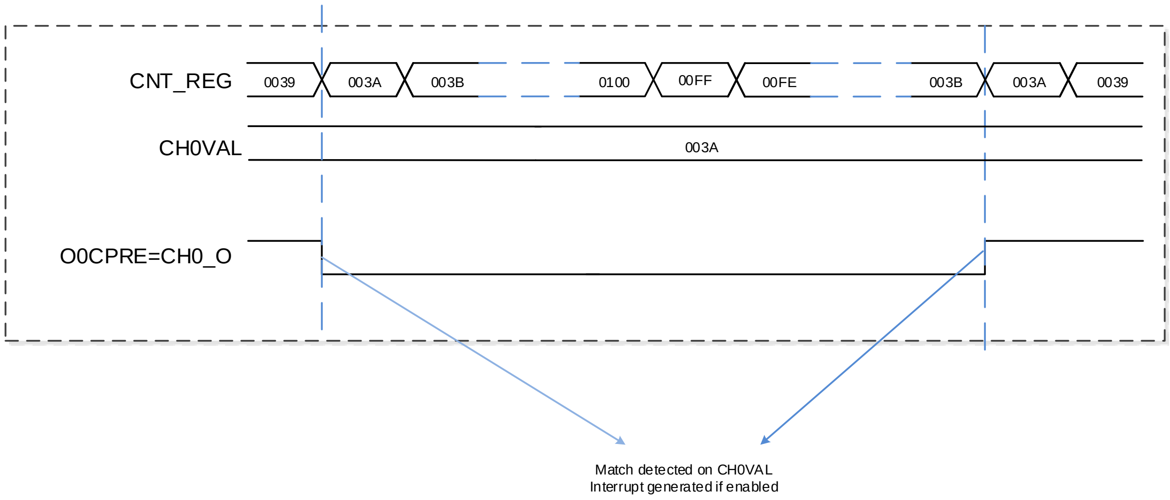


**Figure 9-148. Output compare PWM mode 0 on CH0_O, center-aligned counting mode**



### Channel Output Reference Signal

When the TIMER15/TIMER16 is used in the compare match output mode, the OxCPRE signal (Channel x Output Reference signal) is defined by setting the CHxCOMCTL bits. 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 detailed 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.

The OxCPRE signal can be forced to 0 when the ETIF signal is derived from the external TIMERx_ ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs Complementary and Dead-time

The TIMER15/TIMER16 can output two complementary signals can output by TIMER15/TIMER16. The complementary signals CHx_O and CHx_ON are activated by a combination of several control bits: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register and the POEN, ISOx, ISOxN, IOS and ROS bits in the TIMERx_CCHP and TIMERx_CTL1 registers.The outputs polarity are determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register

If CHxEN, CHxNEN and POEN bits are 1, dead-time should insertion. The rising edge of CHx_O output is delayed relative to the rising edge of OxCPRE and the rising edge of CHx_ON output is delayed relative to the falling edge of OxCPRE.The delay value is a 8-bit dead-time counter determined by DTCFG field in TIMERx_CCHP register.If the delay value is greater than the width of the active output (CHx_O or CHx_ON) then the corresponding pulse is not generated.

**Figure 9-149. Complementary output with dead-time insertion.**

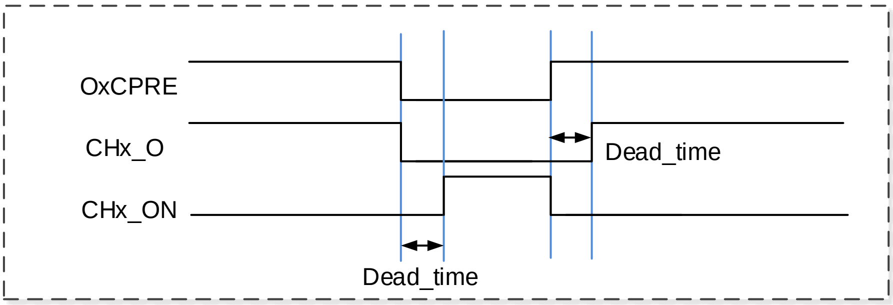


**Figure 9-150. Dead-time waveforms with delay greater than the negative pulse**

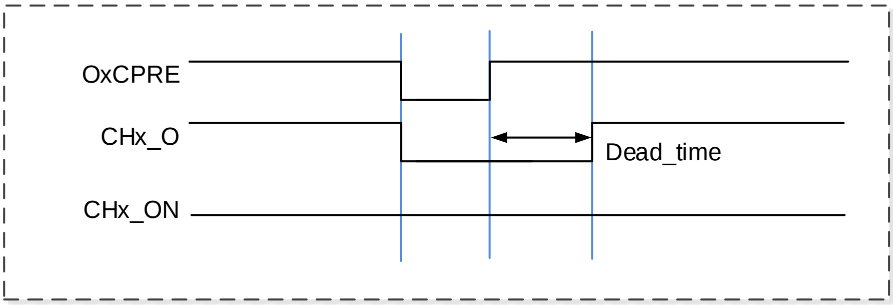
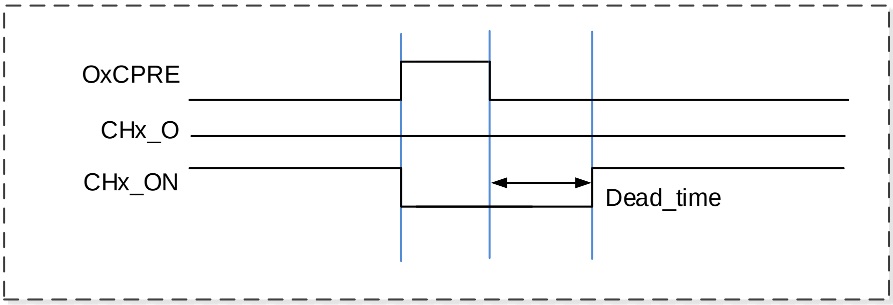


Figure 9-151. Dead-time waveforms with delay greater than the positive pulse



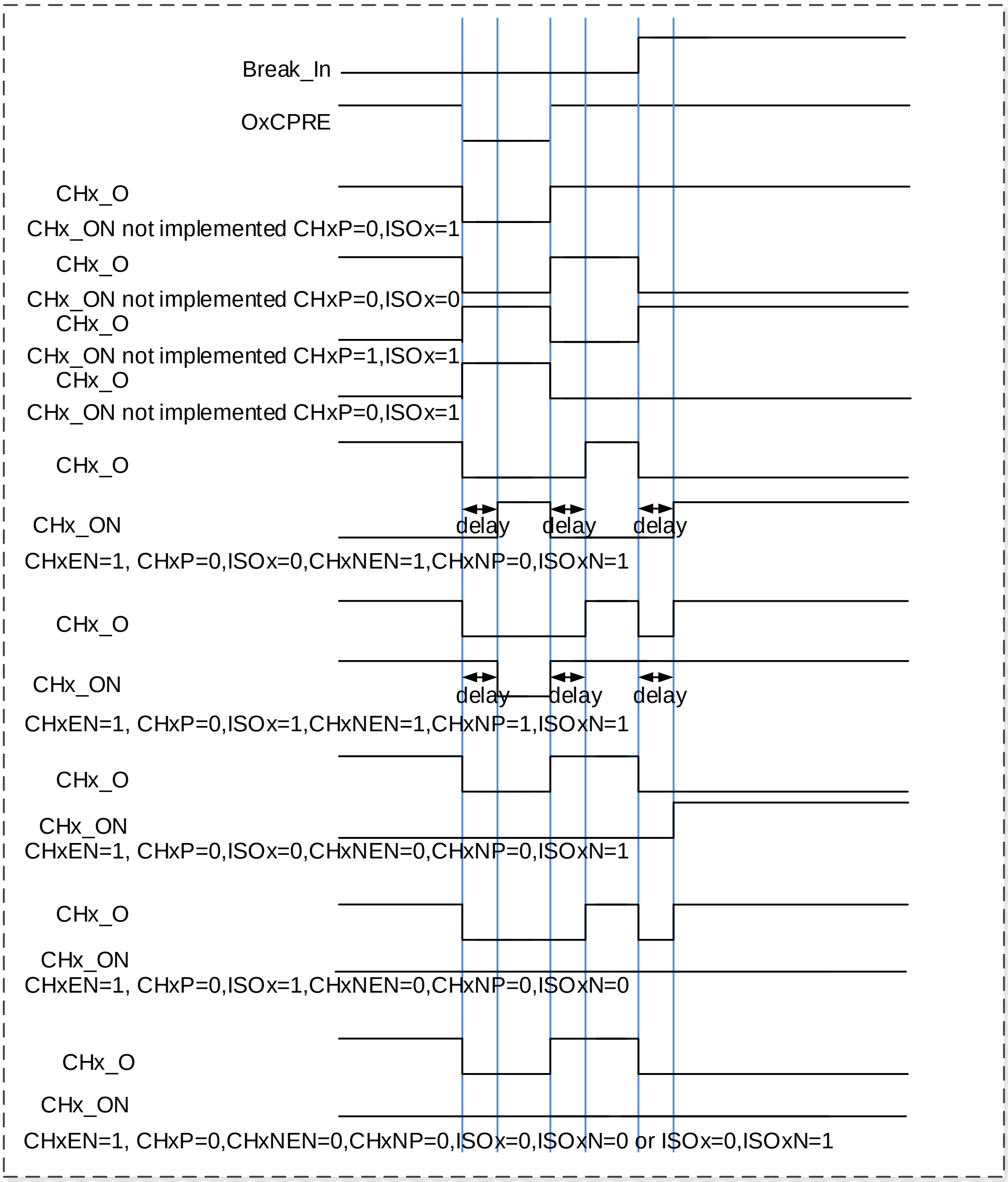
### Break function

In this function, 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 or HXTAL stack 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 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.

**Figure 9-152. Output behavior in response to a break（The break input is acting on high level）**

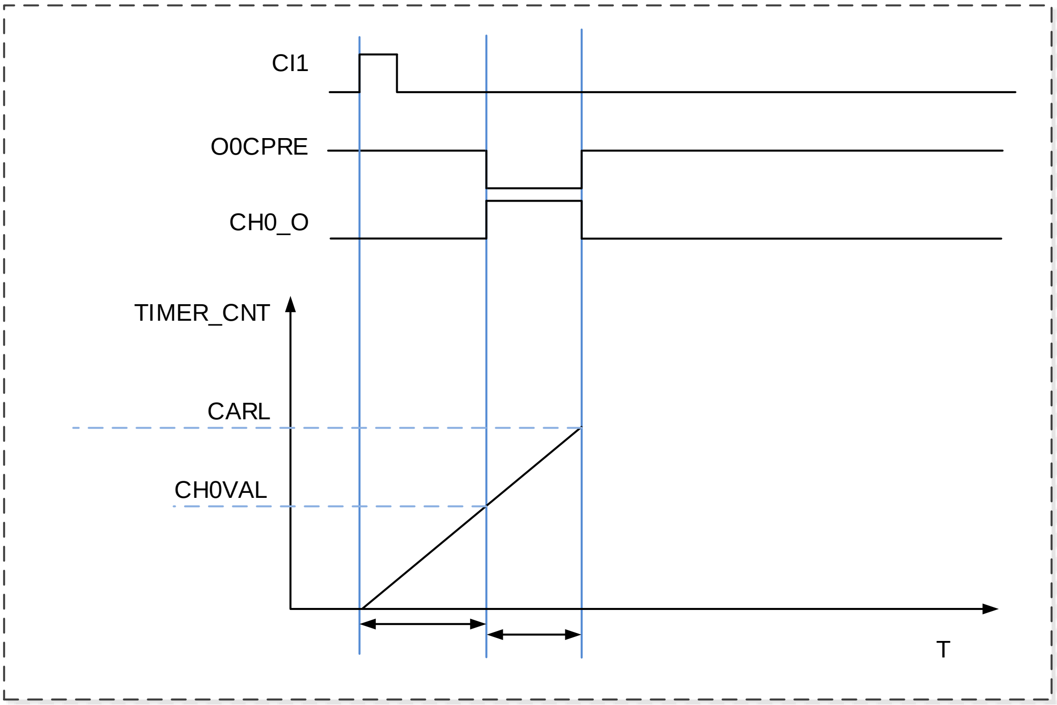

Break_In
OxCPRE
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=0
CHx_O
CHx_ON not implemented CHxP=1,ISOx=1
CHx_O
CHx_ON not implemented CHxP=0,ISOx=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=1,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
delay
delay
delay
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=1,CHxNP=1,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=0,CHxNEN=0,CHxNP=0,ISOxN=1
CHx_O
CHx_ON
CHxEN=1, CHxP=0,ISOx=1,CHxNEN=0,CHxNP=0,ISOxN=0
CHx_O
CHx_ON
CHxEN=1, CHxP=0,CHxNEN=0,CHxNP=0,ISOx=0,ISOxN=0 or ISOx=0,ISOxN=1

## Single Pulse Mode

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. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

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 CHxOFE bit in each TIMERx_CHCTL0 register. After a trigger rising edge 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 CHxOFE 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 9-153. Single pulse mode**



## Timer debug mode

When the Cortex™-M3 halted, and the DBG_TIMERx_STOP configuration bit in DBG module set to 1, the TIMERx counter stops.

## 9.6.4. TIMER15/TIMER16 registers

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:10 | 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 ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.<br>00: $f_{DTS}=f_{TIMER\_CK}$<br>01: $f_{DTS}= f_{TIMER\_CK} /2$<br>10: $f_{DTS}= f_{TIMER\_CK} /4$<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 TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value |
| 3 | SPM | Single pulse mode.<br>0: Counter continues after update event.<br>1: The CEN is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: When enabled, any of the following events generate an update interrupt or DMA request:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request. |
| 1 | UPDIS | Update disable.<br>This bit is used to enable or disable the update event generation. |

| | | |
|---|---|---|
| | | 0: Update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set<br>– The counter generates an overflow or underflow event<br>– The slave mode controller generates an update event.<br>1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UPG bit is set or if the slave mode controller generates a hardware reset event. |
| 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 encoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | ISO0N | ISO0 | Reserved | | | | DMAS | CCUC | Res. | CCSE |
| | | | | | | rw | rw | | | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:11 | 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<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. |
| 7:4 | Reserved | Must be kept at reset value |
| 3 | DMAS | DMA request source selection<br>0: DMA request of channel x is sent when capture/compare event occurs<br>1: DMA request of channel x is sent when update event occurs |
| 2 | CCUC | Commutation control shadow register update control |

| | | |
|---|---|---|
| | | When the Commutation control shadow shadow registers (for CHxEN, CHxNEN and CHxCOMCTL bits) are enabled (CCSE=1), this bit control when these shadow registers update.<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 an 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 register 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.After these bits have been written, they are updated only when CMTG bit is set<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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CH0DEN | UPDEN | BRKIE | Res. | CMTIE | Res. | | | CH0IE | UPIE |
| | | | | | | rw | rw | rw | | rw | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | Reserved | Must be kept at reset value |
| 9 | CH0DEN | Channel 0 Capture/Compare DMA request enable<br>0: Channel 0 Capture/Compare DMA request disabled<br>1: Channel 0 Capture/Compare DMA request enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Update DMA request disabled<br>1: Update DMA request enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Break interrupt disabled<br>1: Break interrupt enabled |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CMTIE | CMT interrupt enable<br>0: CMT interrupt disabled<br>1: CMT interrupt enabled |

| | | |
|---|---|---|
| 4:2 | Reserved | Must be kept at reset value |
| 1 | CH0IE | Channel 0 Capture/Compare interrupt enable<br>0: Channel 0 Capture/Compare interrupt disabled<br>1: Channel 0 Capture/Compare interrupt enabled |
| 0 | UPIE | Update interrupt enable<br>0: Update interrupt disabled<br>1: Update interrupt enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CH0OF | Res. | BRKIF | Res. | CMTIF | Reserved | | | CH0IF | UPIF |
| | | | | | | rc_w0 | | rc_w0 | | rc_w0 | | | | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | Reserved | Must be kept at reset value |
| 9 | CH0OF | Channel 0 Capture overflow 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 Capture overflow interrupt occurred<br>1: Capture overflow interrupt occurred |
| 8 | Reserved | Must be kept at reset value |
| 7 | BRKIF | Break interrupt flag<br>This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active.<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 CMT event occurs, and cleared by software<br>0: No CMT interrupt occurred<br>1: CMT interrupt occurred |
| 4:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IF | Channel 0 interrupt flag |

| | | |
|---|---|---|
| | | 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BRKG | Reserved. | CMTG | Reserved | | | CH0G | UPG |
| | | | | | | | | w | | w | | | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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 the channel control registers (CHxEN, CHxNEN and CHxCOMCTL bits) of the channels having a complementary output are updated.<br>0: No generate CMT event<br>1: Generate CMT 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 0 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 0 capture or compare event<br>1: Generate a channel 0 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 upcounting mode is selected, else (downcounting) 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | CH0COMFEN | CH0MS[1:0] | |
| | | | | | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| | | | | | | | | rw | | | | rw | | | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable.<br>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.<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 behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.<br>000: Frozen.The O0CPRE signal keep stable, independent of the comparison between the output compare register TIMERx_CH0CV and the counter.<br>001: Set high on match.O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>010: Set low on match. O0CPRE signal is forced low when the counter matches the c output compare register TIMERx_CH0CV.<br>011: Toggle on match. O0CPRE toggles when the counter matches the c output |

| | | |
|---|---|---|
| | | compare register TIMERx_CH0CV.<br>100: Force low. O0CPRE is forced low level.<br>101: Force high. O0CPRE is forced high level.<br>110: PWM mode 0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high.<br>111: PWM mode 1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low.<br>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison 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. |
| 3 | CH0COMSEN | Channel 0 output compare 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 validating the shadow register only in one pulse mode (SPM bit set in TIMERx_CTL0 register is set).<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 compare fast disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br>1: Channel 0 output compare fast enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles. |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 0 is configured as output<br>01: Channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: Channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG |

register.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disable, $f_{SAMP}$=$f_{DTS}$, N=1<br>0001: $f_{SAMP}$=$f_{TIMER\_CK}$, N=2<br>0010: $f_{SAMP}$= $f_{TIMER\_CK}$, N=4<br>0011: $f_{SAMP}$= $f_{TIMER\_CK}$, N=8<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8 |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the ratio of the prescaler on channel 0 input.The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is reset.<br>00: prescaler disable, capture is done on each channel input edge<br>01: capture is done every 2 channel input edges<br>10: capture is done every 4 channel input edges<br>11: capture is done every 8 channel input edges |
| 1:0 | CH0MS[1:0] | Channel 0 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 OFF (CH0EN bit in TIMERx_CHCTL2 register is reset).<br>00: channel 0 is configured as output<br>01: channel 0 is configured as input, IC0 is connected to CI0FE0<br>10: Channel 0 is configured as input, IC0 is connected to CI1FE0<br>11: channel 0 is configured as input, IC0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CH0NP | CH0NEN | CH0P | CH0EN |
| | | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:4 | 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>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 channel 0.<br>0: Channel 0 complementary output disabled<br>1: Channel 0 complementary output enabled |
| 1 | CH0P | Channel 0 polarity<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>This bit cannot be modified when PROT[1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 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 channel 0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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

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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC 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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CAR[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

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | 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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CH0VAL[15:0] | Capture or compare value of channel 0<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 Complementary Protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15 | POEN | Primary output enable<br>This bit s set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware assoon as the break input is active. When a channel 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: Channel outputs are disabled or forced to idle state.<br>1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN can be not set by hardware.<br>1: POEN can be set by hardware automatically at the next update event,if the break input is notactive.<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 BRK input signal.<br>0: BRK input active low<br>1: BRK input active high |
| 12 | BRKEN | Break enable<br>This bit can be set to enable the BRK and CCS clock failure event inputs.<br>0: Break inputs disabled<br>1: Break inputs enabled<br>This bit can be modified onlywhen PROT[1:0] bit-filed in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode off-state configure<br>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.<br>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled<br>1: When POEN bit is set, he channel output signals (CHx_O/CHx_ON)are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register<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 configure<br>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode.<br>0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are |

| | | |
|---|---|---|
| | | disabled.<br>1: When POEN bit is reset, he channel output signals (CHx_O/CHx_ON)are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.<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.<br>00: protect disable. No write protection.<br>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.<br>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.<br>11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTLx registers (if the related channel is configured in output) are writing protected.<br>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<br>This bit-field specifies the duration of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG[7:5]=0xx: duration =DTCFG[7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG[7:5]=10x:duration =(64+DTCFG[5:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*2.<br>DTCFG[7:5]=110: duration =(32+DTCFG[4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG[7:5]=111:duration =(32+DTCFG[4:0])x$t_{DT}$,$t_{DT}$=$t_{DTS}$*16.<br>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

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

| 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 |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value |
| 12:8 | DMATC[4:0] | DMA transfer count |

| Bits | Fields | Descriptions |
|---|---|---|
| | | When register access are done through the TIMERx_DMATB address, this 5-bit bit-field specifies the number of transfers. |
| 7:5 | Reserved | Must be kept at reset value |
| 4:0 | DMATA[4:0] | DMA transfer access start address<br>When register access are done through the TIMERx_DMATB address, this bit-field specifies the offset of the starting address from the TIMERx_CTL0 register. |

### DMA Transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C
Reset value: 0x0000

This register can be accessed by half-word(16-bit) or 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 |
|---|---|---|
| 15:0 | DMATB[15:0] | DMA transfer<br>When a read or write operation is assigned to this register, the register located at the address range (DMATA + burst counter) x 4 from TIMERx_CTL0 will be accessed.<br>The burst counter is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC
Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:2 | Reserved | Must be kept at reset value |
| 1 | CCSEL | Write Capture/Compare register selection<br>This bit-field set and reset by software.<br>1: If write the Capture/Compare register, the write value is same as the Capture/Compare value, the write access ignored. |

| | | |
|---|---|---|
| | | 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 |

# 10. Infrared ray port (IFRR)

## 10.1. Introduction

Infrared ray port (IFRR) 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.

## 10.2. Main features

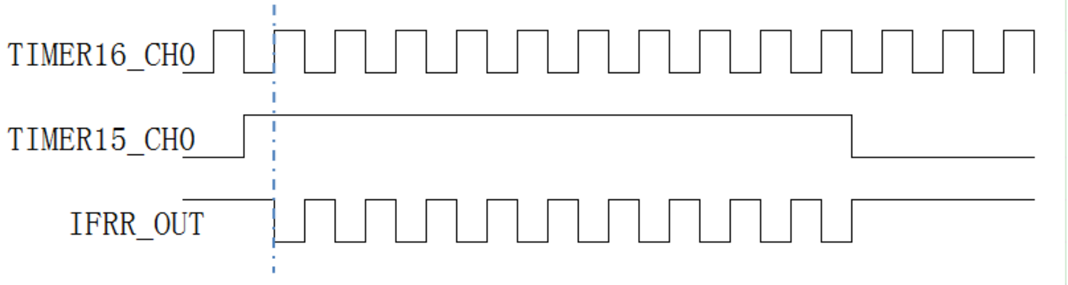
- The IFRR output is enabled by enabling PB9 pin high current capability(set PB9_HCCE)
- The IFRR output signal is decided by TIMER15_CH0 and TIMER16_CH0
- To get correct infrared ray signal, TIMER15 should generate low frequence modulation envelope signal, and TIMER16 should generate high frequence carrier signal
- PB9 should provide high current to control LED interface

## 10.3. Function description

IFRR is a module which could integrate the output of TIMER15 and TIMER16 to generate an infrared ray signal.

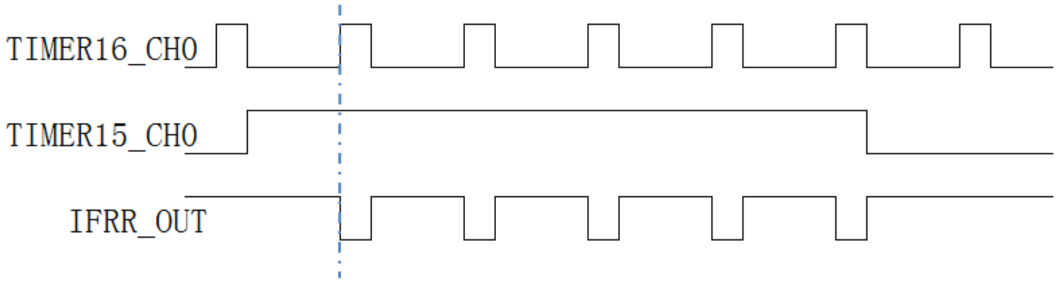
1. The TIMER15's CH0 is programed to generate the low frequence PWM signal which is the modulation evalope signal. The TIMER16's CH0 is programed to generate the high frquence PWM signal which is the carrier signal. And the channel is needed to enable before generating these signals.
2. Program the GPIO remap regisger and enable the pin.
3. If you want to get the high current capacity of output, remapping IFRR_OUT to PB9 and setting the PB9 to Fast Mode by the register in SYS_CFG module are required.

**Figure 10-1. IFRR output timechart 1**



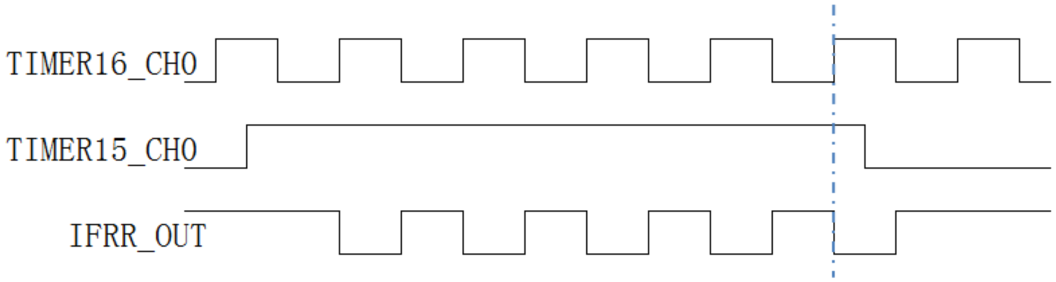
**Note:** IFRR_OUT has one APB clock delay from TIMER16_CH0.

**Figure 10-2. IFRR output timechart 2**



**Note:** Carrier(TIMER15_CH0)'s duty cycle can be changed, and IFRR_OUT has inverted relationship with TIMER16_CH0 when TIMER15_CH0 is high.

**Figure 10-3. IFRR output timechart 3**



**Note:** IFRR_OUT will keep the integrity of TIMER16_CH0, even if evelope signal (TIMER15_CH0) is no active.

# 11. Watchdog timer (WDGT)

The Watchdog Timer (WDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. The GD32F1x0 has two watchdog peripherals, Free watchdog timer and Window watchdog timer. They offer a combination of a high safety level, flexibility of use and timing accuracy. Both watchdog peripherals are offered to resolve malfunctions of software. The Watchdog will generate a reset (or an interrupt in Window watchdog timer) when the counter reaches a given value.

The Watchdog Timer counter can be stopped while the processor is in the debug mode. The register write protection function in Free watchdog timer can be enabled to prevent it from changing the configuration unexpectedly.

## 11.1. Free watchdog timer (FWDGT)

### 11.1.1. Introduction

The Free watchdog timer (FWDGT) has independent clock source (IRC40K). There upon 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 Window watchdog timer (WWDGT) is suitable for the situation that requires an accurate timing.

In addition, the FWDGT has a configurable window value, if the software reloads the counter before the counter reaches the window value, a reset will be generated.

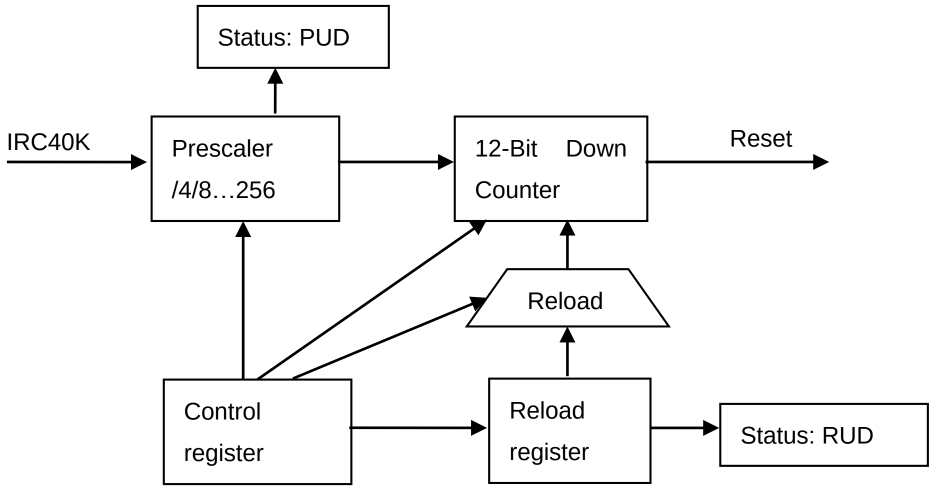
### 11.1.2. Main features

- Free-running 12-bit downcounter.
- Reset when the downcounter reaches 0, if the watchdog is enabled.
- Reset when the downcounter is reloaded outside the window, if watchdog activated.
- Independent clock source，FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog timer bit, automatically start the FWDGT at power on or not depending on this bit.
- FWDGT debug mode congfig bit, the FWDGT continus to work or stoped in debug mode depends on this bit.

### 11.1.3. Function description

The free watchdog timer consists of an 8-stage prescaler and a 12-bit down-counter. Refer to the figure below for the functional blocks of the free watchdog timer module.

**Figure 11-1. Free watchdog timer block diagram**



The free watchdog timer is enabled by writing the value 0xCCCC in the control register (FWDGT_CTL), and the counter starts counting down. When the counter reaches the value 0x000, a reset is generated.

Reload the counter by writing the value 0xAAAA in the control register anytime, the reload value is come from the FWDGT_RLD register. The software prevents 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 timer can automatically start at power on when the hardware free watchdog timer bit in the device option bits is set. Then the software should reload the counter before the counter reaches 0x000.

The FWDGT_PSC register, the FWDGT_RLD register and the FWDGT_WND register are write protected. Before writing these registers, the software should write the value 0x5555 in the control register. These registers will be protected again by writing any other value in the control register. When an update of the prescaler (FWDGT_PSC) or watchdog counter window value (FWDGT_WND) or the reload value (FWDGT_RLD) is on going, the status bit in FWDGT_STAT register is set.

If the FWDGT_HOLD bit in DBG module is cleared, the FWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

**Table 11-1. 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 | 0.1 | 409.6 |
| 1/8 | 001 | 0.2 | 819.2 |
| 1/16 | 010 | 0.4 | 1638.4 |
| 1/32 | 011 | 0.8 | 3276.8 |
| 1/64 | 100 | 1.6 | 6553.6 |
| 1/128 | 101 | 3.2 | 13107.2 |
| 1/256 | 110 or 111 | 6.4 | 26214.4 |

The FWDGT timeout can be more accurately by calibrating the IRC40K.

## 11.1.4. FWDGT registers

### 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] | | | | | | | | | | | | | | | |
| wo | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CMD[15:0] | Write only. These bits have different fuctions when writing 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 timer 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. When a write operation to this register ongoing, the PUD bit in the FWDGT_STAT register is set and the value read from this register is invalid.<br>000: 1/4 001: 1/8 010: 1/16<br>011: 1/32 100: 1/64 101: 1/128<br>110: 1/256 111: 1/256<br>If several prescaler values are used by the application, it is mandatory to wait until PUD bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PUD is reset before continuing code execution except in case of low-power mode entry. |

### 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

| 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 | | | | RLD [11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| 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.<br>These bits are write-protected. Write 0X5555 in the FWDGT_CTL register before writing these bits. When a write operation to this register ongoing, 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 is reset before changing the reload value. However, after updating the reload value it is not necessary to wait until RUD is reset before continuing code execution except in case of low-power mode entry.

### 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 |
| | | | | | | | | | | | | | ro | ro | ro |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | 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>When a write operation to FWDGT_RLD register ongoing, this bit is set and the value read from FWDGT_RLD register is invalid. |
| 0 | PUD | Free watchdog timer prescaler value update<br>When a write operation to FWDGT_PSC register ongoing, this bit is set and the value read from FWDGT_PSC register is invalid. |

### Window register (FWDGT_WND)

Address offset: 0x10

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 | | | | WND[11:0] | | | | | | | | | | | |
| | | | | r | | | | | | | | | | | |

| 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.<br>These bits are write protected. Write 5555h in the FWDGT_CTL register before writing these bits.<br>If several window values are used by the application, it is mandatory to wait until WUD bit is 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. |

## 11.2. Window watchdog timer (WWDGT)

### 11.2.1. Introduction

The Window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions.After the Window watchdog timer start, the 7-bit downcounter reduce progressively. The watchdog causes a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared). The watchdog also cause a reset if 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 has an early wakeup status flag when the counter reaches 0x40. Interrupt occurs if it is enabled

The Window watchdog timer (WWDGT) clock is prescaled from the APB1 clock.

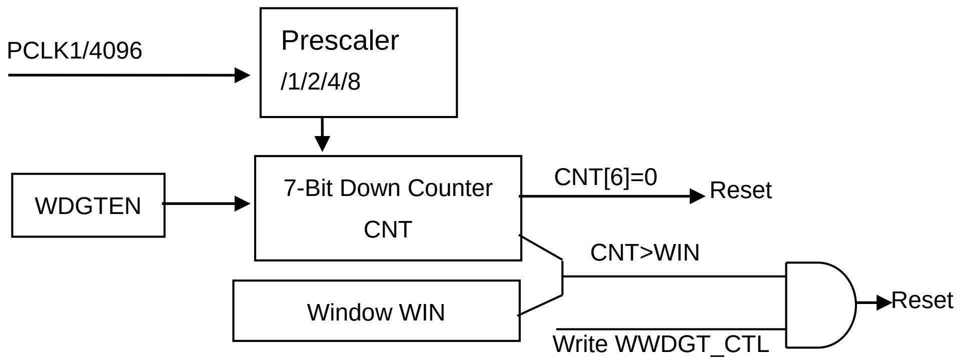
### 11.2.2. Main features

- Programmable free-running 7-bit downcounter
- Conditional reset, 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 occurred when the counter reached 0x40
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

### 11.2.3. Function description

If the Window watchdog timer is enable (set the WDGTEN bit in the WWDGT_CTL), the watchdog cause a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared), or when the counter is refreshed before the counter reached the window register value.

**Figure 11-2. Window watchdog timer block diagram**



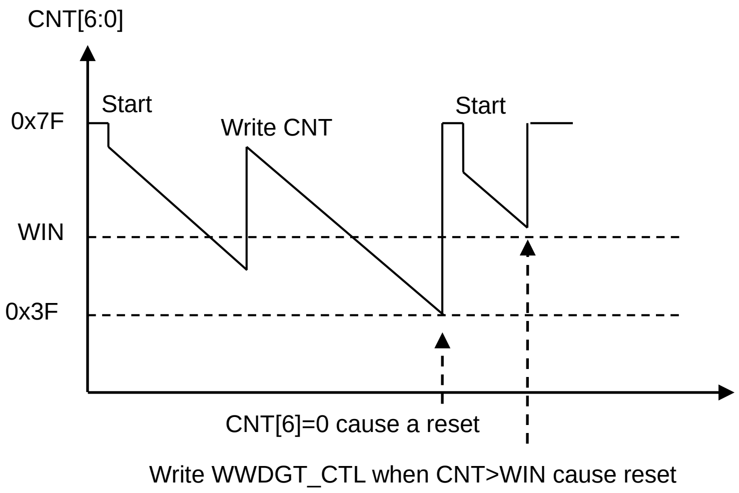
The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDGTEN bit in the WWDGT_CTL register. Whenever Window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, it implies that the CNT[6] bit should be set. The CNT[5:0] determine the maximum time interval of two reloading. The countdown 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 downcounter when 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 occur when the counter reaches 0x40. The software can do something such as communications 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 used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDGT_STAT register.

**Figure 11-3. Window watchdog timer timing diagram**



Calculate the WWDGT timeout by using the formula below.

$t_{WWDGT} = t_{PCLK1} \times 4096 \times 2PSC \times (CNT[5:0] + 1)$ (ms)

where:

$t_{WWDGT}$: WWDGT timeout

$t_{PCLK1}$: APB1 clock period measured in ms

Refer to the table below for the minimum and maximum values of the $t_{WWDG}$.

**Table 11-2. Min-max timeout value at 36 MHz (fPCLK1)**

| Prescaler divider | PSC[1:0] | Min timeout value CNT[6:0] =0x40 | Max timeout value CNT[6:0]=0x7F |
|---|---|---|---|
| 1/1 | 00 | 113 μs | 7.28 ms |
| 1/2 | 01 | 227 μs | 14.56 ms |
| 1/4 | 10 | 455 μs | 29.12 ms |
| 1/8 | 11 | 910 μs | 58.25 ms |

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

## 11.2.4. WWDGT registers

### 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 will occur when the value of this counter decreases from 0x40 to 0x3F. Writing this counter when the value of this counter is greater than the window value also cause 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. An interrupt will occur when the counter reaches 0x40 if the bit is set. It's could be cleared by a hardware reset or software clock reset (refer to 4.3.5. APB1 reset register ) . A write 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 will 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)

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | EWIF | Early wakeup interrupt flag. When the counter has reached 0x40, 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 to it. |

# 12. Analog to digital converter (ADC)

## 12.1. Introduction

The 12 bit ADC is an analog-to-digital converter using successive approximation approach. It has 19 multiplexed channels which are used to measure the signals from 16 external and 3 internal signal sources. Analog watchdog allows the application to detect whether the input voltage exceeds the user's set of high and low threshold. The A/D conversion of each channel can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC will be stored in a 16 bit data register in left or right aligned manner.

## 12.2. Main features

**For GD32F130xx and GD32F150xx devices:**

- 12-bit resolution
- ADC conversion time: 1.0 us (1MS/s)
- Dual clock domain architecture (APB clock and ADC clock)
- Self-calibration time: 83 ADC clock periods
- Programmable sampling time
- Configurable data alignment in data registers
- DMA request for regular data transfer
- Analog input channels: 16 external analog inputs, 1 channel for internal temperature sensor ($V_{SENSE}$), 1 channel for internal reference voltage ($V_{REFINT}$), 1 channel for monitoring external $V_{BAT}$ power supply pin
- Conversion modes: single, scan, continuous and discontinuous
- Analog watchdog
- ADC supply requirements: 2.6V to 3.6V
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{DDA}$
- Interrupt generation at the end of regular and inserted group conversions, and in case of analog watchdog
- External trigger on regular and inserted channels

**For GD32F170xx and GD32F190xx devices:**

- High performance
  - 12-bit, 10-bit, 8-bit, or 6-bit configurable resolution
  - ADC conversion time: 0.5μs for 12-bit resolution (2MS/s), about 0.43μs conversion time for 10-bit resolution, about 0.286μs for 6-bit resolution. Faster conversion time can be obtained by lowering the resolution
  - Self-calibration time: 84 ADC clock cycles
  - Programmable sampling time
  - Configurable data alignment in data registers
  - DMA request for regular data transfer
- Dual clock domain architecture (APB clock and ADC clock)
- Analog input channels
  - 16 external analog inputs
  - 1 channel for internal temperature sensor ($V_{SENSE}$)
  - 1 channel for internal reference voltage ($V_{REFINT}$)
  - 1 channel for monitoring external $V_{BAT}$ power supply pin
- Start of the conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (internal timer events from TIMER0,TIMER1, TIMER2 and TIMER14)
  - External trigger on regular and inserted channels
- Conversion modes:
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of regular and inserted group conversions, and in case of analog watchdog events
- Analog watchdog
- ADC supply requirements: from 3V to 5.5V, and typical power supply voltage is 5V.
- Oversampling

- 16-bit data register
- Oversampling ratio adjustable from 2 to 256x
- Programmable data shift up to 8-bits

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

## 12.3. Function description

Figure 12-1 and Figure 12-2 show the ADC block diagram.

**Figure 12-1. ADC module block diagram of GD32F130xx and GD32F150xx devices**

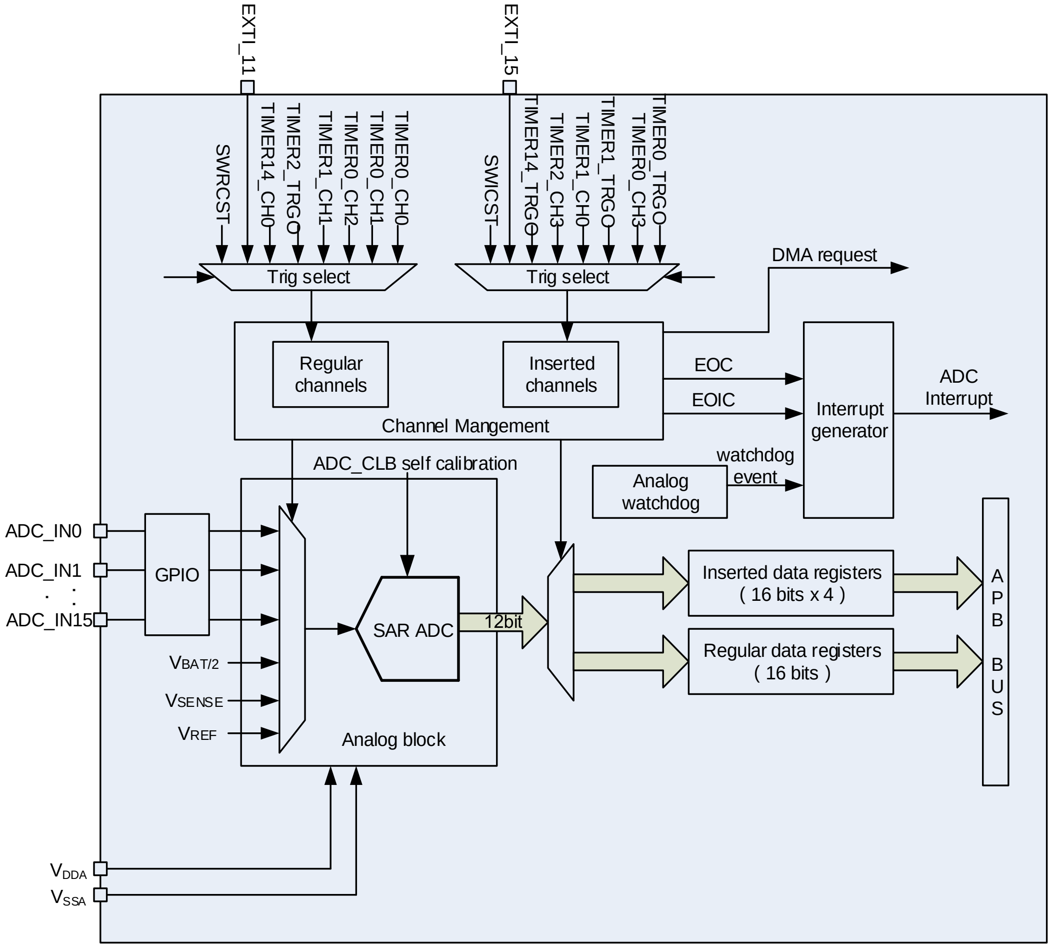


Figure 12-2. ADC module block diagram of GD32F170xx and GD32F190xx devices

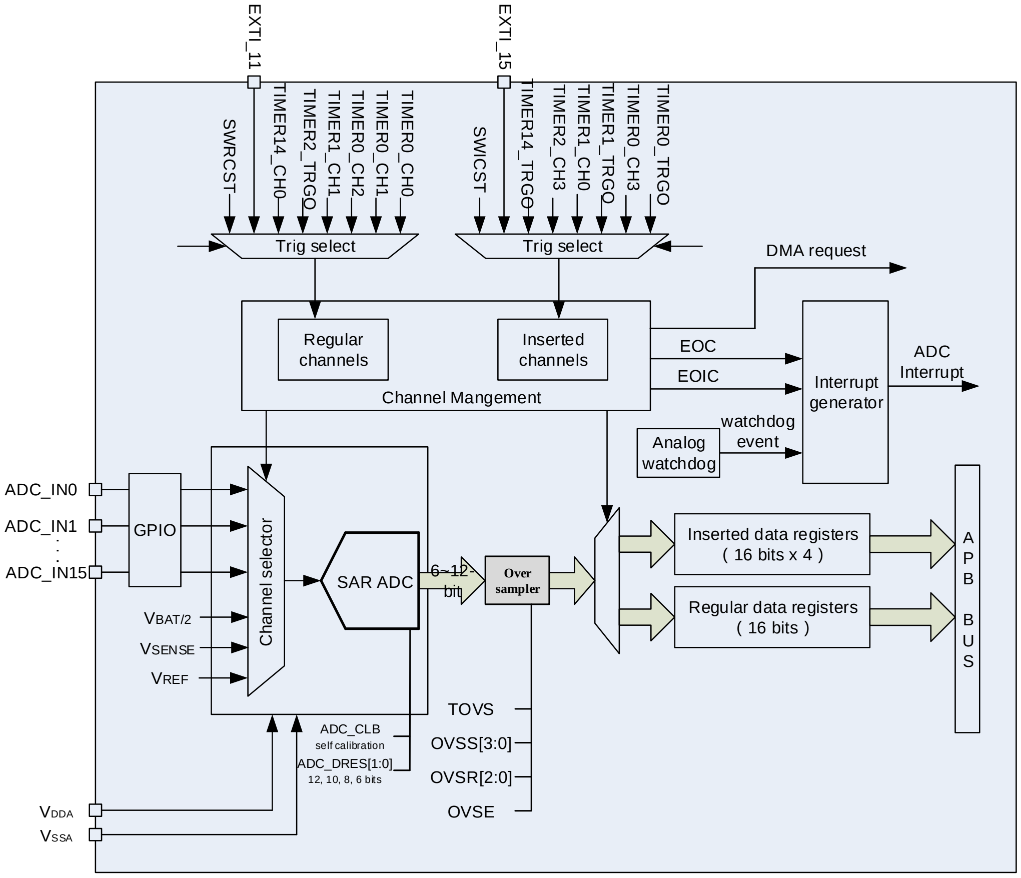


Table 12-2 and Table 12-3 give the ADC internal signals and pins description.

Table 12-1. ADC internal signals

| Internal signal name | Signal type | Description |
|---|---|---|
| $V_{SENSE}$ | Input | Internal temperature sensor output voltage |
| $V_{REFINT}$ | Input | Internal voltage reference output voltage |
| $V_{BAT}/2$ | Input | $V_{BAT}$ pin input voltage divided by 2 |

Table 12-2. ADC pins definition of GD32F130xx and GD32F150xx devices

| Name | Signal type | Remarks |
|---|---|---|
| $V_{DDA}$[1] | Input, analog supply | Analog power supply equal to $V_{DD}$ and 2.6 V ≤$V_{DDA}$≤ 3.6 V |
| $V_{SSA}$[1] | Input, analog supply ground | Ground for analog power supply equal to $V_{SS}$ |
| ADCx_IN[15:0] | Analog signals | Up to 16 analog channels |

Table 12-3. ADC pins definition of GD32F170xx and GD32F190xx devices

| Name | Signal type | Remarks |
|---|---|---|
| $V_{DDA}$[1] | Input, analog supply | Analog power supply equal to $V_{DD}$ and 3 V |

| | | |
|---|---|---|
| | | ≤ $V_{DDA}$ ≤ 5.5 V |
| $V_{SSA}$[1] | Input, analog supply ground | Ground for analog power supply equal to $V_{SS}$ |
| ADCx_IN[15:0] | Analog signals | Up to 16 analog channels |

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

### 12.3.1. Calibration (ADC_CLB)

The ADC has a calibration feature. The application must not use the ADC during calibration and must wait until it is completed. Calibration should be performed before starting A/D conversion. For GD32F130xx and GD32F150xx devices, the calibration needs 83 clock periods to remove the comparator offset error and capacitor mismatch error which may vary from chip to chip due to process variation. But it needs 84 clock periods for GD32F170xx and GD32F190xx devices.

The calibration is initiated by software by setting bit ADC_CLB=1. ADC_CLB bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration completes.

When the ADC's operating conditions change ($V_{DDA}$ changes are the main contributor to ADC offset variations and temperature changes are the secondary contributor), 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 software procedure:

1. Ensure that ADCON=1

2. Set RSTCLB (optional)

3. Set CLB=1

4. Wait until CLB =0

### 12.3.2. 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.

### 12.3.3. Regular and inserted channel groups

The ADC supports 19 multiplexed channels and organizes the conversion results into two groups: a regular channel group and an inserted channel group.

In the regular group, a sequence of up to 16 conversions can be organized in a specific sequence. The ADC_RSQ0~ADC_RSQ2 registers specify the selected channels of the regular group. The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length.

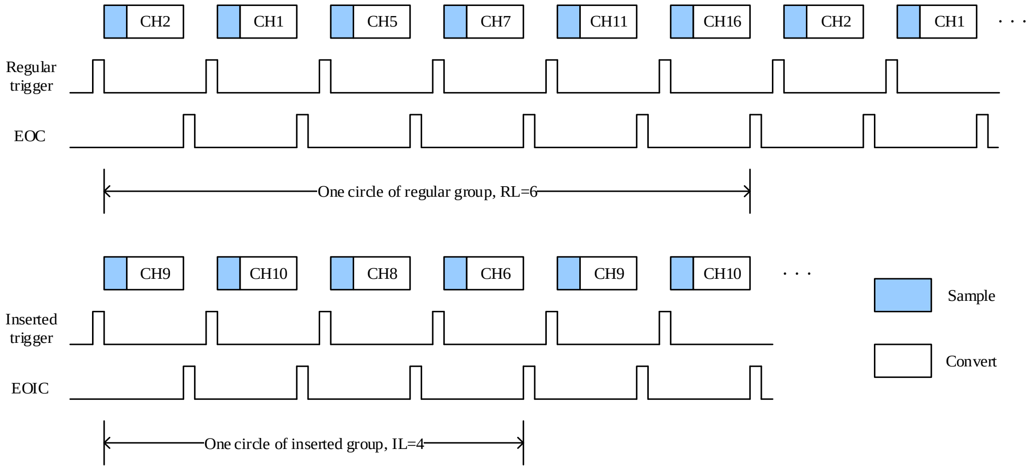
In the inserted group, a sequence of up to 4 conversions can be organized in a specific sequence. The ADC_ISQ register specifies the selected channels of the inserted group. The IL[1:0] bits in the ADC_ISQ register specify the total conversion sequence length.

## 12.3.4. Conversion modes

### Single conversion mode

In the single conversion mode, the ADC performs conversion on the channels with a specific sequence specified in the ADC_RSQ0~ADC_RSQ2 registers or ADC_ISQ register. Once the ADCON is set high, the ADC samples and converts a single channel in the regular or inserted group, when the corresponding software trigger or external trigger is active. After conversion of the single channel, the conversion data will be stored in the ADC_RDATA or ADC_IDATAx register. The EOC or EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set.

**Figure 12-3. Single conversion mode**



### Continuous conversion mode

The continuous conversion mode will be enabled when CTN bit in the ADC_CTL1 register is set. In this mode, the ADC performs conversion on the channels with a specific sequence specified in the ADC_RSQ0~ADC_RSQ2 registers or ADC_ISQ register. Once the ADCON is set high, the ADC samples and converts specified channels one by one in the regular or inserted group, when the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA or ADC_IDATAx register. If scan mode is disabled, the EOC or EOIC will be set after conversion of every channel. If scan mode is enabled, the EOC or EOIC will be set after every circle of the regular or inserted channel group. An interrupt will be generated if the EOCIE or EOICIE bit is set.

**Figure 12-4. Continuous conversion mode, scan disable**

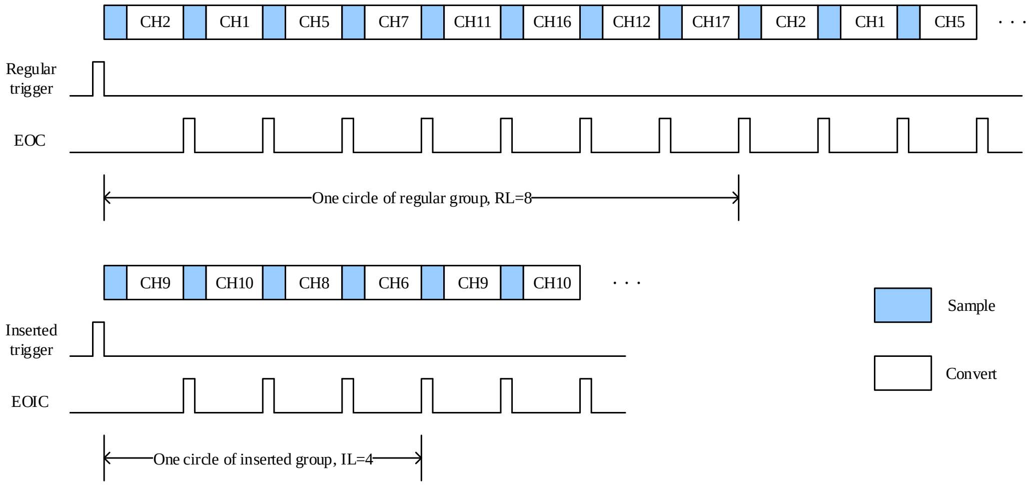
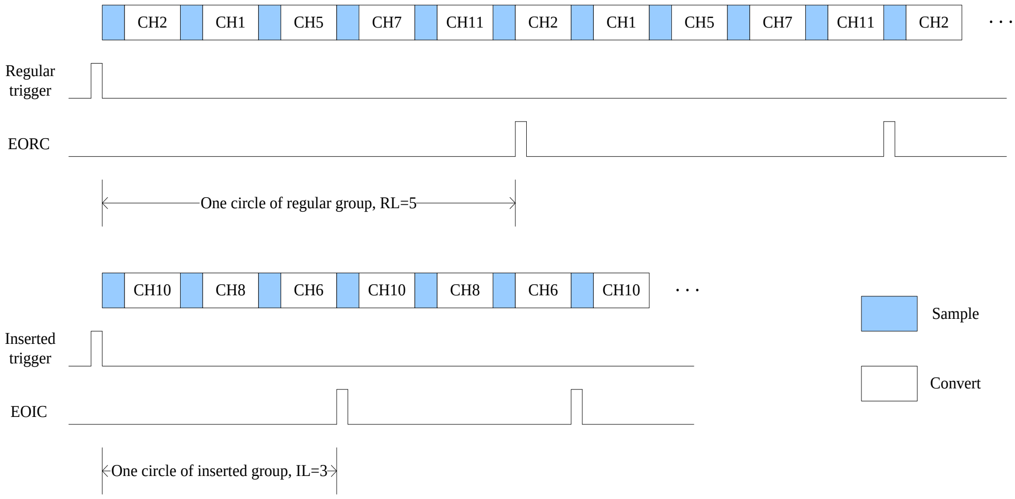


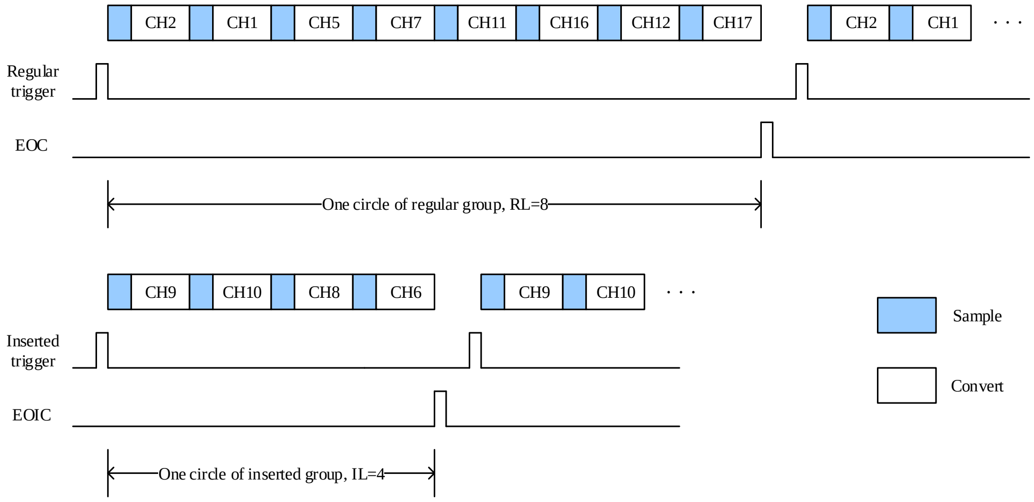
**Figure 12-5. Continuous conversion mode, scan enable**



## Scan conversion mode

The scan conversion mode will be enabled when SM bit in the ADC_CTL0 register is set. In this mode, the ADC performs conversion on the channels with a specific sequence specified in the ADC_RSQ0~ADC_RSQ2 registers or ADC_ISQ register. Once the ADCON is set high, the ADC samples and converts specified channels one by one in the regular or inserted group till the end of the regular or inserted group, when the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA or ADC_IDATAx register. After conversion of the regular or inserted channel group, the EOC or EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set. The DMA bit in ADC_CTL1 register must be set in scan mode.

**Figure 12-6. Scan conversion mode**



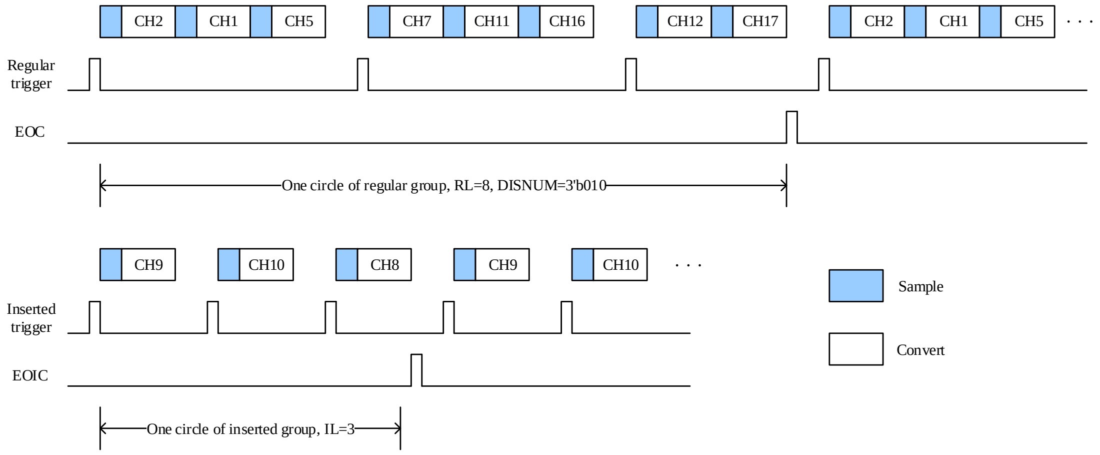
## Discontinuous mode

For regular channel group, the discontinuous conversion mode will be enabled when DISRC bit in the ADC_CTL0 register is set. In this mode, the ADC performs a short sequence of n conversions (n<=8) which is a part of the sequence of conversions selected in the ADC_RSQ0~ADC_RSQ2 registers. The value of n is defined 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 coverts the next n channels selected in the ADC_RSQ0~ADC_RSQ2 registers until all the channels in the regular sequence are done. The EOC will be set after every circle of the regular channel group. An interrupt will be generated if the EOCIE bit is set.

For inserted channel group, the discontinuous conversion mode will be enabled when DISIC bit in the ADC_CTL0 register is set. In this mode, the ADC performs one conversion which is a part of the sequence of conversions selected in the ADC_ISQ register. When the corresponding software trigger or external trigger is active, the ADC samples and coverts the next channel selected in the ADC_ISQ register until all the channels in the inserted sequence are done. The EOIC will be set after every circle of the inserted channel group. An interrupt will be generated if the EOICIE bit is set.

The regular and inserted groups cannot both work in discontinuous conversion mode. Only for one group conversion can be set in discontinuous conversion mode.

**Figure 12-7. Discontinuous conversion mode**



## 12.3.5. Analog watchdog

The analog watchdog is enabled when the RWDEN and IWDEN bits in the ADC_CTL0 register are set for regular and inserted channel groups respectively. When the analog voltage converted by the ADC is below a low threshold or above a high threshold, 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_WLT registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold value is independent of the alignment, which is specified by the DAL bit in the ADC_CTL1 register. One or more channels to be monitored by the analog watchdog are selected by the RWDEN, IWDEN, WDSC and WDCHSEL[4:0] bits in ADC_CTL0 register.

## 12.3.6. Inserted channel management

### Auto-insertion

The inserted group channels are automatically converted after the regular group channels when the ICA bit in ADC_CTL0 register is set. In this mode, external trigger on inserted channels cannot be enabled. A sequence of up to 20 conversions programmed in the ADC_RSQ0~ADC_RSQ2 and ADC_ISQ registers can be used to convert in this mode. In addition to the ICA bit, if the CTN bit is also set, regular channels followed by inserted channels are continuously converted.

The auto insertion mode cannot be enabled when the discontinuous conversion mode is set.

### Triggered insertion

If the ICA bit is cleared and SM bit is set, the triggered insertion occurs if a software or external trigger occurs during the regular group channel conversion. In this situation, the ADC abort from the current conversion and start the conversion of inserted channel sequence in scan mode. After the inserted channel group is done, the regular group channel conversion is resumed from the last aborted conversion.

### 12.3.7. Data alignment

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

After decreased by the user-defined offset written in the ADC_IOFFx registers, the inserted group data value may be a negative value. The sign value is extended.

**Figure 12-8. Data alignment of 12-bit resolution**

Regular group data

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

Inserted group data

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sign | Sign | Sign | Sign | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

DAL=0

Regular group data

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

Inserted group data

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

DAL=1

### 12.3.8. Programmable sample time

The number of ADC_CLK 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 sample time can be specified for each channel. Take the 12-bit resolution as an example, its total conversion time is “sampling time + 12.5” ADC_CLK cycles.

### 12.3.9. External trigger

When the ETERC or ETEIC bit in ADC_CTL1 register is set, conversion of regular or inserted group can be triggered by a rising edge of external trigger input. The ETSRC[2:0] and ETSIC[2:0] control bits are used to specify which out of 8 possible events can trigger conversion for the regular and inserted groups.

**Table 12-4. External trigger for regular channels of ADC**

| ETSRC[2:0] | Trigger Source | Trigger Type |
|---|---|---|
| 000 | TIMER0_CH0 | Internal on-chip signal |
| 001 | TIMER0_CH1 | |
| 010 | TIMER0_CH2 | |
| 011 | TIMER1_CH1 | |
| 100 | TIMER2_TRGO | |

| ETSRC[2:0] | Trigger Source | Trigger Type |
|---|---|---|
| 101 | TIMER14_CH0 | |
| 110 | EXTI_11 | External signal |
| 111 | SWRCST | Software trigger |

**Table 12-5. External trigger for inserted channels of ADC**

| ETSIC[2:0] | Trigger Source | Trigger Type |
|---|---|---|
| 000 | TIMER0_TRGO | Internal on-chip signal |
| 001 | TIMER0_CH3 | |
| 010 | TIMER1_TRGO | |
| 011 | TIMER1_CH0 | |
| 100 | TIMER2_CH3 | |
| 101 | TIMER14_TRGO | |
| 110 | EXTI_15 | External signal |
| 111 | SWICST | Software trigger |

### 12.3.10. DMA request

The DMA request is used to transfer data of regular group for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a regular channel. When this request is received, the DMA will transfer the converted data from the ADC_RDATA register to the destination location which is specified by the user.

### 12.3.11. Temperature sensor and internal reference voltage $V_{REF}$

When the TSVREN bit in ADC_CTL1 register is set, the temperature sensor channel (ADC_IN16) and $V_{REF}$ channel (ADC_IN17) is 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 17.1µs. When this sensor is not in use, it can be put 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 process variation, on this line, the internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. When it is used to detect accurate temperature, an external temperature sensor part should be used.

The internal voltage reference ($V_{REF}$) provides a stable (bandgap) voltage output for the ADC and Comparators. $V_{REF}$ 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 external trigger).

4. Read the resulting temperature data ($V_{temperature}$) in the ADC data register, and get the temperature using the following formula:

$$\text{Temperature (°C)} = \{(V30 – V_{temperature}\,(\text{digit})) / \text{Avg\_Slope}\} + 30.$$

$V_{30}$ : $V_{temperature}$ value at 30°C,the typical value is 1.43 V.

Avg_Slope : Average Slope for curve between Temperature vs. $V_{temperature}$, the typical value is 4.3 mV/°C.

### 12.3.12. Battery voltage monitoring

The $V_{BAT}$ channel can be used to measure the backup battery voltage on the $V_{BAT}$ pin. When the VBATEN bit in ADC_CTL1 register is set, $V_{BAT}$ channel (ADC_IN18) is enabled and a bridge divider by 2 integrated on the $V_{BAT}$ pin is also enabled automatically with it. As $V_{BAT}$ may be higher than $V_{DDA}$, this bridge is used to ensure the ADC correct operation. And it connect $V_{BAT}$/2 to the ADC_IN18 input channel. So, the converted digital value is $V_{BAT}$/2. In order to prevent unnecessary battery energy consumption, it is recommended that the bridge will be enabled only when it is required.

### 12.3.13. ADC interrupts

An interrupt can be produced at the end of conversion for regular and inserted groups and when the analog watchdog status bit is set. The independent interrupt enable bits is used to set the ADC interrupt flexibly.

### 12.3.14. Programmable resolution (RES) - fast conversion mode for GD32F170xx and GD32F190xx devices

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

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the DRES[1:0] bits in the ADC_CTL0 register. Lower resolution allows faster conversion time for applications where high data precision is not required.

The DRES[1:0] bits must only be changed when the ADCON bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeroes.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in Table 12-6.

**Table 12-6. $t_{SAR}$ timings depending on resolution**

| DRES[1:0] bits | $t_{SAR}$ (ADC clock cycles) | $t_{SAR}$ (ns) at $f_{ADC}$=28MHz | $t_{SMPL}$ (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 |

## 12.3.15. Oversampling for GD32F170xx and GD32F190xx devices

The oversampling unit, which is enabled by OVSEN bit in the ADC_OVSAMPCTL register, provides higher data resolution at the cost of lower output data rate.

The oversampling unit 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.

It provides a result with the following form, where N and M can be adjusted:

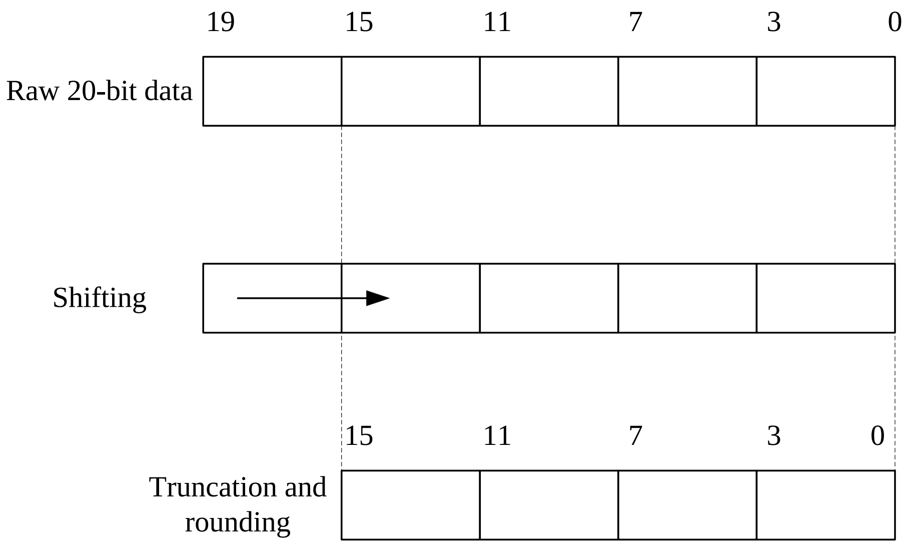
$$\mathrm{Result} = \frac{1}{\mathrm{M}} * \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{N}-1} \mathrm{Conversion}(\mathrm{t_n})$$

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVSR[2:0] bits in the ADC_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_OVSAMPCTL register.

The summation unit can yield a result 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.

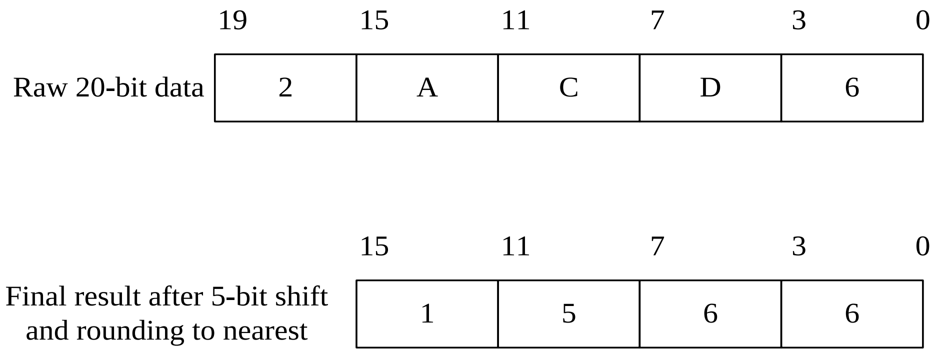
**Figure 12-9. 20-bit to 16-bit result truncation**



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

The Figure 12-10 shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 12-10. Numerical example with 5-bits shift and rounding**



The Table 12-7 below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

**Table 12-7. Maximum output results vs N and M. 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 | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 | 0x0040 | 0x0020 |
| 4x | 0x3FFC | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 | 0x0040 |
| 8x | 0x7FF8 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 | 0x0080 |
| 16x | 0xFFF0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 | 0x0100 |
| 32x | 0x1FFE0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 | 0x0200 |
| 64x | 0x3FFC0 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 | 0x0400 |
| 128x | 0x7FF80 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x0800 |
| 256x | 0xFFF00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF |

The conversion time in oversampling mode do not change compared to standard conversion mode: the sampling time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to N x $t_{ADC}$ = N x ($t_{SMPL}$ + $t_{SAR}$).

### Oversampling work with ADC modes

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

- Regular or inserted channels
- ADC started by softeware or external triggers
- Single or scan, continuous or discontinuous conversion modes
- Programmable sample time
- Analog watchdog

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

## 12.4. ADC registers

### 12.4.1. 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 | STIC | EOIC | EOC | WDE |
| | | | | | | | | | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value |
| 4 | STRC | Start flag of regular channel group<br>0: No regular channel group started<br>1: Regular channel group started<br>Set by hardware when regular channel conversion starts.<br>Cleared by software writing 0 to it. |
| 3 | STIC | Start flag of inserted channel group<br>0: No inserted channel group started<br>1: Inserted channel group started<br>Set by hardware when inserted channel group conversion starts.<br>Cleared by software writing 0 to it. |
| 2 | EOIC | End of inserted group conversion flag<br>0: No end of inserted group conversion<br>1: End of inserted group conversion<br>Set by hardware at the end of all inserted group channel conversion.<br>Cleared by software writing 0 to it. |
| 1 | EOC | End of group conversion flag<br>0: No end of group conversion<br>1: End of group conversion<br>Set by hardware at the end of a regular or inserted group channel 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 |

Set by hardware when the converted voltage crosses the values programmed in the ADC_WLT and ADC_WDHT registers.

Cleared by software writing 0 to it.

## 12.4.2. Control register 0 (ADC_CTL0)

### For GD32F130xx and GD32F150xx devices

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 | | | | | | | | RWDEN | IWDEN | Reserved | | | | | |
| | | | | | | | | rw | rw | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23 | RWDEN | Regular channel analog watchdog enable<br>0: Analog watchdog regular channel disable<br>1: Analog watchdog regular channel enable |
| 22 | IWDEN | Inserted channel analog watchdog enable<br>0: Analog watchdog inserted channel disable<br>1: Analog watchdog inserted channel enable |
| 21:16 | 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 | DISIC | Discontinuous mode on inserted channels<br>0: Discontinuous mode on inserted channels disable<br>1: Discontinuous mode on inserted channels enable |
| 11 | DISRC | Discontinuous mode on regular channels<br>0: Discontinuous mode on regular channels disable<br>1: Discontinuous mode on regular channels enable |
| 10 | ICA | Inserted channel group convert automatically<br>0: Inserted channel group convert automatically disable<br>1: Inserted channel group convert automatically enable |

| | | |
|---|---|---|
| 9 | WDSC | When in scan mode, analog watchdog is effective on a single channel<br>0: Analog watchdog is effective on all channels<br>1: Analog watchdog is effective on a single channel |
| 8 | SM | Scan mode<br>0: scan mode enable<br>1: scan mode disable |
| 7 | EOICIE | Interrupt enable for EOIC<br>0: EOIC interrupt disable<br>1: EOIC interrupt enable |
| 6 | WDEIE | Interrupt enable for WDE<br>0: WDE interrupt disable<br>1: WDE interrupt enable |
| 5 | EOCIE | Interrupt enable for EOC<br>0: EOC interrupt disable<br>1: EOC interrupt enable |
| 4:0 | WDCHSEL[4:0] | Analog watchdog channel select<br>00000: ADC channel0<br>00001: ADC channel1<br>00010: ADC channel2<br>……<br>01111: ADC channel15<br>1xx00: ADC channel16<br>1xx01: ADC channel17<br>1xx1x: ADC channel18<br>“x” means 0 or 1 |

### For GD32F170xx and GD32F190xx devices

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 | IWDEN | Reserved | | | | | |
| | | | | | | rw | | rw | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISNUM[2:0] | | | DISIC | DISRC | ICA | WDSC | SM | EOICIE | WDEIE | EOCIE | WDCHSEL[4:0] | | | | |
| rw | | | rw | rw | 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 | Regular channel analog watchdog enable<br>0: Analog watchdog regular channel disable<br>1: Analog watchdog regular channel enable |
| 22 | IWDEN | Inserted channel analog watchdog enable<br>0: Analog watchdog inserted channel disable<br>1: Analog watchdog inserted channel enable |
| 21:16 | 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 | DISIC | Discontinuous mode on inserted channels<br>0: Discontinuous mode on inserted channels disable<br>1: Discontinuous mode on inserted channels enable |
| 11 | DISRC | Discontinuous mode on regular channels<br>0: Discontinuous mode on regular channels disable<br>1: Discontinuous mode on regular channels enable |
| 10 | ICA | Inserted channel group convert automatically<br>0: Inserted channel group convert automatically disable<br>1: Inserted channel group convert automatically enable |
| 9 | WDSC | When in scan mode, analog watchdog is effective on a single channel<br>0: Analog watchdog is effective on all channels<br>1: Analog watchdog is effective on a single channel |
| 8 | SM | Scan mode<br>0: Scan mode enable<br>1: Scan mode disable |
| 7 | EOICIE | Interrupt enable for EOIC<br>0: EOIC interrupt disable<br>1: EOIC interrupt enable |
| 6 | WDEIE | Interrupt enable for WDE<br>0: WDE interrupt disable<br>1: WDE interrupt enable |
| 5 | EOCIE | Interrupt enable for EOC |

| | | |
|---|---|---|
| | | 0: EOC interrupt disable<br>1: EOC interrupt enable |
| 4:0 | WDCHSEL[4:0] | Analog watchdog channel select<br>00000: ADC channel0<br>00001: ADC channel1<br>00010: ADC channel2<br>……<br>01111: ADC channel15<br>1xx00: ADC channel16<br>1xx01: ADC channel17<br>1xx1x: ADC channel18<br>“x” means 0 or 1 |

## 12.4.3. 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 | | | | | | | VBA<br>TEN | TSV<br>REN | SWR<br>CST | SWIC<br>ST | ETE<br>RC | ETSRC[2:0] | | | Reserved |
| | | | | | | | rw | rw | rw | rw | rw | rw | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value |
| 24 | VBATEN | This bit is set and cleared by software to enable/disable the $V_{BAT}$ channel.<br>0: $V_{BAT}$ channel disabled<br>1: $V_{BAT}$ channel enabled |
| 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 | Start on regular channel.<br>Set 1 on this bit starts a conversion of a group of regular channels if ETSRC is 111. It is set by software and cleared by software or by hardware after the conversion starts. |
| 21 | SWICST | Start on inserted channel.<br>Set 1 on this bit starts a conversion of a group of inserted channels if ETSIC is 111. |

| | | |
|---|---|---|
| | | It is set by software and cleared by software or by hardware after the conversion starts. |
| 20 | ETERC | External trigger enable for regular channel<br>0: External trigger for regular channel disable<br>1: External trigger for regular channel enable |
| 19:17 | ETSRC[2:0] | External trigger select for regular channel<br>000: TIMER0 CH0<br>001: TIMER0 CH1<br>010: TIMER0 CH2<br>011: TIMER1 CH1<br>100: TIMER2 TRGO<br>101: TIMER14 CH1<br>110: EXTI line 11<br>111: SWRCST |
| 16 | Reserved | Must be kept at reset value |
| 15 | ETEIC | External trigger enable for inserted channel<br>0: External trigger for inserted channel disable<br>1: External trigger for inserted channel enable |
| 14:12 | ETSIC[2:0] | External trigger select for inserted channel<br>000: TIMER0 TRGO<br>001: TIMER0 CH3<br>010: TIMER1 TRGO<br>011: TIMER1 CH0<br>100: TIMER2 CH3<br>101: TIMER14 TRGO<br>110: EXTI line15<br>111: SWICST |
| 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 initialize done. |

| | | |
|---|---|---|
| | | 1: Initialize calibration register start |
| 2 | CLB | ADC calibration<br>0: Calibration done<br>1: Calibration start |
| 1 | CTN | Continuous mode<br>0: Continuous mode disable<br>1: Continuous mode enable |
| 0 | ADCON | ADC ON. The ADC will be wake up when this bit is changed from low to high. 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 |

## 12.4.4. Sampling time register 0 (ADC_SAMPT0)

### For GD32F130xx and GD32F150xx devices

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] | | SPT15[2:1] | |
| | | | | | | | | | rw | | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPT15[0] | SPT14[2:0] | | | SPT13[2:0] | | | SPT12[2:0] | | | SPT11[2:0] | | | SPT10[2:0] | | |
| rw | rw | | | rw | | | rw | | | rw | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:0 | SPTn[2:0] | Channel n sampling time<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles<br>**Note**: For GD32F130xx and GD32F150xx devices, the channel 17 and channel 18 sampling time are specified by the SPT17[2:0] bits in the ADC_SAMPT0 |

**For GD32F170xx and GD32F190xx devices**

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 | | | | | SPT18[2:0] | | | SPT17[2:0] | | | SPT16[2:0] | | | SPT15[2:1] | |
| | | | | | rw | | | rw | | | rw | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPT15[0] | SPT14[2:0] | | | SPT13[2:0] | | | SPT12[2:0] | | | SPT11[2:0] | | | SPT10[2:0] | | |
| rw | rw | | | rw | | | rw | | | rw | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value |
| 26:0 | SPTn[2:0] | Channel n sampling time<br>000: 1.5 cycles<br>001: 7.5 cycles<br>010: 13.5 cycles<br>011: 28.5 cycles<br>100: 41.5 cycles<br>101: 55.5 cycles<br>110: 71.5 cycles<br>111: 239.5 cycles |

### 12.4.5. Sampling 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 |
| 29:0 | SPTn[2:0] | Channel sampling time<br>000: 1.5 cycles |

001: 7.5 cycles

010: 13.5 cycles

011: 28.5 cycles

100: 41.5 cycles

101: 55.5 cycles

110: 71.5 cycles

111: 239.5 cycles

### 12.4.6. Inserted channel data offset registers (ADC_IOFFx) (x=0..3)

Address offset: 0x14-0x20

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 | | | | IOFF[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11:0 | IOFF[11:0] | Data offset for inserted channel x<br>These bits will be subtracted from the raw converted data when converting inserted channels. The conversion result can be read from the ADC_IDATAx registers. |

### 12.4.7. Watchdog high threshold register (ADC_WDHT)

Address offset: 0x24

Reset value: 0x0000 0FFF

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 | | | | WDHT[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

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

| | | |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11:0 | WDHT[11:0] | Analog watchdog high threshold<br>These bits define the high threshold for the analog watchdog. |

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

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | WDLT[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

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

### 12.4.9. Regular 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] | | | | RSQ16[4:1] | | | |
| | | | | | | | | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSQ16[0] | RSQ15[4:0] | | | | | RSQ14[4:0] | | | | | RSQ13[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:20 | RL[3:0] | Regular channel group length.<br>The total number of conversion in regular group equals to RL[3:0]+1. |
| 19:0 | RSQn[4:0] | The channel number (0..18) are written to these bits to select a channel at the nth |

conversion in the regular channel group.

### 12.4.10. Regular 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 | | RSQ12[4:0] | | | | | RSQ11[4:0] | | | | | RSQ10[4:1] | | | |
| | | rw | | | | | rw | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSQ10[0] | RSQ9[4:0] | | | | | RSQ8[4:0] | | | | | RSQ7[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:0 | RSQn[4:0] | The channel number (0..18) are written to these bits to select a channel at the nth conversion in the regular channel group. |

### 12.4.11. Regular sequence register 2 (ADC_RSQ2)

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 | | RSQ6[4:0] | | | | | RSQ5[4:0] | | | | | RSQ4[4:1] | | | |
| | | rw | | | | | rw | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSQ4[0] | RSQ3[4:0] | | | | | RSQ2[4:0] | | | | | RSQ1[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:0 | RSQn[4:0] | The channel number (0..18) are written to these bits to select a channel at the nth conversion in the regular channel group. |

### 12.4.12. Inserted sequence register (ADC_ISQ)

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 | | | | | | | | | | IL[1:0] | | ISQ4[4:1] | | | |
| | | | | | | | | | | rw | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISQ4[0] | ISQ3[4:0] | | | | | ISQ2[4:0] | | | | | ISQ1[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value |
| 21:20 | IL[1:0] | Inserted channel group length.<br>The total number of conversion in regular group equals to IL[1:0]+1. |
| 19:0 | ISQn[4:0] | The channel number (0..18) are written to these bits to select a channel at the nth conversion in the inserted channel group.<br>Unlike the regular conversion sequence, the inserted channels are converted starting from (4-IL[1:0]), if IL[1:0] length is less than 4. |

| IL | Insert channel order |
|---|---|
| 11 | ISQ0 >> ISQ1 >> ISQ2 >> ISQ3 |
| 10 | ISQ1 >> ISQ2 >> ISQ3 |
| 01 | ISQ2 >> ISQ3 |
| 00 | ISQ3 |

### 12.4.13. Inserted data registers (ADC_IDATAx) (x= 0..3)

Address offset: 0x3C - 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDATAn[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | IDATAn[15:0] | Inserted number n conversion data<br>These bits contain the number n conversion result, which is read only. |

### 12.4.14. Regular data register (ADC_RDATA)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | RDATA[15:0] | Regular channel data<br>These bits contain the conversion result from regular channel, which is read only. |

### 12.4.15. Oversampling control register (ADC_OVSAMPCTL) of GD32F170xx and GD32F190xx devices

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 oversampled conversions for a channel are done consecutively after a trigger<br>1: Each oversampled conversion for a channel needs a trigger<br>**Note:** Software is allowed to write this bit only when ADCON=0 (which ensures that no conversion is ongoing). |

| | | |
|---|---|---|
| 8:5 | OVSS[3:0] | Oversampling shift<br>This bit is 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 codes reserved<br>**Note:** Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing). |
| 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<br>101: 64x<br>110: 128x<br>111: 256x<br>**Note:** Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing). |
| 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:** Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing). |

# 13. Digital-to-analog converter (DAC)

## 13.1. DAC Introduction

The 12-bit DAC module is a voltage output digital-to-analog converter. The DAC can be configured in 8 or 12 bit mode and may be used in conjunction with the DMA controller. The input data of the DAC could be left-aligned or right-aligned. The output voltage can optionally be buffered for higher drive capability.

For GD32F150xx devices，the DAC module only has one DAC.

For GD32F190xx devices, the DAC module has two DACs, each with its own converter. In DAC concurrent mode, conversions could be done independently or concurrently when both DACs are grouped together for synchronous update operation.

## 13.2. DAC Main features

DAC main features are the following:

- 12-bit resolution. Left or right data alignment
- DMA capability with underrun error detection
- Conversion update synchronously
- Conversion trigged by external triggers
- Configurable internal buffer
- Input voltage reference, $V_{DDA}$

**For GD32F190xx devices, additional features are shown below.**

- Two DAC converters: one output channel each
- DAC independently or concurrently conversions

Figure 13-1 shows the block diagram of DAC and Table 13-1 gives the pin description.

**Figure 13-1. DAC block diagram**

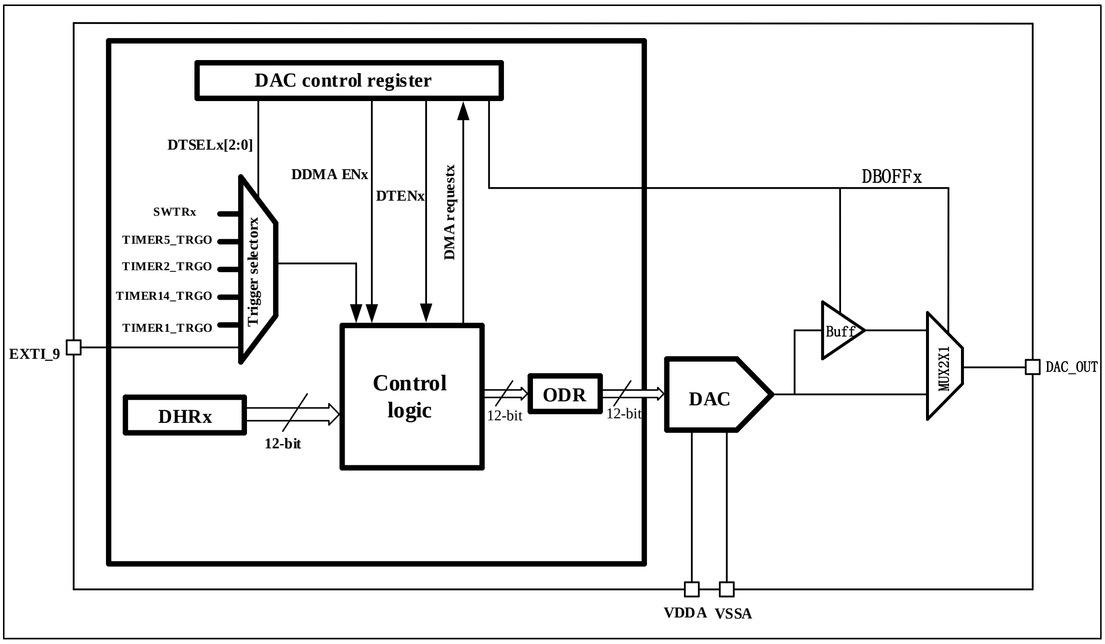


**Table 13-1. DAC pin**

| Name | Description | Signal type |
|---|---|---|
| $V_{DDA}$ | Analog power supply | Input, analog supply |
| $V_{SSA}$ | Ground for analog power supply | Input, analog supply ground |
| DAC_OUTx | DACx analog output | Analog output signal |

**Note:** The corresponding GPIO pin is automatically connected to the analog converter output (DAC_OUTx), Once the DAC is enabled. The GPIO pin should first be configured to analog (AIN to avoid parasitic consumption).

For GD32F150xx devices, corresponding GPIO pin is PA4.

For GD32F190xx devices, corresponding GPIO pin is PA4 or PA5.

# 13.3. DAC Function description

## 13.3.1. DAC enable

The DAC can be powered on by setting the DENx bit in the DAC_CTL register. The DAC is then enabled after a startup time $t_{WAKEUP}$.

## 13.3.2. DAC output buffer enable

The DAC integrates output buffer(For GD32F150xx devices, there is an output buffer; For

GD32F190xx devices, there are two output buffers ) which can be used to reduce the impedance of output, so DAC can be directly connected to external loads without operational amplifier. By the DBOFFx bit in the DAC_CTL register the DAC channel output buffer can be enabled and disabled. Users can enable the output buffer when the load of the DAC is heavy.

### 13.3.3. DAC data format

Depending on the selected configuration mode, the data has to be written in the specified register as described below:

- Single DAC, there are three possibilities

1. right alignment 8-bit: DHx[11:4] bits are stored into DACx_R8DH [7:0] bits
2. right alignment 12-bit: DHx[11:0] bits are stored into DACx_R12DH [11:0] bits
3. left alignment 12-bit: DHx[11:0] bits are stored into DACx_L12DH [15:4] bits

The data can be shifted and stored into the DHx(Data Holding Register x, that are internal non-memory-mapped registers) according to the value of the DACx_yyDH register. When a software trigger or an external event trigger occurs, the DH register will be loaded into DACx_DO register automatically.

- Two DACs, there are three possibilities ( only for GD32F190xx devices)

1. Right alignment 8-bit: data for DAC0, DH0 [11:4] bits are stored into DACC_R8DH [7:0] bits. Data for DAC1, DH1 [11:4] bits are stored into DACC_R8DH [15:8] bits.
2. Right alignment12-bit: data for DAC0 channel, DH0 [11:0] bits are stored into DACC_ R12DH [11:0] bits. Data for DAC1 channel, DH1 [11:0] bits are stored into DACC_ R12DH [27:16] bits.
3. Left alignment 12-bit: data for DAC0 channel, DH0 [11:0] bits are stored into DACC_ L12DH [15:4] bits. Data for DAC1 channel, DH1 [11:0] bits are stored into DACC_ L12DH [31:20] bits.

The data can be shifted and stored into the DAC0_DH and DAC1_DH (Data Holding Register x, that are internal non-memory-mapped registers) according to the value of the DACx_yyDH register. When a software trigger or an external event trigger occurs, the DH register will be loaded into DAC0_DO and DAC1_DO register automatically.

### 13.3.4. DAC conversion

Any data to be transferred by DAC should firstly be stored into the DH registers (DAC0_R8DH, DAC0_R12DH, DAC0_L12DH, DAC1_R8DH, DAC1_R12DH, DAC1_L12DH, DACC_R8DH, DACC_R12DH, DACC_L12DH).

If no hardware trigger is selected (DTENx bit in DAC_CTL register is reset), data stored in the DACx_yyDH register are automatically transferred to the DACx_DO register. However, when a hardware trigger is selected (DTENx bit in DAC_CTL register is set), the transfer is performed after the corresponding trigger occurs.

When the data from the DACx_yyDH register is loaded into the DACx_DO register, after the time $t_{SETTLING}$, the analog output is valid, and the value of $t_{SETTLING}$ is related to the power supply voltage and the analog output load.

### 13.3.5. DAC output voltage

The analog output voltages on the DAC pin are determined by the following equation:

$$\text{DAC output} = V_{DDA}*DO/4095$$

The digital input is linearly converted to an analog output voltage, its range is 0 to $V_{DDA}$.

### 13.3.6. DMA request

Each DAC has a DMA function. For GD32F190xx devices, two DMA channel can be respectively used for DAC DMA requests.

If the DDMAENx bit is set, a DAC DMA request will be generated when an external trigger (not a software trigger) occurs. Then the value in the DACx_yyDH register will be transferred to the DACx_DO register.

If a second external trigger arrives before the acknowledgement of the last request, the new request will not be serviced, because the DAC DMA request is not queued. Than the DDUDRx bit (DACx DMA underrun flag) in the DAC_STAT register is set, error status is reported. DMA data transfer function is disable and no further processing DMA requests. DACx continues to convert old data.

**GD32F190xx devices:**

In concurrent mode, if both DDMAENx bits are set, two DMA requests will be generated.

If you need one DMA request, only the corresponding DDMAENx bit should be set. In addition, the application can handle both DAC channel in concurrent mode using one DMA request and an independent DMA channel.

### 13.3.7. DAC trigger

Conversion can then be triggered by an external event (timer counter, external interrupt line), when the DTENx control bit is set,

The last data in the DACx_yyDH register is transferred into the DACx_DO register, when a DAC interface detects a selected timer TRGO output, or a rising edge on the selected external interrupt line 9.

When the software trigger is selected and the SWTRx bit is set, the conversion will start. SWTRx is reset by hardware.

**Table 13-2. External triggers of DAC**

| DTSELx[2:0] | Trigger Source | Trigger Type |
|---|---|---|
| 000 | TIMER5_TRGO | Internal on-chip signal |
| 001 | TIMER2_TRGO | |
| 010 | reserved | |
| 011 | TIMER14_TRGO | |
| 100 | TIMER1_TRGO | |
| 101 | reserved | |
| 110 | EXTI_9 | External signal |
| 111 | SWTRIG | Software trigger |

### 13.3.8. DAC concurrent conversion for GD32F190xx devices

When the two DACs work at the same time, for more effective utilization of bus bandwidth in applications, three concurrent registers can be used: DACC_R8DH, DACC_R12DH, DACC_L12DH. You just need to access a unique register to realize driving both DAC channels at the same time.

For the two DACs and these special registers, several possible conversion modes are possible. The conversion mode in the case of only use one DAC channel, still can operate through independent DHx registers.

All modes are described in the paragraphs below.

#### Independent trigger

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC trigger enable bits DTEN0 and DTEN1
- Set different values in the DTSEL0[2:0] and DTSEL1[2:0] bits to configure different trigger sources
- Load the DACs channel data into the required DACC_DH register (DACC_R8DH, DACC_R12DH, DACC_L12DH)

When the DAC0 channel is triggered, the value of the DH0 register is transferred to DAC0_DO (three APB1 clock cycles later).

When the DAC1 channel is triggered, the value of the DH1 register is transferred to DAC1_DO (three APB1 clock cycles later).

#### Concurrent software start

In this conversion mode, follow the steps below to set the DAC:

- Load the DACs channel data to the required DACC_DH register (DACC_R8DH, DACC_R12DH, DACC_L12DH)

In this mode, after one APB1 clock cycle, the value of the DH1 and DH2 registers is immediately transferred to DAC0_DO and DAC1_DO.

### Concurrent trigger

In this conversion mode, follow the steps below to set the DAC:

- Set the two DAC trigger enable bits DTEN0 and DTEN1
- Set the DTSEL0[2:0] and DTSEL1[2:0] bit to the same value, to configure the same trigger source for both DAC
- The data of two DACs conversion load to the required DACC_DH register (DACC_R8DH, DACC_R12DH, DACC_L12DH)

If detects a rising edge of a trigger, the value of the DH1 and DH2 registers is immediately transferred to DAC0_DO and DAC1_DO (three APB1 clock cycles later).

## 13.4. DAC registers

### 13.4.1. Control register (DAC_CTL)

#### For GD32F150xx devices

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 | | DDUDRIE0 | DDMAEN0 | Reserved | | | | | | DTSEL0[2:0] | | | DTEN0 | DBOFF0 | DEN0 |
| | | rw | rw | | | | | | | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13 | DDUDRIE0 | DAC0 DMA Underrun Interrupt enable<br>0: DAC0 DMA Underrun Interrupt disabled<br>1: DAC0 DMA Underrun Interrupt enabled |
| 12 | DDMAEN0 | DAC0 DMA enable<br>0: DAC0 DMA mode disabled |

1: DAC0 DMA mode enabled

11:6 Reserved Must be kept at reset value

5:3 DTSEL0[2:0] DAC0 trigger selection

These bits are only used if bit DTEN0 = 1 and select the external event used to trigger DAC0.

000: TIMER5 TRGO event

001: TIMER2 TRGO event

010: Reserved

011: TIMER14 TRGO event

100: TIMER1 TRGO event

101: Reserved

110: External line9

111: Software trigger

2 DTEN0 DAC0 trigger enable

This bit is set and cleared by software to enable/disable DAC0 trigger.

0: DAC0 trigger disabled

1: DAC0 trigger enabled

1 DBOFF0 DAC0 output buffer turn off

This bit is set and cleared by software to enable/disable DAC0 output buffer.

0: DAC0 output buffer enabled

1: DAC0 output buffer turn off

0 DEN0 DAC0 enable

This bit is set and cleared by software to enable/disable DAC0.

0: DAC0 disabled

1: DAC0 enabled

### For GD32F190xx devices

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 | | DDUDRIE1 | DDMAEN1 | Reserved | | | | | | DTSEL1[2:0] | | | DTEN1 | DBOFF1 | DEN1 |
| | | rw | rw | | | | | | | rw | | | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | DDUDRIE0 | DDMAEN0 | Reserved | | | | | | DTSEL0[2:0] | | | DTEN0 | DBOFF0 | DEN0 |
| | | rw | rw | | | | | | | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 29 | DDUDRIE1 | DAC1 DMA Underrun Interrupt enable<br>0: DAC1 DMA Underrun Interrupt disabled<br>1: DAC1 DMA Underrun Interrupt enabled |
| 28 | DDMAEN1 | DAC1 DMA enable<br>0: DAC1 DMA mode disabled<br>1: DAC1 DMA mode enabled |
| 27:22 | Reserved | Must be kept at reset value |
| 21:19 | DTSEL1[2:0] | DAC1 trigger selection<br>These bits are only used if bit DTEN1 = 1 and select the external event used to trigger DAC1.<br>000: TIMER5 TRGO event<br>001: TIMER2 TRGO event<br>010: Reserved<br>011: TIMER14 TRGO event<br>100: TIMER1 TRGO event<br>101: Reserved<br>110: External line9<br>111: Software trigger |
| 18 | DTEN1 | DAC1 trigger enable<br>This bit is set and cleared by software to enable/disable DAC1 trigger.<br>0: DAC1 trigger disabled<br>1: DAC1 trigger enabled |
| 17 | DBOFF1 | DAC1 output buffer turn off<br>This bit is set and cleared by software to enable/disable DAC1 output buffer.<br>0: DAC1 output buffer enabled<br>1: DAC1 output buffer turn offd |
| 16 | DEN1 | DAC1 enable<br>This bit is set and cleared by software to enable/disable DAC1.<br>0: DAC1 disabled<br>1: DAC1 enabled |
| 15:14 | Reserved | Must be kept at reset value |
| 13 | DDUDRIE0 | DAC0 DMA Underrun Interrupt enable<br>0: DAC0 DMA Underrun Interrupt disabled<br>1: DAC0 DMA Underrun Interrupt enabled |
| 12 | DDMAEN0 | DAC0 DMA enable<br>0: DAC0 DMA mode disabled<br>1: DAC0 DMA mode enabled |
| 11:6 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 5:3 | DTSEL0[2:0] | DAC0 trigger selection<br>These bits are only used if bit DTEN0 = 1 and select the external event used to trigger DAC0.<br>000: TIMER5 TRGO event<br>001: TIMER2 TRGO event<br>010: Reserved<br>011: TIMER14 TRGO event<br>100: TIMER1 TRGO event<br>101: Reserved<br>110: External line9<br>111: Software trigger |
| 2 | DTEN0 | DAC0 trigger enable<br>This bit is set and cleared by software to enable/disable DAC0 trigger.<br>0: DAC0 trigger disabled<br>1: DAC0 trigger enabled |
| 1 | DBOFF0 | DAC0 output buffer turn off<br>This bit is set and cleared by software to enable/disable DAC0 output buffer.<br>0: DAC0 output buffer enabled<br>1: DAC0 output buffer turn off |
| 0 | DEN0 | DAC0 enable<br>This bit is set and cleared by software to enable/disable DAC0.<br>0: DAC0 disabled<br>1: DAC0 enabled |

## 13.4.2. Software trigger register (DAC_SWT)

### For GD32F150xx devices

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 | | | | | | | | | | | | | | | SWTR0 |
| | | | | | | | | | | | | | | | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 0 | SWTR0 | DAC0 software trigger, cleared by hardware<br>0: Software trigger disabled<br>1: Software trigger enabled |

**For GD32F190xx devices**

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 | | | | | | | | | | | | | | SWTR1 | SWTR0 |
| | | | | | | | | | | | | | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | SWTR1 | DAC1 software trigger, cleared by hardware<br>0: Software trigger disabled<br>1: Software trigger enabled |
| 0 | SWTR0 | DAC0 software trigger, cleared by hardware<br>0: Software trigger disabled<br>1: Software trigger enabled |

### 13.4.3. DAC0 12-bit right-aligned data holding register (DAC0_R12DH)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DAC0_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11:0 | DAC0_DH[11:0] | DAC0 12-bit right-aligned data |

These bits are written by software which specifies 12-bit data for DAC0.

### 13.4.4. DAC0 12-bit left-aligned data holding register (DAC0_L12DH)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC0_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:4 | DAC0_DH[11:0] | DAC0 12-bit left-aligned data<br>These bits are written by software which specifies 12-bit data for DAC0. |
| 3:0 | Reserved | Must be kept at reset value |

### 13.4.5. DAC0 8-bit right-aligned data holding register (DAC0_R8DH)

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 | | | | | | | | DAC0_DH[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | DAC0_DH[7:0] | DAC0 8-bit right-aligned data<br>These bits are written by software which specifies 8-bit data for DAC0. |

### 13.4.6. DAC1 12-bit right-aligned data holding register (DAC1_R12DH) of GD32F190xx devices

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 | | | | DAC1_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11:0 | DAC1_DH[11:0] | DAC1 12-bit right-aligned data<br>These bits are written by software which specifies 12-bit data for DAC1. |

### 13.4.7. DAC1 12-bit left-aligned data holding register (DAC1_L12DH) of GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC1_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:4 | DAC1_DH[11:0] | DAC1 12-bit left-aligned data<br>These bits are written by software which specifies 12-bit data for DAC1. |
| 3:0 | Reserved | Must be kept at reset value |

### 13.4.8. DAC1 8-bit right-aligned data holding register (DAC1_R8DH) of GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | DAC1_DH[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | DAC1_DH[7:0] | DAC1 8-bit right-aligned data<br>These bits are written by software which specifies 8-bit data for DAC1. |

### 13.4.9. DAC concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) of GD32F190xx devices

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 | | | | DAC1_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DAC0_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27:16 | DAC1_DH[11:0] | DAC1 12-bit right-aligned data<br>These bits are written by software which specifies 12-bit data for DAC1. |
| 15:12 | Reserved | Must be kept at reset value |
| 11:0 | DAC0_DH[11:0] | DAC0 12-bit right-aligned data |

These bits are written by software which specifies 12-bit data for DAC0.

### 13.4.10. DAC concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) of GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC1_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC0_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | DAC1_DH[11:0] | DAC1 12-bit left-aligned data<br>These bits are written by software which specifies 12-bit data for DAC1. |
| 19:16 | Reserved | Must be kept at reset value |
| 15:4 | DAC0_DH[11:0] | DAC0 12-bit left-aligned data<br>These bits are written by software which specifies 12-bit data for DAC1. |
| 3:0 | Reserved | Must be kept at reset value |

### 13.4.11. DAC concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) of GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC1_DH[7:0] | | | | | | | | DAC0_DH[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

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

| | | |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | DAC1_DH[7:0] | DAC1 8-bit right-aligned data<br>These bits are written by software which specifies 8-bit data for DAC1. |
| 7:0 | DAC0_DH[7:0] | DAC0 8-bit right-aligned data<br>These bits are written by software which specifies 8-bit data for DAC0. |

## 13.4.12. DAC0 data output register (DAC0_DO)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DAC0_DO [11:0] | | | | | | | | | | | |
| | | | | r | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11:0 | DAC0_DO [11:0] | DAC0 output data<br>These bits are read-only, they contain data output for DAC0. |

## 13.4.13. DAC1 data output register (DAC1_DO) of GD32F190xx devices

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 | | | | DAC1_DO [11:0] | | | | | | | | | | | |
| | | | | r | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 11:0 | DAC1_DO [11:0] | DAC1 output data<br>These bits are read-only, they contain data output for DAC1. |

## 13.4.14. Status register (DAC_STAT)

### For GD32F150xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DDUDR0 | Reserved | | | | | | | | | | | | |
| | | rc_w1 | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13 | DDUDR0 | DAC0 DMA underrun flag<br>This bit is set by hardware and cleared by software (by writing it to 1).<br>0: No DMA underrun error condition occurred<br>1: DMA underrun error condition occurred (the frequency of the current selected trigger that is driving DAC conversion is higher than the DMA service capability rate) |
| 12:0 | Reserved | Must be kept at reset value |

### For GD32F190xx devices

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 | | DDUDR1 | Reserved | | | | | | | | | | | | |
| | | rc_w1 | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DDUDR0 | Reserved | | | | | | | | | | | | |
| | | rc_w1 | | | | | | | | | | | | | |

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

| | | |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29 | DDUDR1 | DAC1 DMA underrun flag<br>This bit is set by hardware and cleared by software (by writing it to 1).<br>0: No DMA underrun error condition occurred<br>1: DMA underrun error condition occurred (the frequency of the current selected trigger that is driving DAC conversion is higher than the DMA service capability rate) |
| 28:14 | Reserved | Must be kept at reset value |
| 13 | DDUDR0 | DAC0 DMA underrun flag<br>This bit is set by hardware and cleared by software (by writing it to 1).<br>0: No DMA underrun error condition occurred<br>1: DMA underrun error condition occurred (the frequency of the current selected trigger that drives DAC conversion is higher than the DMA service capability rate) |
| 12:0 | Reserved | Must be kept at reset value |

# 14. Inter-integrated circuit (I2C) interface

## 14.1. Introduction

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 at standard or fast speed as well as CRC calculation and checking, SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

## 14.2. Main features

- 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 Speed (up to 100 kHz) and Fast Speed (up to 400 kHz)
- 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

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Support SAM_V mode

## 14.3. Function description

Figure 14-1 below provides details on the internal configuration of the I2C interface.

**Figure 14-1. I2C module block diagram**

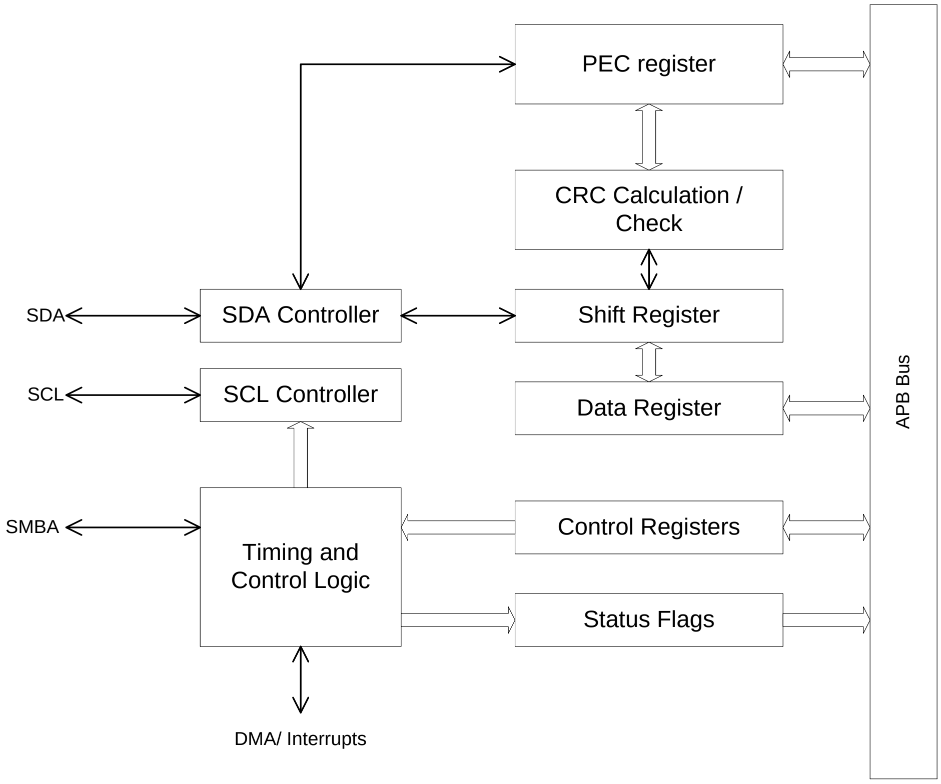


**Table 14-1. Definition of I2C-bus terminology**

| 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 simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted |

### 14.3.1. SDA and SCL lines

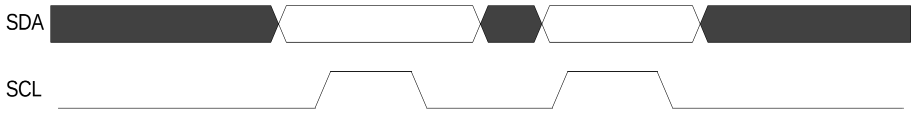
The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus.

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via ccurrent-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 or 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 and up to 400 kbit/s in the fast mode. Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the levels of the logical ‘0’ (LOW) and ‘1’ (HIGH) are not fixed and depend on the associated level of $V_{DD}$.

### 14.3.2. 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 data line can only change when the clock signal on the SCL line is LOW (see Figure 14-2). One clock pulse is generated for each data bit transferred.

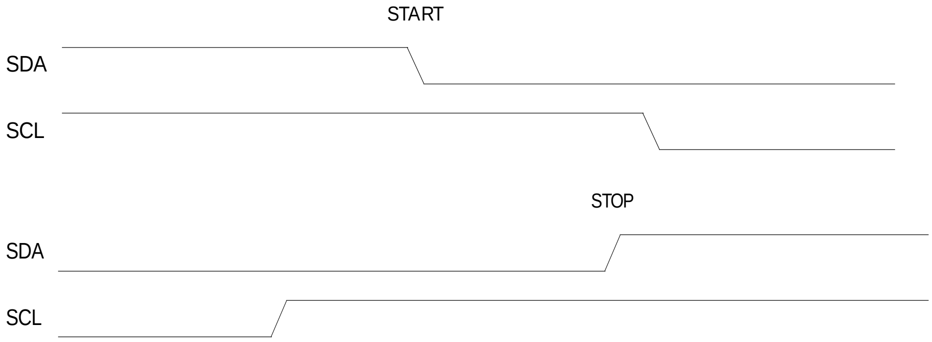
**Figure 14-2. Data validation**



### 14.3.3. START and STOP condition

All transactions begin with a START (S) and are terminated by a STOP (P) (see Figure 14-3). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

**Figure 14-3. START and STOP condition**

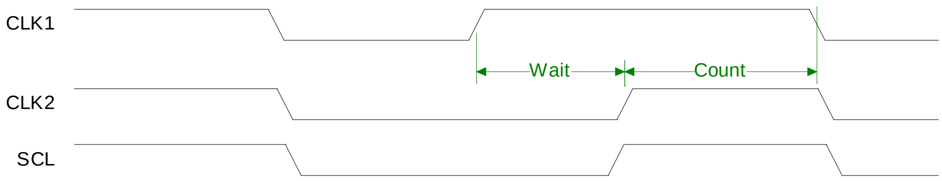


### 14.3.4. Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which takes control of the bus and complete its transmission. This is done by clock synchronization and arbitration. In single master systems, clock synchronization and arbitration are not needed.

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 off 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 14-4). 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 periods enter a HIGH wait-state during this time.

**Figure 14-4. Clock synchronization**



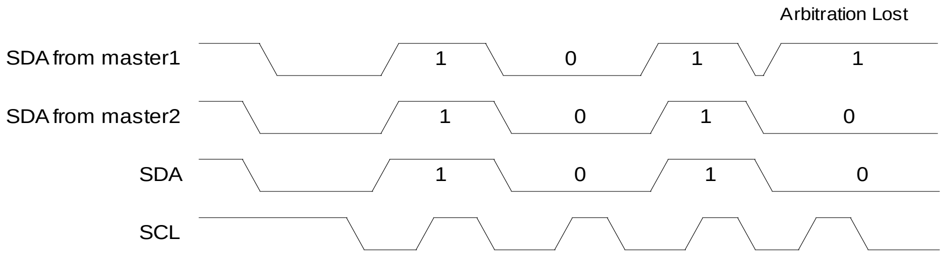
## 14.3.5. Arbitration

Arbitration, like synchronization, refers to a portion of the protocol required only if 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 condition within the minimum hold time of the START condition which results in a valid START condition 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 to see if the SDA level matches what it has sent. This process may take many bits. Two masters can actually complete an entire transaction 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, the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.

**Figure 14-5. SDA Line arbitration**



## 14.3.6. 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 operate as either a transmitter or receiver, depending on the function of the device.

An I2C slave will continue to detect addresses after a START condition on I2C bus and compare the detected address with its own address which is programmable by software. Once the two addresses match, the I2C slave will send an ACK to the I2C bus and responses to the following command on I2C bus: transmitting or receiving desired data. Additionally, if General Call is enabled by software, an I2C slave always responses to a General Call Address (0x00). The I2C block support both 7-bit and 10-bit addresses.

An I2C master always initiates or end a transfer using START or STOP condition and it's also responsible for SCL clock generation.

**Figure 14-6. I2C communication flow with 7-bit address.**

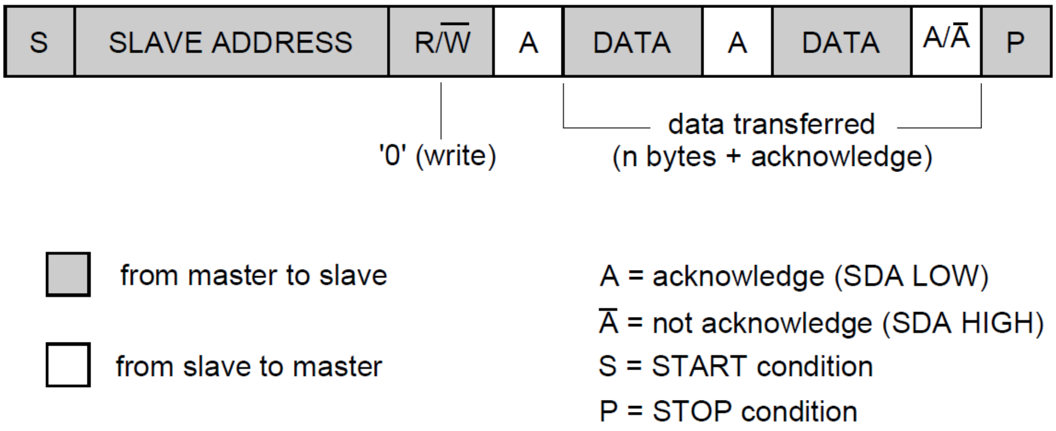
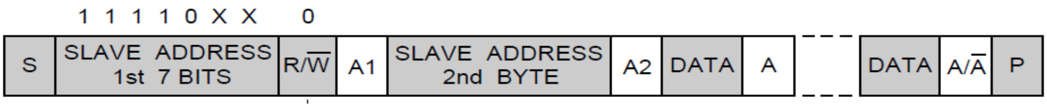


**Figure 14-7. I2C communication flow with 10-bit address.**



## 14.3.7. Programming model

An I2C device such as LCD driver may be only 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 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. If it's programmed by software and finished sending a START condition on I2C bus, it changes into master mode. The I2C changes back to slave mode after it's programmed by software and finished sending a STOP condition on I2C bus.

### Programming model in slave transmitting mode

As it shows in figure below, software should follow these steps to operate I2C block in slave mode for transmitting some data to the I2C bus:

1. First of all, software should 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 condition followed by address on I2C bus.

2. After receiving a START condition 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. Now software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. If the address is in 10-bit format, the I2C master should then generate a repeated START (Sr) condition and send a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START (Sr) condition and the following header. Software also clears the ADDSEND bit 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. Software now write the first byte data to I2C_DATA register, but the TBE is not cleared because the written byte in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as shift register is not empty.

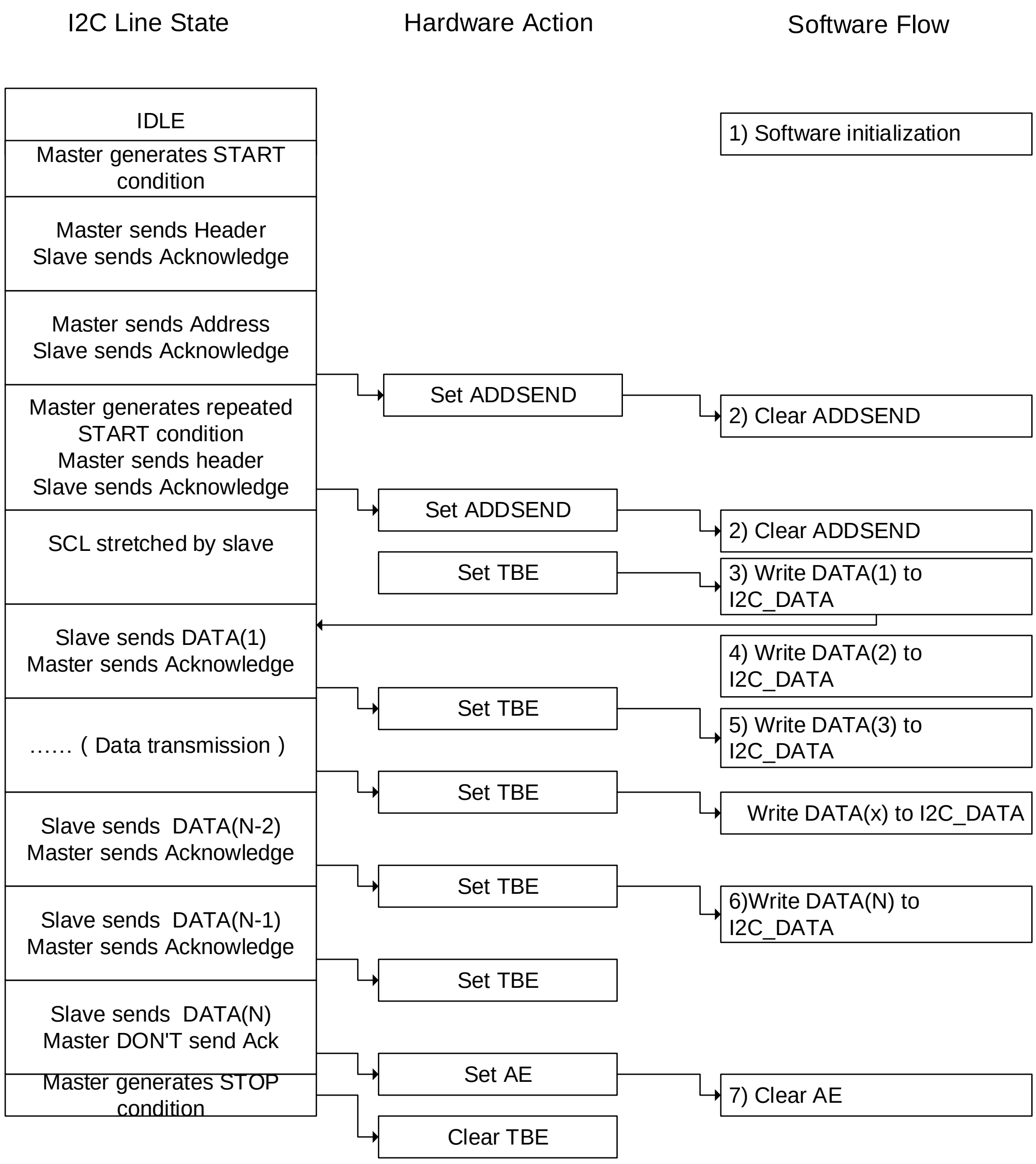
4. During the first byte's transmission, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

5. Any time TBE is set, software can write a byte to I2C_DATA as long as there are still data to be transmitted.

6. During the second last byte's transmission, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be seted after the byte's transmission and not cleared until a STOP condition.

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 condition on I2C bus and sets AERR (Acknowledge Error) bit to notify software that transmission completes. Software clears AERR bit by writing 0 to it.

**Figure 14-8. Programming model for slave transmitting**



## Programming model in slave receiving mode

As it shows in figure below, software should follow these steps to operate I2C block in slave mode for receiving some data from the I2C bus:

1. First of all, software should 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 condition followed by address on I2C bus.

2. After receiving a START condition followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register, which should be monitored by software either by polling or interrupt. Now software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. The I2C begins to receive data to I2C bus as soon as ADDSEND bit is cleared.

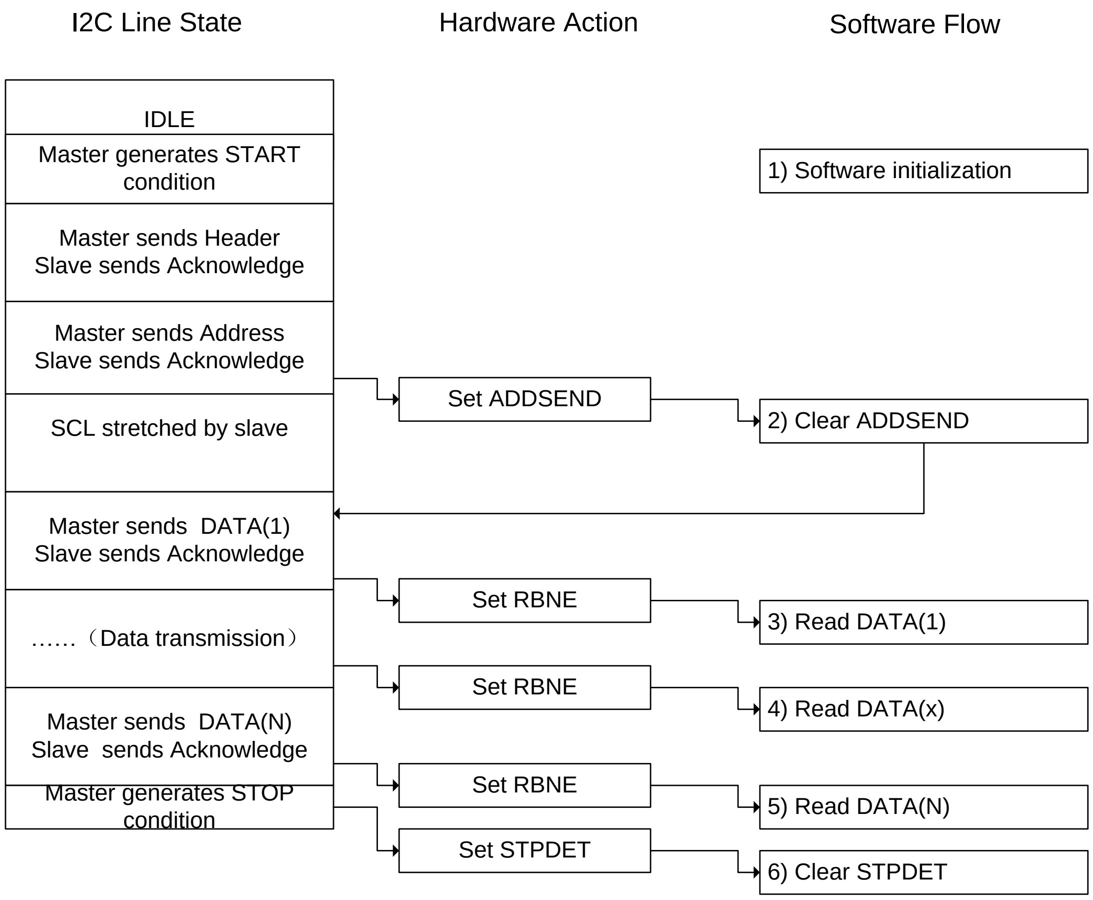
3. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.

4. Any time RBNE is set, software can read a byte from I2C_DATA.

5. After last byte is received, RBNE is set. Software reads the last byte.

6. STPDET bit is set when I2C detects a STOP condition on I2C bus and software reads I2C_STAT0 and then writes I2C_CTL0 to clear the STPDET bit.

**Figure 14-9. Programming model for slave receiving**



## Programming model in master transmitting mode

As it shows in figure below, software should follow these steps to operate I2C block in master mode for transmitting some data to the I2C bus:

1. First of all, software should 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 condition followed by address on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.

3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status 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 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 is 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 write the first byte data to I2C_DATA register, but the TBE is not cleared because the written byte in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as shift register is not empty.

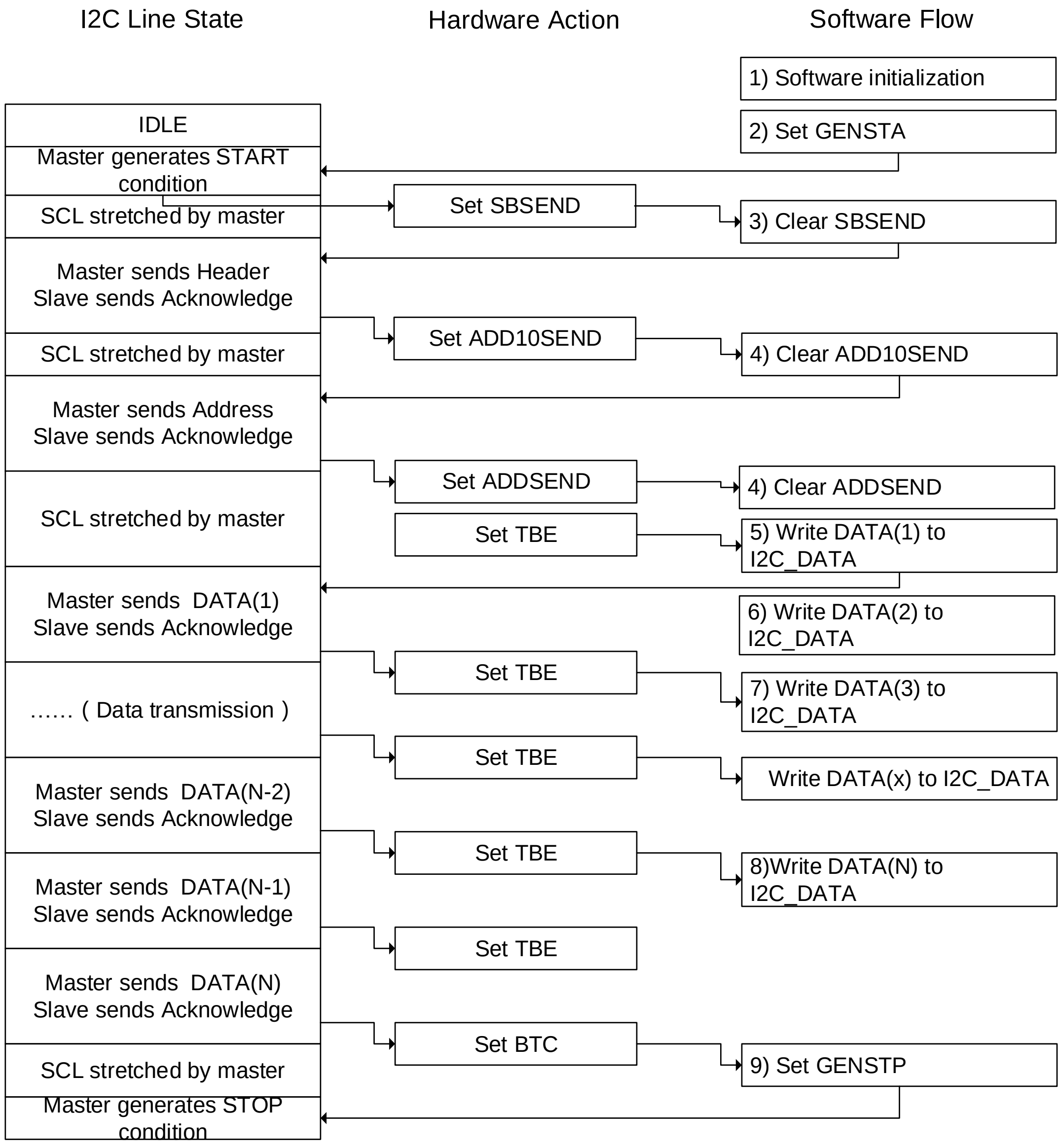
6. During the first byte’s transmission, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

7. Any time TBE is set, software can write a byte to I2C_DATA as long as there are still data to be transmitted.

8. During the second last byte’s transmission, 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 byte’s transmission and not cleared until a STOP condition.

9. After sending the last byte, I2C master sets BTC bit because both shift register and I2C_DATA are empty. Software should program a STOP request now, and the I2C clears both TBE and BTC flags after sending a STOP condition.

**Figure 14-10. Programming model for master transmitting**



### 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 STOP condition on I2C bus. So, special attention should be paid to ensure the correct ending of data receiving. Two solutions for master receiving is provided here for your application: Solution A and B. Solution A requires the software's quick response to I2C events, while Solution B doesn't.

#### Solution A

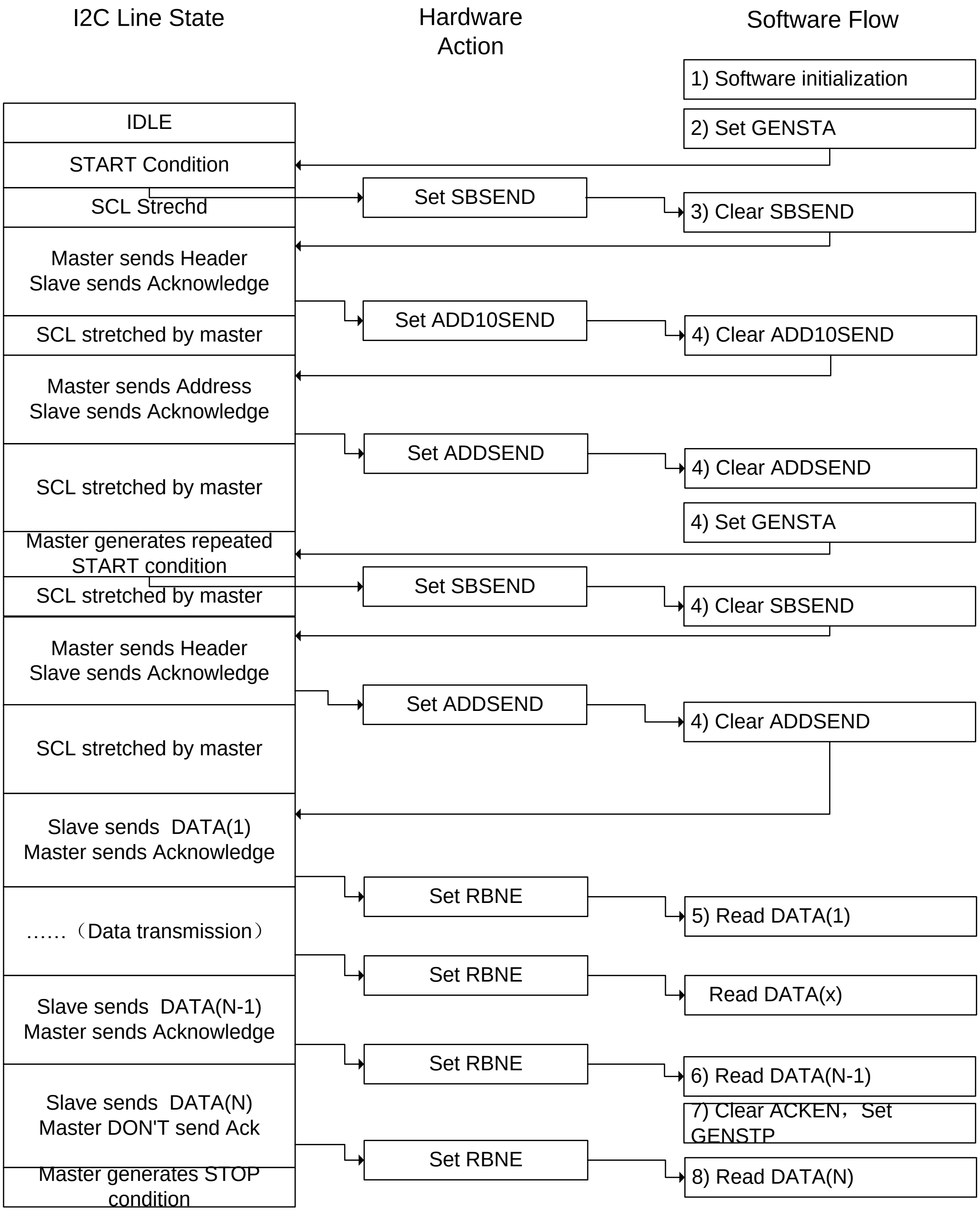
1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured,

I2C operates in its default slave state and waits for START condition followed by address on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.
3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status 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 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 is 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 condition 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 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 is sent for the last byte.
8. After last byte is received, RBNE is set. Software reads the last byte. I2C doesn't send ACK to the last byte and generate a STOP condition after the transmission of the last byte.

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.

**Figure 14-11. Programming model for master receiving using Solution A**



### Solution B

1. First of all, software should 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 condition followed by address on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.

3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status

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 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 is 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 condition 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.
7. As shown in Figure 14-12, 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.
8. 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.
9. After last byte is received, both BTC and RBNE is set again. Software sets STOP bit and master sends out a STOP condition on bus.
10. Software reads the N-1 byte, clearing BTC. After this the last byte is moved from shift register to I2C_DATA.
11. Software reads the last byte, clearing RBNE.

Above steps require that byte number N>2. N=1 or N=2 are similar:

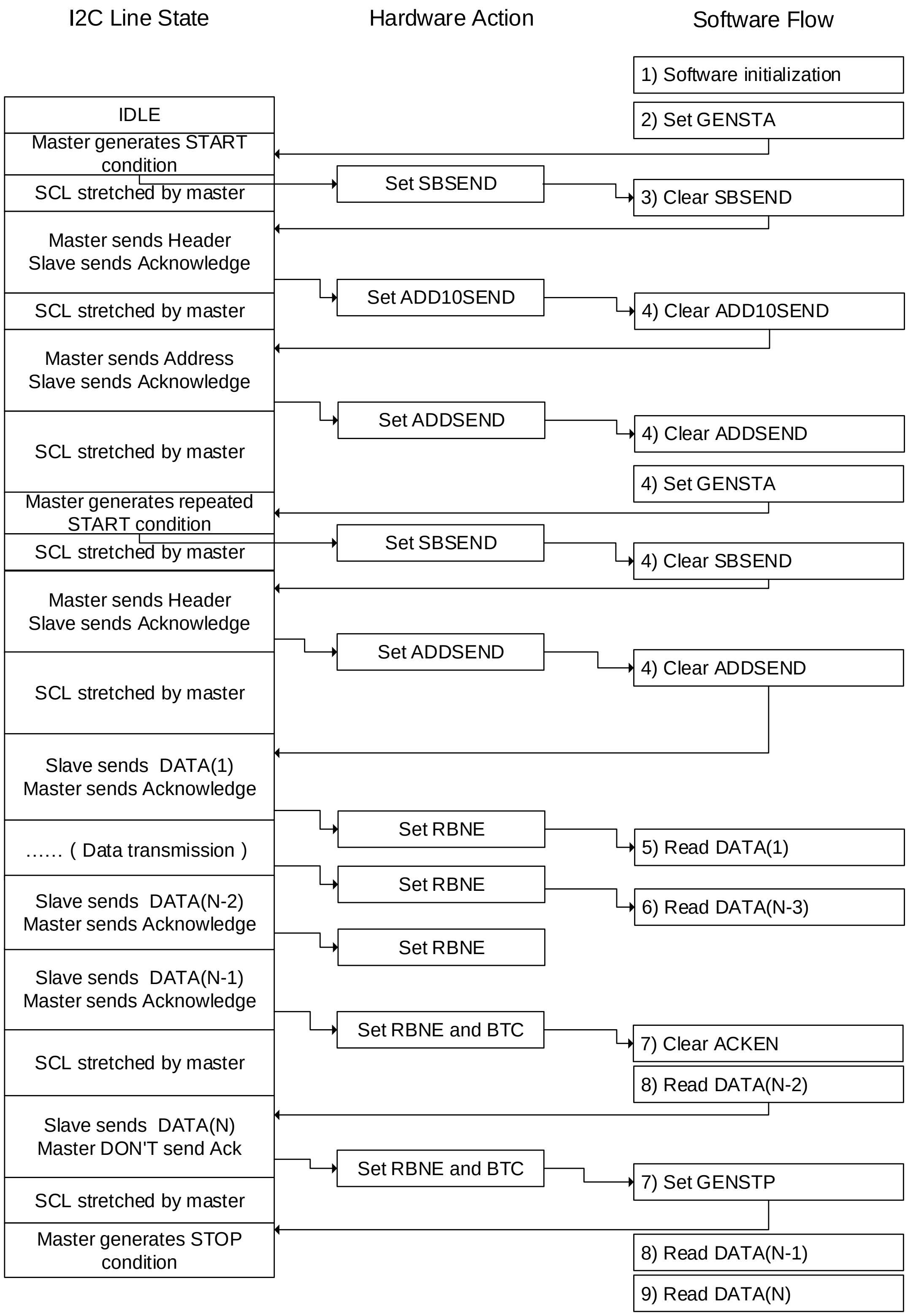
**N=1**

In Step4, software should reset ACK 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 set 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 reads I2C_DATA twice.

**Figure 14-12. Programming model for master receiving using Solution B**



### Programming model for DMA mode

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU’s high overload. DMA can be used to process TBE and RBNE flag: each time TBE or RBNE is asserted. DMA does a read or write operation automatically.

## 14.3.8. Packet error checking

There is a CRC-8 calculator in I2C block to perform 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 is set.

## 14.3.9. 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 the 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 transaction 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 uses I2C hardware and I2C hardware addressing, but adds 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 allow 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 both those protocols there is a very useful distinction made between a System Host and all the other devices in the system that can have the names and functions of masters or slaves.

### Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I2C can be a ‘DC’ bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This will notify to the master that the slave is busy but does not want to lose the communication. The slave device will allow continuation after its task is completed. There is no limit in the I2C bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 35ms. 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 clear this mode. Slave devices are not allowed to hold the clock low too long.

### Packet error checking

SMBus 2.0 and 1.1 allow enabling Packet Error Checking (PEC). In that mode, a PEC (packet error code) byte is appended at the end of each transaction. The byte is calculated as CRC-8 checksum, calculated over 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 but passing more data and building on the I2C multi-master mode.

### 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, response 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 value.
2. In order to support ARP protocol (ARPEN=1), the software should response to HSTSMB flag in SMBus Host Mode (SMBTYPE =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 response to SMBALTS flag and implement the related function.

## 14.3.10. Status, errors and interrupts

There are several status and error flags in I2C, and interrupt may be asserted from these flags by setting some register bits (refer to I2C register for detail).

**Table 14-2. Event status flags**

| Event Flag Name | Description |
|---|---|
| SBSEND | START condition sent (master) |
| ADDSEND | Address sent or received |
| ADD10SEND | Header of 10-bit address sent |
| STPDET | STOP condition detected |
| BTC | Byte transmission completed |
| TBE | I2C_DATA is empty when transmitting |
| RBNE | I2C_DATA is not empty when receiving |

**Table 14-3. I2C error flags**

| I2C Error Name | Description |
|---|---|
| BE | 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 |
| SMBALTS | SMBus Alert |

## 14.4. I2C registers

### 14.4.1. Control register 0 (I2C_CTL0)

Address offset: 0x00

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRESET | Reserved | SALT | PECTRANS | POAP | ACKEN | STOP | START | DISSTRC | GCEN | PECEN | ARPEN | SMBSEL | Reserved | SMBEN | I2CEN |
| rw |  | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |  | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | SRESET | Software reset 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 the reset value |
| 13 | SALT | Software can set and clear this bit and hardware can clear this bit.<br>0: Don’t issue alert through SMBA |

| | | |
|---|---|---|
| | | 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 condition detectedor I2CEN=0<br>0: Don’t transfer PEC value<br>1: Transfer PEC |
| 11 | POAP | Position of ACK/PEC’s meaning<br>This bit is set and cleared by software and cleared by hardware when I2CEN=0<br>0: ACKEN bit decides whether to send ACK or not for the current byte that is being received. PEC bit indicates that PECTRANS is in shift register<br>1: ACKEN bit decides whether to send ACK or not for the next byte, PECTRANS bit indicates that the next byte to be received is PEC |
| 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 condition 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 condition detected.<br>0: STOP will not be sent<br>1: STOP will be sent |
| 8 | START | Generate a START condition on I2C bus<br>This bit is set and cleared by software and and cleared by hardware when START condition detected or I2CEN=0<br>0: START will not be sent<br>1: START will be sent |
| 7 | DISSTRC | 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 response to a General Call<br>1: Slave will response to a General Call |
| 5 | PECEN | PEC Calculation Switch<br>0: PEC Calculation off<br>1: PEC Calculation on |
| 4 | ARPEN | ARP protocol in SMBus switch<br>0: ARP is disabled |

| | | |
|---|---|---|
| | | 1: ARP is enabled |
| 3 | SMBSEL | SMBusType Selection<br>0: Device<br>1: Host |
| 2 | Reserved | Must keep the 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 |

### 14.4.2. Control register 1 (I2C_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DMALST | DMAON | BUFIE | EVIE | ERRIE | Reserved | | I2CCLK[5:0] | | | | | |
| | | | rw | rw | rw | rw | rw | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:13 | Reserved | Must be kept the reset value |
| 12 | DMALST | Flag indicating DMA last transfer<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 disabled<br>1: DMA mode enabled |
| 10 | BUFIE | Buffer interrupt enable<br>0: No interrupt asserted when TBE = 1 or RBNE = 1<br>1: Interrupt asserted when TBE = 1 or RBNE = 1 if ITEVTEN=1 |
| 9 | EVIE | Event interrupt enable<br>0: Event interrupt disabled<br>1: Event interrupt enabled, 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<br>0: Error interrupt disabled<br>1: Error interrupt enabled, means that interrupt will be generated when BE, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALTS flag asserted. |
| 7:6 | Reserved | Must be kept the reset value |
| 5:0 | I2CCLK[5:0] | I2C Peripheral clock frequency<br>I2CCLK[5:0] should be the frequency of input APB clock in MHz which is at least 2.<br>0h - 1h: Not allowed<br>2h - 36h: 2 MHz~36MHz<br>37h - 63h: Not allowed due to the limitation of APB clock |

## 14.4.3. Slave address register 0 (I2C_SADDR0)

Address offset: 0x08

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDFORMAT | Reserved | | | | | ADDRESS[9:8] | | ADDRESS[7:1] | | | | | | | ADDRESS0 |
| rw | | | | | | rw | | rw | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | ADDFORMAT | Address mode for the I2C slave<br>0: 7-bit Address<br>1: 10-bit Address |
| 14:10 | Reserved | Must be kept the 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 |

## 14.4.4. Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | ADDRESS2[7:1] | | | | | | | DUADEN |
| | | | | | | | | rw | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept the reset value |
| 7:1 | ADDRESS2[7:1] | Second I2C address for the slave in Dual-Address mode |
| 0 | DUADEN | Dual-Address mode switch<br>0: Dual-Address mode disabled<br>1: Aual-Address mode enabled |

## 14.4.5. Transfer buffer register (I2C_DATA)

Address offset: 0x10

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept the reset value |
| 7:0 | TRB[7:0] | Transmission or reception data buffer register |

## 14.4.6. Transfer status register 0 (I2C_STAT0)

Address offset: 0x14

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMBALTS | SMBTO | Reserved | PECERR | OUERR | AERR | LOSTARB | BE | 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 |
|---|---|---|
| 15 | SMBALTS | 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 (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 |

| | | |
|---|---|---|
| | | 1: Timeout event occurs (SCL is low for 25 ms) |
| 13 | Reserved | Must keep the 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 and calculated PEC match<br>1: Received PEC and calculated PEC don't match, 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 | BE | A bus error occurs indication an unexpected START or STOP condition 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 | TRBR 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 it. If both BTC and RBNE are asserted, reading I2C_DATA won't clear RBNE because the shift register's byte is moved to I2C_DATA immediately.<br>0: TRBR is empty |

| | | |
|---|---|---|
| | | 1: TRBR is not empty, software can read |
| 5 | Reserved | Must be kept the reset value |
| 4 | STPDET | STOP condition 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 condition not detected in slave mode<br>1: STOP condition 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 sent in master mode<br>1: Header of 10-bit address is sent in master mode |
| 2 | BTC | Byte transmission 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 reading I2C_STAT0 and reading or writing I2C_DATA.<br>0: BTC not asserted<br>1: BTC asserted |
| 1 | ADDSEND | Address is sent in master mode or received and matches in slave mode.<br>This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1.<br>0: No address sent or received<br>1: Address sent out in master mode or a matched address is received in salve mode |
| 0 | SBSEND | START condition 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 condition sent<br>1: START condition sent |

### 14.4.7. Transfer status register 1 (I2C_STAT1)

Address offset: 0x18

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECV[7:0] | | | | | | | | DUMODF | HSTSMB | DEFSMB | RXGC | Reserved | TRS | I2CBSY | MASTER |

r r r r r r r r

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | ECV[7:0] | Packet Error Checking Value that calculated by hardware when PEC is enabled. |
| 7 | DUMODF | Dual Flag in slave mode indicating which address is matched in Dual-Address mode<br>This bit is cleared by hardware after a STOP or a START condition or I2CEN=0<br>0: SADDR0 address matches<br>1: SADDR1 address matches |
| 6 | HSTSMB | SMBus Host Header detected in slave mode<br>This bit is cleared by hardware after a STOP or a START condition or I2CEN=0<br>0: No SMBus Host Header detected<br>1: SMBus Host Header detected |
| 5 | DEFSMB | SMBus host header in slave mode<br>This bit is cleared by hardware after a STOP or a START condition or I2CEN=0.<br>0: SMBus Device has no default address<br>1: Received a default address for SMBus Device |
| 4 | RXGC | General call address (00h) received.<br>This bit is cleared by hardware after a STOP or a START condition or I2CEN=0.<br>0: No general call address (00h) received<br>1: General call address (00h) received |
| 3 | Reserved | Must be kept the reset value |
| 2 | TRS | Whether the I2C is a transmitter or a receiver<br>This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB.<br>0: Receiver<br>1: Transmitter |
| 1 | I2CBSY | Busy flag<br>This bit is cleared by hardware after a STOP condition<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 cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB.<br>0: Slave mode<br>1: Master mode |

### 14.4.8. Clock configure register (I2C_CKCFG)

Address offset: 0x1C

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | FAST | I2C speed selection in master mode<br>0: Standard speed<br>1: Fast speed |
| 14 | DTCY | Duty cycle in fast mode<br>0: $T_{low}/T_{high} = 2$<br>1: $T_{low}/T_{high} = 16/9$ |
| 13:12 | Reserved | Must be kept the 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 if DTCY=0:<br>$T_{high} = CLKC * T_{PCLK1}$ , $T_{low} = 2 * CLKC * T_{PCLK1}$<br>In fast speed mode if DTCY=1:<br>$T_{high} = 9 * CLKC * T_{PCLK1}$ , $T_{low} = 16 * CLKC * T_{PCLK1}$ |

### 14.4.9. Rise time register (I2C_RT)

Address offset: 0x20

Reset value: 0x0002

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:6 | Reserved | Must be kept the reset value |
| 5:0 | RISETIME[5:0] | Maximum rise time in master mode<br>The RISETIME value should be the maximum SCL rise time incremented by 1. |

### 14.4.10. SAM control and status register (I2C_SAMCS) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000

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

| 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 |
| r_w0 | r_w0 | r_w0 | r_w0 | | | r | r | rw | rw | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | RFR | Rxframe rise flag, cleared by software write 0 |
| 14 | RFF | Rxframe fall flag, cleared by software write 0 |
| 13 | TFR | Txframe rise flag, cleared by software write 0 |
| 12 | TFF | Txframe fall flag, cleared by software write 0 |
| 11:10 | Reserved | Must be kept the reset value |
| 9 | RXF | Level of Rxframe signal |
| 8 | TXF | Level of Txframe signal |
| 7 | RFRIE | Rxframe rise interrupt enable<br>0: Disable<br>1: Enable |
| 6 | RFFIE | Rxframe fall interrupt enable<br>0: Disable<br>1: Enable |
| 5 | TFRIE | Txframe rise interrupt enable<br>0: Disable<br>1: Enable |
| 4 | TFFIE | Txframe fall interrupt enable<br>0: Disable<br>1: Enable |
| 3:2 | Reserved | Must be kept the reset value |
| 1 | STOEN | SAM_V interface timeout detect enable<br>0: Disable<br>1: Enable |

| | | |
|---|---|---|
| 0 | SAMEN | SAM_V interface enable<br>0: Disable<br>1: Enable |

# 15. Serial peripheral interface/Inter-IC sound(SPI/I2S)

## 15.1. Introduction

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 data transmit and receive function in both master and slave mode. For GD32F130xx and GD32F150xx devices, the SPI Interface supports single wire configuration in both master and slave mode. But for GD32F170xx and GD32F190xx devices, the SPI Interface also supports quad wire configuration in master mode except supporting single wire configuration. In single wire configuration, the SPI interface uses 4 pins, among which are the serial data input and output lines MISO and MOSI, the clock line (SCK), and the slave select line (NSS). In quad wire configuration, SPI uses 6 pins: MOSI, MISO, IO2, IO3, SCK and NSS. One SPI device acts as a master which controls the data flow using the NSS and SCK signals to indicate the start of the data communication and the data sampling rate. To receive a data byte, the streamed data bits are latched on a specific clock edge and stored in the Data transfer register. Data transmission is carried in a similar way but with a reverse sequence. The configuration fault detection provides a capability for multi-master applications. The SPI interface may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

The inter-IC sound function supports four audio standards, including I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. It can operate in four modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode.

## 15.2. Main features

### 15.2.1. SPI features

- Master or slave operation
- Programmable clock bit rate
- Programmable clock polarity and phase
- Separate transmit and receive buffer, 16 bits wide
- Programmable data frame size, 8 or 16 bits
- Programmable data order, transmit MSB-first or LSB-first
- Hardware CRC calculation and transmit automatic CRC error checking

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines
- NSS work in software mode or hardware mode for both master and slave
- SPI bus busy status flag
- Transmission and reception flags with interrupt capability
- Master configuration fault, overrun and CRC error flags with interrupt capability
- Transmission and reception by DMA capability

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Quad wire configuration available in master mode(SPI1)

### 15.2.2. I2S features

- Supported I2S standards:
  - I2S Phillips standard
  - MSB justified standard
  - LSB justified standard
  - PCM standard (both short and long frame synchronization mode)
- Supported operation modes:
  - Master transmission
  - Master reception
  - Slave transmission
  - Slave reception
- The data length can be 16 bits, 24 bits or 32 bits
- The channel length can be 16 bits or 32 bits
- 16-bit shift register for transmission and reception
- Data direction is always MSB first
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies from 8 kHz to 192 kHz
- Programmable idle state clock polarity
- Master clock can be output to drive an external audio component
- Error flags including the transmission underrun error flag (TXURERR) and the reception

overrun error flag (RXORERR)

- DMA capability for both transmission and reception

## 15.3. SPI function description

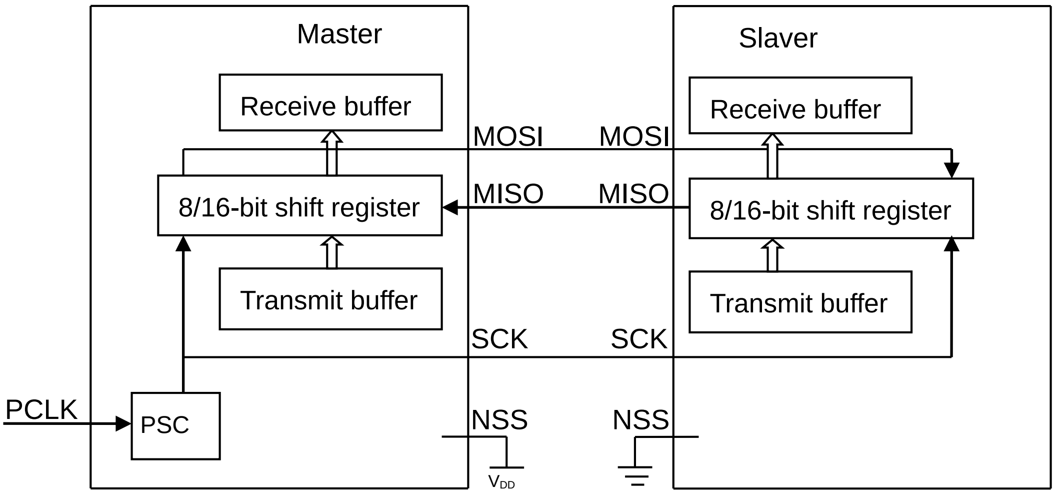
### 15.3.1. Pin configuration (Single wire, default)

The SPI is connected to external devices through 2-4 pins in different modes:

- MISO: This pin is used to receive data in master mode (Master In) or transmit data in slave mode (Slave Out).
- MOSI: This pin is used to transmit data in master mode (Master Out) or receive data in slave mode (Slave In).
- SCK: This pin is used to output clock in master mode or receive clock in slave mode.
- NSS: In hardware mode (SWNSSEN bit is cleared), the NSS pin can be used as an input. The NSS pin should be driven high in master mode or driven low in slave mode. It was a fault if the NSS pin is driven low in master mode, CONFERR bit will be set and MSTMOD will be cleared by hardware. The NSS pin is driven high in slave mode which means the chip is not selected, the SPI would not work until the NSS pin is driven low. In software mode (SWNSSEN bit is set), the SWNSS bit replaces the function of the NSS pin, the NSS pin can be used as a standard IO ports. In addition, the NSS pin can be used as an output in master mode when NSSDRV is set, the NSS pin will be driven low when SPI starts.

Typical interconnections between a single master and a single slave:

**Figure 15-1. Single master/ single slave application**



The MOSI pins are connected, the MISO pins are connected and the SCK pins are connected.

Data is transferred from master to slave by MOSI line and transferred from slave to master by MISO line. The clock is created in master and transferred to slave by SCK. In hardware mode (SWNSSEN bit is cleared), the NSS is driven high in master mode or driven low in slave mode; in software mode (SWNSSEN bit is set), the NSS pin is not used.

The master device controls the communication by the clock. When the master device transmits data to the slave device via MOSI pin, it sends clock to the slave device at same time via SCK pin, the slave receives data via MOSI pin and transmits data via MISO pin according to the clock.

The NSS pin can be used in software mode or hardware mode by the SWNSSEN bit in the SPI_CTL0 register.

- Hardware NSS mode (SWNSSEN = 0)

  Depending on the NSSDRV bit in SPI_CTL1 register, the NSS pin can be used as input or output (only in master mode).

  - NSS output enabled (SWNSSEN = 0, NSSDRV = 1)

    This configuration is used only in master mode. The NSS signal is driven low when the master starts the communication (start to transmit clock) and is kept low until the SPI is disabled.

  - NSS output disabled (SWNSSEN = 0, NSSDRV = 0)

    The NSS pin is used as input. In master mode, the NSS pin should be driven high. In slave mode, the NSS pin acts as a chip select, the slave is selected when NSS is low and deselected when NSS high.

- Software NSS mode (SWNSSEN = 1)

  The SWNSS bit in SPI_CTL0 replace the function of the NSS pin, in master mode, the SWNSS bit should be set. In slave mode, the slave is selected when SWNSS is cleared and deselected when SWNSS is set. The NSS pin is not used in SPI communication.

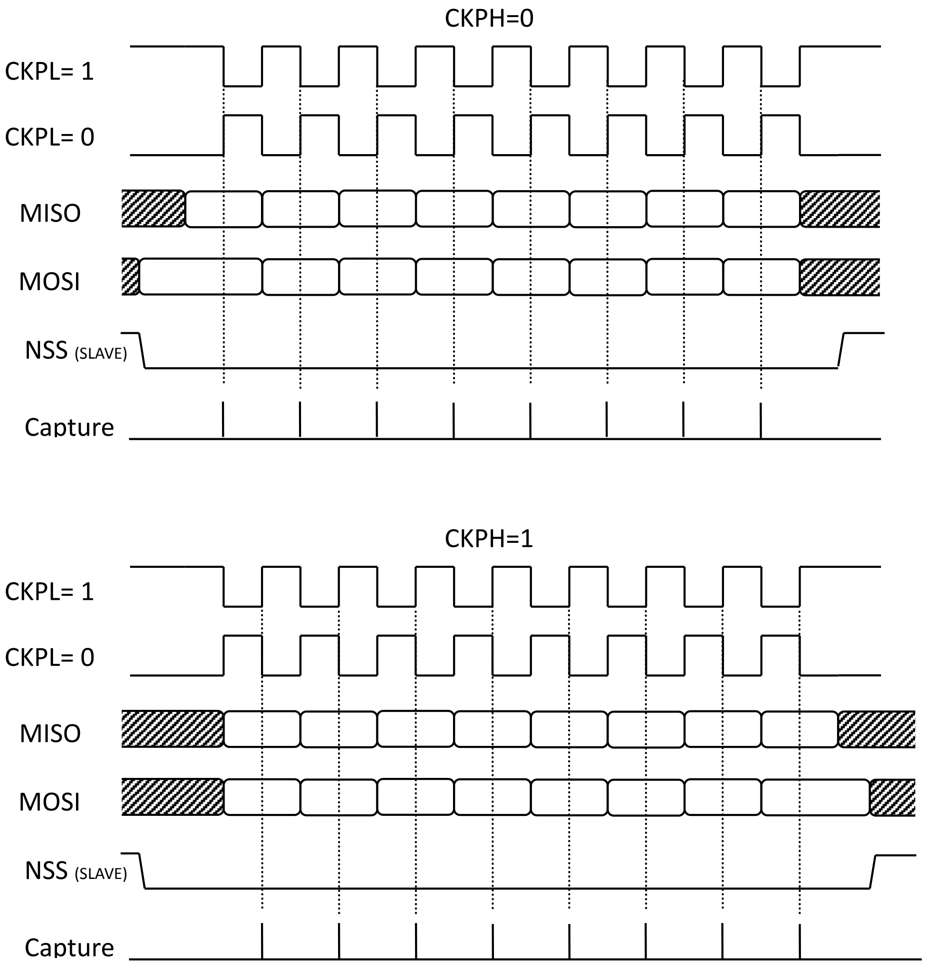
### Clock phase and clock polarity

Using the CKPL and CKPH bits in SPI_CTL0 register, four types of the timing relationship can be configured. The CKPL bit controls the steady state value of the clock. If the CKPL is cleared, the SCK pin is low when idle. If the CKPL is set, the SCK pin is high when idle.

The CKPH determines the timing of capturing the data. If the CKPH is cleared, capturing the first data at the first edge on the SCK pin. If the CKPH is set, capturing the first data at the second edge on the SCK pin.

The following figure shows an SPI transfer with the four types of timing relationship:

**Figure 15-2. SPI data clock timing diagram**



### Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LF bit in the SPI_CTL0 Register.

Depending on the LF bit in SPI_CTL0 register, the MSB (LF=0) or the LSB (LF=1) will be sent out at first. And the length of data is 8 bits or 16 bits depending on the FF16 bit in the SPI_CTL0 register.

## 15.3.2. Pin configuration (Quad wire) for GD32F170xx and GD32F190xx devices

SPI is in single wire configuration by default and enters into quad wire mode after QMOD bit in SPI_QCTL register is set (only available in SPI1).

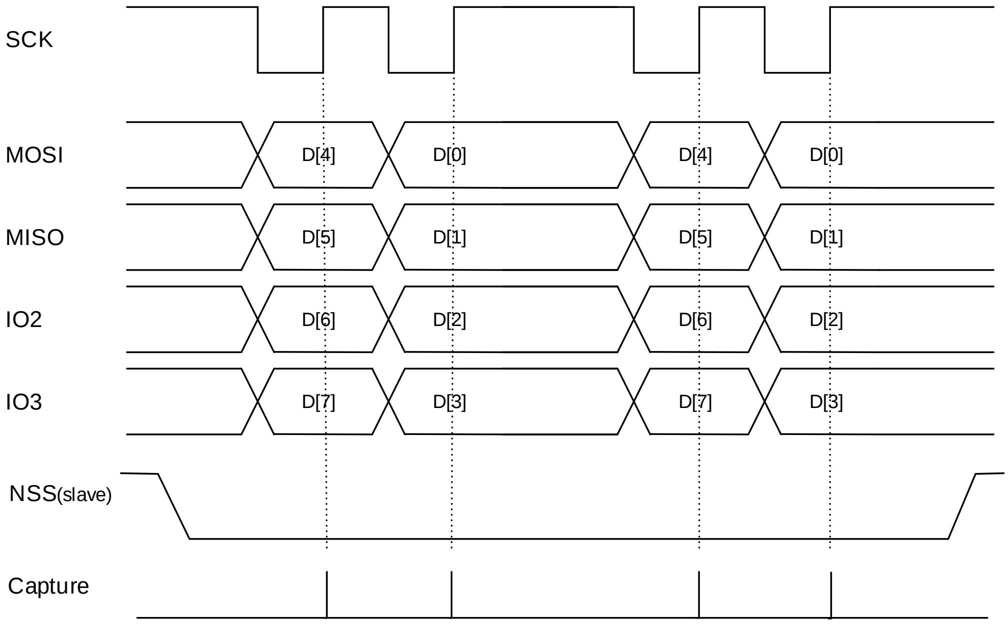
The SPI is connected to external devices through 6 pins in quad wire configuration:

- MOSI: This pin is used to transmit data in quad write mode and receive data in quad read mode.

- MISO: This pin is used to transmit data in quad write mode and receive data in quad read mode.
- IO2: This pin is used to transmit data in quad write mode and receive data in quad read mode.
- IO3: This pin is used to transmit data in quad write mode and receive data in quad read mode.
- SCK: The configuration and behavior of SCK pin is the same as single wire mode except that there are only 2 clock cycles per data frame in quad wire configuration.

Data frame format: The frame length is fixed to 8 bits in quad wire mode, as shown below.

**Figure 15-3. SPI data clock timing diagram in quad wire mode (CKPL=1, CKPH=1)**



## 15.3.3. SPI slave mode

In slave mode, the serial clock is received from master device, the PSC[2:0] bits in the SPI_CTL0 register are useless. The communication is controlled by the master device and the slave should be enabled before the master sends the clock.

SPI slave mode configuration steps:

1. Program data format (FF16 bit in the SPI_CTL0 register).
2. Program the timing relationships (CKPL and CKPH bits in the SPI_CTL0 register). They must be configured in the same way as the master device.
3. Program the frame format (LF bit in the SPI_CTL0 register), it must be same as the master device.

4. Program the NSS mode (SWNSSEN bit in the SPI_CTL0 register). In hardware mode (SWNSSEN=0), the NSS pin must be connected to a low level signal during transmit sequence. In software mode (SWNSSEN=1), the SWNSS bit in the SPI_CTL0 register must be cleared during transmit sequence.

5. Set the slave mode (Clear the MSTMOD bit). Enable the SPI (set the SPIEN bit).

### Transmit sequence

The slave transmit begins when the slave receives the clock via the SCK pin, the LSB or MSB transmit on its MOSI pin, the other bits are loaded from transmit buffer to shift-register. The TBE bit is set and the software writes the second data to the transmit buffer if it is necessary. The hardware transmits the bits in the shift-register according the clock received.

### Receive sequence

After the last sampling clock edge, data transfer is complete, the data in shift register is copied to receive buffer and the RBNE bit in SPI_STAT register is set. Reading SPI_DATA returns the data in receive buffer and the RBNE bit is cleared by reading the SPI_DATA register.

## 15.3.4. SPI master mode

In the master configuration, the serial clock is generated on the SCK pin.

In master mode, the serial clock is generated by the PCLK (PCLK2 for SPI0 or PCLK1 for SPI1), the PSC[2:0] is the baud rate.

SPI master mode configuration steps:

1. Program the PSC[2:0] bits to define the baud rate.

2. Program data format (FF16 bit in the SPI_CTL0 register).

3. Program the timing relationships (CKPL and CKPH bits in the SPI_CTL0 register). They must be configured in the same way as the slave device.

4. Program the frame format (LF bit in the SPI_CTL0 register), it must be same as the slave device.

5. Program the NSS mode (SWNSSEN bit in the SPI_CTL0 register). If the NSS pin is used as input, in hardware mode (SWNSSEN=0), the NSS pin must be connected to a high level signal. In software mode (SWNSSEN=1), the SWNSS bit in the SPI_CTL0 register must be set. If the NSS pin is used as output, the NSSDRV bit should be set, and the NSS pin will be driven low when the transfer begins.

6. Set the master mode (Set the MSTMOD bit). Enable the SPI (set the SPIEN bit).

### Transmit sequence

The master transmit begins when a data is written in the transmit buffer, the LSB or MSB transmit on its MOSI pin, the other bits are loaded from transmit buffer to shift-register. The TBE bit is set after the data is loaded from transmit buffer to shift-register. A continuous data can be transmitted if the data is put in the transmit buffer. Write SPI_DATA register when the TBE bit is set.

#### Receive sequence

After the last sampling clock edge, data transfer is complete, the data in shift register is copied to receive buffer and the RBNE bit in SPI_STAT register is set. Reading SPI_DATA returns the data in receive buffer and the RBNE bit is cleared by reading the SPI_DATA register.

### 15.3.5. SPI quad wire operation in master mode for GD32F170xx and GD32F190xx devices

The SPI quad wire mode is designed to control quad SPI flash, including but not limited to GD25QXX serials flash chip.

In order to enter quad wire mode, the software should first verify that the TBE bit is set and TRANS bit is cleared, and then set QMOD bit in SPI_QCTL register. In quad wire mode, BDEN, BDOEN, CRCERRN, CRCNT, FF16, RO and LF in SPI_CTL0 register should be kept cleared and MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

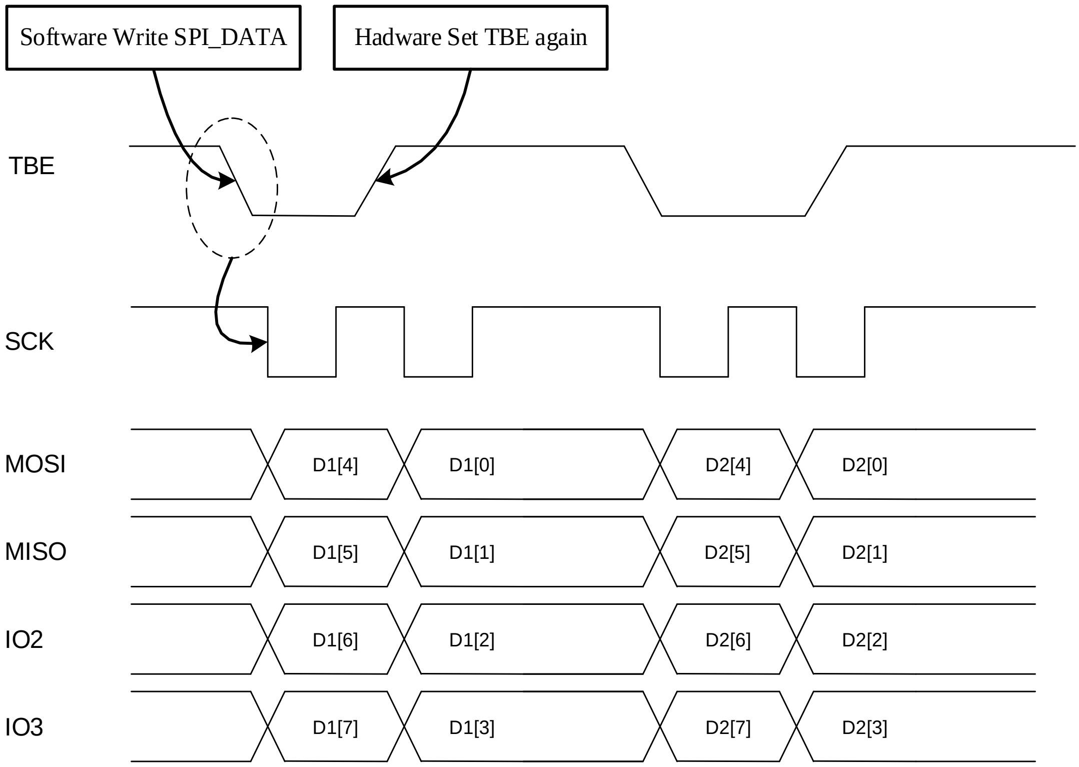
There are 2 operation modes in quad wire configuration: quad write and quad read, decided by QRD bit in SPI_QCTL register.

#### 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 TBE is cleared (SPI_DATA not empty) and SPIEN is set. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition not met.

The operation flow for transmitting in quad mode:

1. Configure clock presales, clock polarity, phase, etc in SPI_CTL0 and SPI_CTL1 based on your application requirements.
2. Set QMOD bit in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0.
3. Write the byte to SPI_DATA register and the TBE will be cleared.
4. Wait until TBE be set by hardware again before writing the next byte.



### Quad read operation

SPI works in quad read mode when QMOD and QRD 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 TBE is cleared (SPI_DATA not empty) and SPIEN is set. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition not met. So, software should always write dummy data into SPI_DATA to make SPI generate SCK and receive data.

The operation flow for receiving in quad mode:

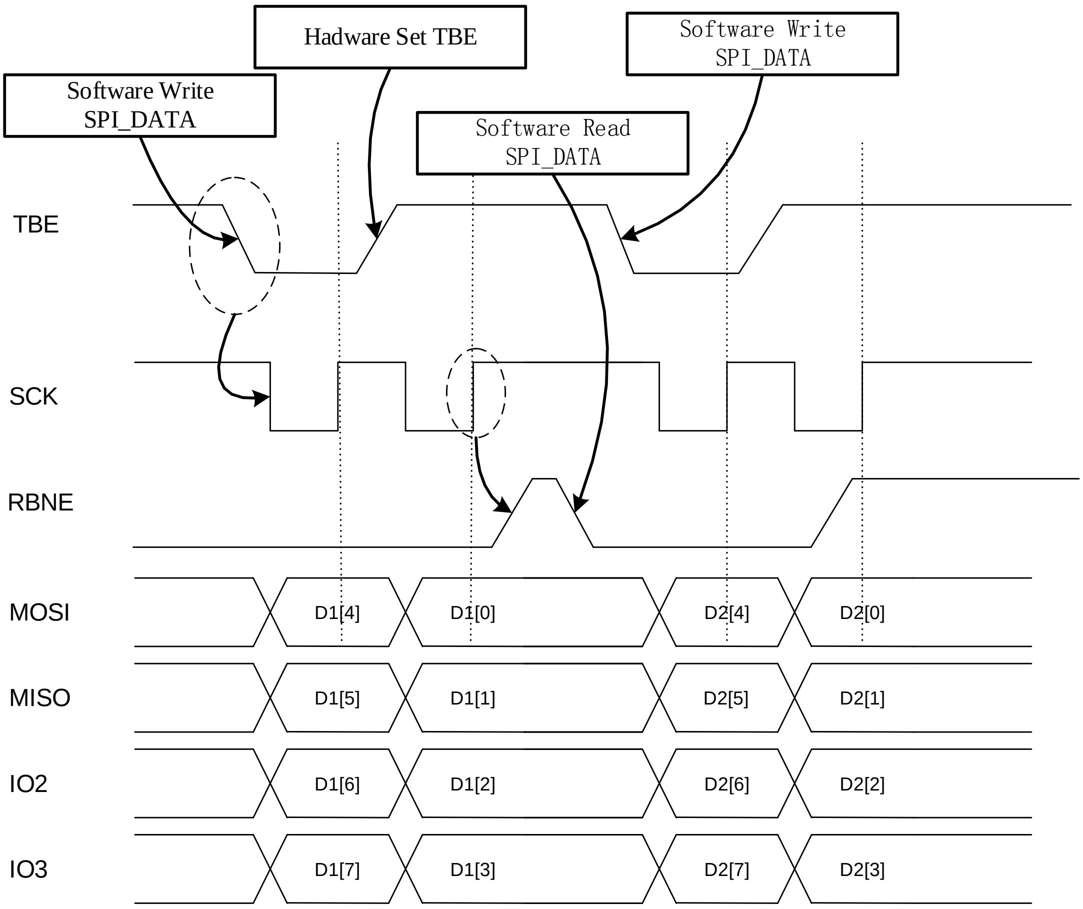
1. Configure clock presales, clock polarity, phase, etc in SPI_CTL0 and SPI_CTL1 based on your application requirements.

2. Set QMOD and QRD bits in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0.

3. Write an arbitrary byte (for example, 0xFF) to SPI_DATA register.

4. Wait the RBNE flag 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.



**Leave quad wire mode or disable SPI**

Before leaving quad wire mode or disabling SPI, software should first check that TBE bit is set and TRANS bit is cleared, then clear the QMOD bit in SPI_QCTL register or SPIEN bit in SPI_CTL0 register.

## 15.3.6. SPI simplex communication

The SPI is able to work in simplex mode in 2 configurations.

1. Set the BDEN bit in the SPI_CTL0 register. The clock is transmitted from the SCK pin of master to the SCK pin of slave. The data is transmitted between the MOSI pin of the master and the MISO pin of the slave. The transfer direction is defined by the BDOEN bit in the SPI_CTL0 register, the BDOEN bit in master and slave must be different. If the BDOEN is set in master and cleared in slave, the data is transmitted from master to slave. If the BDOEN is cleared in master and set in slave, the data is transmitted from slave to master.
2. BDEN is cleared, the RO bit in master and in slave should be different in simplex communication. If the RO bit is cleared in master and set in slave, the data is transmitted

from master to slave, the MOSI pin of master outputs the data and the MOSI pin of slave receives the data, the MISO pins are not used. If the RO bit is set in master and cleared in slave, the data is transmitted from slave to master, the MISO pin of slave outputs the data and the MISO pin of master receives the data, the MOSI pins are not used. Anyway the clock is generated in master and transmitted to slave by SCK pins.

In simplex communication, if the master is configured in receive-only mode and the slave is configured in transmit-only mode, the master starts receiving data immediately when the SPI is enable. The slave should get ready before the master enables the SPI, and write data in transmit buffer in a limited time. The master should disable the SPI (clear the SPIEN bit) after the last second data is received; the last data is being transmitted at that time, and the SPI will be disabled by hardware after the last data received.

### 15.3.7. Data Rx and Tx procedures

There are two buffers for each SPI module, transmit buffer and receive buffer, a write access to the SPI_DATA stores the data into the transmit buffer and a read access to the SPI_DATA returns the data in the receive buffer.

When the previous data is transmitted over, the data in the transmit buffer is copied to the shift register, and the TBE bit is set by hardware, then the software can write the next data to the transmit buffer by writing it to the SPI_DATA register if it is necessary, an interrupt is generated when the TBE is set if the TBEIE bit in the SPI_CTL1 register is set. Writing the SPI_DATA register can clear the TBE bit. Writing the SPI_DATA register when the TBE bit is cleared will cover the data stored in the transmit buffer.

When the last bit of one data is captured on the sampling clock edge, the data is received and stored in shift register, then the hardware copies the data in shift register to the receive buffer and sets the RBNE bit. It’s ready to be read by software. An interrupt is generated when the RBNE is set if the RBNEIE bit in the SPI_CTL1 register is set. Reading the SPI_DATA register can clear the RBNE bit. If one data is received when the RBNE is set (the last data have not be read), the RXORERR bit is set to indicate that was fault.

### 15.3.8. CRC calculation

CRC calculation increases communication reliability. Two CRC calculators are implemented for transmitted data and received data. The calculators calculate the CRC value serially on each bit. The polynomial used in calculation is programmable and stored in SPI_CRCPOLY register.

CRC calculation is enabled by setting the CRCEN bit in the SPI_CTL0 register. In full-duplex or transmit-only mode, the software should set the CRCNT bit immediately after the last data is written to the SPI_DATA, last the SPI_TCRC value will be transmitted after the last data. If the data is transmitted by DMA, the SPI_TCRC value is transmitted by hardware and the CRCNT is not used. When the SPI_TCRC value is being transmitted, the data received is considered to be the CRC value of the received data, the calculator is switched off and the data is compared with the SPI_RCRC, the CRCERR is set if they are different.

In receive-only mode, the software should write the CRCNT bit after the second last data has been received (The last data is being received at that time). The CRC value of the received data is received after the last data and is compared with the SPI_RCRC, the CRCERR is set if they are different.

The CRC value is transmitted only when the transmit buffer is empty. During CRC transmission, the CRC calculator is switched off.

SPI communication using the CRC:

1. Program the CKPL, CKPH, LF, PSC, SWNSSEN, SWNSS and MSTMOD values.
2. Program the polynomial in the SPI_CRCPOLY register.
3. Enable the CRC calculation by setting the CRCEN bit in the SPI_CTL0 register.
4. Enable the SPI by setting the SPIEN bit in the SPI_CTL0 register.
5. Start the communication.
6. In full-duplex or transmit-only mode, set the CRCNT bit immediately after the last data is written to the SPI_DATA. In receive-only mode, set the bit CRCNT after the reception of the second to last data. CRC calculation is switched off during the CRC transfer.
7. The SPI transfers the CRC value after the last data. The received CRC value is compared with the SPI_RCRC, the CRCERR is set if they are different.

If the CRCEN is set in slave mode, CRC calculator is still work even if the NSS pin is pulled high. The CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation. The CRC value (SPI_RCRC and SPI_TCRC) is cleared when the SPIEN bit or the CRCEN is cleared.

## 15.3.9. Status flags and error flags

### Status flags

- ■ Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing it to the SPI_DATA register. If the TBEIE bit is set, an interrupt is generated when TBE is set. The TBE bit is cleared by writing it to the SPI_DATA register.

- ■ Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, one data is received and stored in the receive buffer, and software can read the data by reading the SPI_DATA register. If the RBNEIE bit is set, an interrupt is generated when RBNE is set. The RBNE bit is cleared by reading the SPI_DATA register.

- SPI Transmitting On-Going flag (TRANS)

This TRANS flag is set and cleared by hardware. It indicates the state of the communication layer of the SPI. The TRANS flag is useful to detect the end of a transfer if the software wants to disable the SPI. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The TRANS bit is set when a transfer starts, except in bidirectional receive-only mode of master device (MSTMOD=1 and BDEN=1 and BDOEN=0). It is cleared when the SPI is disabled or a configuration fault (CONFERR=1) occurs. When communication is not continuous, TRANS flag is low between each communication. When communication is continuous, TRANS flag is low between each communication in slave mode and kept high during all the transfers in master mode.

### Error flags

- Configuration Fault Error (CONFERR)

In NSS hardware mode and the NSSDRV is not enable, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the SWNSS bit is low. When the CONFERR is set, an interrupt is generated if the ERRIE bit 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 CONFERR bit is cleared by the following software sequence:

1. Read from or write to the SPI_STAT register.
2. Write to the SPI_CTL0 register.

The SPIEN and MSTMOD bits are in 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. That means, the last data has not be read out and the new data is received. The receive buffer contents will be cover with the newly received data, and the last data is lost. An interrupt is generated when the RXORERR bit is set if the ERRIE bit is set.

The RXORERR bit is cleared by the following software sequence: a read access to SPI_DATA register, then a read access to SPI_STAT register.

- 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. Clear the CRCERR bit by writing 0 to this bit, write 1 to this bit has no effect.

### 15.3.10. Disabling the SPI

When a transfer is finished, the software stops the SPI by clearing the SPIEN bit. In some configurations, the last transfer is ongoing after the SPIEN is cleared. To avoid corrupting the last transfer, follow the steps below:

Full-duplex mode

1. Wait until RBNE=1 to receive the last data
2. Wait until TBE=1
3. Wait until TRANS=0
4. Disable the SPI (SPIEN=0), enter the Halt mode or disable the peripheral clock

Transmit-only mode

1. The last data is written into the SPI_DATA register
2. Wait until TBE=1
3. Wait until TRANS=0
4. Disable the SPI (SPIEN=0), enter the Halt mode or disable the peripheral clock

Receive-only mode of master

1. Wait for the second last RBNE=1
2. Wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPIEN=0)
3. Wait for the last RBNE=1 before entering the Halt mode or disabling the peripheral clock

Receive-only mode of slave

1. The SPI can be disabled (write SPIEN=0) at any time, the SPI is effectively disabled after the current transfer complete
2. Wait until TRANS = 0 then entering the Halt mode or disabling the peripheral clock.

### 15.3.11. DMA requests

Using DMA to transmit or receive data will let the SPI operate at its maximum speed. Read and write SPI_DATA is fast enough and it’s no interstice between data that will be transmitted.

A DMA access is enabled if the DMATEN bit or the DMAREN bit in the SPI_CTL1 register is set. When the TBE is set, a DMA request is issued in transmit channel of DMA, the TBE bit is cleared after the DMA writes a data to SPI_DATA register. When the RBNE is set, a DMA request is issued in receive channel of DMA, the RBNE bit is cleared after the DMA reads a data from SPI_DATA register.

When the SPI is only used for transmitting data, it is possible to enable the SPI Tx DMA channel only. In this case, because the data received are not read, the RXORERR flag is set.

When the SPI is only used for receiving data, it is possible to enable the SPI Rx DMA channel only.

In transmission mode, when all the data to be transmitted has been written by the DMA (flag FTFIF is set in the DMA_INTF register), the TRANS flag can be monitored to ensure that the SPI communication is complete. Before disabling the SPI or entering the halt mode, it is required to avoid corrupting the last transmission. The software must first wait until TBE=1 and then until TRANS=0.

**Figure 15-4. Transmission using DMA**

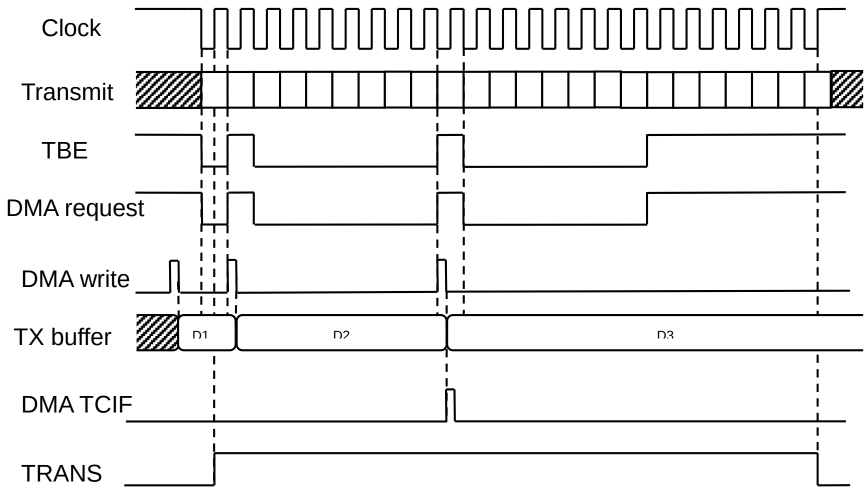
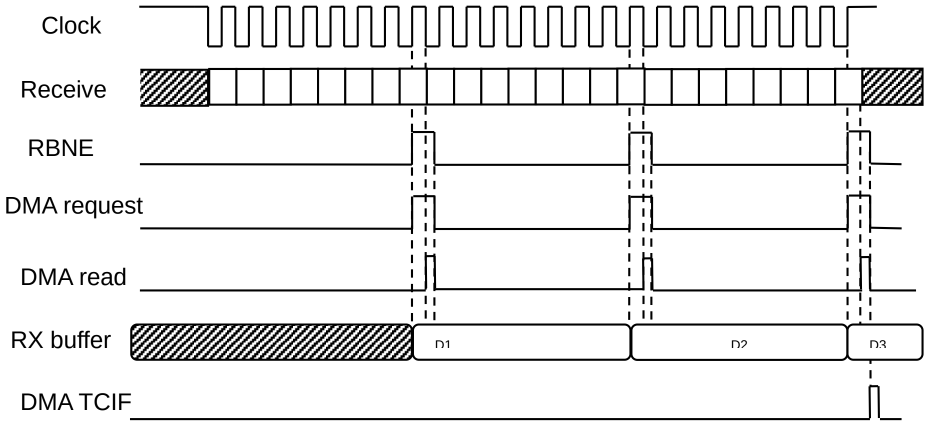


**Figure 15-5. Reception using DMA**



If the CRCEN bit is set when using DMA in SPI communication, the CRC value transmitted and received after the last data are automatic. The CRCNT is no useful. The CRC value in receive buffer should be read to clear the RBNE bit.

## 15.3.12. SPI interrupts

**Table 15-1. SPI interrupt requests**

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| Transmit buffer empty | TBE | TBEIE |

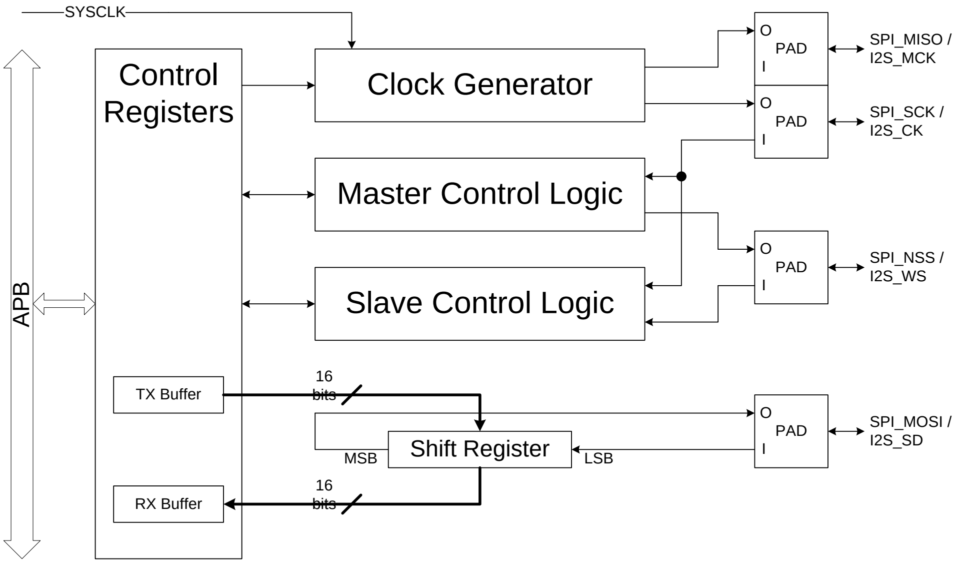
| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| Receive buffer not empty | RBNE | RBNEIE |
| Configuration Fault Error | CONFERR | ERRIE |
| Rx Overrun Error | RXORERR | |
| CRC error | CRCERR | |

## 15.4. I2S function description

### 15.4.1. General description

The block diagram of I2S is shown in the following figure.

**Figure 15-6. I2S block diagram**



The I2S shares the same pins, flags, interrupts, data buffers, and shift register with SPI. When the I2SSEL bit in the SPI_I2SCTL register is set, the resources are occupied by I2S. Or, they are used by SPI.

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 data 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, which shares the same pin with SPI_MISO. I2S_MCK is an optional signal for I2S interface. It produces a frequency rate equal to 256 x Fs, where Fs is the audio sampling frequency.

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.

## 15.4.2. Supported audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI_I2SCTL register. Four audio standards are supported, including I2S Phillips 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 bits 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 a frame. In the case that the data length is 16 bits, only one write or read operation to or from the SPI_DATA register is needed to complete 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 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 Phillips standard

For I2S Phillips standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The timing diagrams for each configuration are shown below.

**Figure 15-7. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

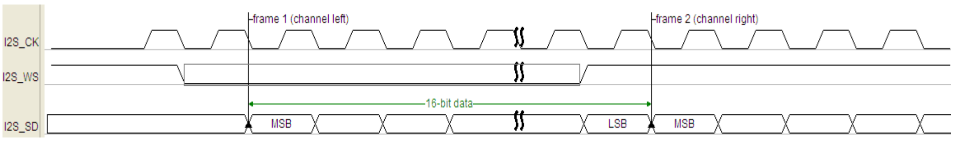
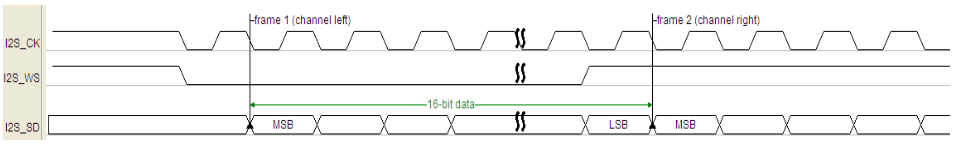


**Figure 15-8. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



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 a frame.

Figure 15-9. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

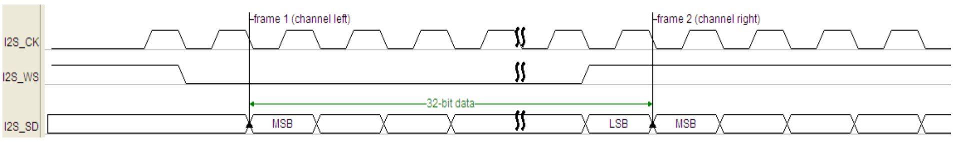
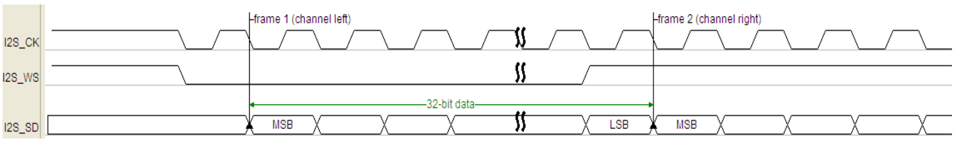


Figure 15-10. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)



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 a frame. In transmission mode, if 0x8899AABB is going to be sent, the first data written to the SPI_DATA register should be 0x8899, and the second one should be 0xAABB. In reception mode, if 0x8899AABB is received, the first data read from the SPI_DATA register should be 0x8899, and the second one should be 0xAABB.

Figure 15-11. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

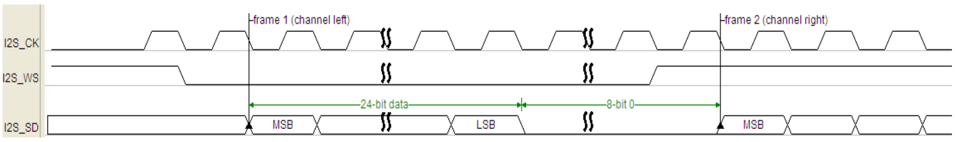
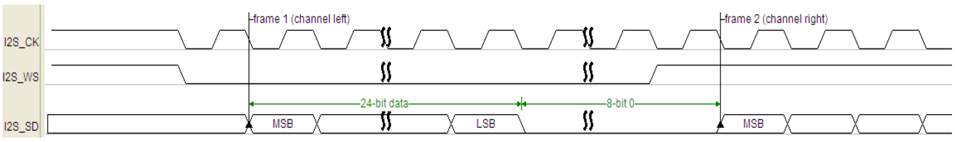


Figure 15-12. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



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 0x8899AA is going to be sent, the first data written to the SPI_DATA register should be 0x8899, and the second one should be 0xAAXX (the 8 LSB could be any value, but forced to 0x00 instead by hardware). In reception mode, if 0x8899AA is received, the first data read from the SPI_DATA register should be 0x8899, and the second one should be 0xAA00.

Figure 15-13. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

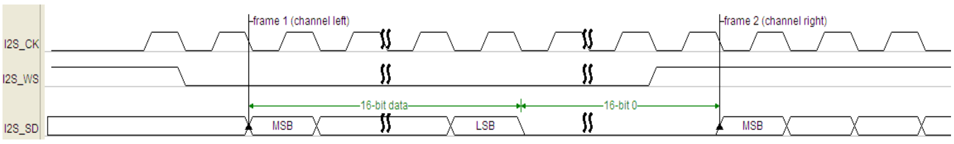
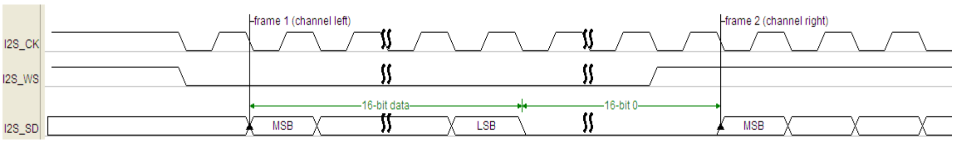


**Figure 15-14. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



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 a frame. The 16 remaining 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 Phillips standard. The timing diagrams for each configuration are shown below.

**Figure 15-15. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

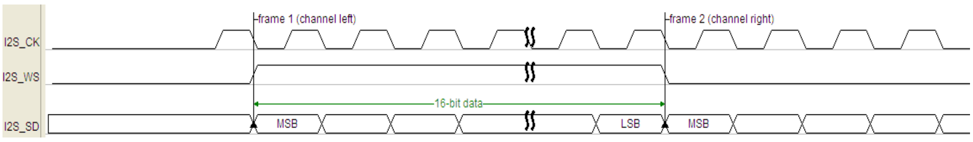


**Figure 15-16. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

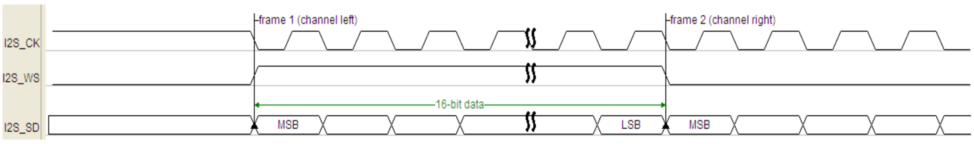


**Figure 15-17. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

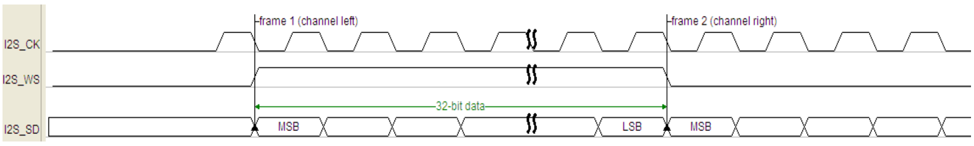


**Figure 15-18. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

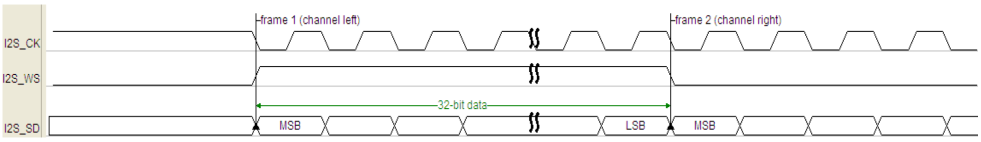


Figure 15-19. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

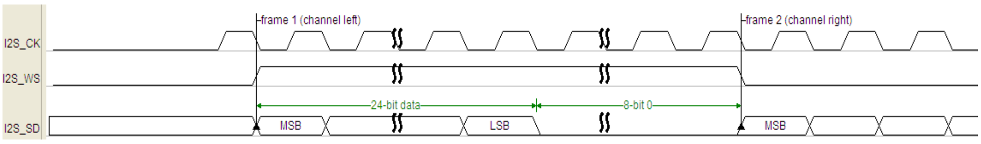


Figure 15-20. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

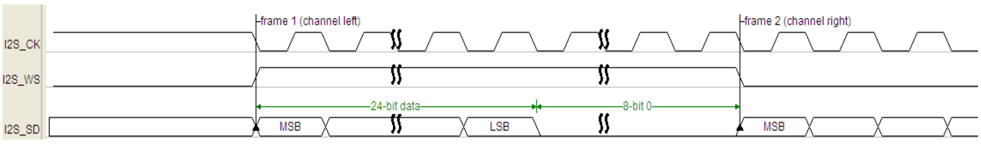


Figure 15-21. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

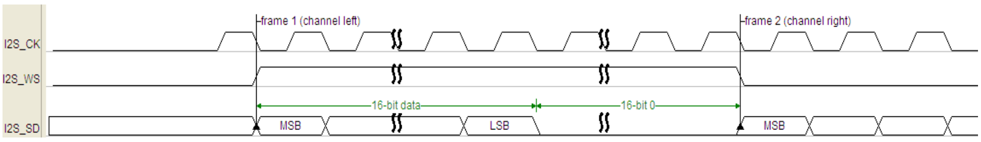
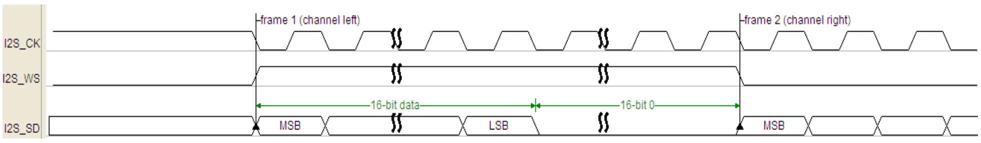


Figure 15-22. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



## 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 15-23. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

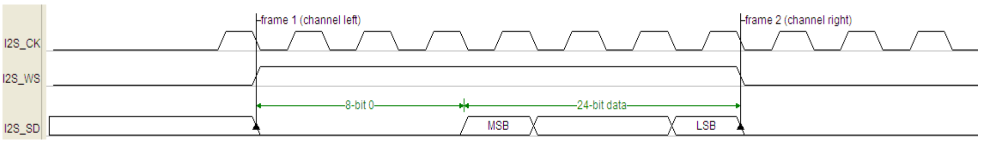
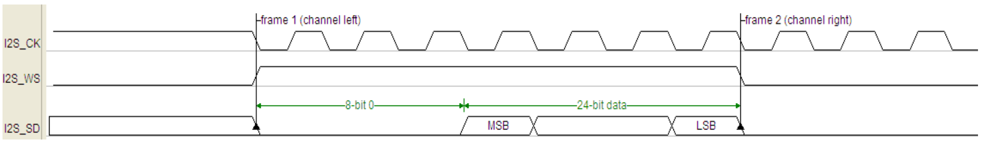


Figure 15-24. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



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 a frame. In transmission mode, if 0x8899AA is going to be sent, the first data written to the SPI_DATA register should be 0xXX88 (the 8 MSB could be any value, but forced to 0x00 instead by hardware), and the second one should be 0x99AA. In reception mode, if 0x8899AA is received, the first data read from the SPI_DATA register should be 0x0088, and the second one should be 0x99AA.

**Figure 15-25. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

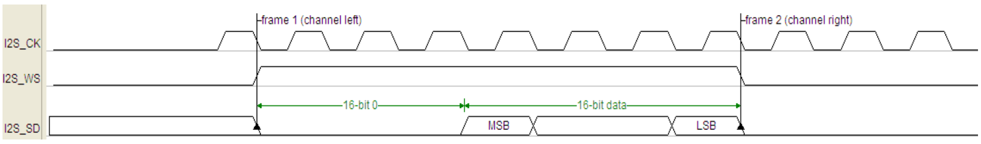
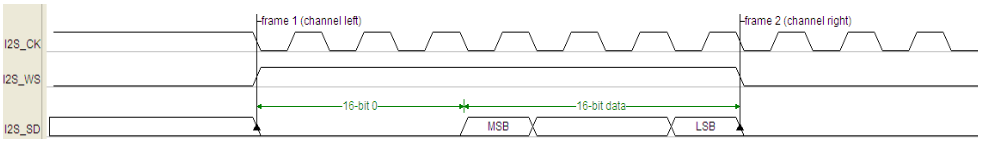


**Figure 15-26. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



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 a frame. The 16 remaining 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 Phillips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

**Figure 15-27. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

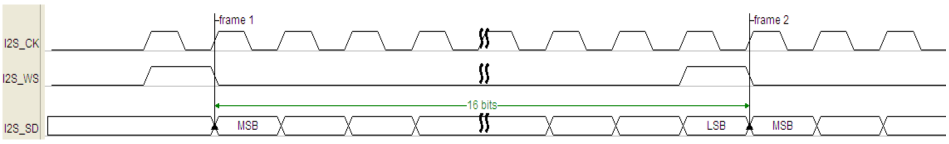


**Figure 15-28. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

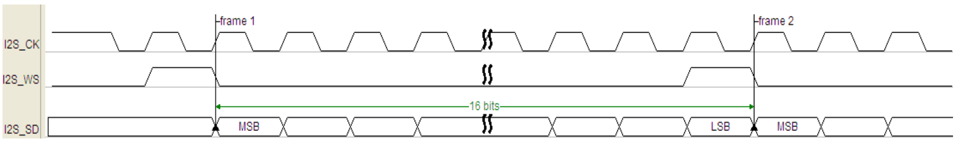


Figure 15-29. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

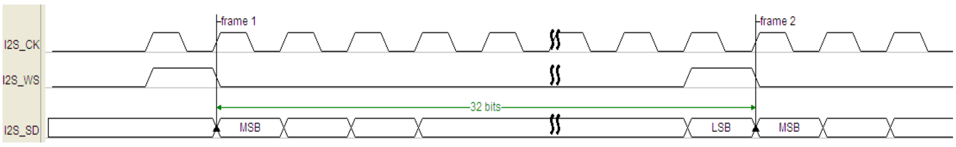


Figure 15-30. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

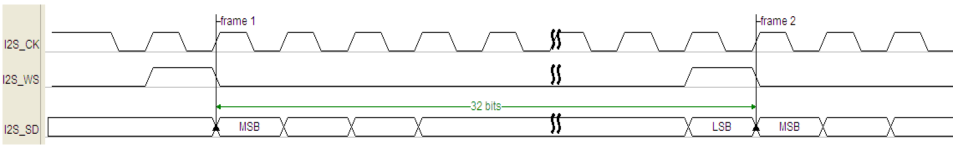


Figure 15-31. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

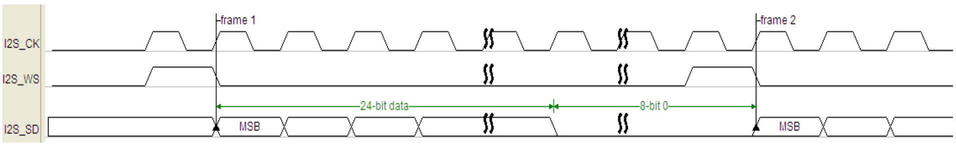


Figure 15-32. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

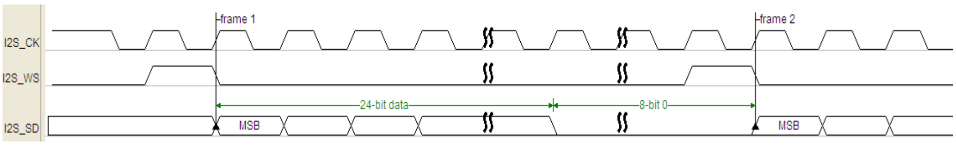


Figure 15-33. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

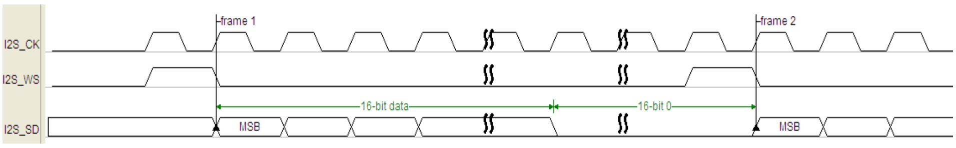
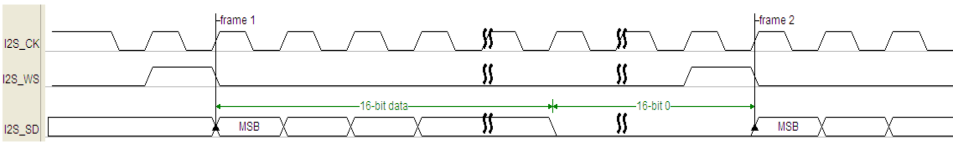


Figure 15-34. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 15-35. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

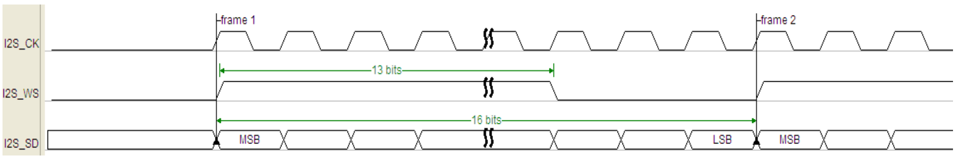


**Figure 15-36. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

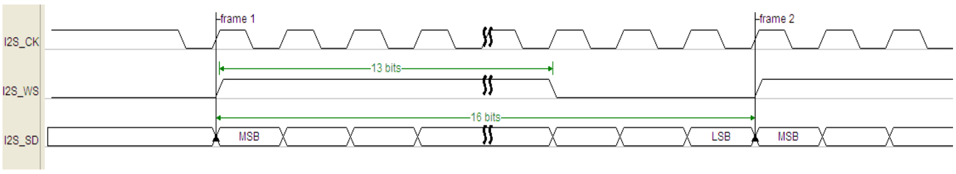


**Figure 15-37. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

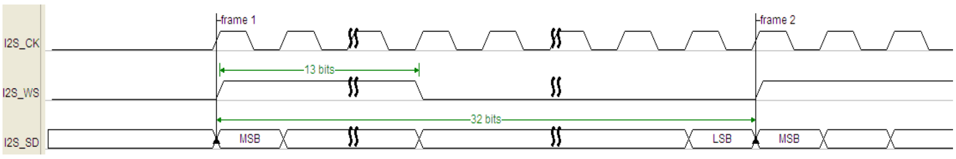


**Figure 15-38. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

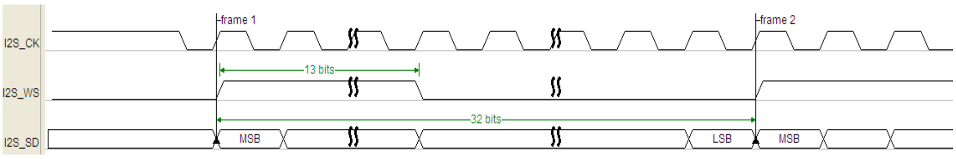


**Figure 15-39. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

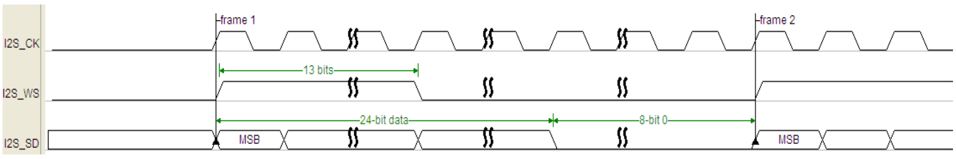


**Figure 15-40. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

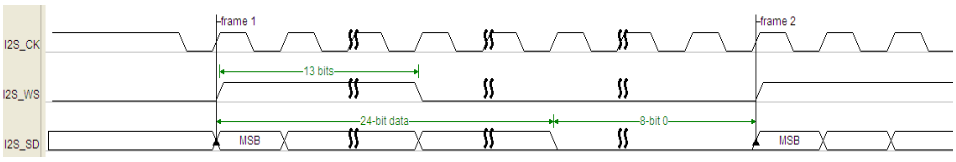


**Figure 15-41. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

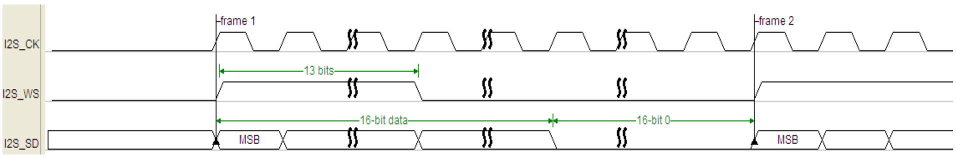
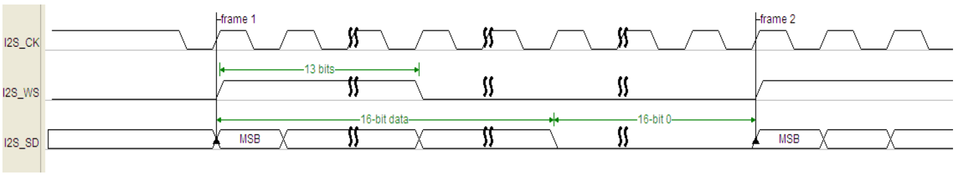
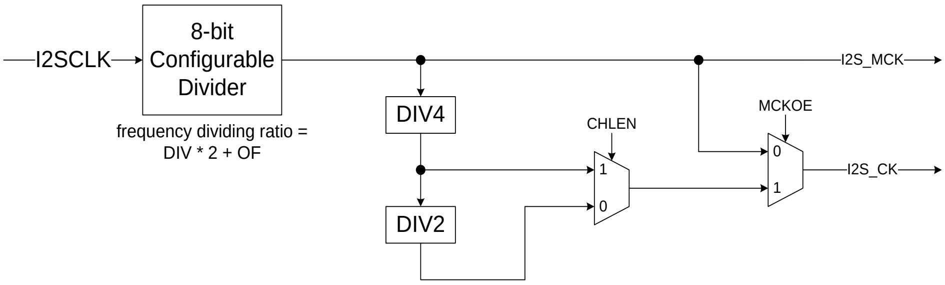


**Figure 15-42. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



## 15.4.3. Clock generator

**Figure 15-43. Block diagram of I2S clock generator**



The block diagram of I2S clock generator is shown above. 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 SYSCLK. The I2S bitrate can be calculated by the formulas shown in the following table.

**Table 15-2. 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 the following table.

**Table 15-3. 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)) |

The configuration and precision of audio sampling frequencies in common use are listed in the following table. The precision is calculated in the case when using standard 8 MHz HXTAL.

**Table 15-4. Audio sampling frequency configuration and precision**

| Target Fs(Hz) | SYSCLK (MHz) | MCKOEN | CHLEN = 0 | | | | CHLEN = 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | DIV | OF | Real Fs(Hz) | Error | DIV | OF | Real Fs(Hz) | Error |
| 96000 | 72 | No | 11 | 1 | 97826.09 | 1.90 % | 6 | 0 | 93750 | 2.34 % |
| 96000 | 72 | Yes | 1 | 1 | 93750 | 2.34 % | 1 | 1 | 93750 | 2.34 % |
| 96000 | 48 | No | 8 | 0 | 93750 | 2.34 % | 4 | 0 | 93750 | 2.34 % |
| 96000 | 48 | Yes | 1 | 0 | 93750 | 2.34 % | 1 | 0 | 93750 | 2.34 % |
| 48000 | 72 | No | 23 | 1 | 47872.34 | 0.27 % | 11 | 1 | 48913.04 | 1.90 % |
| 48000 | 72 | Yes | 3 | 0 | 46875 | 2.34 % | 3 | 0 | 46875 | 2.34 % |
| 48000 | 48 | No | 15 | 1 | 48387.1 | 0.81 % | 8 | 0 | 46875 | 2.34 % |
| 48000 | 48 | Yes | 2 | 0 | 46875 | 2.34 % | 2 | 0 | 46875 | 2.34 % |
| 44100 | 72 | No | 25 | 1 | 44117.65 | 0.04 % | 13 | 0 | 43269.23 | 1.88 % |
| 44100 | 72 | Yes | 3 | 0 | 46875 | 6.29 % | 3 | 0 | 46875 | 6.29 % |
| 44100 | 48 | No | 17 | 0 | 44117.65 | 0.04 % | 8 | 1 | 44117.65 | 0.04 % |
| 44100 | 48 | Yes | 2 | 0 | 46875 | 6.29 % | 2 | 0 | 46875 | 6.29 % |
| 32000 | 72 | No | 35 | 0 | 32142.86 | 0.44 % | 17 | 1 | 32142.86 | 0.44 % |

| Target Fs(Hz) | SYSCLK (MHz) | MCKOEN | CHLEN = 0 | | | | CHLEN = 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | DIV | OF | Real Fs(Hz) | Error | DIV | OF | Real Fs(Hz) | Error |
| 32000 | 72 | Yes | 4 | 1 | 31250 | 2.34% | 4 | 1 | 31250 | 2.34% |
| 32000 | 48 | No | 23 | 1 | 31914.89 | 0.27% | 11 | 1 | 32608.7 | 1.90% |
| 32000 | 48 | Yes | 3 | 0 | 31250 | 2.34% | 3 | 0 | 31250 | 2.34% |
| 22050 | 72 | No | 51 | 0 | 22058.82 | 0.04% | 25 | 1 | 22058.82 | 0.04% |
| 22050 | 72 | Yes | 6 | 1 | 21634.61 | 1.88% | 6 | 1 | 21634.61 | 1.88% |
| 22050 | 48 | No | 34 | 0 | 22058.82 | 0.04% | 17 | 0 | 22058.82 | 0.04% |
| 22050 | 48 | Yes | 4 | 1 | 20833.33 | 5.52% | 4 | 1 | 20833.33 | 5.52% |
| 16000 | 72 | No | 70 | 1 | 15675.75 | 0.27% | 35 | 0 | 16071.43 | 0.45% |
| 16000 | 72 | Yes | 9 | 0 | 15625 | 2.34% | 9 | 0 | 15625 | 2.34% |
| 16000 | 48 | No | 47 | 0 | 15957.45 | 0.27% | 23 | 1 | 15957.45 | 0.27% |
| 16000 | 48 | Yes | 6 | 0 | 15625 | 2.34% | 6 | 0 | 15625 | 2.34% |
| 11025 | 72 | No | 102 | 0 | 11029.41 | 0.04% | 51 | 0 | 11029.41 | 0.04% |
| 11025 | 72 | Yes | 13 | 0 | 10817.3 | 1.88% | 13 | 0 | 10817.3 | 1.88% |
| 11025 | 48 | No | 68 | 0 | 11029.41 | 0.04% | 34 | 0 | 11029.41 | 0.04% |
| 11025 | 48 | Yes | 8 | 1 | 11029.41 | 0.04% | 8 | 1 | 11029.41 | 0.04% |
| 8000 | 72 | No | 140 | 1 | 8007.11 | 0.09% | 70 | 1 | 7978.72 | 0.27% |
| 8000 | 72 | Yes | 17 | 1 | 8035.71 | 0.45% | 17 | 1 | 8035.71 | 0.45% |
| 8000 | 48 | No | 94 | 0 | 7978.72 | 0.27% | 47 | 0 | 7978.72 | 0.27% |
| 8000 | 48 | Yes | 11 | 1 | 8152.17 | 1.90% | 11 | 1 | 8152.17 | 1.90% |

### 15.4.4. Operation

#### Operation modes

The operation mode is selected by the I2SOPMOD 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 following table.

**Table 15-5. 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.

#### Status flags and interrupts

There are six status flags implemented in the SPI_STAT register, including TBE, RBNE, TRANS, I2SCH, TXURERR, and RXORERR. The user can use them to fully monitor the state of the I2S bus.

- Transmit buffer empty flag (TBE)

  This bit is set when the transmit buffer is empty. An interrupt may be generated if the TBEIE bit in the SPI_CTL1 register is set. The software can write the next data to the transmit buffer by writing it to the SPI_DATA register. The TBE bit is cleared by a write operation to the SPI_DATA register.

- Receive buffer not empty flag (RBNE)

  This bit is set when receive buffer is not empty. An interrupt may be generated if the RBNEIE bit in the SPI_CTL1 register is set. It indicates valid data has been received and stored in the receive buffer. The software can read the data by reading the SPI_DATA register. The RBNE bit is cleared by a read operation to the SPI_DATA register.

- Transmitting On-Going flag (TRANS)

  This TRANS flag is set and cleared by hardware. It indicates the state of the communication layer of the I2S. The TRANS flag is useful to detect the end of a transfer if the software wants to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected. The TRANS flag is set when a transfer starts, except in master reception mode where the flag is kept low during reception. It is cleared when the I2S is disenabled or a transfer complete. When communication is not continuous, the TRANS flag is low between each communication. When communication is continuous, the TRANS flag is kept high during all the transfers in master transmission mode, and goes low for one I2S clock cycle between each transfer in slave mode.

- I2S channel side flag (I2SCH)

  In transmission mode, this flag is refreshed at the moment when the TBE flag goes high, indicating the channel side to which the data to transfer belongs. In reception mode, this flag is refreshed at the moment when the RBNE flag goes high, indicating the channel side to which the received data belongs. Notice that in case of error (TXURERR or RXORERR) this flag becomes not reliable and I2S needs to be switched off and switched on before resuming the communication. Besides, this flag has no meaning in the PCM standard.

- Transmission underrun error flag (TXURERR)

  This flag is set when the first clock for data transmission appears while the transmit buffer is still empty in slave transmission mode. An interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. This flag is cleared by a read operation to the SPI_STAT register.

- Reception overrun error flag (RXORERR)

  This flag is set when data are received and the previous data has not been read from the SPI_DATA register yet in reception mode. An interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In the case, the contents in receive buffer are not updated with the newly received data. A read operation to the SPI_DATA register returns the previous correctly received data. All other subsequently received half-words are lost. This flag is cleared by a read access to the SPI_DATA register followed by a read access to the SPI_STAT register.

I2S interrupt events and corresponding enable bits are summed up in the following table.

**Table 15-6. I2S interrupt**

| Interrupt event | Flag | Enable bit |
|---|---|---|
| Transmit buffer empty | TBE | TBEIE |
| Receive buffer not empty | RBNE | RBNEIE |
| Transmission underrun error | TXURERR | ERRIE |
| Reception overrun error | RXORERR | ERRIE |

### Initialization sequence

I2S initialization sequence contains the five steps shown below. In order to initialize I2S working in master mode, all the five steps should be done. In order to initialize I2S working in slave mode, only step 2, step 3, step 4 and step 5 should be done.

- Step 1: Configure the DIV[7:0] bits, the OF bit, and the MCKOEN bit in the SPI_I2SPSC register, in order to define the I2S bitrate and whether I2S_MCK needs to be provided or not.

- Step 2: Configure the CKPL in the SPI_I2SCTL register, in order to define the idle state clock polarity.
- Step 3: Configure the I2SSEL bit, the I2SSTD[1:0] bits, the PCMSMOD bit, the I2SOPMOD[1:0] bits, the DTLEN[1:0] bits, and the CHLEN bit in the SPI_I2SCTL register, in order to define the I2S feature.
- Step 4: Configure the TBEIE bit, the RBNEIE bit, the ERRIE bit, the DMATEN bit, and the DMAREN bit in the SPI_CTL1 register, in order to select the potential interrupt sources and the DMA capabilities. This step is optional.
- Step 5: Set the I2SEN bit in the SPI_I2SCTL register to enable I2S.

### Master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates the transmit buffer is empty, and may generate an interrupt if the TBEIE bit in the SPI_CTL1 register is set. At the beginning, the transmit 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 transmit 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 transmit buffer is loaded into the shift register, and the TBE flag goes back high. To ensure a continuous audio data transmission, it is mandatory to write the SPI_DATA register with the next data to transmit before the end of the current transmission.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the data to transfer belongs. 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 switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

### Master reception sequence

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the receive buffer is not empty, and may generate an interrupt 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 receive buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the receive 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.

Or, 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 mandatory to switch off and switch on I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the received data belongs. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

In order to switch off I2S, specific actions are required to ensure that I2S completes the transfer cycle properly without initiating a new data transfer. The actions depend on the audio standard selected, and on the configuration of the data length and the channel length. The actions for each case are described below.

■ 16-bit data packed in 32-bit frame in the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD = 10)

1. Wait for the second to last RBNE
2. Then wait 17 I2S clock cycles
3. Clear the I2SEN bit

■ 16-bit data packed in 32-bit frame in the audio standards except the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD is not equal to 10)

1. Wait for the last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

■ For all other cases

1. Wait for the second to last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

### Slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The difference between them is 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. To ensure a continuous audio data transmission, it is mandatory to write the SPI_DATA register with the next data to transmit before the end of the current transmission. Or, 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 switch off and switch on I2S before resuming the communication. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

### Slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The difference between them is 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 requests the transfer of data. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

### 15.4.5. DMA features

DMA is working in exactly the same way as for the SPI mode. The only difference is that the CRC feature is not available in I2S mode.

## 15.5. SPI registers

### 15.5.1. Control register 0 (SPI_CTL0)

Address offset: 0x00
Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BDEN | BDOEN | CRCEN | CRCNT | FF16 | RO | SWNSSEN | 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 |
|---|---|---|
| 15 | BDEN | Bidirectional Enable<br>0: 2 line unidirectional transmit mode<br>1: 1 line bidirectional transmit mode. The information transfer between the MOSI pin in master and the MISO pin in 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: CRC calculation is disabled<br>1: CRC calculation is enabled. |
| 12 | CRCNT | CRC Transfer Next<br>0: Next transfer is Data<br>1: Next transfer is CRC value (TCR)<br>When the transfers are 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<br>0: 8-bit data frame format<br>1: 16-bit data frame format |
| 10 | RO | Receive only<br>When BDEN is cleared, this bit determines the direction of transfer.<br>0: Full-duplex<br>1: Receive-only |
| 9 | SWNSSEN | NSS Software Mode Selection<br>0: NSS hardware mode. The NSS pin input depends on IO.<br>1: NSS software mode. The NSS pin input depends on SWNSS bit. |
| 8 | SWNSS | NSS Pin Selection In NSS Software Mode<br>0: NSS pin is pull low<br>1: NSS pin is pull high<br>This bit has an effect only when the SWNSSEN bit is set. |
| 7 | LF | LSB First Mode<br>0: Transmit MSB first<br>1: Transmit LSB first |
| 6 | SPIEN | SPI Enable<br>0: SPI peripheral is disabled<br>1: SPI peripheral is enabled |
| 5:3 | PSC[2:0] | Master Clock Prescaler Selection<br>000: PCLK/2 100: PCLK/32<br>001: PCLK/4 101: PCLK/64<br>010: PCLK/8 110: PCLK/128<br>011: PCLK/16 111: PCLK/256 |

| | | |
|---|---|---|
| | | PCLK means PCLK2 when using SPI0 or PCLK1 when using SPI1 |
| 2 | MSTMOD | Master Selection<br>0: Slave mode<br>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 |

## 15.5.2. Control register 1 (SPI_CTL1)

Address offset: 0x04
Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TBEIE | RBNEIE | ERRIE | Reserved. | | NSSDRV | DMATEN | DMAREN |
| | | | | | | | | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | TBEIE | Transmit Buffer Empty Interrupt Enable<br>0: TBE interrupt is disenabled.<br>1: TBE interrupt is enabled. An interrupt is generated when the TBE bit is set |
| 6 | RBNEIE | Receive Buffer Not Empty Interrupt Enable<br>0: RBNE interrupt is disenabled.<br>1: RBNE interrupt is enabled. An interrupt is generated when the RBNE bit is set |
| 5 | ERRIE | Errors Interrupt Enable.<br>0: Error interrupt is disabled.<br>1: Error interrupt is enabled. An interrupt is generated when the CRCERR bit or the CONFERR bit or the RXORERR bit or the TXURERR bit is set. |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | NSSDRV | Drive NSS Output<br>0: NSS output is disabled.<br>1: NSS output is enabled. If the NSS pin is configured as output, the NSS pin is pulled low in master mode when SPI is enabled.<br>If the NSS pin is configured as input, the NSS pin should be pulled high in master mode, and |

this bit has on effect.

1 DMATEN Transmit Buffer DMA Enable

0: Transmit buffer DMA is disabled

1: Transmit buffer DMA is enabled, when the TBE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.

0 DMAREN Receive Buffer DMA Enable

0: Receive buffer DMA is disabled

1: Receive buffer DMA is enabled, when the RBNE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.

### 15.5.3. Status register (SPI_STAT)

Address offset: 0x08
Reset value: 0x0002

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TRANS | RXORERR | CONFERR | CRCERR | TXURERR | I2SCH | TBE | RBNE |
| | | | | | | | | r | r | r | rc_w0 | r | r | r | r |

**Bits** **Fields** **Descriptions**

15:8 Reserved Must be kept at reset value.

7 TRANS Transmitting On-going Bit

0: SPI or I2S is idle.

1: SPI or I2S is currently transmitting and/or receiving a frame, or the transmit buffer is not empty.

This bit is set and cleared by hardware.

6 RXORERR Reception Overrun Error Bit

0: No reception overrun error occurred.

1: Reception overrun error occurred.

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

0: No configuration fault occurred

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.)

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.

This bit is not used in I2S mode.

4 CRCERR SPI CRC Error Bit

0: The SPI_RCR value is equals to the received CRC data at last.

1: The SPI_RCR value is not equals to the received CRC data at last.

This bit is set by hardware and cleared by software writing 0.

| | | |
|---|---|---|
| | | This bit is not used in I2S mode. |
| 3 | TXURERR | Transmission underrun error bit<br>0: No transmission underrun error occurred.<br>1: Transmission underrun error occurred.<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: Channel Left has to be transmitted or has been received.<br>1: Channel Right has to be transmitted or has been received.<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 Empty<br>0: Transmit buffer is not empty<br>1: Transmit buffer is empty |
| 0 | RBNE | Receive Buffer Not Empty<br>0: Receive buffer is empty<br>1: Receive buffer is not empty |

### 15.5.4. Data register (SPI_DATA)

Address offset: 0x0C
Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | SPI_DATA[15:0] | Data transfer register. The hardware has two buffers, including transmit buffer and receive buffer. Write data to SPI_DATA will save the data to transmit buffer and read data from SPI_DATA will get the data from receive buffer.<br>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, transmit buffer and receive 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, transmit buffer and receive buffer are 16-bit. |

### 15.5.5. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x10
Reset value: 0x0007

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

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

| |
|---|
| CPR [15:0] |
| rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CPR[15:0] | This register contains the CRC polynomial and used for CRC calculation. The default value is 0007h. |

## 15.5.6. Receive CRC register (SPI_RCRC)

Address offset: 0x14
Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RCR[15:0] | When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and save them in RCR register. If the Data frame format is set to 8-bit data, CRC calculation is done based on CRC8 standard, and save the value in RCR[7:0], when the Data frame format is set to 16-bit data, CRC calculation is done based on CRC16 standard, and save the value in RCR[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 or the SPIEN bit of SPI_CTL0 is cleared. |

## 15.5.7. Transmit CRC register (SPI_TCRC)

Address offset: 0x18
Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | TCR[15:0] | When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and save them in TCR register. If the Data frame format is set to 8-bit data, CRC calculation is done based on CRC8 standard, and save the value in TCR[7:0], when the Data frame format is set to 16-bit data, CRC calculation is done based on CRC16 standard, and save the value in TCR[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 format (LF bit of |

the SPI_CTL0) will get different CRC value.

This register is reset when the CRCEN bit or the SPIEN bit of SPI_CTL0 is cleared.

### 15.5.8. I2S control register (SPI_I2SCTL)

Address offset: 0x1C
Reset value: 0x0000

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

| 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 |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value |
| 11 | I2SSEL | I2S mode selection<br>0: SPI mode<br>1: I2S mode<br>This bit should be configured when SPI or I2S is disenabled. |
| 10 | I2SEN | I2S enable<br>0: I2S is disenable<br>1: I2S is enable<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 is disenabled.<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 is disenabled.<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 Phillips standard<br>01: MSB justified standard |

| | | |
|---|---|---|
| | | 10: LSB justified standard<br>11: PCM standard<br>These bits should be configured when I2S is disenabled.<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 is disenabled.<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 is disenabled.<br>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 is disenabled.<br>This bit is not used in SPI mode. |

### 15.5.9. I2S prescaler register (SPI_I2SPSC)

Address offset: 0x20
Reset value: 0x0002

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | MCK<br>OEN | OF | DIV[7:0] | | | | | | | |
| | | | | | | rw | rw | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | Reserved | Must be kept at reset value |
| 9 | MCKOEN | I2S_MCK output enable<br>0: I2S_MCK output is disenable<br>1: I2S_MCK output is enable<br>This bit should be configured when I2S is disenabled.<br>This bit is not used in SPI mode. |
| 8 | OF | Odd factor for the prescaler |

| | | |
|---|---|---|
| | | 0: Real divider value is DIV * 2<br>1: Real divider value is DIV * 2 + 1<br>This bit should be configured when I2S is disenabled.<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 is disenabled.<br>These bits are not used in SPI mode. |

## 15.5.10. Quad wire control register (SPI_QCTL) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | IO23_DRV | QRD | QMOD |
| | | | | | | | | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:3 | Reserved | Must be kept at reset value |
| 2 | IO23_DRV | Drive IO2 and IO3 enable<br>0: IO2 and IO3 are not driven in single wire mode<br>1: IO2 and IO3 are driven to high in single wire mode<br>This bit is only available in SPI1. |
| 1 | QRD | Quad wire read select.<br>0: SPI is in quad wire write mode<br>1: SPI is in quad wire read mode<br>This bit should be only be configured when SPI is not busy (TRANS bit cleared)<br>This bit is only available in SPI1. |
| 0 | QMOD | Quad wire mode enable.<br>0: SPI is in single wire mode<br>1: SPI is in quad wire mode<br>This bit should only be configured when SPI is not busy (TRANS bit cleared).<br>This bit is only available in SPI1. |

# 16. Comparator (CMP)

## 16.1. Introduction

The general purpose comparators, CMP0 and CMP1, can work either standalone (all terminal are available on I/Os) or together with the timers. They can be used for a variety of functions including wakeup from low-power mode when triggered by an analog signal, analog signal conditioning and cycle-by-cycle current control loop when combined with the DAC and a PWM output from a timer.

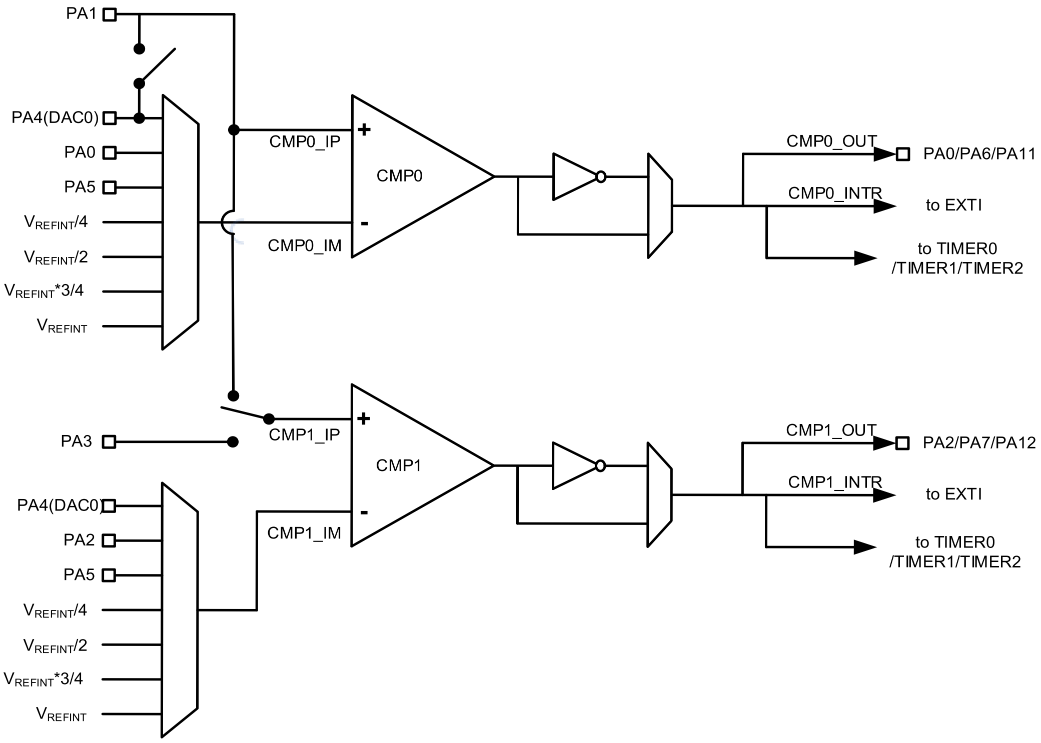
## 16.2. Main features

- Rail-to-rail comparators
- Configurable hysteresis
- Configurable speed and consumption
- Each comparator has configurable analog input source
  - DAC
  - 3 I/O pins
  - The whole or sub-multiple values of internal reference voltage
- Window comparator
- Outputs to I/O
- Outputs to timers for triggering
- Outputs to EXTI

## 16.3. Function description

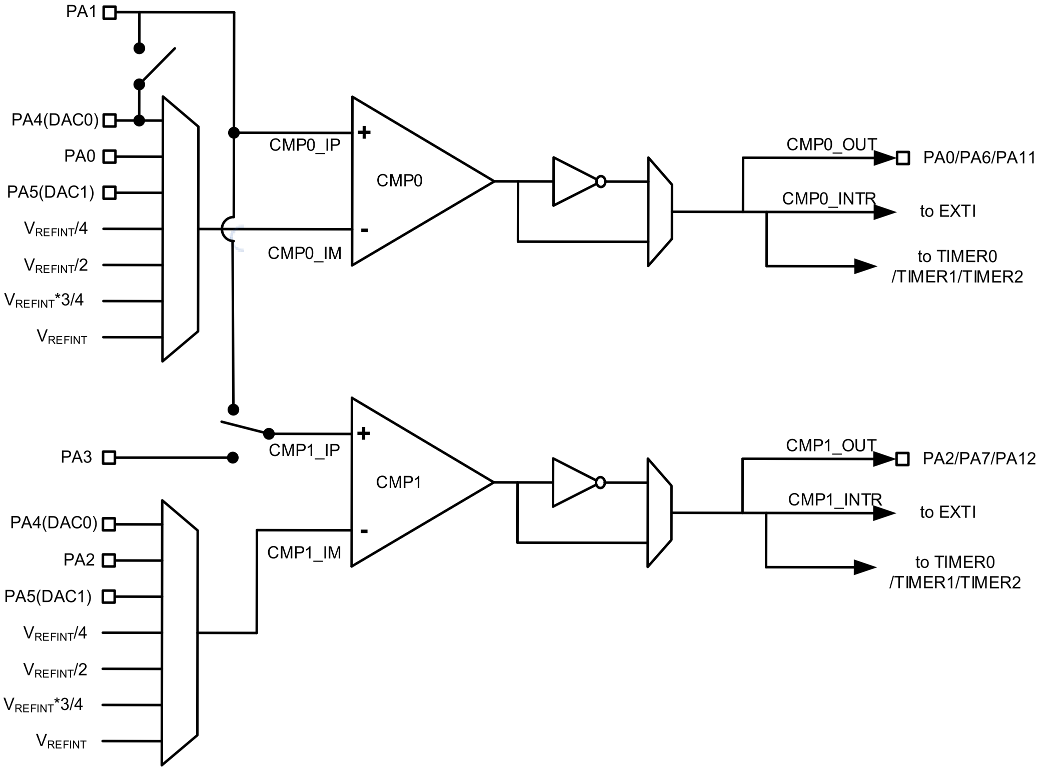
The block diagrams of CMP are shown below.

Figure 16-1. CMP block diagram of GD32F130xx and GD32F150xx devices



**Note**: $V_{REFINT}$ is 1.2V.

Figure 16-2. CMP block diagram of GD32F170xx and GD32F190xx devices



**Note**: $V_{REFINT}$ is 1.2V.

### 16.3.1. CMP clock and reset

The CMP clock provided by the clock controller is synchronous with the PCLK. The CMP share common reset and clock enable bits with SYSCFG.

### 16.3.2. CMP inputs and outputs

The I/Os must be configured in analog mode in the GPIOs registers before they are selected as CMPs inputs.

Considering pin definitions in Datasheet, the CMP output must be connected to corresponding alternate I/Os.

A variety of timer inputs can be internally connected to the CMP output to realize the following functions:

- Input capture for timing measures
- Emergency shut-down of PWM signals, using BKIN
- Cycle-by-cycle current control, using OCPRE_CLR inputs

In order to work even in Deep-sleep mode, the polarity selection logic and the output redirection to the port work independently from PCLK.

It is possible to have the CMP output simultaneously redirected internally and externally.

The CMP outputs are internally connected to the extended interrupts and events controller. Each CMP has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from power saving modes.

### 16.3.3. CMP power mode

For a given application, the CMP power consumption versus propagation delay can be adjusted to have the optimum trade-off by configuring bits CMPxM [1:0] in CMP_CS register. The CMP works fastest with highest power consumption when CMPxM = 2’b00, while works slowest with lowest power consumption when CMPxM = 2’b11.

### 16.3.4. CMP hysteresis

In order to avoid spurious output transitions that caused by the noise signal, the CMP includes a programmable hysteresis to force the hysteresis value using external components. This function can be shut down when you don't need it (for example, exiting from the power saving mode).

### 16.3.5. CMP register write protection

Out of security concerns for applications, such as over-current or thermal protection, and having specific functional safety requirements, it is a need to insure that the CMP's configuration cannot be changed in case of spurious register access or program counter corruption.

For this consideration, the CMP control and status register (CMP_CS) can be entered into the write protect state by setting CMPxLK bit to 1, which should be done once the programming is completed. The whole CMP_CS register will become read-only, including the CMPxLK bit. Only the MCU reset can reset the CMPxLK bit.

## 16.4. CMP registers

### 16.4.1. Control/status register (CMP_CS)

#### For GD32F130xx and GD32F150xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|
| CMP1LK | CMP1O | CMP1HST[1:0] | CMP1PL | CMP1OSEL[2:0] | WNDEN | CMP1MSEL[2:0] | CMP1M | Reserved | CMP1EN |
| rwo | r | rw/r | rw/r | rw/r | rw/r | rw/r | rw/r | | rw/r |

| 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] | CMP0S | CMP0EN |
| rwo | r | rw/r | rw/r | rw/r | | rw/r | rw/r | rw/r | rw/r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CMP1LK | CMP1 lock<br>This bit allows to have all control bits of CMP1 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software.<br>0: CMP_CS[31:16] bits are read-write<br>1: CMP_CS[31:16] bits are read-only |
| 30 | CMP1O | CMP1 output<br>This is a copy of CMP1 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 |
| 29:28 | CMP1HST[1:0] | CMP1 hysteresis<br>These bits are used to control the hysteresis level.<br>00: No hysteresis |

| | | |
|---|---|---|
| | | 01: Low hysteresis<br>10: Medium hysteresis<br>11: High hysteresis |
| 27 | CMP1PL | Polarity of CMP1 output<br>This bit is used to select the CMP1 output.<br>0: Output is not inverted<br>1: Output is inverted |
| 26:24 | CMP1OSEL[2:0] | CMP1 output selection<br>These bits are used to select the destination of the CMP1 output.<br>000: No selection<br>001: TIMER 0 break input<br>010: TIMER 0 channel0 Input capture<br>011: TIMER 0 OCPRE_CLR input<br>100: TIMER1 channel3 input capture<br>101: TIMER1 OCPRE_CLR input<br>110: TIMER2 channel0 input capture<br>111: TIMER2 OCPRE_CLR input |
| 23 | WNDEN | Window mode enable<br>This bit is used to disconnect the CMP1_IP input of CMP1 from PA3 and connect it to CMP0's CMP0_IP input.<br>0: CMP1_IP is connected to PA3<br>1: CMP1_IP is connected to CMP0_IP |
| 22:20 | CMP1MSEL[2:0] | CMP1_M input selection<br>These bits are used to select the source connected to the CMP1_M input of the CMP1.<br>000: $V_{REFINT}$/4<br>001: $V_{REFINT}$ /2<br>010: $V_{REFINT}$ *3/4<br>011: $V_{REFINT}$<br>100: PA4 (DAC0)<br>101: PA5<br>110: PA2<br>111: Reserved |
| 19:18 | CMP1M[1:0] | CMP1 mode<br>These bits are used to control the operating mode of the CMP1 adjust the speed/consumption.<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 |

| | | |
|---|---|---|
| 17 | Reserved | Must be kept at reset value |
| 16 | CMP1EN | CMP1 enable<br>0: CMP1 disabled<br>1: CMP1 enabled |
| 15 | CMP0LK | CMP0 lock<br>This bit allows to have all control bits of CMP0 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software.<br>0: CMP_CS[15:0] bits are read-write<br>1: CMP_CS[15:0] bits are read-only |
| 14 | CMP0O | CMP0 output<br>This 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 CMP0 output.<br>0 : Output is not inverted<br>1 : Output is inverted |
| 10:8 | CMP0OSEL[2:0] | Comparator 0 output selection<br>These bits are used to select the destination of the CMP0 output.<br>000: no selection<br>001: TIMER 0 break input<br>010: TIMER 0 channel0 Input capture<br>011: TIMER 0 OCPRE_CLR input<br>100: TIMER1 channel3 input capture<br>101: TIMER1 OCPRE_CLR input<br>110: TIMER2 channel0 input capture<br>111: TIMER2 OCPRE_CLR input |
| 7 | Reserved | Must be kept at reset value |
| 6:4 | CMP0MSEL[2:0] | CMP0_M input selection<br>These bits are used to select the source connected to the CMP0_M input of the CMP0.<br>000: $V_{REFINT}$ /4<br>001: $V_{REFINT}$ /2<br>010: $V_{REFINT}$ *3/4 |

| | | |
|---|---|---|
| | | 011: $V_{REFINT}$<br>100: PA4 (DAC0)<br>101: PA5<br>110: PA0<br>111: Reserved |
| 3:2 | CMP0M[1:0] | CMP0 mode<br>These bits are used to control the operating mode of the CMP0 adjust the speed/consumption.<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 | CMP0S | CMP0 switch<br>This bit is used to closes a switch between CMP0 non-inverting input on PA0 and PA4 (DAC0) I/O.<br>0: Switch open<br>1: Switch closed |
| 0 | CMP0EN | CMP0 enable<br>0: CMP0 disabled<br>1: CMP0 enabled |

### For GD32F170xx and GD32F190xx devices

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 |
|---|---|---|---|---|---|---|---|---|---|
| CMP1LK | CMP1O | CMP1HST[1:0] | CMP1PL | CMP1OSEL[2:0] | WNDEN | CMP1MSEL[2:0] | CMP1M | Reserved | CMP1EN |
| rwo | r | rw/r | rw/r | rw/r | rw/r | rw/r | rw/r | | rw/r |
| 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] | CMP0S | CMP0EN |
| rwo | r | rw/r | rw/r | rw/r | | rw/r | rw/r | rw/r | rw/r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CMP1LK | CMP1 lock<br>This bit allows to have all control bits of CMP1 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software.<br>0: CMP_CS[31:16] bits are read-write<br>1: CMP_CS[31:16] bits are read-only |

| | | |
|---|---|---|
| 30 | CMP1O | CMP1 output<br>This is a copy of CMP1 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 |
| 29:28 | CMP1HST[1:0] | CMP1 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 |
| 27 | CMP1PL | Polarity of CMP1 output<br>This bit is used to select the CMP1 output.<br>0: Output is not inverted<br>1: Output is inverted |
| 26:24 | CMP1OSEL[2:0] | CMP1 output selection<br>These bits are used to select the destination of the CMP1 output.<br>000: No selection<br>001: TIMER 0 break input<br>010: TIMER 0 channel0 Input capture<br>011: TIMER 0 OCPRE_CLR input<br>100: TIMER1 channel3 input capture<br>101: TIMER1 OCPRE_CLR input<br>110: TIMER2 channel0 input capture<br>111: TIMER2 OCPRE_CLR input |
| 23 | WNDEN | Window mode enable<br>This bit is used to disconnect the CMP1_IP input of CMP1 from PA3 and connect it to CMP0’s CMP0_IP input.<br>0: CMP1_IP is connected to PA3<br>1: CMP1_IP is connected to CMP0_IP |
| 22:20 | CMP1MSEL[2:0] | CMP1_M input selection<br>These bits are used to select the source connected to the CMP1_M input of the CMP1.<br>000: $V_{REFINT}$/4<br>001: $V_{REFINT}$ /2<br>010: $V_{REFINT}$ *3/4<br>011: $V_{REFINT}$<br>100: PA4 (DAC0)<br>101: PA5 (DAC1)<br>110: PA2<br>111: Reserved |
| 19:18 | CMP1M[1:0] | CMP1 mode |

| | | |
|---|---|---|
| | | These bits are used to control the operating mode of the CMP1 adjust the speed/consumption.<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 |
| 17 | Reserved | Must be kept at reset value |
| 16 | CMP1EN | CMP1 enable<br>0: CMP1 disabled<br>1: CMP1 enabled |
| 15 | CMP0LK | CMP0 lock<br>This bit allows to have all control bits of CMP0 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software.<br>0: CMP_CS[15:0] bits are read-write<br>1: CMP_CS[15:0] bits are read-only |
| 14 | CMP0O | CMP0 output<br>This 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 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.<br>000: no selection<br>001: TIMER 0 break input<br>010: TIMER 0 channel0 Input capture<br>011: TIMER 0 OCPRE_CLR input<br>100: TIMER1 channel3 input capture<br>101: TIMER1 OCPRE_CLR input<br>110: TIMER2 channel0 input capture<br>111: TIMER2 OCPRE_CLR input |

| | | |
|---|---|---|
| 7 | Reserved | Must be kept at reset value |
| 6:4 | CMP0MSEL[2:0] | CMP0_M input selection<br>These bits are used to select the source connected to the CMP0_M 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 (DAC0)<br>101: PA5 (DAC1)<br>110: PA0<br>111: Reserved |
| 3:2 | CMP0M[1:0] | CMP0 mode<br>These bits are used to control the operating mode of the CMP0 adjust the speed/consumption.<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 | CMP0S | CMP0 switch<br>This bit is used to closes a switch between CMP0 non-inverting input on PA0 and PA4 (DAC0) I/O.<br>0: Switch open<br>1: Switch closed |
| 0 | CMP0EN | CMP0 enable<br>0: CMP0 disabled<br>1: CMP0 enabled |

# 17. Universal synchronous asynchronous receiver transmitter (USART)

## 17.1. Introduction

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 periperial clock (PCLK1 or PCLK2) to produce a dedicated baudrate clock for the USART transmitter and receiver.

It supports half-duplex single wire synchronous communication, LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specification. It also supports modem operations (CTS/RTS) and multiprocessor communication.

The USART also supports DMA function for high speed data communication.

## 17.2. Main features

- Full duplex, asynchronous communications
- Half duplex single wire communications
- NRZ standard format (Mark/Space)
- 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
  - 1, 1.5 or 2 stop bit generation
- Swappable Tx/Rx pin
- Configurable data polarity
- Auto baud rate detection

- Hardware Modem operations (CTS/RTS) and RS485 drive enable
- Configurable multibuffer communication using centralized DMA
- Separate enable bits for Transmitter and Receiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Error detection: Overrun, Noise, Frame and Parity error
- LIN Break generation and detection
- IrDA Support
- Synchronous mode and transmitter clock output for synchronous transmission
- ISO 7816-3 (T=0 and T=1) compliant smartcard interface
- Multiprocessor communication
  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle line or address mark detection
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition
- Wake up from Deep-sleep mode
  - By standard RXNE interrupt
  - By WUF interrupt
- 14 interrupt sources with flags:
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete

- Receive data register full
- Idle line detected
- Overrun error
- Framing error
- Noise error
- Parity error
- Address/character match
- Receiver timeout interrupt
- End of block interrupt
- Wakeup from Deep-sleep mode

While USART0 is fully implemented, USART1 is only partially implemented with the following features not supported.

- Auto baud rate detection
- Smartcard mode
- IrDA SIR ENDEC block
- LIN mode
- Dual clock domain and wakeup from Deep-sleep mode
- Receiver timeout interrupt
- Modbus communication

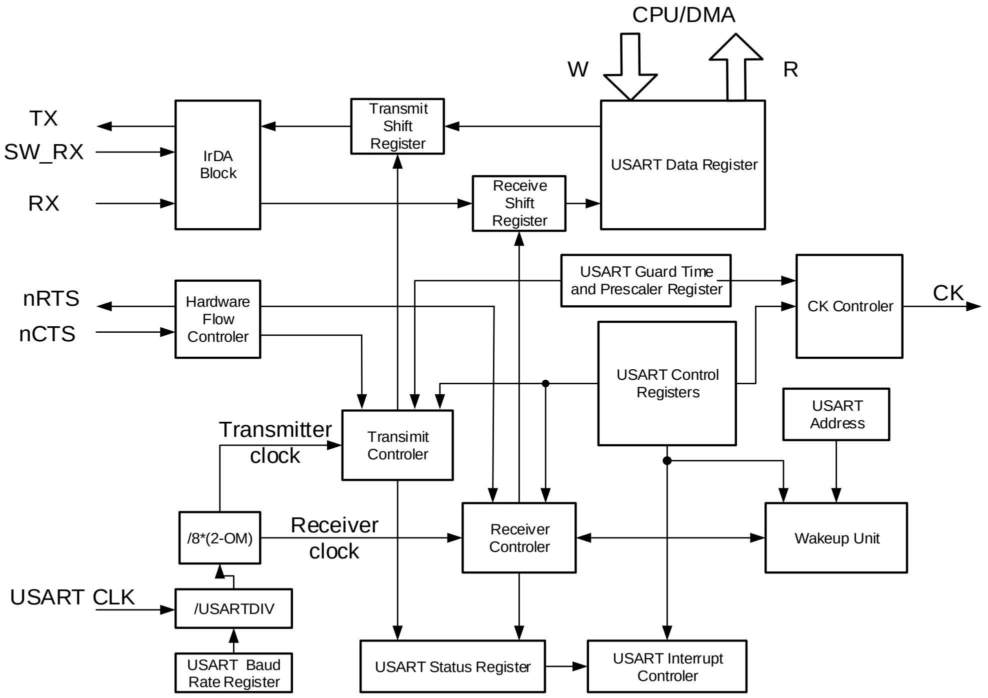
## 17.3. Function description

The interface is externally connected to another device by the main pins listed as following.

**Table 17-1. USART important pins description**

| 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 17-1. USART module block diagram**



## 17.3.1. USART transmitter

When the transmit enable bit (TEN) in USART_CTL0 register is set, the transmitter clock pulses are generated by the baud rate generator, and output on the CK pin. Then the serial bit stream is sent after an idle frame by the transmitter according to the programmed configuration in the control registers.

In case of transmission corruption, the TEN bit should not be disabled when transmission is ongoing.

If the data can be written to the USART_TDATA without overwriting the previous one, the TBE bit is asserted. And it is cleared when the data is written.

If a frame is transmitted and the TBE bit is asserted, the TC bit will be set. An interrupt is generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

Refer to the following procedure for the USART transmission:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the stop bits length in USART_CTL1.
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 in the USART_TDATA register.

9. Wait until TC=1 to finish.

It is necessary to wait for the TC bit asserted before disabling the USART or entering the power saving mode.

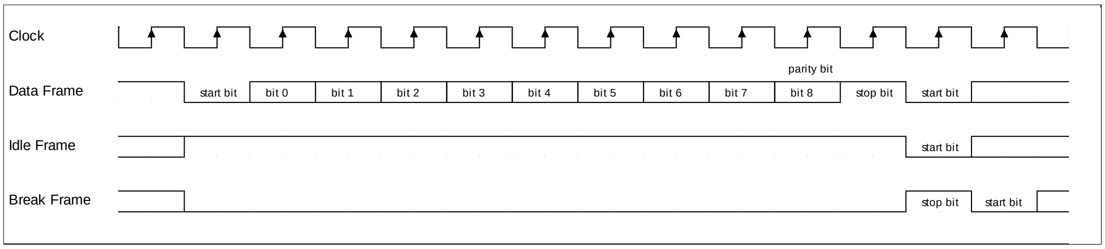
Reading the USART_STAT then writing the USART_TDATA can clear the TC bit. And writing '0' directly to TC bit can also clear the TC bit for multibuffer communication

The break frame is sent when the SBKCMD bit is set, and SBKCMD bit is reset after the transmission.

**Table 17-2. Stop bits configuration**

| Stop bit length (bit) | Description |
|---|---|
| 1 | default value |
| 1.5 | Smartcard mode |
| 2 | normal USART, single-wire and modem modes |

**Figure 17-2. USART character frame (9 bits data and 1 stop bit)**



The MSB bit is taken as the parity bit if parity control is enabled.

If there is even number of '1s' in the data bits (the LSB bits), the parity bit should be '0' (even parity) or '1' (odd parity).

### 17.3.2. USART receiver

The receiver receives a bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed.

If a frame is received and the RBNE bit is asserted, the USART_RDATA and associated bits in USART_STAT can be read. An interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register.

The RBNE bit must be cleared before the next data arrived, or an overrun error will occur.

In case of reception corruption, the REN bit should not be disabled when reception is ongoing.

The RBNE bit can be cleared by directly reading the USART_RDATA in multibuffer communication. DMA read in multibuffer communication can also clear the RBNE bit.

Refer to the following procedure for the USART receiving:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the stop bits length in USART_CTL1.
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 REN bit in USART_CTL0.

### 17.3.3. Reception errors

An overrun error occurs, if the next data arrived or the previous DMA request has not been serviced when the RBNE bit is set. The ORERR bit in the USART_STAT register is set.

The internal baud-rate reference clock is used to over sample RX line for data recovery. If a noise error occurs, the NERR in USART_STAT is set at the rising edge of the RBNE bit and the invalid data is received from the shift register.

A framing error occurs when the stop bit is not detected at the expected time. If a framing error occurs, the FERR in USART_STAT is set at the rising edge of the RBNE bit and the invalid data is received from the shift register.

The three error flag bits can be reset by writting the OREC, NEC and FEC bits in USART_INTC register respectively.

### 17.3.4. Baud rate generation

The baud-rate divisor is a 16-bit number consisting 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 to the system clock:

In case of oversampling by 16, the equation is:

$$\mathrm{USARTDIV} = \frac{\mathrm{UCLK}}{16 \times \mathrm{Baud\ Rate}}$$

In case of oversampling by 8, the equation is:

$$\mathrm{USARTDIV} = \frac{\mathrm{UCLK}}{8 \times \mathrm{Baud\ Rate}}$$

The choice of the USART clock (UCLK) is done through the Clock Control system (see the Reset and clock uint(RCU) section). The clock source must be chosen before enabling the

USART (by setting the UEN bit).

### 17.3.5. Auto baudrate detection

The USART is able to detect and automatically set the USART_BAUD register value based on the reception of one character. There are two methods which can be chosen through the ABDM bits in the USART_CTL1 register. These methods are:

1. The USART will measure the duration of the start bit (falling edge to rising edge). In this case the receiving pattern should be any character starting with a bit at 1.
2. The USART will measure the duration of the start and of the 1st data bit. The measure is done falling edge to falling edge, ensuring a better accuracy in the case of slow signal slopes. In this case, the receiving pattern should be any character starting with 10xx bits.

### 17.3.6. Multi-processor communication

In multiprocessor communication, the mute mode for the unaddressed receivers is introduced. During a reception, the target receiver is active to receive the full message contents while the unintended message recipients are in the mute mode. Idle line detection and address mark detection can be used to enter or exit the mute mode.

In idle line detection, the USART enters the mute mode and the RWU bit in the USART_STAT register is set when the MMCMD bit in USART_CMD is asserted. It exits as soon as the idle frame is detected. The RWU bit in the USART_STAT register is cleared when the IDLEF bit in USART_STAT is not set.

In address mark detection, the bytes with their MSB bit set are recognized as address byte. The 4/7 LSB bits in the address byte are the address of the target receiver. The receivers compare the address byte with the address of their own, then enter or exit the mute mode (set or reset the RWU bit) according to the result.

### 17.3.7. 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 addrress 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.

### 17.3.8. LIN mode

The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN bit in USART_CTL1 and the STPL, SCEN, HDEN, IREN bits in USART_CTL2 should be reset in LIN mode.

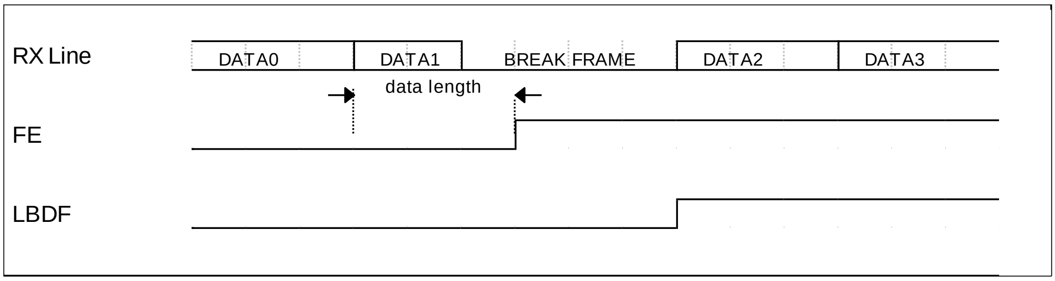
The LIN transmission procedure is almost the same as the normal transmission procedure. The data bits length can only be 8. And the break frame is 13-bit '0s'.

Break detection is totally independent from the normal USART receiver. So a break can be detected during the idle state or during a frame.

When the receiver is enabled and a start bit has been detected, the circuit samples the next bit.

During the break detection, when the framing error occurs, the break detection cancels only until the RX line becomes high level. Then start bit detection begins. If 10/11 (configured by the LBLEN bit in USART_CTL1) consecutive bits are detected as '0' before a delimiter character (high level), the LIN break detection flag is also set in USART_STAT.

**Figure 17-3. Frame error detection and break frame detection in LIN mode**



### 17.3.9. 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 reset in half-duplex communication mode.

Only one wire is used in half-duplex mode. The TX and RX pins are connected together internally. The TX pin should be configured as IO pin. The conflicts should be controlled by the software. When the TEN bit is set, the data in the data register will be sent.

### 17.3.10. 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 reset in synchronous mode. The CK pin is the synchronous USART transmitter clock output, and can be only activated when the TEN bit is enabled. No clock pulse will be sent to the CK pin during the start bit and stop bit transmission. The CLEN bit in USART_CTL1 can be used to determine whether the clock is output or not during the LSB (address index) 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 Mode idle state.

These 3 bits (CPL, CPH, CLEN) should not be changed while the transmitter or the receiver is enabled

The clock is synchronized with the data transmitted. The receiver in synchronous mode samples the data on the transmitter clock without any oversampling.

**Figure 17-4. Example of USART in synchronous mode**

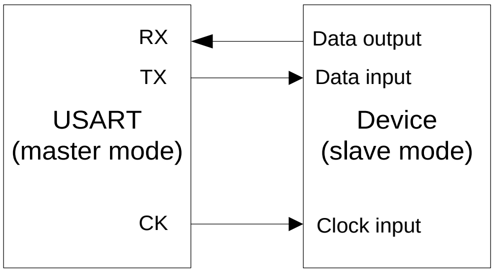
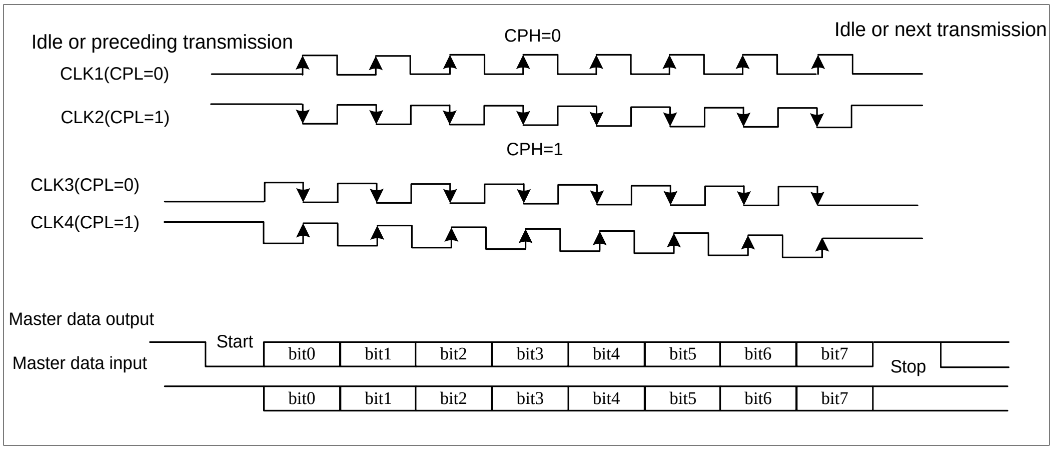


**Figure 17-5. 8-bit format USART synchronous waveform (CLEN=1)**



### 17.3.11. Smartcard (ISO7816) mode

The smartcard mode is an asynchronous mode, which 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.

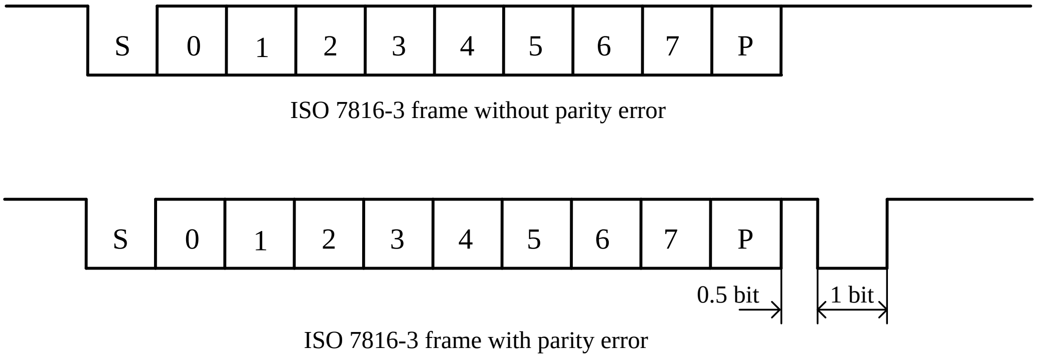
A clock is provided to the smartcard if the CKEN bit is set. The clock can be divided for other use.

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 and drives a bidirectional line that is also driven by the smartcard.

### T=0 mode

**Figure 17-6. ISO7816-3 frame format**


S
0
1
2
3
4
5
6
7
P
ISO 7816-3 frame without parity error
S
0
1
2
3
4
5
6
7
P
0.5 bit
1 bit
ISO 7816-3 frame with parity error

Comparing to the time in normal operation, the transmission time from transmit shift register to the TX pin is delayed half baud clock, and the TC flag assertion time delayed a certain value wrote in the guard time register. The USART can automaticly re-send data according to the protocol by SCRTNUM times. At the end of reception of the last repeated character the TC bit is set without gardtime immediately. The USART will stop transmitting and signal the error as a framing error if it continues receiving the NKEN after the programmed number of retries. The TXFCMD bit in the USART_CMD register can be used to clear the TBE bit.

During USART reception, the TX line is pulled low for a baud clock after finishing receiving the frame if a parity error is detected. This signal is the ‘NKEN’ signal to smartcard. Then a frame error occurred 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 NKEN and signals the error as a parity error if the received character is still erroneous after the maximum number of retries specified in the SCARNUM bit field.

The ‘NKEN’ signal will be sent to the USART if the NKEN bit in USART_CTL2 is set. And the USART will not take the ‘NKEN’ signal as the start bit.

The idle frame and break frame do not apply for the smartcard mode.

### T=1 mode (block mode)

In T=1 (block) mode, the NKEN bit in the USART_CTL2 register should be cleared to deactivate the parity error transmission.

When requesting a read from the smartcard, the 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 received byte.

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 all the characters received. The length of the block, which must be programmed to the BL field in the USART_RT register, is communicated by 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 H 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 an L state on the signal line and parity is even. In this case, the following control bits must be programmed: MSBF=1, DINV=1.

## 17.3.12. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART_CTL2. The LMEN, STPL, CKEN bits in USART_CTL1 and HDEN, SCEN bits in USART_CTL2 should be reset in IrDA mode.

In IrDA SIR physical layer, an infrared light pulse (a Return to Zero signal) represent the logic ‘0’. The pulse width should be 3/16 of a bit period. The IrDA could not detect the pulse if the pulse width is less than 1 PSC clock. While it can detect some pulse by chance if the pulse width is greater than 1 but smaller than 2 times PSC clock.

The USART data frame is modulated in SIR Transmit encoder. The modulated signal is transmitted by the infrared LED. The baud rate should not be larger than 115200 for the encoder.

The infrared detector receives the modulated signal and outputs the data frame decoded. The polarity of modulated signal transmitted by the encoder is opposite to that received by the decoder. Then the decoder input is usually the high level and a start bit is decoded if the input signal is low.

The transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

For power saving mode:

- Transmitter: The pulse width can be 3 times the low-power baud rate.
- Receiver: The same as normal mode.

**Figure 17-7. IrDA SIR ENDEC module**

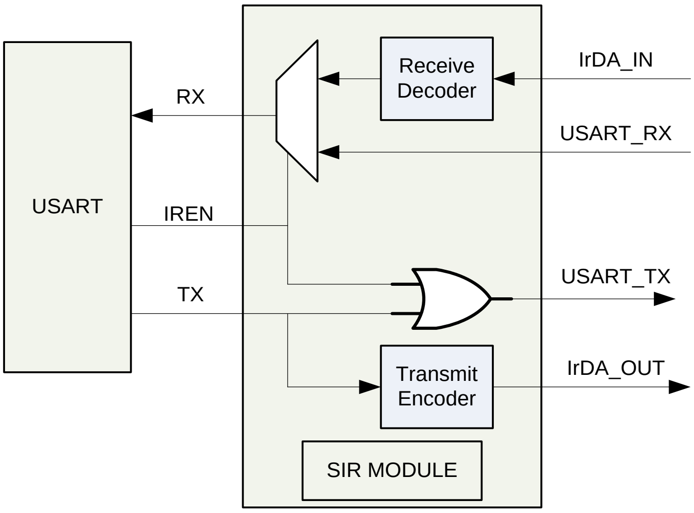
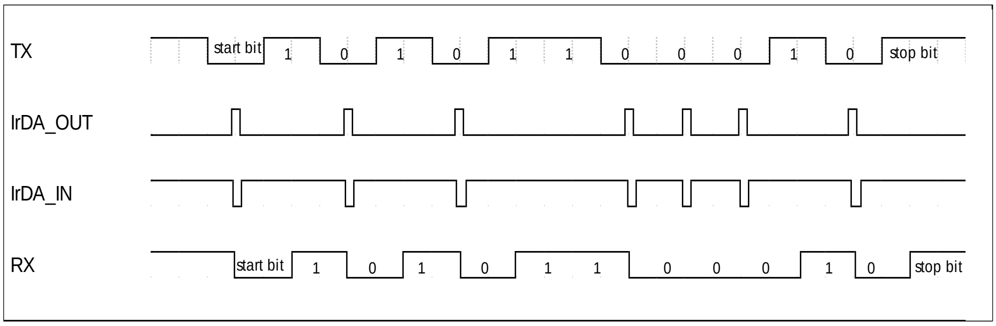


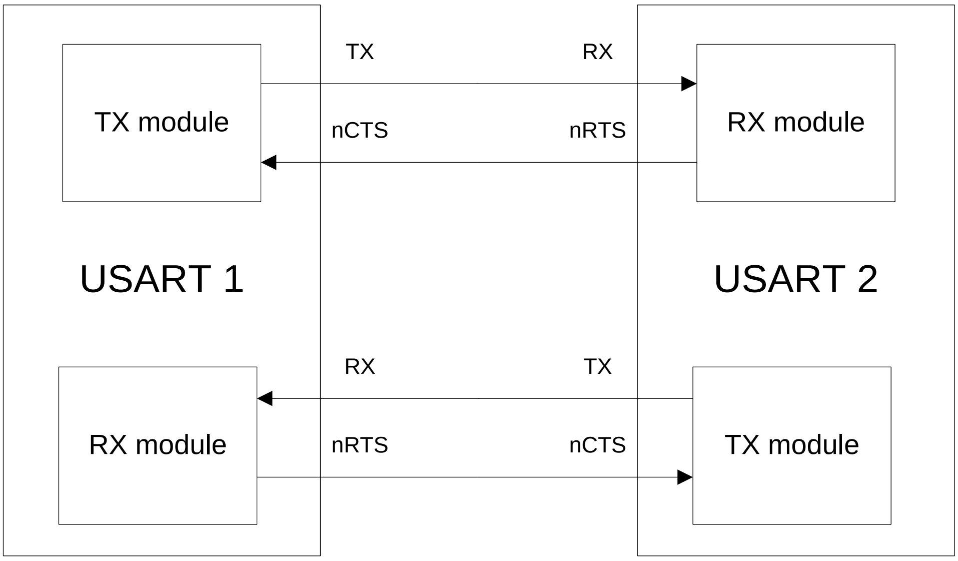
**Figure 17-8. IrDA data modulation**



## 17.3.13. Hardware flow control

Using the nCTS input and the nRTS output to control the serial data flow is called hardware flow control. The RTS flow control is enabled by writing '1' to the RTSEN bit in USART_CTL2 and the CTS flow control is enabled by write '1' to the CTSEN bit in USART_CTL2.

**Figure 17-9. Hardware flow control between two USARTs**



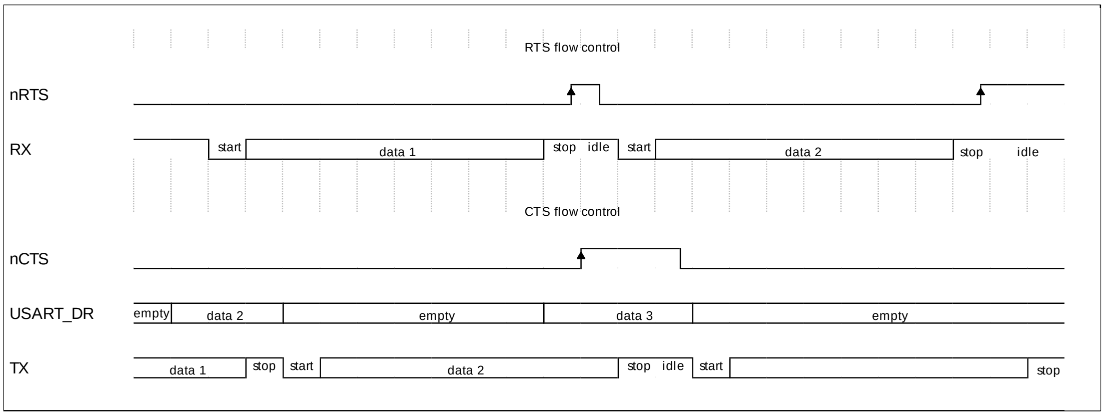
### RTS flow control

USART receiver can receive data only when the nRTS signal is low, and the signal does not go high until the data frame reception is finished. The next reception occurs when the nRTS signal goes low again. The signal keeps high when the receive register is full.

### CTS flow control

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.

**Figure 17-10. 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.

## 17.3.14. DMA requests

DMA can be used for USART continuously communication. 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.

DMA transmission configuration:

1. Configuring the DMA registers, which the destination address (USART_TDATA register address), the source address (memory address), the total number of bytes to be transferred, channel priority, DMA interrupts are written to.
2. Writing ‘0’ to the TCC bit in USART_INTC.
3. Writing ‘1’ to the DENT bit in the DMA control register to enable the DMA channel.

The TC flag in USART_STAT is set later than the time when the FTFIF flag is set in DMA_INTF. It remains reset during the data transfers, and is set when the USART communication finished. Entering the Deep-sleep mode or disabling the USART will lead to the transfer corruption when the TC bit is not set.

DMA reception configuration: configuring the DMA registers, which the destination address (memory address), the source address (USART_RDATA register address), the total number of bytes to be transferred, channel priority, DMA interrupts are written to.

The RBNE event occurs as soon as the data is received.

### 17.3.15. 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 soure 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.

### 17.3.16. USART interrupts

The USART interrupt events and flags are listed in the table below.

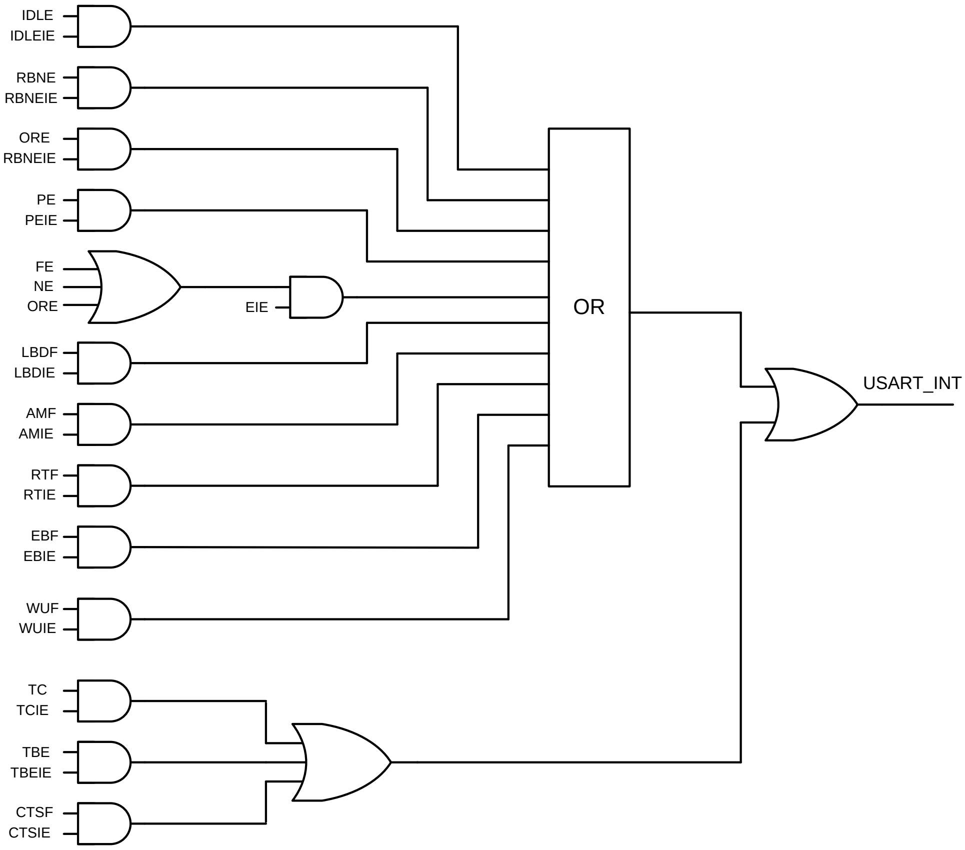
**Table 17-3. USART interrupt requests**

| 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 | |
| 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 | EIE |
| Character match | AMF | AMIE |
| Receiver timeout error | RTF | RTIE |

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| 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 17-11. USART interrupt mapping diagram**



## 17.4. USART registers

### 17.4.1. 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: End of Block interrupt is disabled<br>1: End of Block interrupt is enabled<br>This bit is reserved in USART1. |
| 26 | RTIE | Receiver timeout interrupt enable<br>0: Receiver timeout interrupt is disabled<br>1: Receiver timeout interrupt is enabled<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 deassertion 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<br>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: ADDR match interrupt is disabled<br>1: ADDR match interrupt is enabled |
| 13 | MEN | Mute mode enable<br>0: Mute mode disabled<br>1: Mute mode enabled |
| 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 Mark<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 10 | PCEN | Parity control enable<br>0: Parity control disabled<br>1: Parity control enabled<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: Parity error interrupt is disabled<br>1: An interrupt will occur whenever the PERR bit is set in USART_STAT. |
| 7 | TBEIE | Transmitter register empty interrupt enable<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the TBE bit is set in USART_STAT |
| 6 | TCIE | Transmission complete interrupt enable<br>0: Transmission complete interrupt is disabled<br>1: An interrupt will occur whenever the TC bit is set in USART_STAT. |
| 5 | RBNEIE | Read data buffer not empty interrupt and overrun error interrupt enable<br>0: Read data register not empty interrupt and overrun error interrupt disabled<br>1: 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: IDLE line detected interrupt disabled<br>1: An interrupt will occur whenever the IDLEF bit is set in USART_STAT. |
| 3 | TEN | Transmitter enable<br>0: Transmitter is disabled<br>1: Transmitter is enabled |
| 2 | REN | Receiver enable<br>0: Receiver is disabled<br>1: Receiver is enabled 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. |

| | | |
|---|---|---|
| | | This bit is reserved in USART1. |
| 0 | UEN | USART enable<br>0: USART prescaler and outputs disabled<br>1: USART prescaler and outputs enabled |

## 17.4.2. 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 | ABDM[1:0] | | ABDEN | MSBF | DINV | TINV | RINV |
| rw | | | | | | | | rw | 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 mark 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 ADD[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: Receiver timeout function disabled<br>1: Receiver timeout function enabled<br>This bit is reserved in USART1. |
| 22:21 | ABDM[1:0] | Auto baud rate mode<br>00: Falling edge to rising edge measurement (measurement of the start bit)<br>01: Falling edge to falling edge measurement (the received frame must be in a Start 10xxxxxx frame format)<br>10: Reserved.<br>11: Reserved<br>This bit field cannot be written when the USART is enabled (UEN=1). |

| | | |
|---|---|---|
| | | This bit is reserved in USART1. |
| 20 | ABDEN | Auto baud rate enable<br>0: Auto baud rate detection is disabled<br>1: Auto baud rate detection is enabled<br>This bit is reserved in USART1. |
| 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<br>01: Reserved<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: CK pin disabled<br>1: CK pin enabled |

| | | |
|---|---|---|
| | | This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 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: LIN break detection interrupt is disabled<br>1: An interrupt will occur whenever 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 (ADD[5:0], ADD[6:0] and ADD[7:0]) respectively<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 3:0 | Reserved | Must be kept at reset value |

### 17.4.3. 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: Wakeup from Deep-sleep mode interrupt is disabled<br>1: Wakeup from Deep-sleep mode interrupt is enabled<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 WU (Wakeup from Deep-sleep mode flag) in the USART_STAT register.<br>00: WU active on address match, which is defined by ADDR and ADDM<br>01: Reserved<br>10: WU active on Start bit<br>11: WU 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 transmit and receive.<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.<br>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: DE function is disabled<br>1: DE function is enabled<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: DMA is disabled following a reception error. The DMA request is not asserted 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: Overrun functionality is enabled. 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: Overrun functionality is disabled. 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: CTS interrupt is disabled<br>1: An interrupt will occur whenever the CTS bit is set in USART_STAT |
| 9 | CTSEN | CTS enable<br>0: CTS hardware flow control disabled<br>1: CTS hardware flow control enabled<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 8 | RTSEN | RTS enable<br>0: RTS hardware flow control disabled<br>1: RTS hardware flow control enabled, 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 |

| | | |
|---|---|---|
| | | 0: DMA mode is disabled for transmission<br>1: DMA mode is enabled for transmission |
| 6 | DENR | DMA enable for reception<br>0: DMA mode is disabled for reception<br>1: DMA mode is enabled for reception |
| 5 | SCEN | Smartcard mode enable<br>0: Smartcard Mode disabled<br>1: Smartcard Mode enabled<br>This bit field cannot be written when the USART is enabled (UEN=1).<br>This bit is reserved in USART1. |
| 4 | NKEN | NKEN enable in Smartcard mode<br>0: Disable NKEN transmission when parity error<br>1: Enable NKEN 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: Half duplex mode is disabled<br>1: Half duplex mode is enabled<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: IrDA disabled<br>1: IrDA enabled<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<br>0: Error interrupt disabled<br>1: An interrupt will occur whenever the FERR bit or the ORERR bit or the NERR bit is set in USART_STAT in multibuffer communication |

### 17.4.4. Baud rate generator register (USART_BAUD)

Address offset: 0x0C
Reset value: 0x0000_0000

This register has to be accessed by word (32-bit)
This register cannot be written when the USART is enabled (UEN=1)

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:4 | INTDIV [11:0] | Integer part of baud-rate divider<br>DIV_INT[11:0] = INTDIV [11:0] |
| 3:0 | FRADIV [3:0] | Fraction part of baud-rate divider<br>If OVSMOD = 0, USARTDIV [3:0] = FRADIV [3:0];<br>If OVSMOD = 1, USARTDIV [3:1] = FRADIV [2:0], FRADIV [3] must be reset. |

### 17.4.5. Prescaler and guard time configuration register (USART_GP)

Address offset: 0x10

Reset value: 0x0000_0000

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

This register cannot be written when the USART is enabled (UEN=1)

This register is reserved in USART1

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GUAT[7:0] | | | | | | | | PSC[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | GUAT[7:0] | Guard time value in smartcard mode |
| 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>... |

**In IrDA normal mode**.

00000001: can be set this value only

**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.

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

### 17.4.6. Receiver timeout register (USART_RT)

Address offset: 0x14

Reset value: 0x0000_0000

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

This bit is reserved in USART1

| 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 EOBCF 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. 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.

### 17.4.7. 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 | ABDCMD |
| | | | | | | | | | | | w | 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 make the USART send a BREAK frame, as soon as the transmit machine is idle. |
| 0 | ABDCMD | Auto baudrate detection command<br>Writing 1 to this bit issues an automatic baud rate measurement command on the next received data frame and resets the ABDF flag in the USART_STAT.<br>This bit is reserved in USART1 |

### 17.4.8. 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABDF | ABDE | 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 | 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 WUS 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<br>It is cleared/set by hardware when a wakeup/mute sequence (address or IDLE) is recognized, which is selected by the WAKE 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 IDLE mode is selected. |
| 18 | SBF | Send break flag<br>0: No break character is to be transmitted<br>1: Break character will be transmitted<br>This bit indicates that a send break character was requested. |

| | | |
|---|---|---|
| | | Set by software, by writing 1 to the SBKCMD bit in the USART_CMD register.<br>Clear by hardware during the stop bit of break transmission. |
| 17 | AMF | ADDR match flag<br>0: ADDR do not match the received character<br>1: ADDR matches the received character, An interrupt is generated if AMIE=1in the USART_CTL0 register.<br>Set by hardware, when the character defined by ADDR [7:0] is received.<br>Cleared 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 | ABDF | Auto baudrate detection flag<br>0: No auto baudrate detection complete<br>1: Auto baudrate detection complete<br>Set by hardware when the automatic baud rate has been completed.<br>Cleared by writing 1 to the ABDCMD in the USART_CMD register, to request a new auto baudrate detection.<br>This bit is reserved in USART1. |
| 14 | ABDE | Auto baudrate detection error<br>0: No auto baudrate detection error occured<br>1: Auto baudrate detection error occured<br>Set by hardware if the baud rate out of range or character comparison failed<br>Cleared by software, by writing 1 to the ABDRQ bit in the USART_CTL2 register.<br>This bit is reserved in USART1 |
| 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. |

| | | |
|---|---|---|
| | | 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<br>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_TDATA 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_TDATA 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. |

### 17.4.9. 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 AM bit in the USART_STAT register. |
| 16:13 | Reserved | Must be kept at reset value |
| 12 | EBC | End of timeout clear<br>Writing 1 to this bit clears the EB 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 RT 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<br>Writing 1 to this bit clears the NERR bit in the USART_STAT register. |
| 1 | FEC | Frame error flagclear<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. |

### 17.4.10. Receive data register (USART_RDATA)

Offset: 0x24

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 | | | | | | | | | | | | | | | |

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

| 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 (PERR bit set to 1 in the USART_CTL0 register). |

### 17.4.11. Transmit data register (USART_TDATA)

Offset: 0x28

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 | | | | | | | | | | | | | | | |

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

rw

| 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 (PERR bit set to 1 in the USART_CTL0 register).<br>This register must be written only when TBE bit in USART_STAT reigister is set. |

# 18. Debug (DBG)

## 18.1. Introduction

The DBG module helps debugger to debug power saving mode, TIMER, I2C, RTC, WWDGT, FWDGT and CAN. When corresponding bit set, provide clock when in power saving mode or hold the state for TIMER, WWDGT, FWDGT, RTC, I2C or CAN. (The contents of CAN is only for GD32F170xx and GD32F190xx devices)

## 18.2. Function description

### 18.2.1. Debug support for power saving mode

When STB_HOLD bit in DBG control register 0 (DBG_CTL0) 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 exitting the standby mode, a system reset generated.

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

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

When the core halted and the corresponding bit in DBG control register 0 (DBG_CTL0) or DBG control register 1 (DBG_CTLR1) is set, the following behaved will be occurred

For TIMER, the timer counters stopped and hold for debug.

For I2C, SMBUS timeout hold for debug.

For WWDGT or FWDGT, the counter clock stopped for debug.

For RTC, the counter stopped for debug.

For CAN, the receive register stopped counting for debug.

# 18.3. DBG registers

## 18.3.1. ID code register (DBG_ID)

Address: 0xE004 2000

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 be read by software, These bits are unchanged constant. |

## 18.3.2. Control register 0(DBG_CTL0)

### For GD32F130xx and GD32F150xx devices

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 | I2C2_HOLD | I2C1_HOLD |
| | | | | rw | | | | | | | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2C0_HOLD | Reserved | | TIMER2_HOLD | TIMER1_HOLD | TIMER0_HOLD | WWDGT_HOLD | FWDGT_HOLD | Reserved | | | | | STB_HOLD | DSLP_HOLD | SLP_HOLD |
| rw | | | rw | rw | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27 | TIMER13_HOLD | TIMER13 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER13 counter for debug when core halted |
| 26:20 | Reserved | Must be kept at reset value |
| 19 | TIMER5_HOLD | TIMER5 hold register<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No effect<br>1: Hold the TIMER5 counter for debug when core halted |
| 18 | Reserved | Must be kept at reset value |
| 17 | I2C2_HOLD | I2C2 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C2 SMBUS timeout for debug when core halted |
| 16 | I2C1_HOLD | I2C1 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C1 SMBUS timeout for debug when core halted |
| 15 | I2C0_HOLD | I2C0 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C0 SMBUS timeout for debug when core halted |
| 14:13 | Reserved | Must be kept at reset value |
| 12 | TIMER2_HOLD | TIMER2 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER2 counter for debug when core halted |
| 11 | TIMER1_HOLD | TIMER1 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER1 counter for debug when core halted |
| 10 | TIMER0_HOLD | TIMER0 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER0 counter for debug when core halted |
| 9 | WWDGT_HOLD | WWDGT hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the WWDGT counter clock for debug when core halted |
| 8 | FWDGT_HOLD | FWDGT hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the FWDGT counter clock for debug when core halted |
| 2 | STB_HOLD | Standby mode hold register<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No effect<br>1: At the standby mode, the system clock and HCLK are provided by CK_IRC8M, a system reset generated when exiting standby mode |
| 1 | DSLP_HOLD | Deep-sleep mode hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: At the Deep-sleep mode, the system clock and HCLK are provided by CK_IRC8M |
| 0 | SLP_HOLD | Sleep mode hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: At the sleep mode, the HCLK is on |

### For GD32F170xx and GD32F190xx devices

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 | | | | | CAN1_HOLD | Reserved | TIMER5_HOLD | Reserved | I2C2_HOLD | I2C1 HOLD |
| | | | | rw | | | | | | rw | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2C0_HOLD | CAN0_HOLD | Reserved | TIMER2_HOLD | TIMER1_HOLD | TIMER0_HOLD | WWDGT_HOLD | FWDGT_HOLD | Reserved | | | | | STB_HOLD | DSLP_HOLD | SLP_HOLD |
| rw | rw | | rw | rw | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value |
| 27 | TIMER13_HOLD | TIMER13 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER13 counter for debug when core halted |
| 26:22 | Reserved | Must be kept at reset value |
| 21 | CAN1_HOLD | CAN1 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the CAN1 for debug when core halted |
| 20 | Reserved | Must be kept at reset value |
| 19 | TIMER5_HOLD | TIMER5 hold register<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No effect<br>1: Hold the TIMER5 counter for debug when core halted |
| 18 | Reserved | Must be kept at reset value |
| 17 | I2C2_HOLD | I2C2 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C2 SMBUS timeout for debug when core halted |
| 16 | I2C1_HOLD | I2C1 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C1 SMBUS timeout for debug when core halted |
| 15 | I2C0_HOLD | I2C0 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the I2C0 SMBUS timeout for debug when core halted |
| 14 | CAN0_HOLD | CAN0 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the CAN0 for debug when core halted |
| 13 | Reserved | Must be kept at reset value |
| 12 | TIMER2_HOLD | TIMER2 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER2 counter for debug when core halted |
| 11 | TIMER1_HOLD | TIMER1 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER1 counter for debug when core halted |
| 10 | TIMER0_HOLD | TIMER0 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER0 counter for debug when core halted |
| 9 | WWDGT_HOLD | WWDGT hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the WWDGT counter clock for debug when core halted |
| 8 | FWDGT_HOLD | FWDGT hold register<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No effect<br>1: Hold the FWDGT counter clock for debug when core halted |
| 2 | STB_HOLD | Standby mode hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: At the standby mode, the system clock and HCLK are provided by CK_IRC8M, a system reset generated when exit stanby mode |
| 1 | DSLP_HOLD | Deep-sleep mode hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: At the Deep-sleep mode, the system clock and HCLK are provided by CK_IRC8M |
| 0 | SLP_HOLD | Sleep mode hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: At the sleep mode, the HCLK is on |

### 18.3.3. 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 | TIMER16 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER16 counter for debug when core halted |
| 17 | TIMER15_HOLD | TIMER15 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER15 counter for debug when core halted |

| | | |
|---|---|---|
| 16 | TIMER14_HOLD | TIMER14 hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the TIMER14 counter for debug when core halted |
| 15:11 | Reserved | Must be kept at reset value |
| 10 | RTC_HOLD | RTC hold register<br>This bit is set and reset by software.<br>0: No effect<br>1: Hold the RTC counter for debug when core halted |
| 9:0 | Reserved | Must be kept at reset value |

# 19. Universal serial bus full-speed device interface (USBD)

This chapter applies to GD32F150xx devices, mainly describes the USB full-speed device controller and its interface. As this module is different from other serial ports, so the higher complexity of it requires developers to know about its background knowledge. If developers want to get its latest information, please refer to the official specification of USB 2.0 version which is published by the USB Implements Forum (USB-IF).

## 19.1. Introduction

The Universal Serial Bus (USB) is a four-wire bus, which are the power line VBus, the data line D+ (DP), the data line D- (DM) and the ground line GND. It supports communication between a host and the function device implemented by the USB controller. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. It communicates data with device in polling method. The USB bus supports hot plugging and dynamic configuration of the devices. All interactive processes are initiated by the host.

Every transmission between the USB host and the device is accomplished by transaction. Transactions can be classified into three types: Setup transaction, Data-IN transaction and Data-out transaction. Transaction consists of three kinds of data packets: token packet, data packet and optional handshake packet.

The USB device interface implements the communication between a full-speed USB 2.0 bus and the APB1 bus. According to USB standard request, it can deal with all USB packets: such as detecting token packets, handling data packet transfer and processing handshake packets. The format of transaction is performed by the hardware, including CRC generation and checking. According to the USB protocol, each device can have a maximum of 16 logical or 32 physical endpoints. The USB controller supports up to 8 bidirectional endpoints or 16 mono-directional endpoints.

Each USB endpoint has a corresponding buffer located in a dedicated 512 byte SRAM. The data transfer between host PC and the system memory is based on the data buffer. The data transceiver of endpoint in its own buffer must comply with the condition that the state of endpoint is valid. For isochronous transfer which is need to be in stable rate, and bulk transfer with high throughput, the USB controller provides special handling and implements a double buffer algorithm, which allows the isochronous endpoint or bulk endpoint to transmit and receive data continuously without waiting for the state of endpoint to become valid, since that buffers of these endpoints are always available, one is handling data, meanwhile, MCU can use the other one.

The USB module can be placed in low-power mode (SUSPEND mode) by writing in the Control register (USBD_CTL) whenever required. At this time, all static power dissipation is avoided and the USB clock can be slowed down or stopped. It will be resumed when detect activity at the USB bus.

## 19.2. Main features

- USB 2.0 full-speed device controller
- Support USB 2.0 Link Power Management (LPM)
- Support up to 8 configurable endpoints
- Support double-buffered bulk/isochronous endpoints
- Each endpoint supports control, bulk, isochronous or interrupt transfer types(exclude endpoint 0)
- Support USB suspend/resume operations
- A dedicated 512-byte SRAM used for data packet buffer
- Integrated USB PHY

### 19.2.1. Implementation

Table 19-1 describes the USB implementation in GD32F150xx devices.

**Table 19-1. GD32F150xx USB implementation**

| USB features | GD32F150xx USB |
|---|---|
| Number of endpoints | 8 bidirectional / 16 mono-directional endpoints |
| Size of dedicated data packet buffer(SRAM) | 512 bytes |
| Dedicated data packet buffer access scheme | 2×16bits / word |
| USB 2.0 Link Power Management(LPM)support | YES |

### 19.2.2. Clock

According to the USB standard definition, the USB full-speed module adopt fixed 48MHz clock. For using the USB module, it is need to open the two clock, one is the USB controller clock, its frequency must be configured to 48MHz, and the other one is the APB1 USB interface clock, that is APB1 bus clock, its frequency can be above or below 48MHz.

**Note:** in order to meet the system requirements of packet buffer interface and USB data transfer rate, the frequency of the APB1 bus clock must be greater than 24MHz, so as to avoid data buffer overflow and underflow.

48MHz clock of USB controller can be generated by MCU internal or external crystal oscillator, through PLL frequency multiplication:

- Regard two frequency division of 8MHz internal oscillator as the input of the PLL, then 12 frequencies doubling of the clock

- Regard 8MHz external oscillator as the input of the PLL, firstly frequency doubling, then adopt USB Frequency divider to divide frequency

When the USB clock is generated by external crystal, only four USB frequency division coefficients are 1, 1.5, 2 and 2.5 (useless, MCU frequency cannot reach 120 MHz).Thus, for obtaining 48MHz clock, PLL frequencies doubling can only be 48MHz, 72MHz and 96MHz.

**Note:** Regardless of internal or external crystal oscillator generate USB clock, the clock accuracy must reach ±500ppm. If the accuracy of the USB clock cannot meet the condition, data transfer may not conform to the requirements of the USB specification, and even it may lead USB cannot work directly.

### 19.2.3. Pin description

The USB controller in device mode uses 3 pins, DP, DM and VBus. DP and DM are data pins, which are mainly used to send and receive data and fixed on the controller. Before enabling USB, these pins default to ordinary GPIO after reset, after enabling USB, they will become USB pins. As bus power supply USB device, the VBus pin is mainly used to supply power for the controller, while it is not used in that USB controller is self powered device.

The pins DP and DM don’t need configure because they are connected to the USB internal transceiver automatically as soon as the USB is enabled.

The signal pinouts are shown in the table below:

**Table 19-2. GD32F150xx USB signal pinouts**

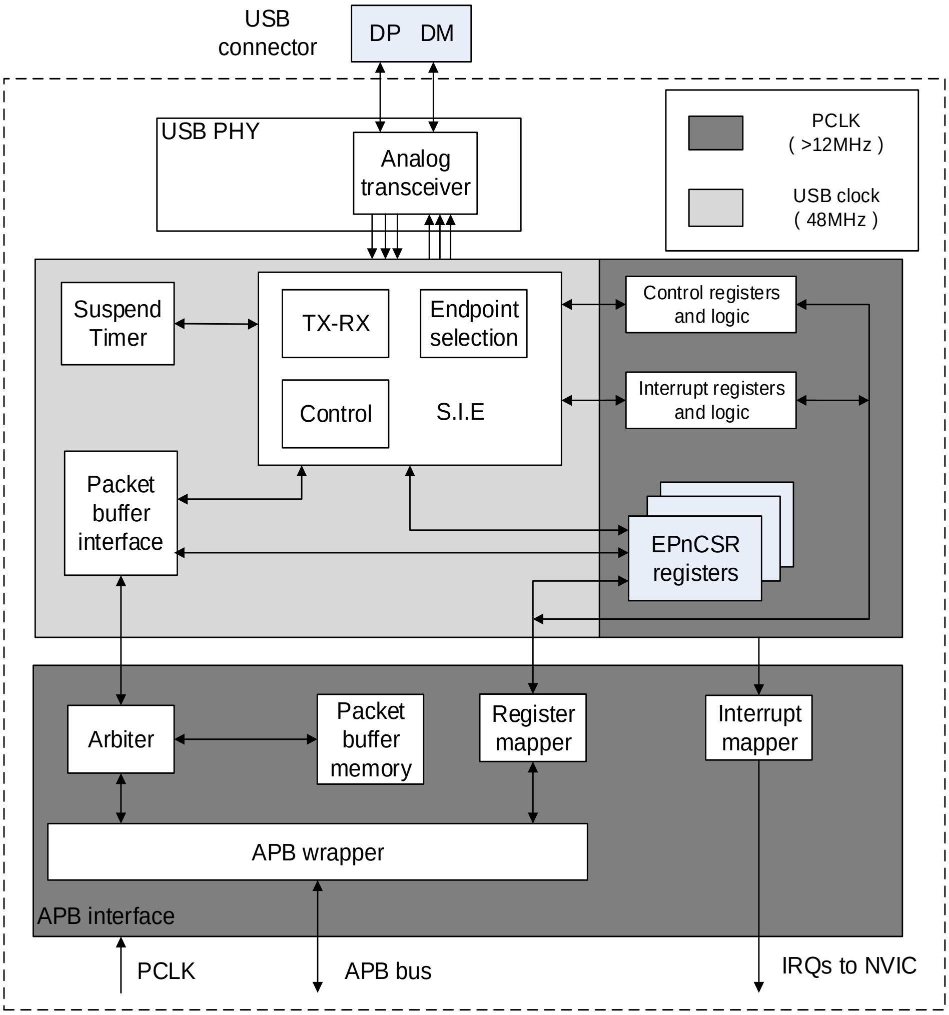
| USB pinout | GPIO pin | GPIO configuration |
|---|---|---|
| USBDP | PA12 | As soon as the USB is enabled, these pins are connected to the USB internal transceiver automatically. |
| USBDM | PA11 | |

## 19.3. Function description

### 19.3.1. Basic functional block

Figure 19-1,shows the block diagram of the USB peripheral

**Figure 19-1. USB peripheral block diagram**



## Controller block

The USB controller includes the following blocks:

- Built-in analog transceiver (ATX): The USB ATX connects to the USB pins DP and DM to send/receive the bi-directional signals of the USB bus.
- Serial interface engine (SIE): The SIE implements integrated full-speed USB protocol layer. In order to ensure the speed, it is implemented by hardware without software intervention. SIE is in charge of data transfer between endpoint buffers and USB bus. The module features include: decodes and encodes the serial data, corrects error, bit stuffing and relief stuffing, isochronous pattern recognition, CRC verifying and generating, PID verifying and generating, parallel and serial conversion, address recognition, and evaluation and generation of handshake signals.
- Timer: The timer generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no activity has occurred on the USB bus for 3ms.

- Packet buffer interface: this module mainly manages data packet buffers for endpoints transmission and reception. It could select proper buffer for endpoints to meet SIE demand and locate them in the right memory addresses according to endpoint register data. It progressive increase address with each byte exchanging until the end of data packet, as well as to follow up on the number of exchanged bytes for preventing buffer overrun.
- Endpoint control and status registers: Each endpoint has an associated register containing its controlling information and current status. For mono-directional/single-buffer endpoints, a single endpoint register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or 7 double-buffer endpoints in any combination. For example, the USB peripheral can be programmed to have two double buffer endpoints and 12 mono-directional/single-buffer endpoints.
- Control registers: These registers contain information about the status and controlling of the whole USB device.
- Interrupt flag registers: These registers contain the interrupt flag and records of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear a pending interrupt.

### APB1 interface block

The USB APB1 interface includes the following blocks:

- Data Packet buffer: The block is dedicated 512 bytes SRAM. There is reserved memory for each implemented USB endpoint. The size of total buffer depends on the number of endpoints, the packet maximum length of endpoint and whether to support double buffering. This module is used by the data packet buffer interface and the corresponding data structure is created. Application can directly access the buffer. It is composed of 256 16-bits words. However, the access mode of MCU is 32-bits, so the actual occupancy memory size is 1024 bytes.
- Arbiter: Since the USB module is connected to the APB1 bus through the APB1 interface, the APB1 bus has a conflict with the USB interface when they all have memory access request. The arbiter is used to resolve the conflict. It will give the APB1 bus higher priority, and always keep half of the memory bandwidth for USB complete transmission. Through this time division multiplexing strategy, the virtual dual port SRAM is realized, which allows the application to access memory at the same time of the USB transmission. At the same time, it allows any length of multi byte APB1 transmission in the transmission process.
- Register Mapper: This module collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bits wide word set addressed by the APB1.
- APB1 Wrapper: This module provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.

- Interrupt Mapper: This module is used to select interrupts which are generated by the possible USB events and map them to different lines of the NVIC.

### 19.3.2. Buffer

USB buffer transfers data between MCU core and SIE. It is implemented by a special virtual dual port SRAM memory, and it is mapped to the APB1 peripheral memory, from 0x4000 6000 to 0x4000 6400. The total capacity is 1KB, but USB uses actually only 512 bytes

#### Buffer descriptor table

Each USB endpoint has a relevant buffer descriptor only for control endpoint buffer. In application, it is generally treated as a special function “register”. Since these registers are not hardware mapped, their addresses are not constant. Register mainly control endpoints related memory address, buffer size, and number of transfer bytes. If the endpoint which corresponds to buffer descriptor is not enabled, the register of endpoint cannot come into use, and then buffer descriptor turn into available SRAM. A set of these registers is known as the buffer descriptor table, which provides a flexible method for the user to construct and control various length and configuration endpoint buffer.

The buffer descriptor table is composed of many data structures that define the buffer address and the length of the buffer. The first table item is defined by USBD_BADDR. An effective endpoint usually has two packet buffers, which are used to send and receive, so it correspond two table items. Each table item includes four 16-bits words, and thus is aligned with the 8 byte boundary.

#### Double buffer

In general, each endpoint has only one send buffer and one receive buffer, thus, for both sending and receiving, USB endpoint adopt signal buffer. However, to meet the following two kinds of USB transfer, the USB module implements a double buffer data transfer mode:

- Bulk transfer: require high data throughput, ensure can achieve the fastest rate when the bus is in idle state.

- Isochronous transfer: which is real-time streaming transfer, require data must be at a constant rate transfer, or arrive within a specific time limit.

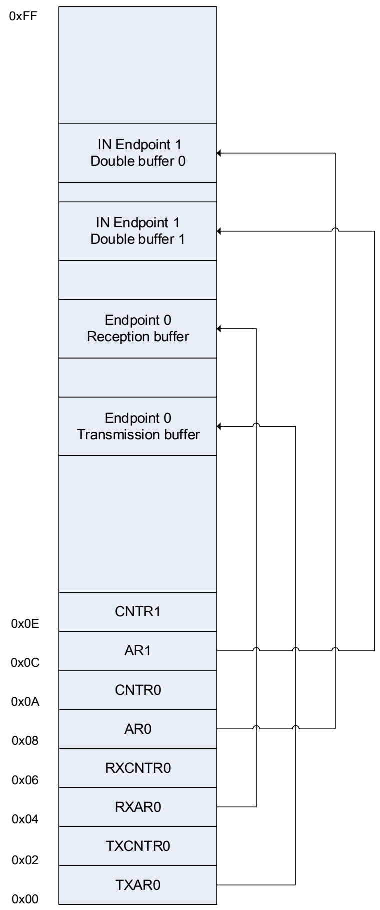
As an endpoint register can be used to send and receive data, so endpoint register actually control two buffers. For implementing double buffer, it is necessary to change two buffers which are controlled by endpoint register into two send buffers or receive buffers. Actually, the course of change from one buffer to double buffer is complex, referring to the “double buffer endpoint” section for details.

Figure 19-2 describes the relationship between the buffer description table and the packet buffer, including endpoint 0 which using single buffer and endpoint 1 which using double buffer.

**Note:** this figure is not drawn on the actual scale, and it is addressed through the USB bus

16-bits mode.

**Figure 19-2. An example with buffer descriptor table usage (USBD_BADDR = 0)**



## 19.3.3. Endpoint

The USB module supports dynamic configuration of the endpoint, except for the endpoint 0 (which is only configured to be control transfer mode), each endpoint could be configured to be any one of four kinds of USB transfer mode. Except that endpoint 0 is only configured with single buffer, other endpoints could be configured with single buffer or double buffer.

### Single buffer endpoint

In general, if endpoint is configured with single buffer, there is only one buffer in each transfer direction. Once the transaction is completed, the endpoint state will be automatically set to the NAK state by hardware. At this time, the USB device needs to handle the previous data transfer, however, if this USB endpoint meanwhile receives a new packet, USB device will reply host with NAK packet. It leads host to retransmit the same data constantly until the device could handle new data, and then to respond the ACK packet. Such a heavy retransmission takes up lots of bandwidth, which affects the transmission rate in terms of the bulk transfer, which need high throughput, and isochronous transfer, which require higher real-time performance. Therefore, the dual buffer is needed.

### Double buffer endpoint

Double buffer endpoint use two buffers to send and receive data, one of them is used by hardware, and the other one is used by application. In the course of using double buffer, USB peripherals need to know which buffer is used by applications for avoiding conflict. Since there are 2 data rollover bits in the USBD_EPxCS register, the USB device uses only 1 bit to handle hardware data processing (double buffer function constraint one direction transfer),so application program use the other bit to indicate which buffer is used at this time, that is what is called "SW_BUF".

In the following table, the correspondence between USBD_EPxCS register bits and DTOG/SW_BUF definition is explained.

**Table 19-3. Double-buffering buffer flag definition**

| Buffer flag | Tx endpoint | Rx endpoint |
|---|---|---|
| DTOG | TX_DTG(USBD_EPxCS bit 6) | RX_DTG(USBD_EPxCS bit 14) |
| SW_BUF | USBD_EPxCS bit 14 | USBD_EPxCS bit 6 |

The DTOG bit and the SW_BUF bit are responsible for the stream control. When a transfer completes, the USB peripheral toggle the DTOG bit; when the data have been copied, the application software need to toggle the SW_BUF bit. Except the first time, if the value of DTOG bit equal to the SW_BUF's, the transfer will pause, and NAK will be sent to host. When the two bits do not equal, the transfer resume.

To enable the double-buffered feature, EP_CTL and EP_KCTL need to be configured:

- Setting EP_CTL = '00
- Setting EP_KCTL = '1

**Table 19-4. Double buffer usage**

| Endpoint Type | DTOG | SW_ BUF | Packet buffer used by the USB peripheral | Packet buffer used by the application software |
|---|---|---|---|---|
| OUT | 0 | 1 | RXARn_0 / RXCNTRn_0 buffer description table locations. | RXARn_1 / RXCNTRn_1 buffer description table locations. |
| | 1 | 0 | RXARn_1 / RXCNTRn_1 buffer description table locations. | RXARn_0 / RXCNTRn_0 buffer description table locations. |
| IN | 0 | 1 | TXARn_0 / TXCNTn_0 buffer description table locations. | TXARn_1 / TXCNTn_1 buffer description table locations. |
| | 1 | 0 | TXARn_1 / TXCNTn_1 buffer description table locations. | TXARn_0 / TXCNTn_0 buffer description table locations. |

### Endpoint initialization

The USB endpoints must be initialized before they are used, initialization process is as follows:

- Initialize the TADDRx/RADDRx registers with transmission/reception data buffer address;
- Configure The EP_CTL and EP_KCTL bits in the USBD_EPxCS register according to the endpoint usage;
- For transmission, initiate the TXCNTn and enable TX_STA ; for reception, initiate the RXCNTRn, BLKSIZ and BLKNUM, then enable RX_STA.

For isochronous and double-buffered bulk endpoints, both transmission and reception relevant fields are need to be initialized. Once the endpoint is enabled, register USBD_EPxCS, buffer address and COUNT filed should not be modified by the application software.

### Endpoint request

If there are multiple endpoints trigger interrupt request at the same time, the hardware will respond to the highest priority endpoint. The priority definition method for endpoint is as follows:

- Firstly, depending on the basis of the endpoint attributes, synchronous endpoint and double buffering endpoint has a high priority, the other endpoints are low priority
- Then, according to the same attribute endpoint's endpoint number to arrange, the smaller the endpoint number, the higher the priority

Application should conform to the above priority policy order for processing endpoint interrupt request.

## 19.3.4. Interrupt handling

There are two kinds of USB interrupt event: device interrupt and endpoint interrupt. Device interrupt event includes the reset, suspend, wakeup and so on. Endpoint interrupt event only includes successful transfer interrupt. All interrupt events are captured through interrupt flag register, When there is an interrupt condition occur, the corresponding interrupt status bit will be set by the hardware (regardless of whether the interrupt enable bit is set). The software can program the corresponding bits of interrupt enable register, thereby NVIC triggers the USB interrupt event. NVIC can configure three kinds of USB interrupt as following:

- USB low-priority interrupt: triggered by all USB events
- USB high-priority interrupt: triggered only by a successful transfer event for isochronous and double buffer bulk transfer
- USB wakeup interrupt: triggered by the wakeup events.

### USB low-priority interrupts handling

Checking interrupt flag bits in the interrupt service program is to confirm the specific interrupt event. After the corresponding operation implemented, interrupt flag bit must be cleared, otherwise the same interrupt is triggered again. Even if the multiple interrupt flags are set, only one interrupt will be triggered. The sequence to check each interrupt flag of the interrupt flag register in the interrupt service program, which determines the priority of each interrupt event, processing procedures in accordance with this priority to handle the interrupt event.

### USB high-priority interrupts handling

It is only for handling isochronous transfer and dual buffering bulk transfer. Surely, the two transfers can also be configured as lower priority interrupt, but if it is configured as high priority interrupt, they will be handled in high priority interrupt process instead of lower priority process. Since there is only normally successful transfer interrupt in the interrupt service program, its handling speed is faster than lower priority interrupt.

### USB wakeup interrupt handling

In general MCU sleep is awaked by external interrupt event. However, USB is awaked by both external interrupt event and internal interrupt event. The external interrupt source is used to wake up the USB by triggering the interrupt line 18. The internal wake event is mainly aroused by the signal. Specific wake-up interrupt handling process is directly performed through the USB low priority interrupt processing service program.

The USB clock will be restored and the interrupt flag bit will be cleared in the wake up interrupt service program.

## 19.3.5. Reset events

### System reset and power on reset

Upon the system or the power reset, the USB controller and the interface are in the closed state. At this time, all the endpoints are in a disabled state, the USB module will not respond to any group. Applications need to be carried out as follows:

Firstly, the USB controller is configured to 48MHz, and the APB1 interface clock is turned on, so as to activate the register unit clock and then clear the reset signal;

Secondly, open the analog part of the USB transceiver, which opens the internal voltage required for a period of time;

Finally, the configuration of the register and buffer description table is required, so that the USB module can carry out normal interrupt processing and data transfer.

The USB firmware should do as follows:

- Reset CLOSE bit in USBD_CTL register
- Wait for the internal reference voltage stability
- Clear SETRST bit in USBD_CTL register

- Clear the IFR register to remove the redundant pending interrupt and then enable other unit

### Reset interrupt

Once reset occurs, the USB module will reset the internal protocol state machine, and the sending and receiving parts will be banned until the reset interrupt bit is cleared. All the configuration registers will keep the original state and will not be reset to ensure that correct execution can transfer immediately after reset, however, device address and endpoint register will be reset.

The USB firmware should do as follows:

- Set USBEN bit in USBD_DADDR register to enable USB module in 10ms
- Initialize the EP0CS register and its related packet buffers

## 19.3.6. Transfer procedure

### USB transaction summary

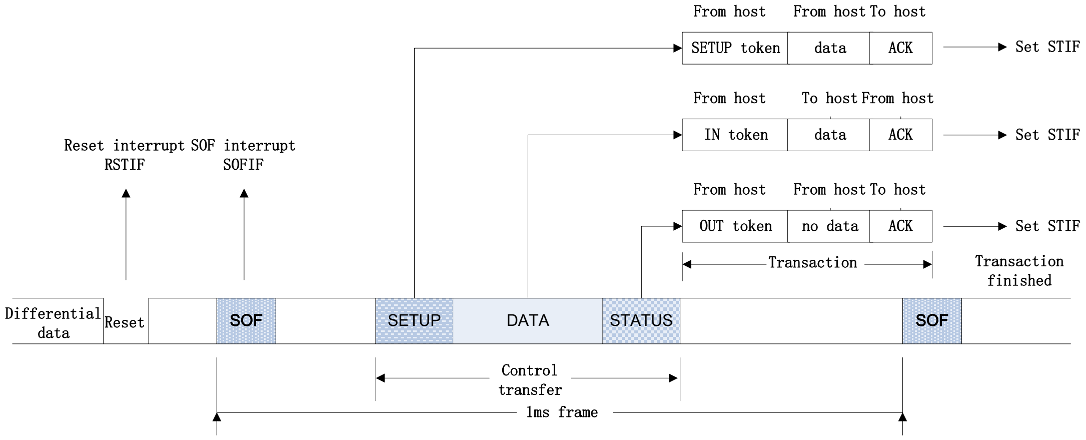
USB module transfers data between the device endpoint and the host through the transaction. Host defines the direction of the transaction: IN and OUT are relative to the host, the IN transaction is to transfer data from the device to the host, the OUT transaction is to transfer data from the host to the device. All the transactions are initiated by the host controller.

The process of a transaction is generally as follows: when the host sends a valid function / endpoint token packet that can be recognized by the USB module, the USB device is properly received and needs to send or receive data, which will be followed by the associated data transfer process. The USB module is used to realize the data exchange between the special buffer and the port through the buffer description block and the endpoint register, and then realizes the data exchange with the host through the port. After all the data transfer is completed, if required, according to the direction of transmission, send or receive the corresponding handshake packet.

After each successful transaction, it will trigger an interrupt related to the endpoint. In the case of enabling interrupt related to the endpoint, the Interrupt flag register can be read to handle the corresponding interrupt event.

Figure 19-3 describes the relationship between the USB transaction and the interrupt event:

**Figure 19-3 .USB transaction and the interrupt event**



### Data transmission (IN transaction)

When a valid endpoint receives an IN token packet, the SIE will begin to read TXCNTn bytes of parallel data from the endpoint associated TADDRx buffer. After the IN packet transmission, the data will be loaded into the output shift register and converted to serial data to be sent through the USB ATX to the USB bus and send to host. At the end of the data transmission, the hardware will send the computed CRC to the host. If the endpoint indicated by the IN group is invalid, a NAK or STALL handshake is sent according to the TX_STA value. The USB peripheral transmit the data packet from the address indicated by TADDRx by LSB order.

When receiving the ACK sent from the host, then the USB peripheral will toggle the TX_DTG, set TX_STA=’10 (NAK) and TX_ST bit. The application software can identify which endpoint has completed transfer by checking the EPNUM and DIR bits in the USBD_INTF register. After filled the data packet memory with data, the application software can start next transfer by re-enable the endpoint by setting TX_STA=’11 (VALID).

### Data reception (OUT and SETUP transaction)

When a valid endpoint receives an OUT or SETUP token packet, the USB ATX will receive the bidirectional data on the USB bus. SIE receives serial data from the ATX and converts them into parallel data streams. These data streams are stored in the buffer by the RADDRx controlled. And the controller updates RXCNTRn with actually received number of bytes. The values of BLKSIZ and BLKNUM are used to compute the BUF_COUNT, which is used to detect the buffer overrun. The USB peripheral receives the data from the host by LSB order and the computed CRC. When detecting the end of DATA packet, the computed CRC and received CRC are compared. If no errors occur, an ACK handshake packet is sent to the host.

If any error happens during reception, the USB peripheral set the ERRIF bit and still copy data into the packet memory buffer, but not send the ACK packet. The USB peripheral itself can recover from reception errors and continue to handle next transfer. The USB peripheral never access outside the packet memory buffer, which is defined by BLKSIZ and BLKNUM. The received 2-byte CRC which follow data bytes is also copied to the packet memory buffer. If the length of data is greater than actually allocated length, the excess data are not copied.

This is a buffer overrun condition. A STALL handshake is sent, and this transaction fails.

If an addressed endpoint is invalid, a NAK or STALL handshake packet is sent instead of the ACK according to bits RX_STA in the USBD_EPxCS register, no data is written in the reception memory buffers. Reception memory buffers are written starting from the address contained in the RADDRx, the number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BLKSIZ and BLKNUM. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

If no error happens, the USB peripheral send the ACK handshake packet to the host, toggle RX_DTG bit, set RX_STA=‘10 (NAK) and RX_ST bit. The application software checks the EPNUM and DIR bits in the USBD_INTF register to determine which endpoint to trigger the RX_ST. After handling the received data, the application software can initiate another transaction by setting RX_STA=’11 (Valid).

### Control transfers

Control transfers require that a SETUP transaction be started from the host to a device to describe the type of control access that the device should perform. The SETUP transaction is followed by zero or more control DATA transactions that carry the specific information for the requested access. Finally, a STATUS transaction completes the control transfer and allows the endpoint to return the status of the control transfer to the client software. After the STATUS transaction for a control transfer is completed, the host can urge the next control transfer for the endpoint.

To initialize the control transfer, TX_DTG and RX_DTG bits are set to 1 and 0 respectively, and both TX_STA and RX_STA are set to ‘10 (NAK). According to the SETUP contents, the application software can determine if the next transactions are IN or OUT. When RX_ST event happens during control transfer, the SETUP bit must be firstly checked. If it is 1, it means a SETUP transaction, or else a normal OUT one. The USB peripheral is aware of the number and direction of data transfer by analyzing the contents of SETUP transaction, and is required to set the unused direction to STALL except the last data stage.

At the last data stage, the application software set the control endpoint opposite direction to NAK. This will keep the host waiting for the completion of the control operation. If the operation completes successfully, the software will change NAK to VALID, otherwise to STALL. If the status stage is an OUT transaction, the STATUS_OUT bit should be set, so that a status transaction with non-zero data will be answered STALL to indicate an error happen.

The USB specification requires that the SETUP packet of the host has an automatic retransfer function, which means that the device cannot respond to the SETUP request of the host with a non ACK packet (NAK packet or STALL packet). When the SETUP packet transmission fails, the next SETUP packet transmission will be triggered.

The RX_STA bit of the control endpoint cannot be set to '00 (disabled) state. When the SETUP token is received, the USB receives the data, performs the requested data transmission and sends back an ACK handshake packet. If that endpoint has a previously issued RX_ST request not yet handled by the application, the USB will discard the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. It is to avoid missing another SETUP packet transfer which following RX_ST interrupt.

### Isochronous transfers

Isochronous transfers can guarantee constant data rate and bounded latency, but do not support data retransmission in response to errors on the bus. A receiver can check that a transmission error occurred. The bottom USB protocol does not allow handshakes to be returned to the transmitter of an isochronous pipe. Normally, handshakes would be returned to tell the transmitter whether a packet was successfully received or not. Consequently, the isochronous transaction does not have a handshake phase, and have no ACK packet after the data packet. Data toggling is not supported, and only DATA0 PID is used to start a data packet.

The endpoint is defined as isochronous endpoint by setting the EP_CTL bits to ‘10. The STA bits have two possible values: ‘00 (Disabled) and ‘11 (Valid), any other value is illegal. The application software can implement double-buffering to improve performance. By swapping transmission and reception data packet buffer on each transaction, the application software can copy the data into or out of a buffer, at the same time the USB peripheral handle the data transmission or reception of data in another buffer. The DTOG bit indicates that the USB peripheral is currently using which buffer.

The application software initializes the DTOG according to the first buffer to be used. At the end of each transaction, the RX_ST or TX_ST bit is set depending on the enabled direction, regardless of CRC errors and buffer-overrun condition (if errors happen, the ERRIF bit will be set). At the same time, The USB peripheral will toggle the DTOG bit, but not affect the STA bits.

## 19.3.7. Power management

### Power mode

USB applications tend to have several different power requirements and configurations. The most common mode of power supply is as follows:

1. Bus power supply mode: the USB protocol insists on power management by the USB device which requires the device to draw power from the bus. The following constraints should be met by the bus-powered device.

   - A device in the non-configured state should draw a maximum of 100mA from the USB bus.

- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500mA.
- A suspended device should draw a maximum of 500uA.

2. Self powered mode: in this mode, USB is applied to provide power for itself, only a very small part of the power supply comes from USB
3. Dual power with self-power dominance mode: some applications may require a dual power option. This allows the application to use internal power primarily, but switch to power from the USB when no internal power is available.

The USB module supports both the bus power supply and the self powered mode, and the switching which requires manual control between the two modes.

### Suspend and wakeup

A device will go into the SUSPEND state if there is no activity on the USB bus for more than 3ms. As the full speed mode of the USB host sends a start frame packet (SOF) every millisecond, the suspend mode can be determined by detecting 3 consecutive SOF packet loss events. A suspended device wakes up, if RESUME signaling is detected. RESUME signaling is derived from the RESUME sequence, which can be initiated by the host, and can also be triggered by the device itself. Under the SUSPEND state, the hardware do not detect the suspend signal until wakeup process complete.

The USB peripheral also supports software initiated remote wakeup. Remote wakeup always means device to wakeup host. At this time, it requires to wakeup USB module first, then wakeup host. To initiate remote wakeup, the application software must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wakeup signal upstream.

Setting the SETSPS bit to 1 enables the SUSPEND mode, and it will disable the check of SOF reception. Setting the LOWM bit to 1 will shut down the static power consumption in the analog USB transceivers, but the resume signal is still able to be detected.

Setting the RSREQ bit to 1 and clearing it in 10ms will initiate the RESUME sequence. (The time interval of 10ms can be implemented with ESOF interrupt, the interrupt occurs once per 1ms in the kernel.)

### Link Power Management (LPM) Level L1

A device will go into the LPM L1 state (sleep state) if the host sends the EXT PID and LPM SubPID when the SUB_STA bits in USB sub endpoint registers is 0x3(Valid).

Setting the SETSPS bit to 1 enables the power saving mode, and it will disable the check of SOF reception. Setting the LOWM bit to 1 will shut down the static power consumption in the analog USB transceivers, but the resume signal is still able to be detected. When the resume signal is detected, sleep state will exit.

## 19.4. USB registers

### 19.4.1. Control register (USBD_CTL)

Address offset: 0x40

Reset value: 0x0003

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STIE | PMOUIE | ERRIE | WKUPIE | SPSIE | RSTIE | SOFIE | ESOFIE | Reserved | | | RSREQ | SETSPS | LOWM | CLOSE | SETRST |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | STIE | Successful transfer interrupt enable<br>0: Successful transfer interrupt disabled<br>1: Interrupt generated when STIF bit in USBD_INTF register is set |
| 14 | PMOUIE | Packet memory overrun / underrun interrupt enable<br>0: No interrupt generated when packet memory overrun / underrun<br>1: Interrupt generated when PMOUIF bit in USBD_INTF register is set |
| 13 | ERRIE | Error interrupt enable<br>0: Error interrupt disabled<br>1: Interrupt generated when ERRIF bit in USBD_INTF register is set |
| 12 | WKUPIE | Wakeup interrupt enable<br>0: Wakeup interrupt disabled<br>1: Interrupt generated when WKUPIF bit in USBD_INTF register is set |
| 11 | SPSIE | Suspend state interrupt enable<br>0: Suspend state interrupt disabled<br>1: Interrupt generated when SPSIF bit in USBD_INTF register is set |
| 10 | RSTIE | USB reset interrupt enable<br>0: USB reset interrupt disabled<br>1: Interrupt generated when RSTIF bit in USBD_INTF register is set |
| 9 | SOFIE | Start of frame interrupt enable<br>0: Start of frame interrupt disabled<br>1: Interrupt generated when SOFIF bit in USBD_INTF register is set. |
| 8 | ESOFIE | Expected start of frame interrupt enable register<br>0: Expected start of frame interrupt disabled<br>1: Interrupt generated when ESOFIF bit in USBD_INTF register is set |

| | | |
|---|---|---|
| 7:5 | Reserved | Must be kept at reset value |
| 4 | RSREQ | The software set a resume request to the USB host, and the USB host should drive the resume sequence according the USB specifications<br>0: No resume request<br>1: Send resume request. |
| 3 | SETSPS | The software should set suspend state when SPSIF bit in USBD_INTF register is set<br>0: Not set suspend state.<br>1: Set suspend state. |
| 2 | LOWM | When set this bit, the USB goes to low-power mode at suspend state. If resume from suspend state, the hardware reset this bit<br>0: No effect<br>1: Go to low-power mode at suspend state |
| 1 | CLOSE | When this bit set, the USB goes to close state, and completely close the USB and disconnected from the host<br>0: Not in close state<br>1: In close state. |
| 0 | SETRST | When this bit set, the USB peripheral should be reset<br>0: No reset<br>1: A reset generated |

### 19.4.2. Interrupt flag register (USBD_INTF)

Address offset: 0x44

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STIF | PMOUIF | ERRIF | WKUPIF | SPSIF | RSTIF | SOFIF | ESOFIF | Reserved | | | DIR | EPNUM[3:0] | | | |
| r | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | | r | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | STIF | Successful transfer interrupt flag<br>This bit set by hardware when a successful transaction completes |
| 14 | PMOUIF | Packet memory overrun / underrun interrupt flag<br>This bit set by hardware to indicate that the packet memory is inadequate to hold transfer data. The software writes 0 to clear this bit. |
| 13 | ERRIF | Error interrupt flag<br>This bit set by hardware when an error happens during transaction. The software writes 0 to clear this bit. |
| 12 | WKUPIF | Wakeup interrupt flag |

| | | |
|---|---|---|
| | | This bit set by hardware in the SUSPEND state to indicate that activity is detected. The software writes 0 to clear this bit. |
| 11 | SPSIF | Suspend state interrupt flag<br>When no traffic happen in 3 ms, hardware set this bit to indicate a SUSPEND request. The software writes 0 to clear this bit. |
| 10 | RSTIF | USB reset interrupt flag<br>Set by hardware when the USB RESET signal is detected. The software writes 0 to clear this bit. |
| 9 | SOFIF | Start of frame interrupt flag<br>Set by hardware when a new SOF packet arrives, The software writes 0 to clear this bit. |
| 8 | ESOFIF | Expected start of frame interrupt flag<br>Set by the hardware to indicate that a SOF packet is expected but not received. The software writes 0 to clear this bit. |
| 7:5 | Reserved | Must be kept at reset value |
| 4 | DIR | Direction of transaction<br>Set by the hardware to indicate the direction of the transaction<br>0: IN type<br>1: OUT type |
| 3:0 | EPNUM[3:0] | Endpoint Number<br>Set by the hardware to identify the endpoint which the transaction is directed to |

### 19.4.3. Status register (USBD_STAT)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RX_DP | RX_DM | LOCK | SOFLN[1:0] | | FCNT[10:0] | | | | | | | | | | |
| r | r | r | r | | r | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | RX_DP | Receive data + line status<br>Represent the status on the DP line. |
| 14 | RX_DM | Receive data - line status<br>Represent the status on the DM line. |
| 13 | LOCK | Locked the USB<br>Set by the hardware indicate that at the least two consecutive SOF have been received. |

| | | |
|---|---|---|
| 12:11 | SOFLN[1:0] | SOF lost number<br>Increment every ESOFIF happens by hardware.<br>Cleared once the reception of SOF. |
| 10:0 | FCNT[10:0] | Frame number counter<br>The Frame number counter incremented every SOF received. |

### 19.4.4. Device address register (USBD_DADDR)

Address offset: 0x4C

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value |
| 7 | USBEN | USB device enable<br>Set by software to enable the USB device.<br>0: The USB device disabled. No transactions handled.<br>1: The USB device enabled. |
| 6:0 | USBADDR[6:0] | USB device address<br>After bus reset, the address is reset to<br>0x00. If the enable bit is set, the device will respond on packets<br>for function address DEV_ADDR. |

### 19.4.5. Buffer address register (USBD_BADDR)

Address offset: 0x50

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:3 | BAR[15:3] | Buffer address<br>Start address of the allocation buffer(512byte on-chip SRAM), used for buffer descriptor table, packet memory. |

| | | |
|---|---|---|
| 2:0 | Reserved | Must be kept at reset value |

### 19.4.6. Endpoint n control/status register (USBD_EPxCS), n=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RX_ST | RX_DTG | RX_STA[1:0] | | SETUP | EP_CTL[1:0] | | EP_KCTL | TX_ST | TX_DTG | TX_STA[1:0] | | EP_AR[3:0] | | | |
| rc_w0 | t | t | | r | rw | | rw | rc_w0 | t | t | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | RX_ST | Reception Successful Transferred<br>Set by hardware when a successful OUT/SETUP transaction complete.<br>Cleared by software by writing 0. |
| 14 | RX_DTG | Reception Data PID Toggle<br>This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint.<br>Used to implement the flow control for double-buffered endpoint.<br>Used to swap buffer for isochronous endpoint. |
| 13:12 | RX_STA[1:0] | Reception status bits<br>Toggle by writing 1 by software.<br>Remain unchanged by writing 0.<br>Refer to the table below |
| 11 | SETUP | Setup transaction completed<br>Set by hardware when a SETUP transaction completed. |
| 10:9 | EP_CTL[1:0] | Endpoint type control<br>Refer to the table below |
| 8 | EP_KCTL | Endpoint kind control<br>The exact meaning depends on the endpoint type.<br>Refer to the table below |
| 7 | TX_ST | Transmission Successful Transfer<br>Set by hardware when a successful IN transaction complete.<br>Clear by software. |
| 6 | TX_DTG | Transmission Data PID Toggle<br>This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint.<br>Used to implement the flow control for double-buffered endpoint.<br>Used to swap buffer for isochronous endpoint. |

| | | |
|---|---|---|
| 5:4 | TX_STA[1:0] | Status bits, for transmission transfers<br>Refer to the table below |
| 3:0 | EP_AR | Endpoint address<br>Used to direct the transaction to the target endpoint. |

**Table 19-5. Reception status encoding**

| RX_STA[1:0] | Meaning |
|---|---|
| 00 | **DISABLED:** ignore all reception requests of this endpoint |
| 01 | **STALL**: STALL handshake status |
| 10 | **NAK**: NAK handshake status |
| 11 | **VALID**: enable endpoint for reception. |

**Table 19-6. Endpoint type encoding**

| EP_CTL[1:0] | Meaning |
|---|---|
| 00 | BULK |
| 01 | CONTROL |
| 10 | ISO |
| 11 | INTERRUPT |

**Table 19-7. Endpoint kind meaning**

| EP_CTL[1:0] | | EP_KCTL Meaning |
|---|---|---|
| 00 | BULK | DBL_BUF |
| 01 | CONTROL | STATUS_OUT |

**Table 19-8. Transmission status encoding**

| TX_STA[1:0] | Meaning |
|---|---|
| 00 | **DISABLED:** ignore all transmission requests of this endpoint |
| 01 | **STALL**: STALL handshake status |
| 10 | **NAK**: NAK handshake status |
| 11 | **VALID**: enable this endpoint for transmission. |

## 19.4.7. Transmission buffer address register x (USBD_TADDRx)

Address offset: [USBD_BADDR] + n*16

USB local address: [USBD_BADDR] + n*8

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXARn[15:1] | | | | | | | | | | | | | | | TXARn0 |
| rw | | | | | | | | | | | | | | | rw |

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

| | | |
|---|---|---|
| 15:1 | TXARn[15:1] | Transmission buffer address<br>Start address of the packet buffer containing data to be sent when receive next IN token. |
| 0 | TXARn[0] | Must be set to 0 |

### 19.4.8. Transmission byte count register x (USBD_TCNTx)

Address offset: [USBD_BADDR] + n*16 + 4
USB local Address: [USBD_BADDR] + n*8 + 2

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | Reserved | Must be kept at reset value |
| 9:0 | TXCNTn[9:0] | Transmission bytes count<br>The number of bytes to be transmitted at next IN token |

### 19.4.9. Reception buffer address register x (USBD_RADDRx)

Address offset: [USBD_BADDR] + n*16 + 8
USB local Address: [USBD_BADDR] + n*8 + 4

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXARn[15:1] | | | | | | | | | | | | | | | RXARn0 |
| rw | | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:1 | RXARn[15:1] | Reception buffer address<br>Start address of packet buffer containing the data<br>received by the endpoint at the next OUT/SETUP token. |
| 0 | RXARn[0] | Must be set to 0 |

### 19.4.10. Reception byte count register x (USBD_RCNTx)

Address offset: [USBD_BADDR] + n*16 + 12
USB local Address: [USBD_BADDR] + n*8 + 6

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | BLKSIZ | Block size<br>0: Block size is 2 bytes<br>1: Block size is 32 bytes |
| 14:10 | BLKNUM[4:0] | Block number<br>The number of blocks allocated to the packet buffer. |
| 9:0 | RXCNTRn[9:0] | Reception bytes count<br>The number of bytes to be received at next OUT/SETUP token. |

### 19.4.11. Sub-endpoint x registers (USBD_SEPx), n=[0..7]

Address offset: 0x100 to 0x11C

Reset value: 0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SUB_ST | Reserved | SUB_STA[1:0] | | Reserved | SUBPID_ATTR | | | | | | | | | | |
| rc_w0 | | t | | | r | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | SUB_ST | Successful Receive for LPM Token<br>Set by hardware when a successful receiving for LPM token completely.<br>Cleared by software by writing 0. |
| 14 | Reserved | Must be kept at reset value |
| 13:12 | SUB_STA[1:0] | Status bits, for the handshake of receiving subpid LPM<br>Toggle by writing 1 by software.<br>Remain unchanged by writing 0.<br>Refer to the table below |
| 11 | Reserved | Must be kept at reset value |
| 10:0 | SUBPID_ATTR | LPM Token bmAttribute Field.<br>These bits write by hardware and read by software. |

**Table 19-9. Sub-endpoints status**

| SUB_STA[1:0] | Meaning |
|---|---|
| 00 | **DISABLED:** ignore all LPM token reception requests of this endpoint |
| 01 | **STALL**: STALL handshake status |

| | |
|---|---|
| 10 | **NYET**: NYET handshake status |
| 11 | **VALID**: enable endpoint for reception LPM token |

### 19.4.12. LPM control register (USBD_LPMCTL)

Address offset: 0x140

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | LPMSTIE | LPM token successful transfer interrupt enable<br>This bit set and reset by software<br>0: No interrupt generated<br>1: Interrupt generated when LPM token successful transferred |
| 14:0 | Reserved | Must be kept at reset value |

### 19.4.13. LPM interrupt flag register (USBD_LPMINTF)

Address offset: 0x144

Reset value: 0x0000

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

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

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | LPMSTIF | LPM token Correct transfer interrupt flag<br>This bit set and reset by software<br>0: No effect<br>1: LPM token Correct transferred |
| 14:0 | Reserved | Must be kept at reset value |

# 20. Real-time clock(RTC)

## 20.1. Introduction

The RTC provides a time which includes hour/minute/second/sub-second and a calendar includes year/month/day/week day. The time and calendar are expressed in BCD code except sub-second. Sub-second is expressed in binary code. Hour adjust for daylight saving time. Works in power saving mode and smart wakeup for software configurable. Support extern accurate low frequency clock for higher calendar accuracy.

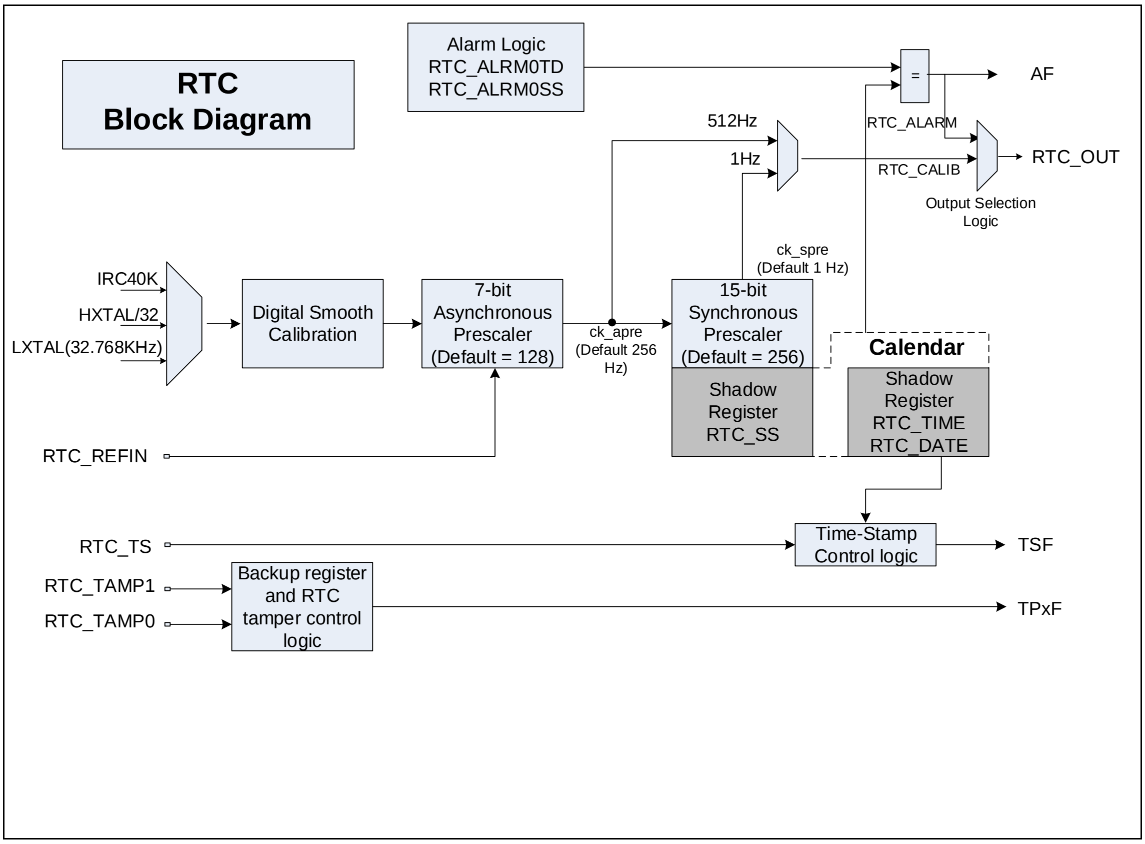
## 20.2. Main features

- Daylight saving compensation support by software
- External high accurate low frequency(50Hz or 60Hz) clock for higher calendar accuracy performed by reference clock detection option function
- Atomic clock adjust(max adjust 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
- 2 Tamper source can be choice and tamper type is configurable
- Programmable calendar and field maskable alarm
- Maskable interrupt source:
  - Alarm
  - Time-stamp detection
  - Tamper detection
- Five 32-bit universal backup registers which can keep data under power saving mode. Backup register will be reset if tamper event detected

## 20.3. Function description

### 20.3.1. Block diagram

**Figure 20-1.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)
  - 1Hz(default prescale)
  - Alarm event(polarity is configurable)
- 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)

### 20.3.2. RTC pin

PC13, PC14 and PC15 can be directly controlled by RTC through the register RTC_TAMP and RTC_CTL.

The function priority of PC13 is obey the below table:

| PC13 Function | OS | COEN | TP1EN | TSEN | PC13 MODE | PC13 VALUE |
|---|---|---|---|---|---|---|
| RTC_ALARM output OD | 1 | X | X | X | X | 0 |
| RTC_ALRM output PP | 1 | X | X | X | X | 1 |
| RTC_CALIB output PP | 0 | 1 | X | X | X | X |
| RTC_TAMP0 Input Floating | 0 | 0 | 1 | 0 | X | X |
| RTC_TS and RTC_TAMP0 Input Floating | 0 | 0 | 1 | 1 | X | X |
| RTC_TS Input Floating | 0 | 0 | 0 | 1 | X | X |
| Output PP forced | 0 | 0 | 0 | 0 | 1 | PC13 value bit |
| Wakeup pin or Standard GPIO | 0 | 0 | 0 | 0 | 0 | X |

**Note:** (1) OD means open drain, PP means push-pull

(2) ‘X’ means no effect

The function priority of PC14 is obey the below table:

| PC14 Function | LXTALEN in RCU_BDCTL | LXTALBPS in RCU_BDCTL | PC14 MODE | PC14 VALUE |
|---|---|---|---|---|
| LXTAL Oscillator Analog Input | 1 | 0 | X | X |
| LXTAL Bypass Analog Input | 1 | 1 | X | X |
| Output PP | 0 | X | 1 | PC14 value bit |

| | | | | |
|---|---|---|---|---|
| forced | | | | |
| Standard GPIO | 0 | X | 0 | X |

**Note:** (1) OD means open drain, PP means push-pull

(2) 'X' means no effect

The function priority of PC15 is obey the below table:

| PC15 Function | LXTALEN in RCU_BDCTL | LXTALBPS in RCU_BDCTL | PC15 MODE | PC15 VALUE |
|---|---|---|---|---|
| LXTAL Oscillator Analog Input | 1 | 0 | X | X |
| Output PP forced | 1 | 1 | 1 | PC15 value bit |
| | 0 | X | | |
| Standard GPIO | 0 | X | 0 | X |

**Note:** (1) OD means open drain, PP means push-pull

(2) 'X' means no effect

### 20.3.3. Clock and prescalers

RTC unit has three independent clock sources: LXTAL IRC40K and HXTAL with 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 other is a 15-bit synchronous prescaler. Asynchronous prescaler is mainly used for reducing the power consuming. The asynchronous prescaler is recommended to set as high as possible if both the prescalers are used.

The frequency formula of two prescalers are shown as below:

$$f_{\text{ck\_apre}} = \frac{f_{\text{rtcclk}}}{\text{FACTOR\_A} + 1}$$

$$f_{\text{ck\_spre}} = \frac{f_{\text{ck\_apre}}}{\text{FACTOR\_S} + 1} = \frac{f_{\text{rtcclk}}}{(\text{FACTOR\_A} + 1)*(\text{FACTOR\_S} + 1)}$$

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 is reached 0 it will automatically reload to FACTOR_S. The ck_spre clock is used to driven the calendar registers. Each clock will make second plus one.

### 20.3.4. Real-time clock and calendar

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 to real calendar register every two RTC clock and at the same time RSYNF bit is 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) if software wants to read 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 ($f_{RTCCLK}$).

Systerm reset will reset the shadow calendar registers.

### 20.3.5. 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/disabled by ALRM0EN bit in RTC_CTL. If all the alarm fields value match the corresponding calendar value when ALRM0EN=1, the ALRM0F flag will set.

**Note:** If the seconds field is selected (MSKS bit reset in RTC_ALRM0TD), the synchronous prescaler division factor value in the RTC_PSC register must be set at least 3 to ensure correct behavior.

If a field is masked, the field is seen as matched in logic. If all the fields are masked the ALRM0F will assert 3 RTC clock later after ALRM0EN is set.

### 20.3.6. 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 below steps 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.

### Calendar initialization and configuration

The prescaler and calendar value can be programmed by following steps:

1. Set INITM bit to 1 to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.

2. Circularly reading INITF bit. The initialization phase mode is really entered if INITF is set to 1. It takes around 2 RTCCLK clock cycles (because of clock synchronization).

3. Program both the asynchronous and synchronous prescaler factors in RTC_PSC register.

4. 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).

5. Clear INITM bit for exiting the initialization mode.

About 4 RTCCLK 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 is already set to 1.

YCM flag indicates whether the calendar has been initialized by checking the year field of calendar is 0x00(YCM=0) or not (YCM=1).

### Daylight saving Time

RTC unit supports daylight saving time adjust function 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 adjust operation. After setting the S1H/A1H, subtract/add 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. Clear ALRM0EN in RTC_CTL to disable Alarm

2. Set the Alarm registers needed(RTC_ALRM0SS/ RTC_ALRM0TD)

3. Set ALRM0EN in the RTC_CTL register to enable Alarm function

### 20.3.7. Calendar reading

#### Reading calendar registers under BPSHAD=0

When BPSHAD=0, caldendar 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 whatever happens.

When APB1 bus clock frequency is not equal to or greater than 7 times the RTC clock, the calendar reading flow should be obeyed:

1. reading calendar time register and date register twice
2. if the twice value is equal, the value can be seen as the correct value
3. if the twice value is 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 calendar register will be updated to current calendar time and date.

To ensure consistency of the 3 values (RTC_SS, RTC_TIME, 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 before reading again.

In below situations, software should wait RSYNF bit asserted be for reading calendar register(RTC_SS, RTC_TIME, RTC_DATE):

1. after a system reset
2. after an initialization
3. after synchronization

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 maintained to 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 with no any delay(max to 2 RTC clock) 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 twice data are the same, the data is coherent and correct.

### 20.3.8. 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 register (RTC_SHIFTCTL)
- RTC timestamp registers (RTC_SSTS/RTC_TTS/RTC_DTS)
- RTC tamper and alternate function configuration register (RTC_TAMP)
- RTC backup registers (RTC_BKPx)
- RTC Alarm registers (RTC_ALRM0SS/RTC_ALRM0TD)

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.

### 20.3.9. RTC synchronization

When the user has a remote clock with higher degree of precision and RTC time has an offset in a fraction of a second with remote clock, RTC unit provides a function named synchronization shift 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 adding sub-seconds value (by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0]) or subtracting(by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] and set A1S at the same time) sub-seconds value 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 is FACTOR_S, the higher is adjust 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 and at the same time the lower FACTOR_A means the higher power consuming.

**Note:** Before using synchronization shift function, the software must check the 15-bit of SS in RTC_SS(SSC[15]) and confirm it is 0.

After writing RTC_SHIFTCTL register, the SOPF bit in RTC_STAT will set at once. When synchronization shift operation is complete, SOPF bit is cleared by hardware. System reset does not affect SOPF bit.

Synchronization shift operation only works correctly when REFEN=0.

Software must not write to RTC_SHIFTCTL if REFEN=1.

### 20.3.10. RTC reference clock detection

RTC reference clock detection is another way supported to increase the precision of RTC second. To enable this function, you should has a external clock source(50Hz or 60 Hz) which has the higher precision than LXTAL clock source.

After enabling this function (REFEN=1), each second update clock(1Hz) 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 edge are misaligned for the reason of LXTAL precision, the RTC reference clock detection function will shift the 1Hz clock edge a little to make future 1Hz aligned to reference clock edge.

When REFEN=1, a time window around every second update time for detection is performed. Different detection state will use different window period.

7 ck_apre window periods are used when the detection state is detecting the first reference clock edge and 3 ck_apre window periods are 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) edge are aligned, this reload operation has no any effect for 1Hz calendar updating 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 and the external reference clock removed(no reference clock edge found in 3 ck_apre window), the calendar updating also 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 ck_spre(1Hz) clock edge.

**Note:** Software must configure the FACTOR_A to 0x7F and FACTOR_S to 0xFF before enabling reference detection function(REFEN=1)

Reference detection function does not work in Standby Mode.

### 20.3.11. RTC smooth digital calibration

RTC calibration function is a way to calibrate the RTC frequency based RTC clock cycle number in a configurable period time.

This calibration is equally executed in a period time and after finish calibrating once, the cycle number of the RTC clock in the period time will be added or subtracted some individual RTCCLK cycles. 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 220/219/218 RTC clock cycles which stands for 32/16/8 seconds if RTC input frequency is 32.768KHz.

The calibration configure register (RTC_HRFC) specifies the number of RTCCLK clock cycles to be calibrated during the period time:

- Setting CMSK[0] to 1 causes 1 cycle to be masked during the period time
- Setting CMSK[1] to 1 causes 2 additional cycles to be masked
- Setting CMSK[2] to 1 causes 4 additional cycles to be masked
- Setting CMSK[8] to 1 causes 256 additional cycles to be masked

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 211/210/29(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 a 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:

$$f_{cal}=f_{rtcclk}*[1+(FREQI\times 512-CMSK)/(2^{N} +CMSK-FREQI\times 512)]$$

**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 32.768KHz, the corresponding FACTOR_S should be set as following rule:

FACTOR_A = 2: 2 less than nominal FACTOR_S(8189 with 32.768KHz)

FACTOR_A = 1: 4 less than nominal FACTOR_S(16379 with 32.768KHz)

FACTOR_A = 0: 8 less than nominal FACTOR_S(32759 with 32.76KHz)

When the FACTOR_A is less than 3,the formula of calibration frequency is as follows:

$$f_{cal}=f_{rtcclk}*[1+(256-CMSK)/(2^{N}+CMSK-256)]$$

**Note:** If FACTOR_A is less than 3, the calibration midpoint of CMSK is 0x100 when RTC clock is exactly 32.768KHz.

### Verifying the RTC calibration

Calibration 1Hz output is provided to assist software to measure and verify the RTC precision.

Up to 2 RTCCLK 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 bit is set to 0

2) Write the new value into RTC_HRFC register

3) After 3 ck_apre clocks, the new calibration settings take effect

## 20.3.12. 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 seen 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.

## 20.3.13. 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 that remains powered-on by $V_{BAT}$ even if $V_{DD}$ power is switched off. The wake up action from Standby Mode or system reset are not affect these registers.

These registers are only affected to reset by detected tamper event or when reading protection of the flash is changed from level 1 to level 0.

### 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 as 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 on tamper0 (PC13) when $V_{DD}$ power is switched off if tamper0 is enabled.

### Level detection mode with configurable filtering on tamper input detection

When FLT bit is not set to 0x0, the tamper detection is set to level detection mode and FLT bit determines the consecutive number of time(2,4 or 8) for sampling.When DISPU is set to 0(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 interval time between each sampling is also configurable. Through adjusting the sampling frequency(FREQ), software can balance between the power consuming and tamper detection latency.

## 20.3.14. Calibration clock output

Calibration reference clock can be output on the PC13 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 for there are maybe a light jitter on falling edge.

When the COS bit is set to 1, the RTC_CALIB frequency is ：

$$f_{RTCCLK}/((FACTOR\_S+1)*(FACTOR\_A+1))$$

When the RTCCLK is 32.768KHz, RTC_CALIB output is corresponding to 1Hz if prescaler are default values.

### 20.3.15. Alarm output

When OS control bits are set to 0x01, RTC_ALARM alternate function output is enabled. This function will directly output the content of ALRM0F bit in RTC_STAT.

The OPOL bit in RTC_CTL can configure the polarity of the ALRM0F output which means that whether the RTC_ALARM output is the opposite of ALRM0F bit.

### 20.3.16. 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 |

### 20.3.17. 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 interrupt:

1) Configure and enable the corresponding line to RTC alarm/tamper/timestamp event of EXIT and set the rising edge for triggering

2) Configure and enable the RTC global interrupt

3) Configure and enable the RTC alarm/tamper/timestamp function

| Interrupt | Event flag | Control Bit | Exit Sleep | Exit Deep-sleep | Exit Standby |
|---|---|---|---|---|---|
| Alarm | 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(*) |

**Note:** Only active when RTC clock source is LXTAL or IRC40K

## 20.4. RTC registers

### 20.4.1. Time of day 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)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="9">Reserved</td><td>PM</td><td colspan="2">HRT[1:0]</td><td colspan="4">HRU[3:0]</td></tr>
<tr><td colspan="9"></td><td>rw</td><td colspan="2">rw</td><td colspan="4">rw</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Reserved</td><td colspan="3">MNT[2:0]</td><td colspan="4">MNU[3:0]</td><td>Reserved</td><td colspan="3">SCT[2:0]</td><td colspan="4">SCU[3:0]</td></tr>
<tr><td></td><td colspan="3">rw</td><td colspan="4">rw</td><td></td><td colspan="3">rw</td><td colspan="4">rw</td></tr>
</table>

| 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 |

### 20.4.2. 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)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">YRT[3:0]</td><td colspan="4">YRU[3:0]</td></tr>
<tr><td colspan="8"></td><td colspan="4">rw</td><td colspan="4">rw</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">DOW[2:0]</td><td>MONT</td><td colspan="4">MONU[3:0]</td><td colspan="2">Reserved</td><td colspan="2">DAYT[1:0]</td><td colspan="4">DAYU[3:0]</td></tr>
<tr><td colspan="3">rw</td><td>rw</td><td colspan="4">rw</td><td colspan="2"></td><td colspan="2">rw</td><td colspan="4">rw</td></tr>
</table>

| 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[2:0] | Month units in BCD code |
| 7:6 | Reserved | Must be kept at reset value |
| 5:4 | DAYT[1:0] | Date tens in BCD code |
| 3:0 | DAYU[3:0] | Date units in BCD code |

## 20.4.3. Control register (RTC_CTL)

Address offset: 0x08

System reset: not affected

Backup domain reset value: 0x0000 0000

This register is write protected

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Reserved</td><td>COEN</td><td colspan="2">OS[1:0]</td><td>OPOL</td><td>COS</td><td>DSM</td><td>S1H</td><td>A1H</td></tr>
<tr><td colspan="8"></td><td>rw</td><td colspan="2">rw</td><td>rw</td><td>rw</td><td>rw</td><td>w</td><td>w</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>TSIE</td><td colspan="2">Reserved</td><td>ALRM0IE</td><td>TSEN</td><td colspan="2">Reserved</td><td>ALRM0EN</td><td>Reserved</td><td>CS</td><td>BPSHAD</td><td>REFEN</td><td>TSEG</td><td colspan="3">Reserved</td></tr>
<tr><td>rw</td><td colspan="2"></td><td>rw</td><td>rw</td><td colspan="2"></td><td>rw</td><td></td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td colspan="3"></td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23 | COEN | Calibration output enable<br>0: Disable calibration output |

| | | |
|---|---|---|
| | | 1: Enable calibration output |
| 22:21 | OS[1:0] | Output selection<br>This bit is used for selecting flag source of RTC_ALARM output<br>0x0: Disable output RTC_ALARM<br>0x1: Enable alarm 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 adjust. |
| 17 | S1H | Subtract 1 hour(winter time change)<br>One hour will be subtracted from now time if now hour time 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 now 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 |

| | | |
|---|---|---|
| | | 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 |
| 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 |

### 20.4.4. Status register (RTC_STAT)

Address offset: 0x0C
System reset: Only INITM, INITF and RSYNF bit are set to 0. others are not affected
Backup domain reset value: 0x0000 0007

This register is write 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | TP1F | TP0F | TSOVRF | TSF | Reserved | | ALRM0F | INITM | INITF | RSYNF | YCM | SOPF | Res. | | ALRM0WF |
| | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rw | r | rc_w0 | r | rc_w0 | | | 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 calibration configuration is taken into account. |
| 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 now time/date matches the time/date of alarm 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 and calendar register 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 write after ALRM0EN bit has reset.<br>0: Alarm registers programming is not allowed<br>1: Alarm registers programming is allowed |

### 20.4.5. Time prescaler register (RTC_PSC)

Address offset: 0x10

System reset: not effected

Backup domain reset value: 0x007F 00FF

This register can only be wrote in initialization mode.

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | 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) |

### 20.4.6. Alarm 0 Time and date register (RTC_ALRM0TD)

Address offset : 0x1C

System reset : not effected

Backup domain 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSKD | DOWS | DAYT[1:0] | | DAYU[3:0] | | | | MSKH | PM | HRT[1:0] | | HRU[3:0] | | | |
| | | 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[3:0] 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 |

### 20.4.7. Write protection key register (RTC_WPK)

Address offset : 0x24

Reset value : 0x0000 0000

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

| 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 | | | | | | | | WPK[7:0] | | | | | | | |
| | | | | | | | | w | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | WPK[7:0] | Key for write protection |

### 20.4.8. 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)

| 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. Second fraction value is calculated by the below formula:<br>Second fraction = ( FACTOR_S - SSC ) / ( FACTOR_S + 1 ) |

### 20.4.9. Shift function control register (RTC_SHIFTCTL)

Address offset: 0x2C

Reset value: 0x0000 0000

This register 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 one second to the clock/calendar.<br>This bit is jointly use with SFS bit 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 just only use SFS, the clock will delay because the synchronous prescaler is a down counter:<br>Delay (seconds) = SFS / ( FACTOR_S + 1 )<br>When jointly use 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.

## 20.4.10. 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<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 |

## 20.4.11. 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)

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DOW[2:0] | | | MONT | MONU[3:0] | | | | Reserved | | DAYT[1:0] | | DAYU[3:0] | | | |
| r | | | r | r | | | | | | r | | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:13 | DOW[2:0] | Week day units |
| 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] | Date tens in BCD code |
| 3:0 | DAYU[3:0] | Date units in BCD code |

### 20.4.12. 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. |

### 20.4.13. High resolution frequency compensation registor (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 equivalently 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. |

### 20.4.14. 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: PC15 is controlled by GPIO configuration.<br>1: PC15 is forced to push-pull output if LXTALEN=0. |
| 22 | PC15VAL | PC15 value<br>PC15 outputs this value when PC15MDE=1 and LXTALEN=0. |
| 21 | PC14MDE | PC14 mode<br>0: PC14 is controlled by GPIO configuration.<br>1: PC14 is forced to push-pull output if LXTALEN=0. |
| 20 | PC14VAL | PC14 value<br>PC14 outputs this value when PC14MODE=1 and LXTALEN=0. |
| 19 | PC13MDE | 0: PC13 is controlled by GPIO configuration.<br>1: PC13 is forced to push-pull output if all RTC alternate functions are disabled.. |
| 18 | PC13VAL | Alarm output type control/PC13 output value<br>When RTC alternate function enabled, output RTC_ALARM:<br>0: RTC_ALARM output is open-drain mode<br>1: RTC_ALARM output is push-pull mode<br>When all RTC alternate function are disabled and PC13MODE=1,this bit control the PC13 output value |
| 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 precharge RTC_TAMPx pin<br>1:Disable precharge duration |
| 14:13 | PRCH[1:0] | Precharge duration time of RTC_TAMPx<br>This setting determines the prechagre 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. Precharge duration is disabled automatically<br>0x1: Detecting tamper event using level mode.2 consecutive valid level samples will make a 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 a effective tamper event |

| | | |
|---|---|---|
| 10:8 | FREQ[2:0] | Sample 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 for RTC_TAMP1 input<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 for RTC_TAMP0 input<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 TAMPxE before change the tamper configuration.

### 20.4.15. 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>0x10: SSC[9:0] is to be compared and all others are ignored<br>0x11: SSC[10:0] is to be compared and all others are ignored<br>0x12: SSC[11:0] is to be compared and all others are ignored<br>0x13: SSC[12:0] is to be compared and all others are ignored<br>0x14: SSC[13:0] is to be compared and all others are ignored<br>0x15: 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 bit. |

### 20.4.16. Backup register (RTC_BKPx)(x=0,1,2,3,4)

Address offset: 0x50 to 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] | Backup domain registers.<br>These registers can be wrote or read by software. The content remains valid even in power saving mode because they can powered-on by $V_{BAT}$. Tamper detection flag TPxF assertion or disable Flash readout protection will reset these registers. |

# 21. Touch sensing interface(TSI)

## 21.1. Introduction

Touch Sensing Interface (TSI) provides a convenient solution for touch keys, sliders and capacitive proximity sensing applications. The controller builds on charge transfer method. Placing a finger near fringing electric fields adds capacitance to the system and TSI is able to measure this capacitance change using charge transfer method.

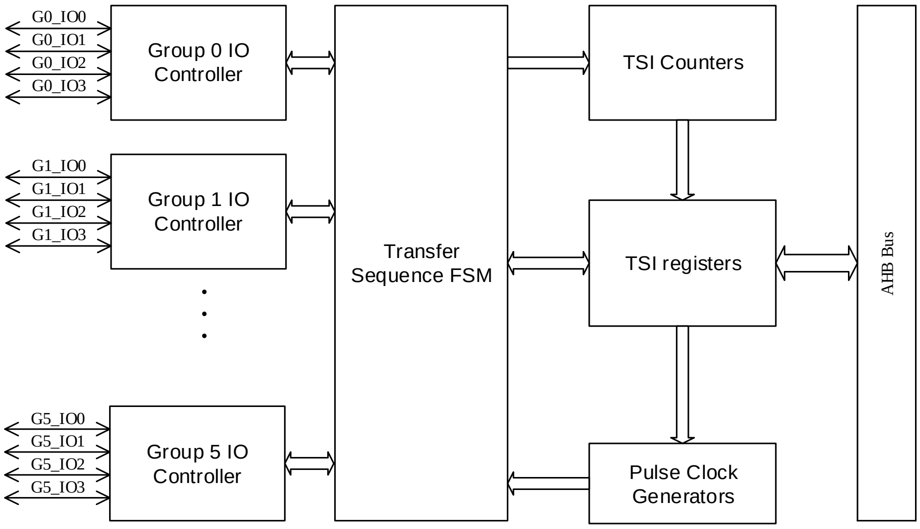
## 21.2. Main features

- Transfer sequence fully controlled by hardware
- 6 fully parallel groups implemented
- 18 IOs configurable for capacitive sensing Channel Pins and 6 for Sample Pins
- Configurable transfer sequence frequency
- Possible to implement the user specific transfer sequences
- Sequence end and error flags / configurable interrupts
- Spread spectrum function implemented

## 21.3. Function description

### 21.3.1. TSI block diagram

**Figure 21-1. Block diagram of TSI module**

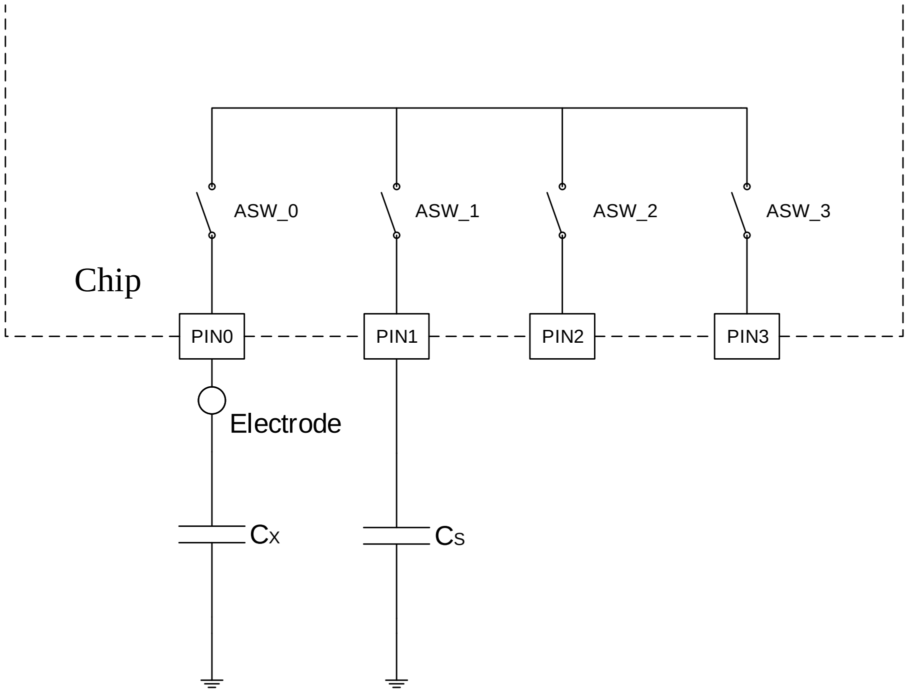


### 21.3.2. Touch sensing technique overview

There are different technologies for touch sensing, such as optical, resistive, capacitive, strain, etc. Detecting the change of a system is the key problem and goal in these technologies. The TSI module is designed to use charge transfer method which detects the capacitive change of an electrode when touched by or a finger close to it. In order to detect the capacitive change, TSI performs a charge transfer sequence including several charging, transfer steps. The number of these steps indicates the capacitance of an electrode. So the application is able to detect the change of capacitance by monitoring the step number of each transfer sequence.

As shown in Figure 21-2, there are 4 PINs in one group and each PIN has an analog switch connected to a common point which is the key component to implement charge transfer. There should be a sample pin and one or more channel pin(s) configured in one group. In Figure 21-2, PIN0 is a channel pin and PIN1 is a sample pin while PIN2 and PIN3 are unused. An electrode connecting PIN0 is designed on PCB board. The A sample capacitor $C_S$ connected to sample pin PIN1 is also required. Now the capacitance of the channel pin PIN0 includes $C_x$ and the capacitance introduced by the electrode, so capacitance of PIN0 increases when a finger is touching while the capacitance of PIN1 remains unchanged. Thus, the finger's touching can be detected if the capacitance of PIN0 can be measured. In TSI module, a charge-transfer sequence is performed to measure the capacitance of the channel pin(s) in a group, which will be detailed in next section.

**Figure 21-2. Block diagram of Sample pin and Channel Pin**



## 21.3.3. Charge transfer sequence

In order to measure the capacitance of a channel pin, charge transfer sequence is performed in chip. The sequence shown in Table 21-1 is described based on the connection of Figure 21-2, i.e. PIN0 is channel pin and PIN1 is sample pin.

Table 21-1. Pin and analog switch state in a charge-transfer sequence

| Step | Name | ASW_0 | ASW_1 | Pin0 | Pin1 |
|---|---|---|---|---|---|
| 1 | Discharge | Close | Close | Input Floating | Pull Down |
| 2 | Buffer Time1 | Open | Open | Input Floating | Input Floating |
| 3 | Charge | Open | Open | Output High | Input Floating |
| 4 | Extend Charge | Open | Open | Output High | Input Floating |
| 5 | Buffer Time2 | Open | Open | Input Floating | Input Floating |
| 6 | Charge Transfer | Close | Close | Input Floating | Input Floating |
| 7 | Buffer Time3 | Open | Open | Input Floating | Input Floating |
| 8 | Compare | Open | Open | Input Floating | Input Floating |

### 1. Discharge

Both $C_x$ and $C_s$ are discharged by closing ASW_0 and ASW_1 and configuring PIN1 to pull down. This step is the initial operation for a correct charge transfer sequence and is performed by software before starting a charge transfer sequence. Discharging time in this step should be guaranteed to ensure that the voltage of $C_x$ and $C_s$ are discharged to zero.

### 2. Buffer Time1

Buffer time with ASW_0 and ASW_1 open, PIN0 is configured to input floating.

### 3. Charge

Channel pin PIN0 is configured to output high, in order to charge $C_x$ . ASW_0 and ASW_1 remain open during this step. The charging time should be configured (see Register Section for detail) to ensure that the voltage of $C_x$ is charged to $V_{DD}$.

### 4. Extend Charge

This is an optional step in a charge-transfer sequence and the behavior of all pins and analog switches in this step is the same as Step 3. The only difference between this and step 3 is the duration time, which is configurable in TSI registers. The duration of this step changes in each loop of a charge-transfer sequence, spreading the spectrum.

### 5. Buffer Time2

Buffer time with ASW_0 and ASW_1 open, PIN0 is configured to input floating.

### 6. Charge transfer

ASW_0 and ASW_1 are closed and PIN0 is configured to input floating to transfer charge from $C_x$ to $C_s$. The transfer time should be configured (see Register Section for detail) to ensure the full transfer after that the voltage of $C_x$ and $C_s$ will be equal.

### 7. Buffer Time3

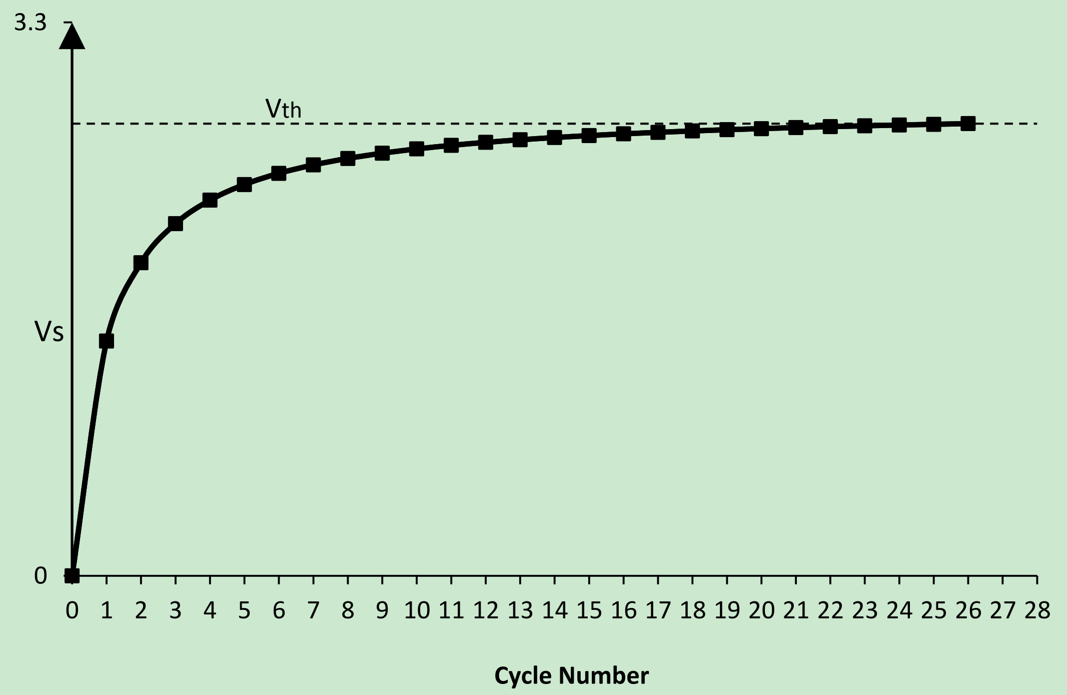
Buffer time with ASW_0 and ASW_1 open, PIN0 is configured to input floating.

### 8. Compare

ASW_0, ASW_1 and PIN0 remain the configuration of Step7. At this step, the voltage of sample pin PIN1 is compared to a threshold called $V_{th}$. If voltage of PIN1 is lower than $V_{th}$, the sequence returns to Step2 and continues, otherwise, the sequence ends.

The voltage of sample pin $V_s$ is zero after initial step and increases after each charge cycle, as shown in Figure 21-3. A larger $C_x$ will cause a greater increase during a cycle. The sequence stops when $V_s$ reaches $V_{th}$. Each group has a counter which records the number of cycles performed on it to reach $V_{th}$. At the end of charge-transfer sequence, the group counter is read out to estimate the $C_x$, i.e. a smaller counter values indicates a larger $C_x$.

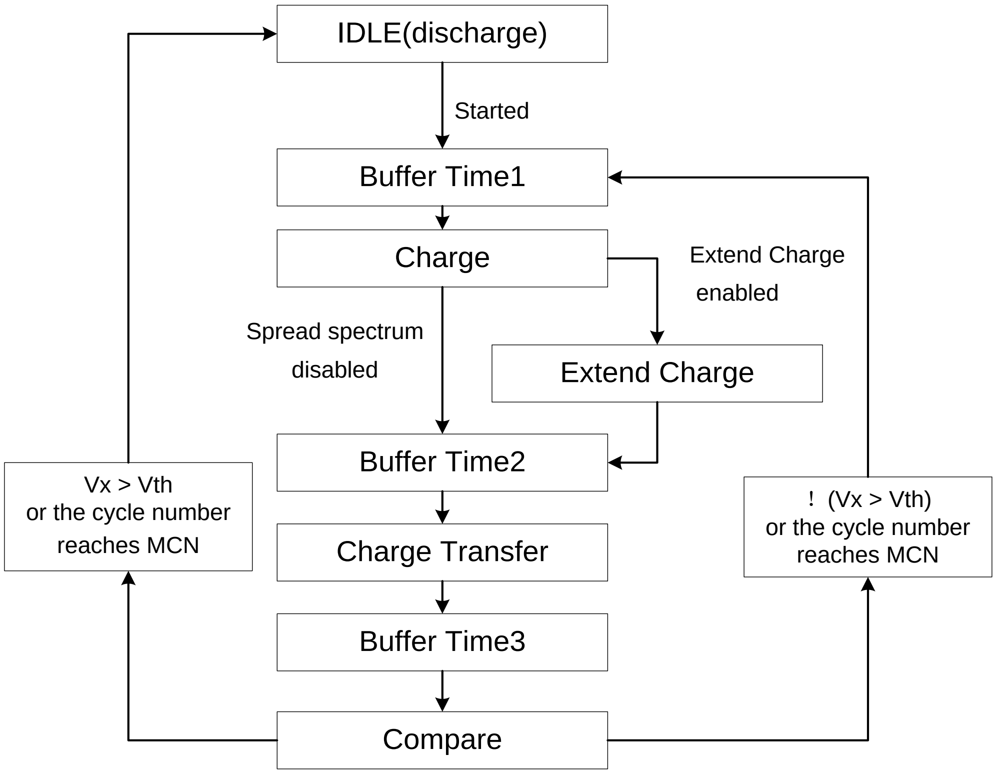
**Figure 21-3. Voltage of a sample pin during charge-transfer sequence**



### 21.3.4. Charge transfer sequence FSM

A hardware FSM is designed in chip to perform the charge transfer sequence described in the previous section as shown in Figure 21-4.

**Figure 21-4. FSM flow of a charge-transfer sequence**



This FSM remains in IDLE state after reset. There are 2 kinds of start condition defined by TRGMOD bit in TSI_CTL register:

**TRGMOD = 0:** Software Trigger Mode. In this mode, the FSM starts after TSIS bit in TSI_CTL register is written 1 by software.

**TRGMOD = 1:** Hardware Trigger Mode. In this mode the FSM starts when a falling or rising edge on TSITG pin is detected.

Once started, the FSM runs following the flow described in Figure 21-4. The FSM leaves a state if the duration time of this state reaches defined value, and goes into the next state.

The Extend Charge state is present only if the ECEN bit is set in TSI_CTL register. This state is designed to implement spread spectrum function.

In comparing sate, the FSM compares voltage of the sample pin in every enabled group and the threshold voltage. If all sample pins' voltage reach the threshold, FSM returns IDLE state and stops, otherwise, it returns to Buffer Time 1 state and begins the next cycle.

As shown in Figure 21-4, after 27 cycles, $V_s$ (the voltage of sample pin) reaches $V_{th}$ (the threshold voltage).

There is also a max cycle number defined by MCN in TSI_CTL register. When the cycle number reaches MCN, FSM returns to IDLE state and stops after Compare State, whether $V_s$ reaches $V_{th}$ or not.

### 21.3.5. Clock and duration time of states

There are 3 clocks in TSI module: HCLK, CTCLK and ECCLK. HCLK is system clock and drives TSI's register and FSM. CTCLK, which is divided from HCLK with division factor defined by register CTCDIV is the clock used for calculating the duration time of the charge state and Charge Transfer state. ECCLK, which is divided from HCLK with division factor defined by register ECCDIV is the clock used to calculate the maximum duration time of Extend Charge state.

The duration time of each state except state Extend Charge state is fixed in each loop according to the configuration of the register.

The duration time of Buffer Time1, Buffer Time2 and Buffer Time3 are fixed to 2 HCLK periods. The duration time of Charge state and Charge Transfer state is defined by CDT and CTDT bits (see TSC_CTL register section for detail).

The duration time of Extend Charge state changes in each cycle of the charge-transfer FSM and the maximum duration time are defined by ECDT[6:0] in TSI_CTL register.

The duration time of Extend Charge state in each cycle is presented in Table 21-2.

**Table 21-2. Duration time of Extend Charge state in each cycle**

| Cycle Number | Number of ECCLKs in Extend Charge state |
|---|---|
| 1 | 0 |
| 2 | 1 |
| … | |
| ECDT | ECDT-1 |
| ECDT+1 | ECDT |
| ECDT+2 | ECDT+1 |
| ECDT+3 | ECDT |
| ECDT+4 | ECDT-1 |
| … | … |
| 2*ECDT+1 | 2 |
| 2*ECDT+2 | 1 |
| 2*ECDT+3 | 0 |
| 2*ECDT+4 | 1 |
| 2*ECDT+5 | 2 |
| … | … |

### 21.3.6. PIN mode control of TSI

There are 4 pins in each group and each of these pins is able to be used as a sample pin or channel pin. Only one pin in a group should be configured as sample pin, and channel pins can be more than one. The sample pin and channel pin(s) should not be configured as the same pin in any case.

When a PIN is configured in GPIO (see chapter GPIO) used by TSI, the pin's mode is controlled by TSI. Generally, each pin has 3 modes: input, output high and output low.

The mode of a channel pin or a sample pin during a charge-transfer sequence is described in Table 21-1 in which PIN0 represents a channel pin and PIN1 represents a sample pin, i.e. the charge-transfer FSM take control of these channels or sample pins' mode and the states of related analog switches when the sequence is on-going. When the sequence is in IDLE state, PINMOD bit in TSI_CTL register defines the mode of these pins. Pins that are configured in GPIO used by TSI but neither sample nor channel in TSI register is called free pins whose mode is defined by PINMOD bit in TSI_CTL, too.

### 21.3.7. ASW and hysteresis mode

A channel or sample pin's analog switch is controlled by charge-transfer sequence when FSM is running, as shown in Table 21-1. When the FSM is IDLE, these pins' analog switches are controlled by GxPy bits in TSI_ASW register. All free pin's analog switches are controlled by GxPy bits too.

TSI takes control of the analog switches when FSM is IDLE, even if these pins are not configured to be used by TSI in GPIO. The user is able to perform user-defined charge-transfer sequence by writing GxPy bits to control these analog switches, while controlling pin mode directly in GPIO.

Each TSI pin's hysteresis mode is configurable by GxPy bit in TSI_PHM register.

### 21.3.8. TSI operation flow

The normal operation flow of TSI is listed below:

System initialization, such as system clock configuration, TSI related GPIO configuration, etc.

Program TSI_CTL, TSI_CHCFG, TSI_INTEN, TSI_SAMPCFG and GEx bits of TSI_GCTL register according to demand.

Enable TSI by setting TSIEN bit in TSI_CTL register.

Optional for software trigger mode: program TSIS bit to start charging transfer sequence. In hardware trigger mode, TSI is started by falling/rising edge on the trigger pin.

Wait for the CTCF or MNERR flag in TSI_INTF and clear these flags by writing TSI_INTC.

Read out the CYCN bits in TSI_GxCYCN registers.

### 21.3.9. TSI flags and interrupts

**Table 21-3. TSI errors and flags**

| Flag Name | Description | Cleared by |
|---|---|---|

| | | |
|---|---|---|
| CTCF | TSI stops because all enabled samplers' sample pins reach $V_{th}$. | CCTCF bit in TSI_INTC |
| MNERR | TSI stops because the cycle number reaches the maximum value. | CMNERR bit in TSI_INTC |

### 21.3.10. TSI GPIOs

**Table 21-4. TSI pins**

| TSI Group | TSI Pins | GPPIN pins |
|---|---|---|
| TSI_GRP0 | PIN0 | PA0 |
| | PIN1 | PA1 |
| | PIN2 | PA2 |
| | PIN3 | PA3 |
| TSI_GRP1 | PIN0 | PA4 |
| | PIN1 | PA5 |
| | PIN2 | PA6 |
| | PIN3 | PA7 |
| TSI_GRP2 | PIN0 | PC5 |
| | PIN1 | PB0 |
| | PIN2 | PB1 |
| | PIN3 | PB2 |
| TSI_GRP3 | PIN0 | PA9 |
| | PIN1 | PA10 |
| | PIN2 | PA11 |
| | PIN3 | PA12 |
| TSI_GRP4 | PIN0 | PB3 |
| | PIN1 | PB4 |
| | PIN2 | PB6 |
| | PIN3 | PB7 |
| TSI_GRP5 | PIN0 | PB11 |
| | PIN1 | PB12 |
| | PIN2 | PB13 |
| | PIN3 | PB14 |

## 21.4. TSI registers

### 21.4.1. Control register (TSI_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CDT[3:0] | | | | CTDT[3:0] | | | | ECDT[6:0] | | | | | | | ECEN |
| rw | | | | rw | | | | rw | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECDIV | CTCDIV[2:0] | | | Reserved | | | | MCN[2:0] | | | PINMOD | EGSEL | TRGMOD | TSIS | TSIEN |
| rw | rw | | | | | | | rw | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | CDT[3:0] | Charge State Duration Time<br>CDT[3:0] is set and clear by software. These bits controls the duration time of Charge State in a charge-transfer sequence.<br>0000: $1 \times t_{CTCLK}$<br>0001: $2 \times t_{CTCLK}$<br>0010: $3 \times t_{CTCLK}$<br>….<br>1111: $16 \times t_{CTCLK}$ |
| 27:24 | CTDT[3:0] | Charge Transfer State Duration Time<br>CTDT[3:0] is set and clear by software. These bits control the duration time of Charge Transfer State in a charge-transfer sequence.<br>0000: $1 \times t_{CTCLK}$<br>0001: $2 \times t_{CTCLK}$<br>0010: $3 \times t_{CTCLK}$<br>….<br>1111: $16 \times t_{CTCLK}$ |
| 23:17 | ECDT[6:0] | Extend Charge State Maximum Duration Time<br>ECDT[6:0] is set and clear by software. These bits control the maximum Duration Time duration time of Extend Charge Transfer State in a charge-transfer sequence. Extend Charge State is only present when ECEN bit in TSI_CTL register is set.<br>0000000: $1 \times t_{ECCLK}$<br>0000001: $2 \times t_{ECCLK}$<br>0000010: $3 \times t_{ECCLK}$<br>….<br>1111111: $128 \times t_{ECCLK}$ |
| 16 | ECEN | Extend Charge State Enable.<br>0: Extend Charge disabled<br>1: Extend Charge enabled |
| 15 | ECDIV | ECCLK clock division factor.<br>ECCLK in TSI is divided from HCLK and ECDIV defines the division factor. |

| | | |
|---|---|---|
| | | 0: $\mathrm{f_{ECCLK}} = \mathrm{f_{HCLK}}$<br>1: $\mathrm{f_{ECCLK}} = 0.5\mathrm{f_{HCLK}}$ |
| 14:12 | CTCDIV[2:0] | CTCLK clock division factor.<br>CTCLK in TSI is divided from HCLK and CTCDIV defines the division factor.<br>000: $\mathrm{f_{CTCLK}} = \mathrm{f_{HCLK}}$<br>001: $\mathrm{f_{CTCLK}} = \mathrm{f_{HCLK}}/2$<br>010: $\mathrm{f_{CTCLK}} = \mathrm{f_{HCLK}}/4$<br>011: $\mathrm{f_{CTCLK}} = \mathrm{f_{HCLK}}/8$<br>….<br>111: $\mathrm{f_{CTCLK}} = \mathrm{f_{HCLK}}/128$ |
| 11:8 | Reserved | Must be kept at reset value |
| 7:5 | MCN[2:0] | Max cycle number of a sequence<br>MCN [2:0] defines the max cycle number of a charge-transfer sequence FSM which stops after reaching this number.<br>000: 255<br>001: 511<br>010: 1023<br>011: 2047<br>100: 4095<br>101: 8191<br>110: 16383<br>111: Reserved |
| 4 | PINMOD | Pin mode<br>This bit defines a TSI pin’s mode when charge-transfer sequence is IDLE.<br>0: TSI pin will output low when IDLE<br>1: TSI pin will keep input mode when IDLE |
| 3 | EGSEL | Edge selection<br>This bit defines the edge type in hardware trigger mode.<br>0: Falling edge<br>1: Rising edge |
| 2 | TRGMOD | Trigger mode selection<br>0: Software Trigger Mode, sequence will start after TSIS bit is set.<br>1: Hardware Trigger Mode, sequence will start after a falling/rising edge on trigger pin detected. |
| 1 | TSIS | TSI start<br>This bit is set by software to start a charge-transfer sequence in software trigger mode and reset by hardware when the sequence stops. After setting this bit, software can reset it to stop the started sequence manually.<br>0: TSI is not started<br>1: TSI is started. |

| | | |
|---|---|---|
| 0 | TSIEN | TSI enable<br>0: TSI module is enabled<br>1: TSI module is disabled |

### 21.4.2. Interrupt enable register (TSI_INTEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register can 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 | | | | | | | | | | | | | | MNERRIE | CTCFIE |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | MNERRIE | Max Cycle Number Error Interrupt Enable<br>0: MNERR interrupt is disabled<br>1: MNERR interrupt is enabled |
| 0 | CTCFIE | Charge-transfer complete flag Interrupt Enable<br>0: CTCF interrupt is enabled<br>1: CTCF interrupt is disabled |

### 21.4.3. Interrupt flag clear register (TSI_INTC)

Address offset: 0x08

Reset value: 0x0000 0000

This register can 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 | | | | | | | | | | | | | | CMNERR | CCTCF |
| | | | | | | | | | | | | | | w | w |

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

| | | |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | CMNERR | Clear max cycle number error<br>0: Reserved<br>1: Clear MNERR |
| 0 | CCTCF | Clear charge-transfer complete flag<br>0: Reserved<br>1: Clear CTCF |

## 21.4.4. Interrupt flag register (TSI_INTF)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can 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 | | | | | | | | | | | | | | MNERR | CTCF |
| | | | | | | | | | | | | | | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | MNERR | Max Cycle Number Error<br>This bit is set by hardware after charge-transfer sequence stops because it reaches the max cycle number defined by MCN[2:0]. This bit is cleared by writing 1 to CMNERR bit in TSI_INTC register.<br>0: No Max Count Error<br>1: Max Count Error |
| 0 | CTCF | Charge-Transfer complete flag<br>This bit is set by hardware after charge-transfer sequence stops because all enabled group’s sample pins reach the threshold voltage or because the cycle number reaches the value defined by MCN[2:0]. This bit is cleared by writing 1 to CCTCF bit in TSI_INTC register.<br>0: Charge-Transfer not complete<br>1: Charge-Transfer complete |

## 21.4.5. Pin hysteresis mode register (TSI_PHM)

Address offset: 0x10

Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | G5P3 | G5P2 | G5P1 | G5P0 | G4P3 | G4P2 | G4P1 | G4P0 |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G3P3 | G3P2 | G3P1 | G3P0 | G2P3 | G2P2 | G2P1 | G2P0 | G1P3 | G1P2 | G1P1 | G1P0 | G0P3 | G0P2 | G0P1 | G0P0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:0 | GxPy | Pin hysteresis mode<br>This bit is set and cleared by software.<br>0: Pin GxPy Schmitt trigger hysteresis mode disabled<br>1: Pin GxPy Schmitt trigger hysteresis mode enabled |

### 21.4.6. Analog switch register (TSI_ASW)

Address offset: 0x18

Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | G5P3 | G5P2 | G5P1 | G5P0 | G4P3 | G4P2 | G4P1 | G4P0 |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G3P3 | G3P2 | G3P1 | G3P0 | G2P3 | G2P2 | G2P1 | G2P0 | G1P3 | G1P2 | G1P1 | G1P0 | G0P3 | G0P2 | G0P1 | G0P0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:0 | GxPy | Analog switch state.<br>This bit is set and cleared by software.<br>0: Analog switch of GxPy is open<br>1: Analog switch of GxPy is closed |

### 21.4.7. Sample configuration register (TSI_SAMPCFG)

Address offset: 0x20

Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | G5P3 | G5P2 | G5P1 | G5P0 | G4P3 | G4P2 | G4P1 | G4P0 |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G3P3 | G3P2 | G3P1 | G3P0 | G2P3 | G2P2 | G2P1 | G2P0 | G1P3 | G1P2 | G1P1 | G1P0 | G0P3 | G0P2 | G0P1 | G0P0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:0 | GxPy | Sample pin mode<br>This bit is set and cleared by software.<br>0: Pin GxPy is not a sample pin<br>1: Pin GxPy is a sample pin |

### 21.4.8. Channel configuration register (TSI_CHCFG)

Address offset: 0x28

Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | G5P3 | G5P2 | G5P1 | G5P0 | G4P3 | G4P2 | G4P1 | G4P0 |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G3P3 | G3P2 | G3P1 | G3P0 | G2P3 | G2P2 | G2P1 | G2P0 | G1P3 | G1P2 | G1P1 | G1P0 | G0P3 | G0P2 | G0P1 | G0P0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 23:0 | GxPy | Channel pin mode<br>This bit is set and cleared by software.<br>0: Pin GxPy is not a channel pin<br>1: Pin GxPy is a channel pin |

### 21.4.9. Group control register (TSI_GCTL)

Address offset: 0x30

Reset value: 0x0000 0000

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | GC5 | GC4 | GC3 | GC2 | GC1 | GC0 |
| | | | | | | | | | | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | GE5 | GE4 | GE3 | GE2 | GE1 | GE0 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value |
| 21:16 | GCx | Group complete<br>This bit is set by hardware when charge-transfer sequence for an enabled group is complete. It is cleared by hardware when a new charge-transfer sequence starts.<br>0: Charge-transfer for group x is not complete<br>1: Charge-transfer for group x is complete |
| 15:6 | Reserved | Must be kept at reset value |
| 5:0 | GEx | Group enable<br>This bit is set and cleared by software.<br>0: Group x is disabled<br>1: Group x is enabled |

### 21.4.10. Group x cycle number registers (TSI_GxCYCN) (x= 0..5)

Address offset: 0x30 + 0x04 *(x + 1)

Reset value: 0x0000 0000

This register can 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 | | CYCN[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13:0 | CYCN[13:0] | Cycle number<br>These bits reflect the cycle number for a group as soon as a charge-transfer sequence completes. They are cleared by hardware when a new charge-transfer sequence starts. |

# 22. HDMI-CEC controller(HDMI-CEC)

## 22.1. Introduction

Consumer Electronics Control (CEC) belongs to a part of HDMI (High-Definition Multimedia Interface) standard. CEC, as a kind of protocol, it provides the advanced control functions of all kinds of audio-visual products in a user environment. The CEC protocol gets hardware support from the HDMI-CEC controller.

## 22.2. Main features

- HDMI-CEC controller complies with HDMI-CEC v1.4 Specification
- Tow clock source options for 32.768KHz CEC clock:
  1) LXTAL oscillator
  2) IRC8M oscillator with settled prescaler (IRC8M/244)
- For ultra low-power applications ,HDMI-CEC controller can work in Deep-sleep mode
- Programmable SFT(Signal Free Time) value for arbitration priority:
  1) User configure
  2) Auto configure by controller as HDMI-CEC protocol specification
- Programmable own address(OADR)
- Listen mode supports user receiving messages on the CEC line but not disturb the CEC line.
- RX bit-tolerance function support for higher compatibility
- Bit Error Detection
  1) Short bit period error(RSBPE)
  2) Long bit period error(RLBPE)
  3) Bit rising error(RBRE)
- Programmable error-bit generation
  1) RSBPE detection will always generate error-bit
  2) RLBPE detection will generate error-bit if RLBPEGEN=1
  3) RBRE detection will generate error-bit if RBREGEN=1
- Transmission error detection (TERR)
- Transmission underrun detection (TU)
- Reception overrun detection (RO)
- Arbitration lost detection (LSTARB),if asserted controller will automatic retry transmission

## 22.3. Function description

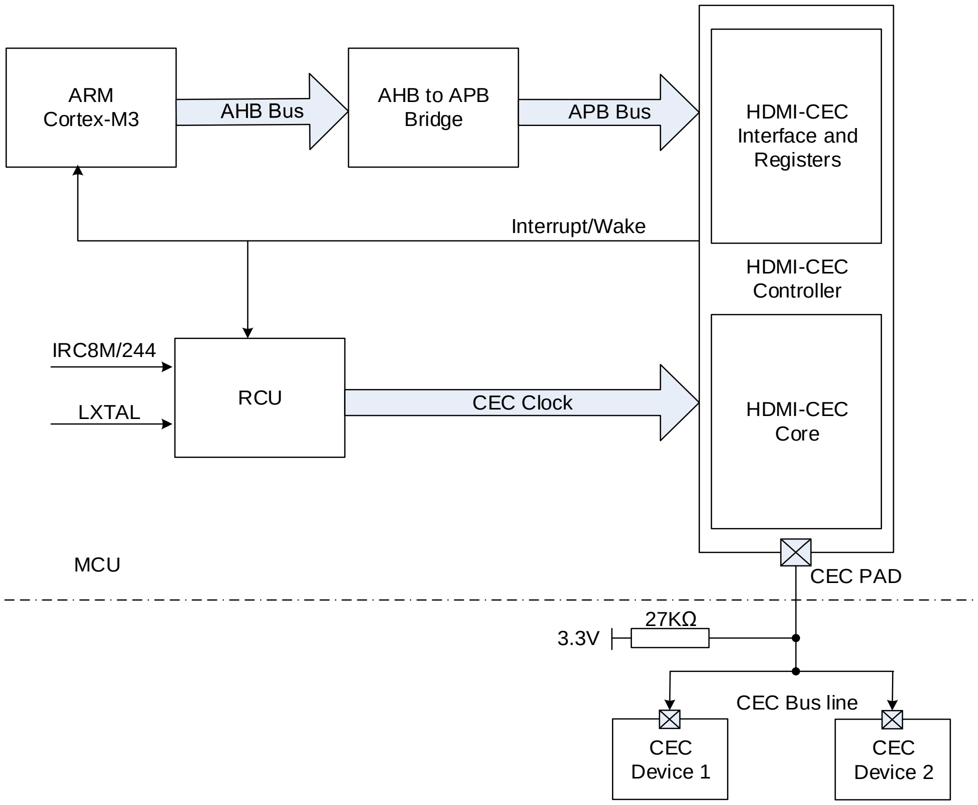
### 22.3.1. CEC bus pin

The CEC device only use one bidirectional line to connect to others. The CEC pin through a

27KΩ pull-up resister connected to a +3.3V supply voltage. When the CEC device is in the output state, in order to allow a wired-and connection it must have an open-drain or open-collector.

Using HDMI-CEC controller needs configuring CEC pin as alternate function open drain and a 27kΩ resister is also needed for pulling-up the CEC pin.

**Figure 22-1. HDMI-CEC Controller Block Diagram**



## 22.3.2. Message description

A complete message includes one or more frames and the message structure is shown as below:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bus<br>High | START<br>Bit | Header<br>Frame | Data<br>Frame | .... | Data<br>Frame | Data<br>Frame | Bus<br>High |

The frame has two types:

1) **Header frame**: The first frame in the message which followed the start-bit, it consists of the source logical address field and the destination logical address field.

2) **Data frame**: All frames in the message except header frame. All frames are ten bits long and have the same basic structure as shown below:

| Frame Structure | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |

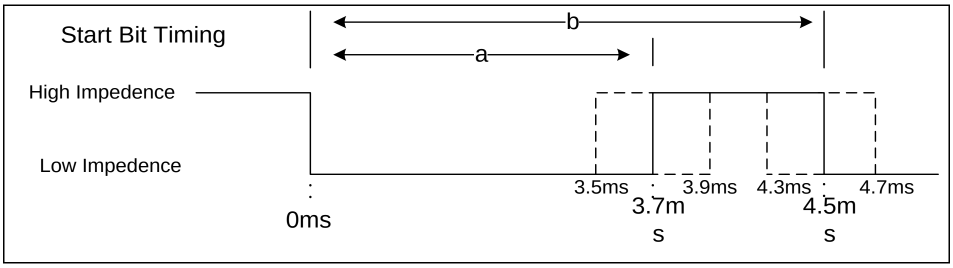
| | | |
|---|---|---|
| Information bits | ENDOM | ACK |

The information bits are data, opcodes or addresses, dependent on context. The control bits, ENDOM and ACK, are always present and always have the same usage.

### 22.3.3. Bit timing description

All bits timing in the message are divided into two types: Start bit and Data bit.

1) Start Bit: The start bit has to be validated by its low duration(a) and its total duration(b) showed as below:



2) Data Bit: The valid data bit timing is constrained as below:

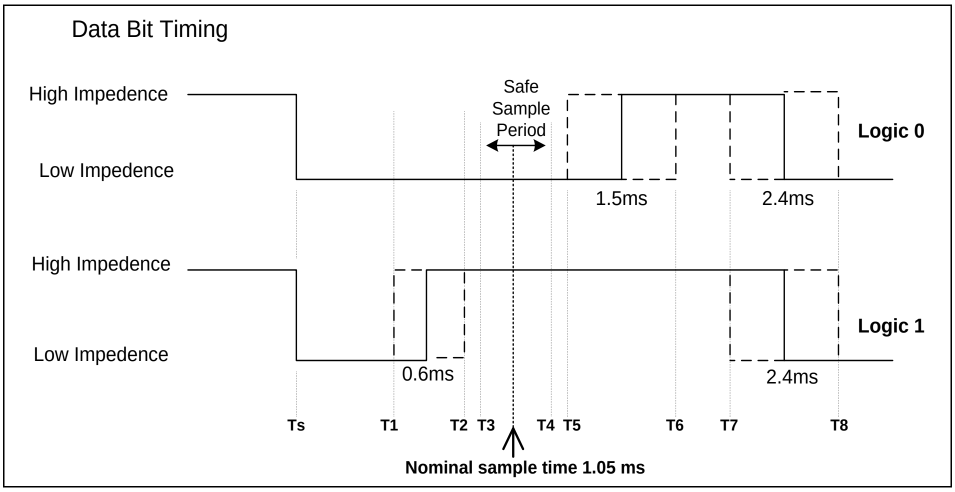


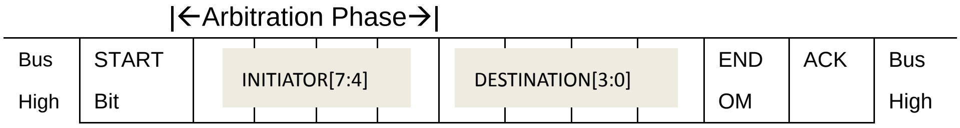
**Table 22-1. Data Bit Timing Parameter Table**

| Ts | Time (ms) | The bit start event. |
|---|---|---|
| T1 | 0.4ms | When indicating a logical 1,T1 as the earliest time for a low - high transition. |
| T2 | 0.8ms | When indicating a logical 1,T2 as the latest time for a low - high transition. |
| T3 | 0.85ms | The earliest time it is safe to sample the signal line to determine its state. |
| T4 | 1.25ms | The latest time it is safe to sample the signal line to determine its state. |
| T5 | 1.3ms | T5 as the earliest time that a device is allowed to return to a high impedance state(logical 0). |

| Ts | Time (ms) | The bit start event. |
|---|---|---|
| T6 | 1.7ms | T6 as the latest time that a device is allowed to return to a high impedance state(logical 0). |
| T7 | 2.05ms | T7 as the earliest time for the start of a following bit. |
|  | 2.4ms | As a nominal data bit period. |
| T8 | 2.75ms | T8 as the latest time for the start of a following bit. |

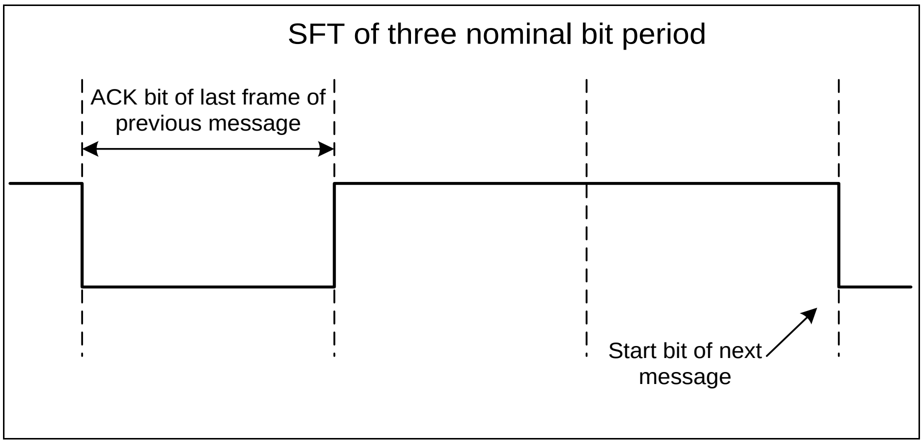
### 22.3.4. Arbitration

CEC line arbitration starts from the front edge of the start bit to the end of the Initiator address bits among the Header Block. In the meantime the Initiator should monitors the CEC line.During this period , if low impedance be detected when whilst in high impedance state then it should assume that it has lost the arbitration to a second Initiator.



Before attempting to transmit or re-transmit a frame, a device shall ensure that the CEC line has been inactive for a number of bit periods.

This signal free time is defined as the time since the start of the final bit of the previous frame.


SFT of three nominal bit period
ACK bit of last frame of previous message
Start bit of next message

The length of the required signal free time depends on the current status of the control signal line and the initiating device. If SFT=0x0, the HDMI-CEC controller's SFT will perform as table below:

| Precondition | Signal Free Time (nominal data bit periods) |
|---|---|
| Present Initiator wants to send another message immediately after its previous message | ≥7 |
| New Initiator wants to send a message | ≥5 |
| Previous attempt to send message unsuccessful | ≥3 |

This means that there is an opportunity for other devices to gain access to the CEC line during the periods mentioned above to send their own messages after the current device has finished sending its current message.

If SFT is not 0x0,the corresponding user configure SFT will be performed.

### 22.3.5. SFT option bit description

SFT option bit support another way for saving bus inactive time through setting more SFT counter's start time point.

When SFTOPT = 0, the SFT timer will start at the time SOM bit asserted when the controller is in the idle state.

When SFTOPT = 1, the SFT timer will start at the time CEC bus is in idle state and the SFT time will be saved if you configure the SOM after SFT done because the controller will start transmit without any latency.

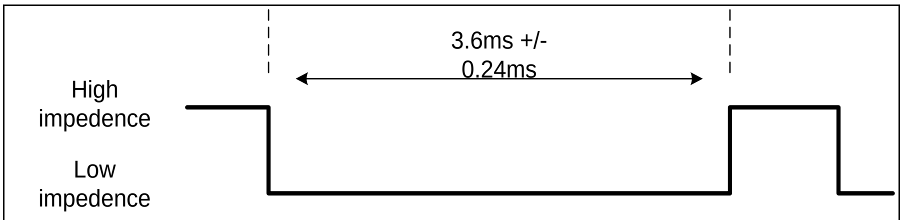
When SFTOPT = 1, some other cases will also start the SFT counter:

- In case of regular TX/RX complete(TEND/REND asserted)
- In case of transmission not complete such as the time TERR,TAERR or TU asserted.
- In case of receiving progress, if some error detected and error bit is generated, the SFT timer will start when output error bit finished.

### 22.3.6. Error definition

#### Error-Bit

If some errors are occurred and corresponding generation configure is enabled the HDMI-CEC controller will generate an error bit on the CEC pin for indicating. Error bit period definition is shown as below:



Error-Bit Timing

#### Frame error

CEC protocol defines that each frame of message need the acknowledgement to confirm the communication is successful. For broadcast(destination address=0xF), the ACK bit should be logic 1 and for singlecast(destination address<0xF), the ACK bit should be logic 0,otherwise the frame error occurs(TAERR/RAE flag asserted).

Another frame error situation is that the CEC bus pin voltage is different from CEC pad when HDMI-CEC controller is under initiator state(TERR flag asserted).

### Bit rising error(RBRE)

RBRE flag can be asserted if rising edge detected during the RBRE checking window and RBRE flag will also generate CEC interrupt if the RBREIE=1.

If RBRESTP=1,the controller will stop receiving message and if RBREGEN=1 the error-bit will be generated.

If RBRESTP=1 in broadcast the RBRE flag asserted, the error bit will be generated to notify the initiator the error.If you do not want to generate the error bit for RBRE detection you can configure the RBREGEN=0 and BCNG=1.

**Note:** The RBRESTP=0 and RBREGEN=1 configuration must be avoided.

### Short bit period error(RSBPE)

RSBPE is set when the period of the neighboring falling edge is shorter than expected. RSBPE flag will also generate CEC interrupt if the RSBPEIE=1.

If the RSBPE flag asserted, the error bit will must be generated except one of the cases below:

1) BCNG = 1
2) LMEN = 1
3) Receiving broadcast

### Long bit period error(RLBPE)

RLBPE is set when the period of the neighboring falling edge is longer than expected. RLBPE flag will also generate CEC interrupt if the RLBPEIE=1.

When RLBPE asserted, controller will stop receiving message and generate error bit if in one of the cases below:

1) RLBPEGEN=1 in both singlecast and broadcast
2) BCNG=0 in broadcast

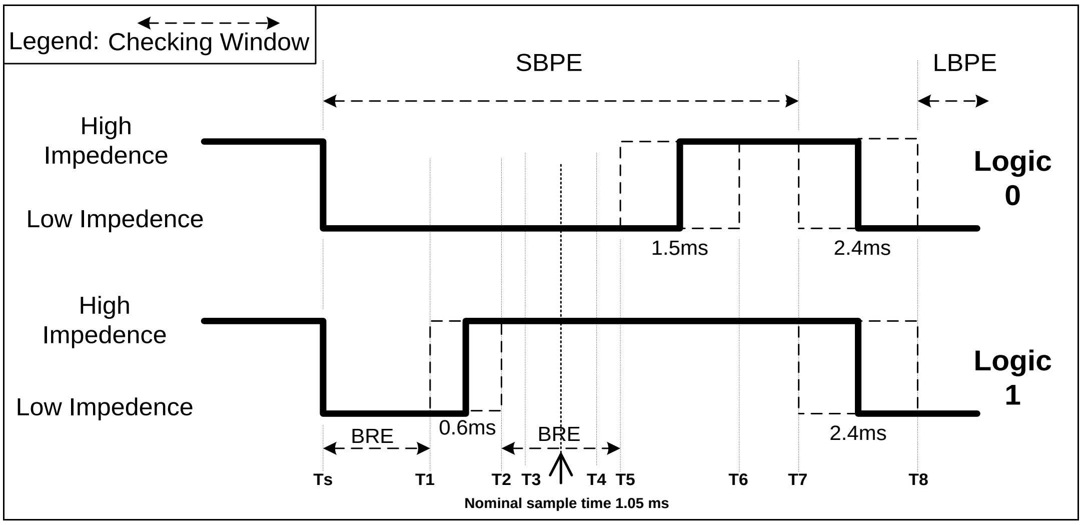


**Table 22-2. Error Handling Timing Parameter Table**

| Symbol | RTOL | Time(ms) | Description |
|---|---|---|---|
| Ts | - | 0ms | The bit start event. |
| T1 | 1 | 0.3ms | When indicating a logical 1,T1 as the earliest time for a low - high transition. |
|  | 0 | 0.4ms |  |
| T2 | 0 | 0.8ms | When indicating a logical 1,T2 as the latest time for a low - high transition. |
|  | 1 | 0.9ms |  |
| T3 | - | 0.85ms | The earliest time it is safe to sample the signal line to determine its state. |
| T4 | - | 1.25ms | The latest time it is safe to sample the signal line to determine its state. |
| T5 | 1 | 1.2ms | T5 as the earliest time that a device is allowed to return to a high impedance state(logical 0). |
|  | 0 | 1.3ms |  |
| T6 | 0 | 1.7ms | T6 as the latest time that a device is allowed to return to a high impedance state(logical 0). |
|  | 1 | 1.8ms |  |
| T7 | 1 | 1.85ms | T7 as the earliest time for the start of a following bit. |
|  | 0 | 2.05ms |  |
|  |  | 2.4ms | As a nominal data bit period. |
| T8 | 0 | 2.75ms | T8 as the latest time for the start of a following bit. |
|  | 1 | 2.95ms |  |

### Transmission error detection(TERR)

The TERR is set when the initiator find low impedance on the CEC bus when it is transmitting high impedance. TERR will also generate CEC interrupt if TERRIE=1.

When TERR asserted the transmission is aborted and the software can retry the transmission.

TERR check window is depending on the different bit state of the frame shown as below:

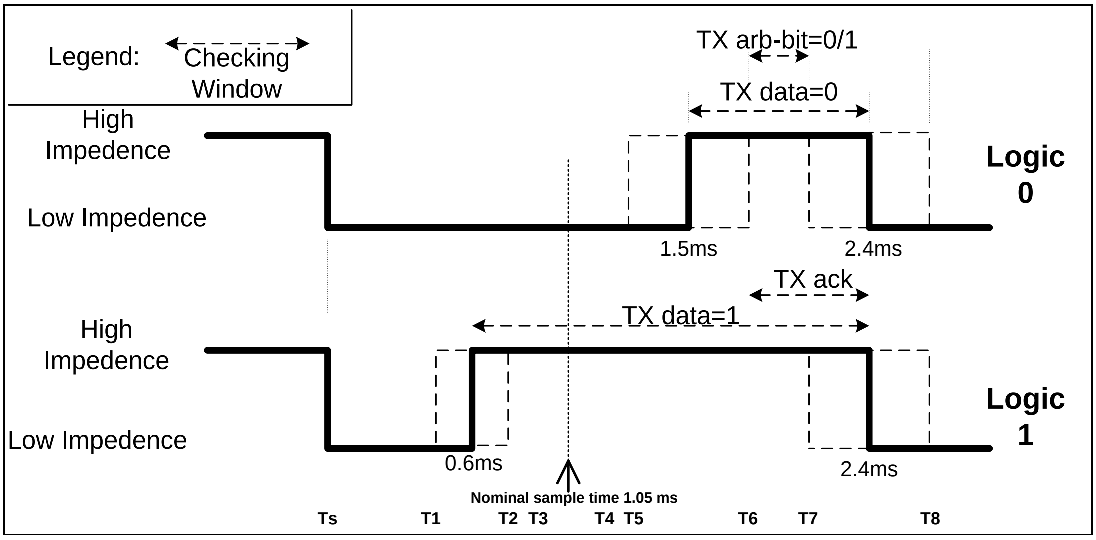


**Table 22-3. TERR Timing Parameter Table**

| Symbol | RTOL | Time(ms) | Description |
|---|---|---|---|
| Ts | - | 0ms | The bit start event. |
| T1 | 1 | 0.3ms | The earliest time for a low - high transition when indicating a logical 1. |
|  | 0 | 0.4ms |  |
| T2 | 0 | 0.8ms | The latest time for a low - high transition when indicating a logical 1. |
|  | 1 | 0.9ms |  |
| T3 | - | 0.85ms | The earliest time it is safe to sample the signal line to determine its state. |
| T4 | - | 1.25ms | The latest time it is safe to sample the signal line to determine its state. |
| T5 | 1 | 1.2ms | The earliest time a device is permitted return to a high impedance state (logical 0). |
|  | 0 | 1.3ms |  |
| T6 | 0 | 1.7ms | The latest time a device is permitted return to a high impedance state (logical 0). |
|  | 1 | 1.8ms |  |
| T7 | 1 | 1.85ms | The earliest time for the start of a following bit. |
|  | 0 | 2.05ms |  |
|  |  | 2.4ms | The nominal data bit period. |
| T8 | 0 | 2.75ms | The latest time for the start of a following bit. |
|  | 1 | 2.95ms |  |

### 22.3.7. HDMI-CEC interrupt

There 13 interrupts in HDMI-CEC controller and each are made up of corresponding flag and interrupt enable bit:

| No. | Interrupt event in HDMI-CEC | Event flag | Interrupt enable bit |
|---|---|---|---|
| 1 | Arbitration lost | LSTARB | LSTARBIE |
| 2 | TX Byte Request | TBR | TBRIE |

| No. | Interrupt event in HDMI-CEC | Event flag | Interrupt enable bit |
|---|---|---|---|
| 3 | Transmission end | TEND | TENDIE |
| 4 | TX Byte buffer underrun | TU | TUIE |
| 5 | TX error | TERR | TERRIE |
| 6 | TX acknowledge error | TAERR | TAERRIE |
| 7 | RX Byte Received | RBR | RBRIE |
| 8 | Reception end | REND | RENDIE |
| 9 | RX Overrun | RO | ROIE |
| 10 | RX-Bit rising error | RBRE | BREIE |
| 11 | RX-Short Bit Period Error | RSBPE | RSBPEIE |
| 12 | RX-Long Bit Period Error | RLBPE | RLBPEIE |
| 13 | RX acknowledge error | RAE | RAEIE |

**Note:** Any HDMI-CEC interrupt will wake up the chip from Deep-sleep Mode.

## 22.4. HDMI-CEC registers

### 22.4.1. Control register (CEC_CTL)

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 | | | | | | | | | | | | | ENDOM | SOM | CECEN |
| | | | | | | | | | | | | | rs | rs | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value |
| 2 | ENDOM | ENDOM bit value in the next frame in TX mode.<br>ENDOM can only be software wrote when CECON=1.ENDOM is cleared by hardware in the same situation of clear SOM.<br>0: Next frame send 0 in ENDOM bit position<br>1: Next frame send 1 in ENDOM bit position |
| 1 | SOM | Start of sending a message.<br>SOM can only be software wrote when CECON=1.SOM is cleared by hardware if any of flags asserted: TEND, TU, TAERR, TERR and CECON=0.<br>If the message consists of only one frame, ENDOM should be set before configuring txdata. After setting SOM, the SFT counter will start and when SFT is |

| | | |
|---|---|---|
| | | done the Start-bit will performance one CEC line. Software can abort sending the message through clear CECON bit under SOM=1.<br>0: No CEC transmission is on-going<br>1: CEC transmission is pending or executing |
| 0 | CECEN | Enable/disable HDMI-CEC controller bit.<br>CECON bit is configured by software.<br>0: Disable HDMI-CEC controller. Abort any sending message state and clear ENDOM/SOM<br>1: Enable CEC controller and go into receiving state if SOM=0 |

### 22.4.2. Configuration register (CEC_CFG)

Address offset: 0x04

Reset value: 0x0000 0000

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

**Note:** This register can only be write when CECON=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LMEN | OADR [14:0] | | | | | | | | | | | | | | |
| rw | rw | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | SFTOPT | BCNG | RLBPEGEN | RBREGEN | RBRESTP | RTOL | SFT[2:0] | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LMEN | Listen Mode Enable Bit<br>This bit is set and cleared by software.<br>0: Only receive broadcast and singlecast in OADR address with appropriate ACK<br>1: Receive broadcast and singlecast in OADR address with appropriate ACK and receive message whose destination address is not in OADR without feedback ACK |
| 30:16 | OADR[14:0] | Own Address<br>Each bit of OADR represents one destination address. For example: if oar[0]=1,the controller will receive the message sent address=0x0.This means the controller can be configured to multiple own address. Broadcast message is always received.<br>After received destination address(last 4 bit of HEADER frame), if it is valid in the oar, the controller will feedback with positive acknowledge ,if it is not valid in the oar and LMEN=1,the controller will receive the message with no acknowledge, if it is not valid in the oar and LMEN=0,the controller will not receive the message. |
| 15:9 | Reserved | Must be kept at reset value |
| 8 | SFTOPT | The SFT start option bit |

| | | |
|---|---|---|
| | | This bit is set and cleared by software.<br>0: SFT counter starts counting when SOM is asserted<br>1: SFT counter starts automatically after transmission/reception en |
| 7 | BCNG | Do not generate Error-bit in broadcast message<br>This bit is set and cleared by software.<br>0: In broadcast mode, RBRE and RLBPE will generate Error-bit on CEC line and if LMEN=1, RSBPE will also generate Error-bit<br>1: Error-bit is not generated in the same condition as above |
| 6 | RLBPEGEN | Generate Error-bit when detected RLBPE in singlecast<br>This bit is set and cleared by software.<br>0: Not generate Error-bit on CEC line when detected RLBPE in singlecast<br>1: Generate Error-bit on CEC line when detected RLBPE in singlecast |
| 5 | RBREGEN | Generate Error-bit when detected RBRE in singlecast<br>This bit is set and cleared by software.<br>0: Not generate Error-bit on CEC line when detected RBRE in singlecast<br>1: Generate Error-bit on CEC line when detected RBRE in singlecast |
| 4 | RBRESTP | Whether stop receive message when detected RBRE<br>This bit is set and cleared by software.<br>0: Do not stop reception for RBRE and data bit is sampled at nominal time(1.05ms)<br>1: Stop reception for RBRE |
| 3 | RTOL | Reception bit timing tolerance<br>This bit is set and cleared by software.<br>0: Standard bit timing tolerance<br>1: Extended bit timing tolerance |
| 2:0 | SFT[2:0] | Signal Free Time<br>This bit is set and cleared by software.<br>If SFT=0x0, the SFT time will perform as HDMI-CEC protocol description and if not, the SFT time is fixed configured by software. The start point is the falling edge of the ACK bit.<br>0x0:<br>- 3 Standard data-bit period if SFT counter is start because of unsuccessful transmission(LSTARB=1,TERR=1,TU=1 or TAERR=1)<br>- 5 Standard data-bit period if CEC controller is the new initiator<br>- 7 Standard data-bit period if CEC controller has successful completed transmission<br>0x1: 1.5 nominal data bit periods<br>0x2: 2.5 nominal data bit periods<br>0x3: 3.5 nominal data bit periods<br>0x4: 4.5 nominal data bit periods<br>0x5: 5.5 nominal data bit periods<br>0x6: 6.5 nominal data bit periods |

0x7: 7.5 nominal data bit periods

### 22.4.3. Transmit data register (CEC_TDATA)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TXDATA[7:0] | | | | | | | |
| | | | | | | | | w | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | TXDATA[7:0] | Tx data register<br>This bit is write only. |

### 22.4.4. Receive data register (CEC_RDATA)

Address offset: 0xC

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 | | | | | | | | RXDATA[7:0] | | | | | | | |
| | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7:0 | RXDATA[7:0] | Rx data register<br>This bit is read only and contains the last data byte which has been received from the CEC line. |

### 22.4.5. Interrupt Flag Register (CEC_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 | | | TAERR | TERR | TU | TEND | TBR | LSTARB | RAE | RLBPE | RSBPE | RBRE | RO | REND | RBR |
| | | | 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 |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value |
| 12 | TAERR | Tx ACK Error flag.<br>This bit is set by hardware and cleared by software write at 1.<br>The ACK bit is received 1 in singlecast and is received 0 in broadcast will assert the flag. TAERR will stop sending message and clear SOM and ENDOM. |
| 11 | TERR | Tx-Error<br>This bit is set by hardware and cleared by software write at 1.<br>TERR is asserted if the controller is in initiator state and find the CEC line is low impedance but it does not pull it down. TERR will stop sending message and clear SOM and ENDOM. |
| 10 | TU | Tx data buffer underrun<br>This bit is set by hardware and cleared by software write at 1.<br>TU is asserted if the software does not write data before sending the next byte. TU will stop sending message and clear SOM and ENDOM. |
| 9 | TEND | Transmission successfully end<br>This bit is set by hardware and cleared by software write at 1.<br>TEND is asserted if the all frames of the message are successfully transmitted. TEND will clear SOM and ENDOM bit. |
| 8 | TBR | Tx-Byte data request<br>This bit is set by hardware and cleared by software write at 1.<br>TBR is asserted when the 4th bit of current frame is transmitted and software should write data into txdata within 6 nominal data-bit periods |
| 7 | LSTARB | Arbitration lost<br>This bit is set by hardware and cleared by software write at 1.<br>LSTARB is asserted when either situation is occurs: external CEC device pull down the CEC line for sending start bit when controller is in SFT state or the |

| | | |
|---|---|---|
| | | controller and CEC device sending the start bit at the same time but the controller's initiator address priority is lower.<br>If LSTARB is asserted, the controller will get into reception state and after finish receiving the message the controller will retry to send message. During receiving and sending message, the SOM will keep set. |
| 6 | RAE | Rx ACK Error<br>This bit is set by hardware and cleared by software write at 1.<br>RAE is asserted if ACK=0 in broadcast or ACK=1 in singlecast under LMEN=1 and destination address is not in OADR.<br>RAE will stop receiving message. |
| 5 | RLBPE | Long Bit Period Error<br>This bit is set by hardware and cleared by software write at 1.<br>RLBPE is asserted when the data-bit is out of the maximum period. RLBPE will stop receiving message and generate error-bit if RLBPEGEN=1 in singlecast or BCNG=0 in broadcast. |
| 4 | RSBPE | Short Bit Period Error<br>This bit is set by hardware and cleared by software write at 1.<br>RSBPE is asserted if a data-bit period is less than the minimal period. |
| 3 | RBRE | Bit Rising Error<br>This bit is set by hardware and cleared by software write at 1.<br>RBRE is asserted if the rising edge in a period is occurs in unexpected time. |
| 2 | RO | RX Overrun<br>This bit is set by hardware and cleared by software write at 1.<br>RO is asserted when a new byte is received and RBR is also set.<br>RO will stop receiving message and send an incorrect ACK bit. |
| 1 | REND | End of Reception<br>This bit is set by hardware and cleared by software write at 1.<br>REND is asserted if the controller received the whole message with feedback correct ACK. REND is asserted at the same time of RBR. |
| 0 | RBR | Rx-Byte data received<br>This bit is set by hardware and cleared by software write at 1.<br>RBR is asserted if the controller received the whole message with feedback correct ACK. When RBR is asserted, the rxdata is valid. |

### 22.4.6. Interrupt enable register (CEC_INTEN)

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 | | | TAERRIE | TERRIE | TUIE | TXENDIE | TBRIE | LSTARBIE | RAEIE | RLBPEIE | RSBPEIE | RBREIE | ROIE | RENDIE | RBRIE |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value |
| 12 | TAERRIE | TAERR Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: TAERR interrupt disable<br>1: TAERR interrupt enable |
| 11 | TERRIE | TERR Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: TERR interrupt disable<br>1: TERR interrupt enable |
| 10 | TUIE | TU Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: TU interrupt disable<br>1: TU interrupt enable |
| 9 | TENDIE | TEND Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: TEND interrupt disable<br>1: TEND interrupt enable |
| 8 | TBRIE | TBR Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: TBR interrupt disable<br>1: TBR interrupt enable |
| 7 | LSTARBIE | ALRLST Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: LSTARB interrupt disable<br>1: LSTARB interrupt enable |
| 6 | RAEIE | RAE Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: RAE interrupt disable<br>1: RAE interrupt enable |
| 5 | RLBPEIE | RLBPE Interrupt Enable. |

| | | |
|---|---|---|
| | | This bit is set by and cleared by software.<br>0: RLBPE interrupt disable<br>1: RLBPE interrupt enable |
| 4 | RSBPEIE | RSBPE Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: RSBPE interrupt disable<br>1: RSBPE interrupt enable |
| 3 | RBREIE | RBRE Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: RBRE interrupt disable<br>1: RBRE interrupt enable |
| 2 | ROIE | RO Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: RO interrupt disable<br>1: RO interrupt enable |
| 1 | RENDIE | REND Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: REND interrupt disable<br>1: REND interrupt enable |
| 0 | RBRIE | RBR Interrupt Enable.<br>This bit is set by and cleared by software.<br>0: RBR interrupt disable<br>1: RBR interrupt enable |

# 23. Segment LCD controller (SLCD)

**This chapter applies to GD32F170xx and GD32F190xx devices.**

## 23.1. Introduction

The SLCD controller directly drives LCD displays by creating the AC segment and common voltage signals automatically. It can drive the monochrome passive liquid crystal display (LCD) which composed of a plurality of segments (pixels or complete symbols) that can be converted to visible or invisible. The SLCD controller can support up to 32 segments and 8 commons.

## 23.2. Main features

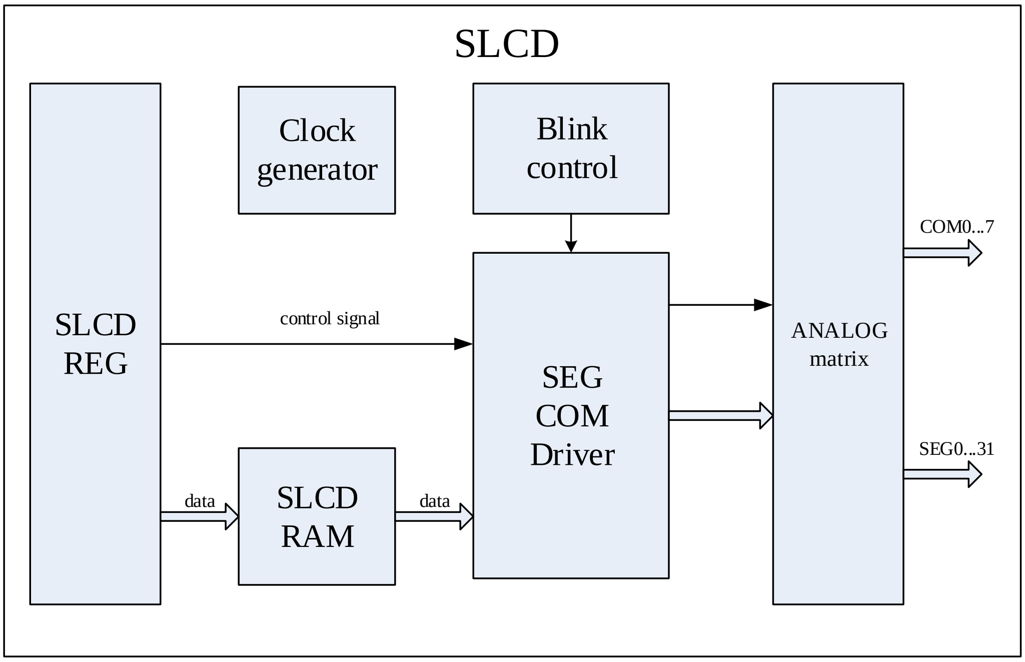
- Configurable frame frequency
- Blinking of individual segments or all segments
- Supports Static, 1/2, 1/3, 1/4, 1/6 and 1/8 duty
- Supports 1/2, 1/3 and 1/4 bias
- Double buffer up to 8x32 bits registers to store SLCD_DATAx
- The contrast can also be adjusted by configuring dead time
- VSLCD rails decoupling capability

## 23.3. Function description

### 23.3.1. SLCD Architecture

The block diagram of the SLCD controller is shown as follows.

**Figure 23-1. SLCD Block Diagram**



The SLCD REG is the register of SLCD controller, which configured by APB bus, and generate interrupt to CPU. It includes SLCD_CTL, SLCD_CFG, SLCD_STAT, SLCD_STATC, SLCD_DATAx registers.

The Clock generator generates SLCD clock from input clock. The SLCD clock drivers the blink control and SEG/COM driver. The Blink control generates blink frequency and blink pixels. The SEG/COM driver generates segment and common signals to ANALOG matrix. The ANALOG matrix implements segment and common voltages.

### 23.3.2. Clock generator

SLCD input clock is the same as RTCCLK, 3 different clock sources: LXTAL, IRC40K and HXTAL divided by 32 can be selected by RTCSRC bits in RCU_BDCTL register. The input clock frequency varies from 32KHz to 1MHz.

The SLCD controller uses the input clock signal from the integrated clock divider to generate the timing for common and segment lines. The SLCD clock frequency is selected with the PSC and DIV bits in SLCD_CTL registers. The resulting SLCD clock frequency is calculated by:

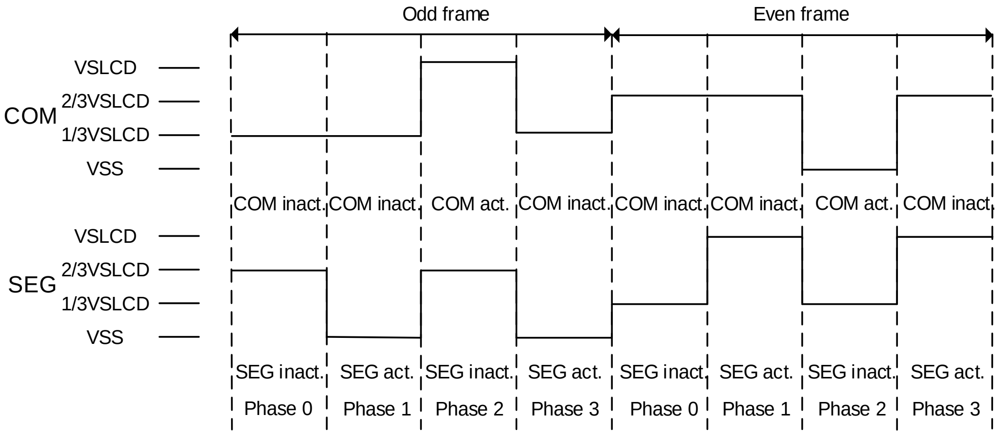
$$f_{SLCD} = \frac{f_{in\_clk}}{2^{PSC} \times (DIV + 16)}$$

The SLCD clock is the time base for the SLCD controller. The frequency of SLCD clock is equivalent to the phase frequency. One SLCD frame is one odd frame or one even frame, both of them have several phases as many as active common terminals. The duty is the Number defined as 1/ (the number of common terminals on a given SLCD display). So the frame frequency is calculated by:

$$f_{frame} = f_{SLCD} \times Duty$$

The SOF bit in SLCD_STAT register is set by the hardware at the start of the frame, and the SLCD interrupt is executed if the SOFIE bit in SLCD_CFG is set. SOF is cleared by writing 1 to the SOFC bit in SLCD_STATC register.

**Figure 23-2. 1/3 Bias, 1/4 Duty**



## 23.3.3. Blink control

The SLCD controller also supports blinking. The blinking mode controlled by BLKMOD bits in SLCD_CFG register. BLKMOD = 01 allows to blink individual segment on SEG0 with COM0, with BLKMOD = 10 all commons on SEG0 are blinking, with BLKMOD = 11 all segments with all commons are blinking, and with BLKMOD = 00 blinking is disabled.

The blink frequency is generated from SLCD clock and selected with BLKDIV bit in SLCD_CFG registers. The resulting BLINK frequency is calculated by:

$$f_{BLINK} = \frac{f_{SLCD}}{2^{(BLKDIV+3)}}$$

After a blinking mode (BLKMOD = 01, 10 or 11) is selected, the enabled segments or all segments go blank at the next frame boundary and stay off for half a BLKCLK period. Then they go active at the next frame boundary and stay on for another half BLKCLK period before they go blank again at a frame boundary.

## 23.3.4. SEG/COM Driver

SEG/COM Driver generates segments and commons signals.

**BIAS generator:**

The bias is selected by setting BIAS bits in SLCD_CTL registers. It has its max amplitude VSLCD or VSS only in the corresponding phase of a frame cycle. The odd frame voltage and even frame voltage are shown as follows:

**Table 23-1. The odd frame voltage**

| BIAS | Static | 1/2 bias | 1/3 bias | 1/4 bias |
|---|---|---|---|---|
| COM active | VSLCD | VSLCD | VSLCD | VSLCD |

| BIAS | Static | 1/2 bias | 1/3 bias | 1/4 bias |
|---|---|---|---|---|
| COM inactive | / | 1/2 VSLCD | 1/3 VSLCD | 1/4 VSLCD |
| SEG active | VSS | VSS | VSS | VSS |
| SEG inactive | VSLCD | VSLCD | 2/3 VSLCD | 1/2 VSLCD |

**Table 23-2. The even frame voltage**

| BIAS | Static | 1/2 bias | 1/3 bias | 1/4 bias |
|---|---|---|---|---|
| COM active | VSS | VSS | VSS | VSS |
| COM inactive | / | 1/2 VSLCD | 2/3 VSLCD | 3/4 VSLCD |
| SEG active | VSLCD | VSLCD | VSLCD | VSLCD |
| SEG inactive | VSS | VSS | 1/3 VSLCD | 1/2 VSLCD |

**COM signal:**

The common signal is selected by DUTY bits in SLCD_CTL register. When DUTY is 000, static duty selected. Only COM[0] used and only one phase in odd frame or even frame, so COM[0] driver active signal always. When DUTY is 001, only COM[1:0] and 2 phases used. When DUTY is 010, only COM[2:0] and 3 phases used. When DUTY is 011, only COM[3:0] and 4 phases used. When DUTY is 100, COM[7:0] and 8 phases used. When DUTY is 101, COM[5:0] and 6 phases used. The all common signal driver is shown as follows:

**Table 23-3. The all common signal driver**

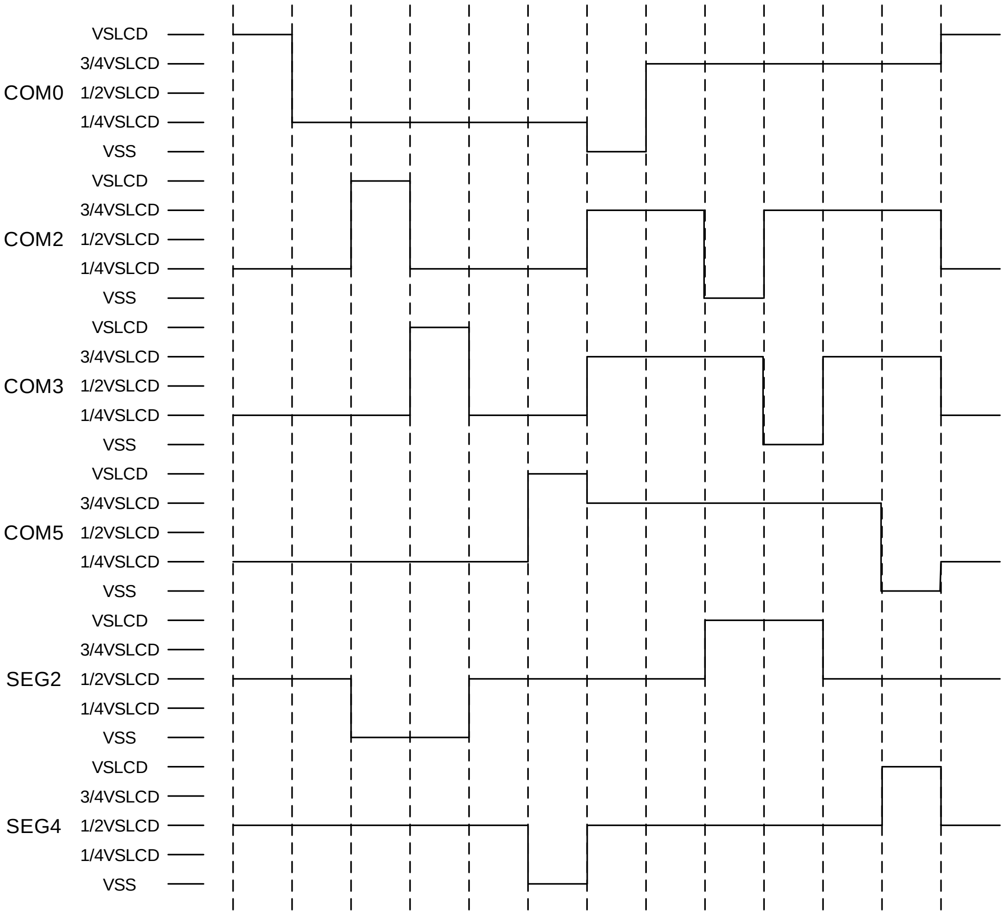
| phase | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| COM0 | active | inactive | inactive | inactive | inactive | inactive | inactive | inactive |
| COM1 | inactive | active | inactive | inactive | inactive | inactive | inactive | inactive |
| COM2 | inactive | inactive | active | inactive | inactive | inactive | inactive | inactive |
| COM3 | inactive | inactive | inactive | active | inactive | inactive | inactive | inactive |
| COM4 | inactive | inactive | inactive | inactive | active | inactive | inactive | inactive |
| COM5 | inactive | inactive | inactive | inactive | inactive | active | inactive | inactive |
| COM6 | inactive | inactive | inactive | inactive | inactive | inactive | active | inactive |
| COM7 | inactive | inactive | inactive | inactive | inactive | inactive | inactive | active |

**SEG signal:**

The segment signals are read from SLCD_DATAx registers. The segment signals data are SLCD_DATAx when phase x. When the value is 1, the corresponding segment drives active signal. When the value is 0, the corresponding segment drives inactive signal.

For example, if the application need to active the pixel COM2 SEG2, COM3 SEG2, and COM5 SEG4. It should set the bit2 in the SLCD_DATA2, the bit2 in the SLCD_DATA3, and the bit4 in the SLCD_DATA5. Then the SEG2 signal will active at the third and fourth phase of each odd and even frame, the SEG4 signal will active at the sixth phase of each odd and even frame. The active and inactive voltages are shown in Table 23-1 and Table 23-2. The segment signals show in figure 23-3 is the result to the above configuration when bias is 1/4 and duty is 1/6.

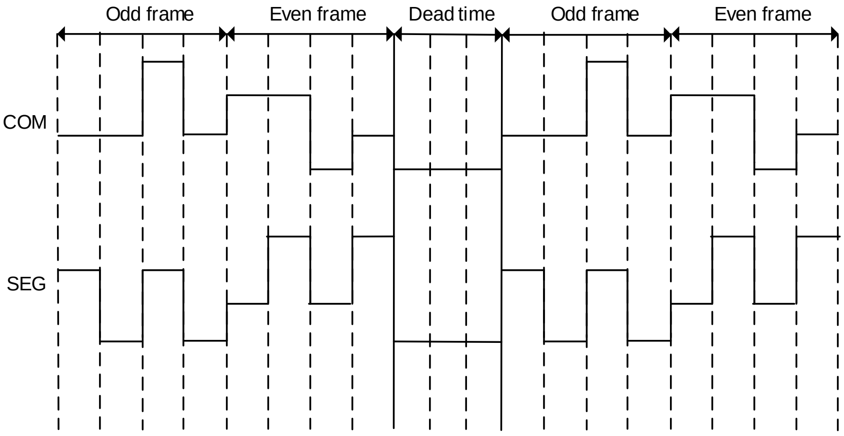
**Figure 23-3. 1/4 Bias, 1/6 Duty**



**DEAD time:**

The dead time is using DTD bits in SLCD_CFG register. It inserts VSS after each even frame. The number of phase inserted is defined by DTD bits. The application can adjust the contrast according to the configuration of dead time.

**Figure 23-4. SLCD dead time (1/3 Bias, 1/4 Duty)**



### 23.3.5. Double buffer memory

The double buffer memory is used to ensure the coherency of the displayed information.

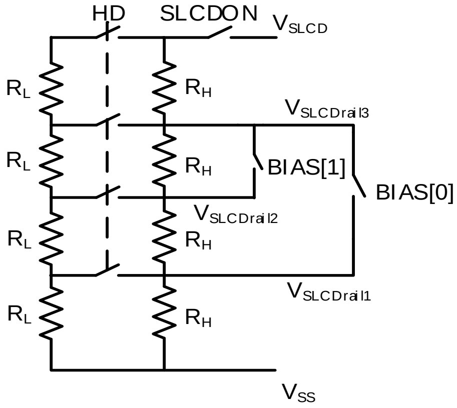
The application access the first buffer according to modify the SLCD_DATAx registers. After writing the displayed information into the SLCD_DATAx registers, the application need to set the UPRF bit in SLCD_STAT register, then the hardware will transfer the data from the first buffer to the seconed buffer, during this time, the UPRF keeps set and the SLCD_DATAx registers are write protected. When the transfer is completed, the UPRF is cleared and the UPDF is set by the hardware, an interrupt will be generated if the UPDIE is set. The segment signal is driven by the data in the second buffer, so the displayed information will not be influenced by writing SLCD_DATAx.

If the UPRF is set when the display is disabled (SLCDON = 0), the transfer will not occur until the SLCDON is set.

### 23.3.6. ANALOG matrix

The analog matrix supplies SLCD voltage. The SLCD voltage levels are generated by the VSLCD pin or by the internal voltage step-up converter (depending on the VSRC bit in the SLCD_CTL register). When using the internal voltage, the VSLCD value can be selected from VSLCD0 to VSLCD7 by the CONR[2:0] bits in the SLCD_CFG register (Refer to the product datasheet for the VSLCDx values). The application can adjust the contrast according to the change of VSLCD value. The other way to adjust the contrast is by using the deadtime.
The analog matrix supplies intermediate voltage levels (1/3 VSLCD, 2/3 VSLCD or 1/4 VSLCD, 2/4 VSLCD, 3/4 VSLCD) between VSS and VSLCD through an internal resistor divider network as shown in the figure 23-5.

**Figure 23-5. Resistr divider network**



During the transitions, the low value resistors ($R_L$) are switched on to increase the current in order to quickly reach the static state. Then the low value resistors ($R_L$) are switched off, the high value resistors ($R_H$) are used to reduce the power. The length of the time during $R_L$ is switched on depend on the PULSE[2:0] bits in the SLCD_CFG register. The $R_L$ can be always switched on according to setting the HDEN bit in the SLCD_CFG register.

**External decoupling:**

The VSLCD intermediate voltage rails (VSLCDrail1, VSLCDrail2, VSLCDrail3 in the figure 23-5) can be connect to the GPIOs by configuring the SLCD_DECA bits of the SYSCFG_CFG1 register. Adding decoupling capacitors on these GPIOs helps to get a steady voltage resulting in a higher contrast. Thus the PULSE[2:0] is allowed to select low value to reduce the power.

**Note:** When using VSLCDrail function, GPIO should be configured to analog input mode. The segment AFIO and VSLCDrail decoupling functions cannot be used simultaneously.

**Table 23-4. VSLCDrail connect pins**

| | BIAS | | | Pin |
|---|---|---|---|---|
| | 1/2 | 1/3 | 1/4 | |
| **VSLCDrail1** | 1/2VSLCD | 1/3VSLCD | 1/4VSLCD | PB12 |
| **VSLCDrail2** | 1/2VSLCD | 2/3VSLCD | 1/2VSLCD | PB2 |
| **VSLCDrail3** | 1/2VSLCD | 2/3VSLCD | 3/4VSLCD | PB0 |

## 23.4. SLCD registers

### 23.4.1. Control register (SLCD_CTL)

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 | | | | | | | | COMS | BIAS[1:0] | | DUTY[2:0] | | | VSRC | SLCDON |
| | | | | | | | | rw | rw | | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 7 | COMS | common/segment pad select<br>This bit is used to common/segment pad selection. When duty selects 1/8 or 1/6, SLCD_COM[7:4] pad is always select SLCD_COM[7:4] function whatever this bit is set or reset.<br>0: SLCD_COM[7:4] pad select SLCD_COM[7:4]<br>1: SLCD_COM[7:4] pad select SLCD_SEG[31:28] |
| 6:5 | BIAS[1:0] | Bias select<br>Bias is the number of voltage levels used when driving a SLCD. It is defined as 1/ (number of voltage levels used to drive an SLCD display - 1). |

| | | |
|---|---|---|
| | | 00: 1/4 Bias (5 voltage levels: VSS, 1/4VSLCD, 1/2VSLCD, 3/4VSLCD, VSLCD)<br>01: 1/2 Bias (3 voltage levels: VSS, 1/2VSLCD, VSLCD)<br>10: 1/3 Bias (4 voltage levels: VSS, 1/3VSLCD, 2/3VSLCD, VSLCD)<br>11: Reserved |
| 4:2 | DUTY[2:0] | Duty select<br>These bits determine the duty cycle. Duty is the number defined as 1/(number of common terminals on a given SLCD display).<br>000: Static duty<br>001: 1/2 duty<br>010: 1/3 duty<br>011: 1/4 duty<br>100: 1/8 duty<br>101: 1/6 duty<br>110: Reserved<br>111: Reserved |
| 1 | VSRC | SLCD Voltage source<br>Set this bit determines which is the SLCD voltage source.<br>0: Internal source<br>1: External source (VSLCD pin) |
| 0 | SLCDON | SLCD controller start<br>Set this bit by software to start SLCD controller. Clear this bit by software to stop SLCD controller and the SLCD controller stop at the beginning of the next frame.<br>0: SLCD Controller stop<br>1: SLCD Controller start |

### 23.4.2. Configuration register (SLCD_CFG)

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 | | | | | | PSC[3:0] | | | | DIV[3:0] | | | | BLKMOD[1:0] | |
| | | | | | | rw | | | | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BLKDIV[2:0] | | | CONR[2:0] | | | DTD[2:0] | | | PULSE[2:0] | | | UPDIE | Reserved | SOFIE | HDEN |
| rw | | | rw | | | rw | | | rw | | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value |
| 25:22 | PSC[3:0] | SLCD clock prescaler |

| | | |
|---|---|---|
| | | Set these bits define the prescaler of SLCD clock.<br>0000: $f_{PSC} = f_{in\_clk}$<br>0001: $f_{PSC} = f_{in\_clk}/2$<br>0010: $f_{PSC} = f_{in\_clk}/4$<br>...<br>1111: $f_{PSC} = f_{in\_clk}/32768$ |
| 21:18 | DIV[3:0] | SLCD clock divider<br>Set these bits define the division factor of the DIV divider.<br>0000: $f_{SLCD} = f_{PSC}/16$<br>0001: $f_{SLCD} = f_{PSC}/17$<br>0010: $f_{SLCD} = f_{PSC}/18$<br>...<br>1111: $f_{SLCD} = f_{PSC}/31$ |
| 17:16 | BLKMOD[1:0] | Blink mode<br>00: No Blink<br>01: Blink on SEG[0], COM[0] (1 pixel)<br>10: Blink on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)<br>11: Blink on all SEGs and all COMs (all pixels) |
| 15:13 | BLKDIV[2:0] | Blink frequency divider<br>000: $f_{BLINK} = f_{SLCD}/8$<br>001: $f_{BLINK} = f_{SLCD}/16$<br>010: $f_{BLINK} = f_{SLCD}\ /32$<br>011: $f_{BLINK} = f_{SLCD}\ /64$<br>100: $f_{BLINK} = f_{SLCD}/128$<br>101: $f_{BLINK} = f_{SLCD}/256$<br>110: $f_{BLINK} = f_{SLCD}/512$<br>111: $f_{BLINK} = f_{SLCD}/1024$ |
| 12:10 | CONR[2:0] | Contrast ratio<br>When chosing the internal voltage source (VSRC=0), these bits specify the VSLCD voltage. It ranges from VSLCD0 to VSLCD7(typical 2.75 V to 5.20V), Refer to the product datasheet for the VSLCDx values. When chosing the external voltage source (VSRC=1), these bits is invalid.<br>000: VSLCD0<br>001: VSLCD1<br>010: VSLCD2<br>011: VSLCD3<br>100: VSLCD4<br>101: VSLCD5<br>110: VSLCD6<br>111: VSLCD7 |
| 9:7 | DTD[2:0] | Dead time duration |

| | | |
|---|---|---|
| | | Set these bits configure the length of the dead time between frames.<br>000: No dead time<br>001: 1 phase dead time<br>010: 2 phase dead time<br>...<br>111: 7 phase dead time |
| 6:4 | PULSE[2:0] | Pulse ON duration<br>Set these bits define the pulse duration in terms of PSC pulses.<br>000: 0<br>001: $1/f_{PSC}$<br>010: $2/f_{PSC}$<br>011: $3/f_{PSC}$<br>100: $4/f_{PSC}$<br>101: $5/f_{PSC}$<br>110: $6/f_{PSC}$<br>111: $7/f_{PSC}$ |
| 3 | UPDIE | SLCD update done interrupt enable<br>This bit is set and cleared by software.<br>0: SLCD Update Done interrupt disabled<br>1: SLCD Update Done interrupt enabled |
| 2 | Reserved | Must be kept at reset value |
| 1 | SOFIE | Start of frame interrupt enable<br>This bit is set and cleared by software.<br>0: SLCD Start of Frame interrupt disabled<br>1: SLCD Start of Frame interrupt enabled |
| 0 | HDEN | High drive enable<br>This bit is set and cleared by software.<br>0: Permanent high drive disabled. The time during which $R_L$ is enabled is configured by the PULSE[2:0].<br>1: Permanent high drive enabled. RL is always switched on, and the PULSE[2:0] is invalid. |

### 23.4.3. Status flag register (SLCD_STAT)

Address offset: 0x08
Reset value: 0x0000 0020

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 | | | | | | | | | | SYNF | VRDYF | UPDF | UPRF | SOF | ONF |
| | | | | | | | | | | r | r | r | rs | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value |
| 5 | SYNF | SLCD_CFG register synchronization flag<br>This bit is set when SLCD_CFG register update to SLCD clock domain, and It is cleared by hardware when writing to the SLCD_CFG register.<br>0: SLCD_CFG Register not yet synchronized<br>1: SLCD_CFG Register synchronized to SLCD clock domain |
| 4 | VRDYF | SLCD voltage ready flag<br>This bit is set and cleared by the hardware according to the SLCD voltage.<br>0: SLCD voltage Is not ready<br>1: Step-up converter is enabled and ready to provide the correct voltage |
| 3 | UPDF | Update SLCD data done flag<br>This bit is set by hardware when update SLCD data done. It is cleared by writing 1 to the UPDC bit in the SLCD_STATC register.<br>0: No effect<br>1: SLCD data update done |
| 2 | UPRF | Update SLCD data request flag<br>After modifying the the first buffer by the SLCD_DATAx registers, the application should set this bit to transfer the data to the second buffer. This bit stays set until the transfer is complete, the SLCD_DATAx register is write protected during this time.<br>0: No effect<br>1: Request SLCD data update |
| 1 | SOF | Start of frame flag<br>This bit is set by hardware at the beginning of a new frame. It is cleared by writing 1 to the SOFC bit in the SLCD_STATC register.<br>0: No effect<br>1: Start of Frame flag |
| 0 | ONF | SLCD controller on flag<br>This bit is set by hardware when SLCDON is set to 1, it is cleard by hardware after the SLCDON is cleared and the last frame is displayed.<br>0: SLCD Controller disabled.<br>1: SLCD Controller enabled |

### 23.4.4. Status flag clear register (SLCD_STATC)

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 | | | | | | | | | | | | UPDC | Reserved | SOFC | Reserved |
| | | | | | | | | | | | | rc_w1 | | rc_w1 | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3 | UPDC | SLCD data update done clear bit<br>Set this bit to clear the UPDF flag in SLCD_STAT register.<br>0: No effect<br>1: Clear UPDF flag |
| 2 | Reserved | Must be kept at reset value |
| 1 | SOFC | Start of frame flag clear<br>Set this bit to clear the SOF flag in the SLCD_STAT register.<br>0: No effect<br>1: Clear SOF flag |
| 0 | Reserved | Must be kept at reset value |

### 23.4.5. Display data registers (SLCD_DATAx, x=0~7)

Address offset: 0x14+0x08*(x+1)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEG_DATAx[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SEG_DATAx[31:0] | Each bit corresponds to one pixel to display.<br>0: Pixel inactive<br>1: Pixel active |

# 24. Operational amplifiers (OPA)/ Programmable current and voltage reference (IVREF)

**This chapter applies to GD32F170xx and GD32F190xx devices.**

## 24.1. Operational amplifiers (OPA)

### 24.1.1. Introduction

There are three operational amplifiers in the MCU and the operational amplifiers can be able to route to external or internal follower.

When an operational amplifier is enabled, there will be a corresponding ADC channel used to output measurement outcome.

### 24.1.2. Main features

- Rail-to-rail input and output voltage range
- Low input bias current
- Low input offset voltage
- Low power mode

### 24.1.3. Function description

**Signal routing**

The connections with dedicated I/O are listed below:

- OPA0_VINP——>PA1
- OPA0_VINM——>PA2
- OPA0_VOUT——>PA3
- OPA1_VINP——>PA6
- OPA1_VINM——>PA7
- OPA1_VOUT——>PB0
- OPA2_VINP——>PC1
- OPA2_VINM——>PC2
- OPA2_VOUT——>PC3

Figure 24-1. OPA0 Signal Route

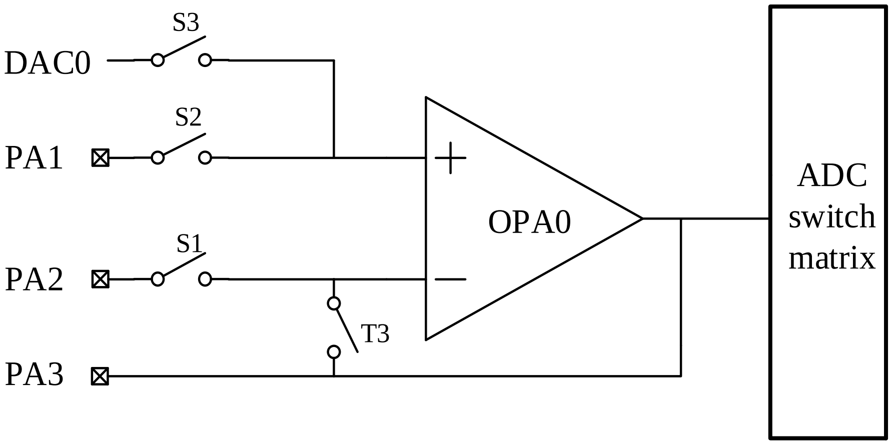


Figure 24-2. OPA1 Signal Route

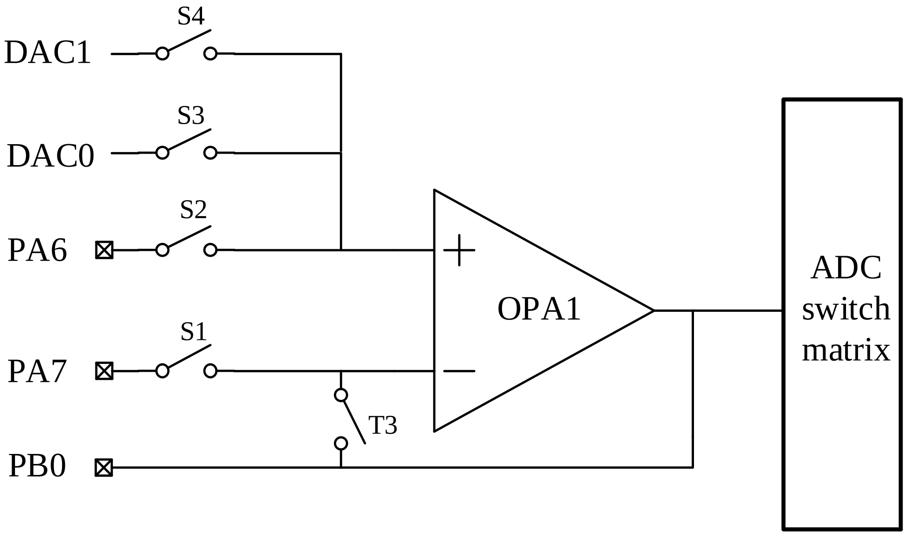
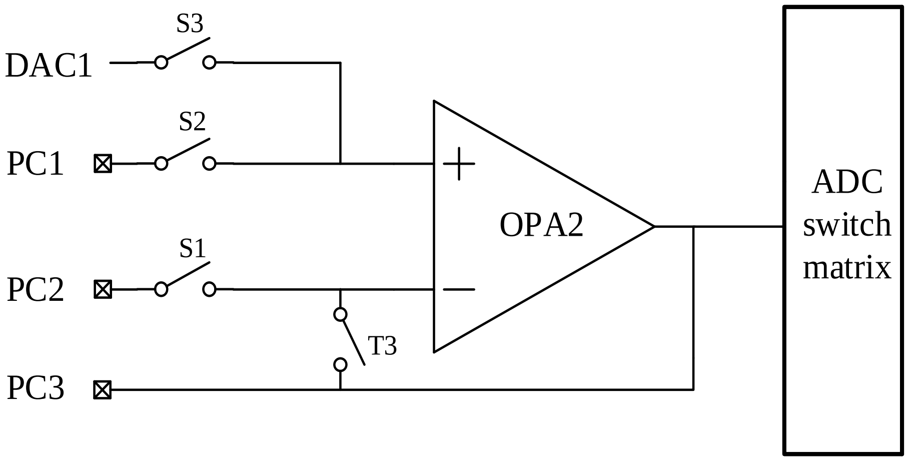


Figure 24-3. OPA2 Signal Route



OPA_CTL register can select the routing for the three operational amplifiers.

For OPA0-2, S3/S4 is used to connect DAC0/DAC1 to their positive input.

The OPAx_PD bit can be used to power down all operational amplifiers.

### Calibration

Before leaving factory, the default factory trimming values are stored in nonvolatile memory and used to initialize the operational amplifier offset. But also the chip support software using the user value to trim the operational amplifier offset.

During trimming operation, all switches related to the inputs of each operational amplifier must be disconnected.

Each operational amplifier has two operation modes: normal mode and low-power mode. The different mode can configure different user trimming value in different register. The trimming value is a 30-bit value for each mode of each operational amplifier. Writing 1 to OT_USER bit will switch the trimming value to user configured values. This will take effect for all the operational amplifiers.

Following steps are the commendatory procedure for either one of operational amplifier:
1). Configure control register to disconnected all the switches to the operational amplifier
2). Set the OT_USER bit to 1
3). Choose one calibration mode in below table:
    For example: Operational amplifier-1, Normal mode offset cal low
    Configure like this:
    OPA0PD=0, OPA0LPM=0, OPA0CAL_H=0, OPA0CAL_L=1
4). Write trimming values in OPA0_TRIM_LOW. The writing value should be start from 00000b to the first value that causes the OPA0CALOUT bit set to 1.

**Note:** After writing trimming values into trimming register and before check CALOUT bit, software should wait for the $t_{offtrimmax}$ delay specified in datasheet.

When got the correct trimming values, the values must be kept in trimming register.

Repeat step 3 and 4 for complete the whole operational amplifier calibration procedure:
- Normal mode calibration high,
- Low-power mode calibration low,
- Low-power mode calibration high.

**Note:** During the whole process of calibration, the external connection of operational amplifier output must not pull-up or pull-down current higher than 500uA.

**Table 24-1. Operating mode and calibration**

| Mode | Control Bits | | | | Output | |
|---|---|---|---|---|---|---|
| | OPAxPD | OPAxLPM | OPAxCAL_H | OPAxCAL_L | $V_{OUT}$ | CALOUT |
| Normal | 0 | 0 | 0 | 0 | analog | 0 |
| | | | 1 | 1 | | |
| Low-power | 0 | 1 | 0 | 0 | analog | 0 |
| | | | 1 | 1 | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Power-down** | 1 | X | X | X | Z | 0 |
| **Offset cal-high** | 0 | X | 1 | 0 | analog | X |
| **Offset cal-low** | 0 | X | 0 | 1 | analog | X |

## 24.1.4. OPA register

### Control register (OPA_CTL)

Address offset: 0x5C

Reset value: 0x0001 0101

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPA2CALOUT | OPA1CALOUT | OPA0CALOUT | OPA_RANGE | S4OPA1 | Reserved | | | OPA2LPM | OPA2CAL_H | OPA2CAL_L | S3OPA2 | S2OPA2 | S1OPA2 | T3OPA2 | OPA2PD |
| r | r | r | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPA1LPM | OPA1CAL_H | OPA1CAL_L | S3OPA1 | S2OPA1 | S1OPA1 | T3OPA1 | OPA1PD | OPA0LPM | OPA0CAL_H | OPA0CAL_L | S3OPA0 | S2OPA0 | S1OPA0 | T3OPA0 | OPA0PD |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | OPA2CALOUT | OPA2 calibration output<br>When in calibration mode, if offset is trimmed the flag will assert. |
| 30 | OPA1CALOUT | OPA1 calibration output<br>When in calibration mode, if offset is trimmed the flag will assert. |
| 29 | OPA0CALOUT | OPA0 calibration output<br>When in calibration mode, if offset is trimmed the flag will assert. |
| 28 | OPA_RANGE | Power supply range<br>This bit can be set and cleared by software when the operational amplifiers are in powered down. It select the operational amplifier power supply range for stability<br>0: Low range ($V_{DDA}$ < 3.3 V)<br>1: High range ($V_{DDA}$ > 3.3 V) |
| 27 | S4OPA1 | S4 switch enable for OPA1<br>0: S4 switch opened<br>1: S4 switch closed |
| 26:24 | Reserved | Must be kept at reset value |
| 23 | OPA2LPM | OPA2 low power mode |

| | | |
|---|---|---|
| | | 0: OPA2 low power mode off<br>1: OPA2 low power mode on |
| 22 | OPA2CAL_H | OPA2 offset calibration for N diff<br>0: OPA2 offset calibration for N diff OFF<br>1: OPA2 offset calibration for N diff ON if OPA2CAL_L = 0 |
| 21 | OPA2CAL_L | OPA2 offset calibration for P diff<br>0: OPA2 offset calibration for P diff OFF<br>1: OPA2 offset calibration for P diff ON if OPA2CAL_H = 0 |
| 20 | S3OPA2 | S3 switch enable for OPA2<br>0: S3 switch opened<br>1: S3 switch closed |
| 19 | S2OPA2 | S2 switch enable for OPA2<br>0: S2 switch opened<br>1: S2 switch closed |
| 18 | S1OPA2 | S1 switch enable for OPA2<br>0: S1 switch opened<br>1: S1 switch closed |
| 17 | T3OPA2 | T3 switch enable for OPA2<br>0: T3 switch opened<br>1: T3 switch closed |
| 16 | OPA2PD | OPA2 power down<br>0: OPA2 enabled<br>1: OPA2 disabled |
| 15 | OPA1LPM | OPA1 low power mode<br>0: OPA1 low power mode off<br>1: OPA1 low power mode on |
| 14 | OPA1CAL_H | OPA1 offset calibration for N diff<br>0: OPA1 offset calibration for N diff OFF<br>1: OPA1 offset calibration for N diff ON if OPA1CAL_L = 0 |
| 13 | OPA1CAL_L | OPA1 offset calibration for P diff<br>0: OPA1 offset calibration for P diff OFF<br>1: OPA1 offset calibration for P diff ON if OPA1CAL_H = 0 |
| 12 | S3OPA1 | S3 switch enable for OPA1<br>0: S3 switch opened<br>1: S3 switch closed |
| 11 | S2OPA1 | S2 switch enable for OPA1<br>0: S2 switch opened |

| | | |
|---|---|---|
| | | 1: S2 switch closed |
| 10 | S1OPA1 | S1 switch enable for OPA1<br>0: S1 switch opened<br>1: S1 switch closed |
| 9 | T3OPA1 | T3 switch enable for OPA1<br>0: T3 switch opened<br>1: T3 switch closed |
| 8 | OPA1PD | OPA1 power down<br>0: OPA1 enabled<br>1: OPA1 disabled |
| 7 | OPA0LPM | OPA0 low power mode<br>0: OPA0 low power mode off<br>1: OPA0 low power mode on |
| 6 | OPA0CAL_H | OPA0 offset calibration for N diff<br>0: OPA0 offset calibration for N diff OFF<br>1: OPA0 offset calibration for N diff ON if OPA0CAL_L = 0 |
| 5 | OPA0CAL_L | OPA0 offset calibration for P diff<br>0: OPA0 offset calibration for P diff OFF<br>1: OPA0 offset calibration for P diff ON if OPA0CAL_H = 0 |
| 4 | S3OPA0 | S3 switch enable for OPA0<br>0: S3 switch opened<br>1: S3 switch closed |
| 3 | S2OPA0 | S2 switch enable for OPA0<br>0: S2 switch opened<br>1: S2 switch closed |
| 2 | S1OPA0 | S1 switch enable for OPA0<br>0: S1 switch opened<br>1: S1 switch closed |
| 1 | T3OPA0 | T3 switch enable for OPA0<br>0: T3 switch opened<br>1: T3 switch closed |
| 0 | OPA0PD | OPA0 power down<br>0: OPA0 enabled<br>1: OPA0 disabled |

### Bias trimming register for normal mode (OPA_BT)

Address offset: 0x60

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OT_USER | Reserved | OA2_TRIM_HIGH[4:0] | | | | | OA2_TRIM_LOW[4:0] | | | | | OA1_TRIM_HIGH[4:1] | | | |
| w | | rw | | | | | rw | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OA1_TRIM_HIGH[0] | OA1 _TRIM_LOW[4:0] | | | | | OA0 _TRIM_HIGH[4:0] | | | | | OA0 _TRIM_LOW[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | OT_USER | This bit is set and cleared by software; it is always read as 0. It is used to select if the OPAx offset is trimmed by the preset factory-programmed trimming values or the user programmed trimming value.<br>0: Trim the OPA offset using default factory values<br>1: Trim the OPA offset using user programmed values |
| 30 | Reserved | Must be kept at reset value |
| 29:25 | OA2_TRIM_HIGH[4:0] | OPA2, normal mode 5-bit offset trim value for NMOS pairs |
| 24:20 | OA2_TRIM_LOW[4:0] | OPA2, normal mode 5-bit offset trim value for PMOS pairs |
| 19:15 | OA1_TRIM_HIGH[4:0] | OPA1, normal mode 5-bit offset trim value for NMOS pairs |
| 14:10 | OA1_TRIM_LOW[4:0] | OPA1, normal mode 5-bit offset trim value for PMOS pairs |
| 9:5 | OA0_TRIM_HIGH[4:0] | OPA0, normal mode 5-bit offset trim value for NMOS pairs |
| 4:0 | OA0_TRIM_LOW[4:0] | OPA0, normal mode 5-bit offset trim value for PMOS pairs |

### Bias trimming register for low power mode (OPA_LPBT)

Address offset: 0x64

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 | | OA2_TRIM_LP_HIGH[4:0] | | | | | OA2_TRIM_ LP_LOW[4:0] | | | | | OA1_TRIM_ LP_HIGH[4:1] | | | |
| | | rw | | | | | rw | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OA1_TRIM_LP_HIGH[0] | OA1 _TRIM_ LP_LOW[4:0] | | | | | OA0 _TRIM_ LP_HIGH[4:0] | | | | | OA0 _TRIM_ LP_LOW[4:0] | | | | |
| rw | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29:25 | OA2_TRIM_LP_HIGH[4:0] | OPA2, low-power mode 5-bit offset trim value for NMOS pairs |
| 24:20 | OA2_TRIM_ LP_LOW[4:0] | OPA2, low-power mode 5-bit offset trim value for PMOS pairs |
| 19:15 | OA1_TRIM_ LP_HIGH[4:0] | OPA1, low-power mode 5-bit offset trim value for NMOS pairs |
| 14:10 | OA1_TRIM_ LP_LOW[4:0] | OPA1, low-power mode 5-bit offset trim value for PMOS pairs |
| 9:5 | OA0_TRIM_ LP_HIGH[4:0] | OPA0, low-power mode 5-bit offset trim value for NMOS pairs |
| 4:0 | OA0_TRIM_ LP_LOW[4:0] | OPA0, low-power mode 5-bit offset trim value for PMOS pairs |

## 24.2. Programmable Current and Voltage Reference (IVREF)

### 24.2.1. Introduction

An application programmable current reference module is included in the MCU.

When users want to use current reference, there are two different running modes are supplied.
One mode named Low Power Mode and another one named High Current Mode.
The difference between two modes is the current step and maximum current.
The former’s (LPM) step current is 1uA and the latter’s (HCM) 8uA.
The former’s (LPM) maximum current is 63uA and the latter’s (HCM) 504uA.

There is still a voltage reference module in the MCU which can provide a 2.5V voltage.

### 24.2.2. Main features

Current Reference feature:

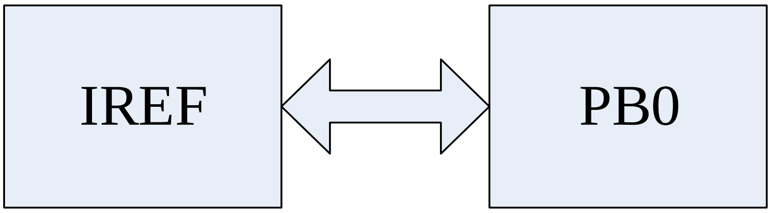
- Programmable current
- Programmable Source or sink
- Low Power Mode and High Current Mode

Voltage Reference feature:

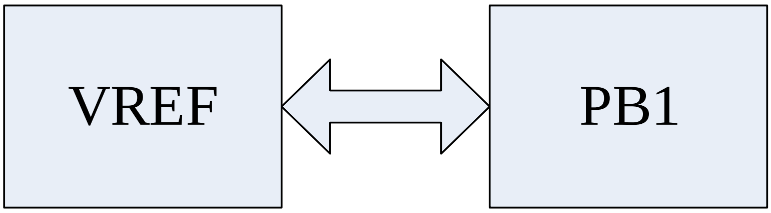
- User programmable trim

## 24.2.3. Function description

### Signal routing



When IREF is used, the PB0 pin should be configured to analog input mode.



When VREF is used, the PB1 pin should be configured as an analog I/O and external VREF decoupling capacitors are recommended.

### User Trimming

User can trim the IREF outputting current by programming IREF_TRIM and can trim the voltage by programming VREF_TRIM.

## 24.2.4. IVREF registers

### Control register (IVREF_CTL)

Address offset: 0x300

Reset value: 0x1000 0F00

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VREN | DECAP | Reserved | VPT[4:0] | | | | | Reserved | | | | | | | |
| rw | rw | | rw | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CREN | SSEL | Reserved | CPT[4:0] | | | | | SCMOD | Reserved | CSDT[5:0] | | | | | |
| rw | rw | | rw | | | | | rw | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | VREN | Voltage Reference Enable<br>0: Disable Voltage Reference<br>1: Enable Voltage Reference |

| | | |
|---|---|---|
| 30 | DECAP | Disconnect external capacitor<br>0: Connect external capacitor<br>1: Disconnect external capacitor |
| 29 | Reserved | Must be kept at reset value |
| 28:24 | VPT[4:0] | Voltage precision trim<br>00000: -6.4%<br>00001: -6.0%<br>….<br>11110: +5.6%<br>11111: +6% |
| 23:16 | Reserved | Must be kept at reset value |
| 15 | CREN | Current Reference Enable.<br>0: Disable current reference<br>1: Enable current reference |
| 14 | SSEL | Step selection<br>0: Low power, 1uA step<br>1: High current, 8uA step |
| 13 | Reserved | Must be kept at reset value |
| 12:8 | CPT[4:0] | Current precision trim<br>00000: -15%<br>….<br>11111: +16% |
| 7 | SCMOD | Sink current mode<br>0: Source current<br>1: Sink current |
| 6 | Reserved | Must be kept at reset value |
| 5:0 | CSDT[5:0] | Current step data<br>000000: Default value.<br>000001: Step * 1<br>….<br>111111: Step * 63 |

# 25. Controller area network (CAN)

**This chapter applies to GD32F170xx and GD32F190xx devices.**

## 25.1. Introduction

CAN bus (for controller area network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer.
The Basic Extended CAN (CAN), interfaces the CAN network. It supports the CAN protocols version 2.0A and B. The CAN interface handles the transmission and the reception of CAN frames fully autonomously. The CAN provides 28 scalable/configurable identifier filter banks. The filters are for selecting the incoming messages the software needs and discarding the others. Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware. Two receive FIFOs are used by hardware to store the incoming messages. The CAN controller also provides all hardware functions for supporting the time-triggered communication option for safety-critical applications.

## 25.2. Main features

- Supports CAN protocols version 2.0A, B
- Baud rates up to 1 Mbit/s
- Supports the time-triggered communication
- Maskable interrupts

**Transmission**

- 3 transmit mailboxes
- Prioritization of messages
- Time Stamp on SOF transmission

**Reception**

- 2 receive FIFOs with 3 messages deep
- 28 scalable/configurable identifier filter banks
- FIFO lock

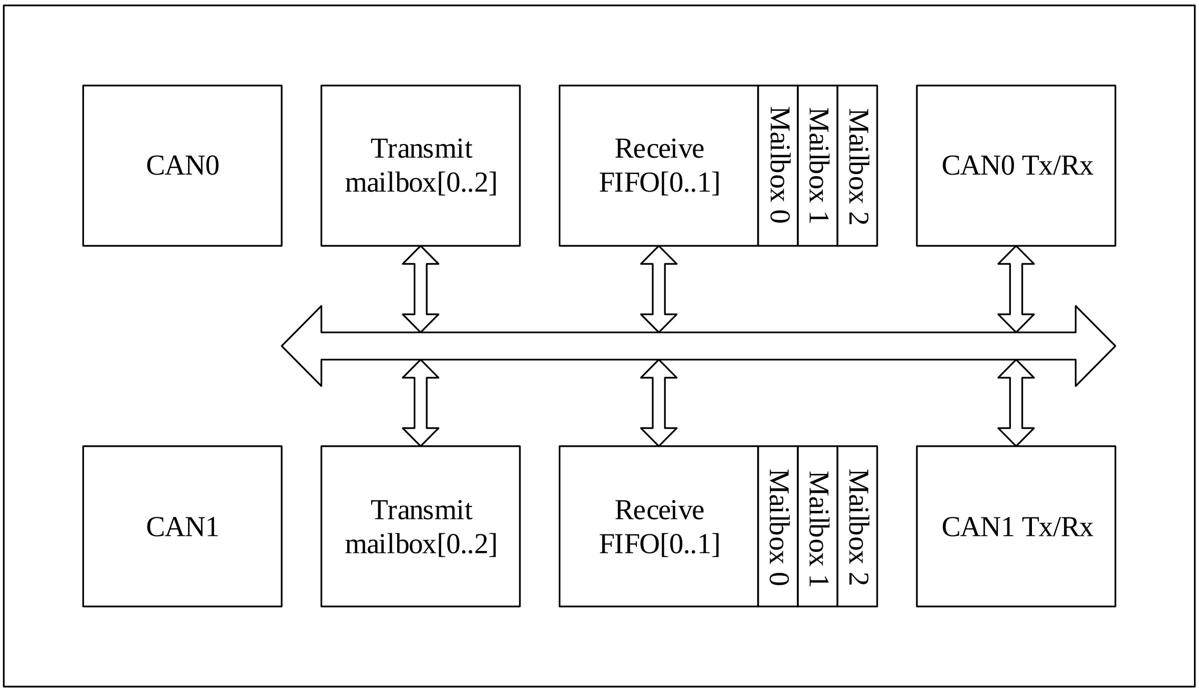
**Time-triggered communication**

- Disable retransmission automatically
- 16-bit free timer
- Time Stamp on SOF reception
- Time Stamp sent in last two data bytes

## 25.3. Function description

Figure below shows the CAN block diagram.

**Figure 25-1. CAN module block diagram**



### 25.3.1. Working mode

The CAN interface has three working modes:

- Sleep working mode
- Initial working mode
- Normal working mode

#### Sleep working mode

Sleep working mode is the default mode after reset. Also, sleep working mode is in the low-power status while the CAN clock is stopped.

When SLPWMOD bit in CAN_CTL register is set, the CAN enters the sleep working mode. Then the SLPWS bit in CAN_STAT register is set.

To leave sleep working mode automatically: the AWU bit in CAN_CTL register is set. To leave sleep working mode by software: clear the SLPWMOD bit in CAN_CTL register.

### Initial working mode

The CAN enters initial working mode whenever the options of CAN bus communication need to be changed.

Set IWMOD bit in CAN_CTL register to enter initial working mode or clear it in order to leave.

### Normal working mode

After initialization, the CAN can enter normal working mode and is ready to communicate with other CAN communication nodes.

To enter normal working mode: clear IWMOD bit in CAN_CTL register.

## 25.3.2. Communication modes

The CAN interface has four communication modes:

- Silent communication mode
- Loopback communication mode
- Loopback and silent communication mode
- Normal communication mode

### Silent communication mode

Silent communication mode means reception available and transmission disable.

The Rx pin of the CAN can get the signal from the network and the Tx pin always holds logical one.

When the SCMOD bit in CAN_BT register is set, the CAN enters the silent communication mode. When it is cleared, the CAN leaves silent communication mode.

Silent communication mode is useful on monitoring the network messages.

### Loopback communication mode

Loopback communication mode means the sending messages are transferred into the reception FIFOs.

Set LCMOD bit in CAN_BT register to enter loopback communication mode or clear it to leave.

Loopback communication mode is useful on self-test.

### Loopback and silent communication mode

Loopback and silent communication mode means the RX and TX pins are disconnected from the CAN network while the sending messages are transformed into the reception FIFOs.

Set LCMOD and SCMOD bit in CAN_BT register to enter loopback and silent communication mode or clear them to leave.

Loopback and silent communication mode is useful on self-test. The TX pin holds logical one. The RX pin holds high impedance state.

#### Normal communication mode

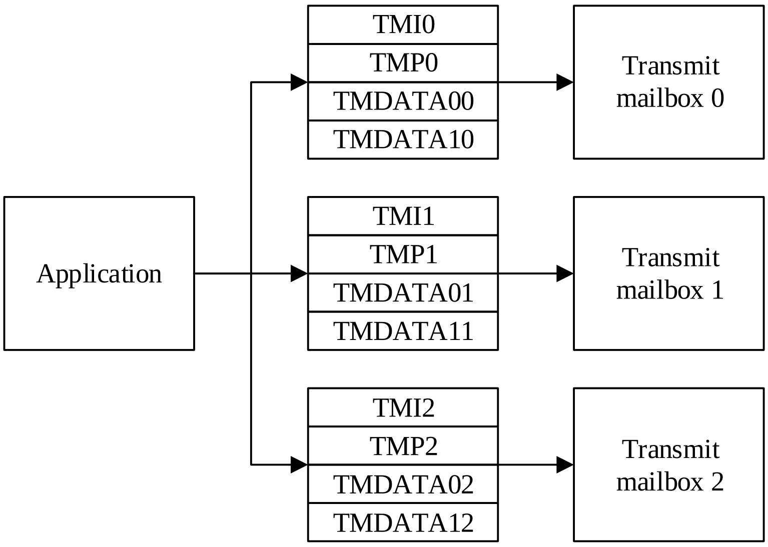
Normal communication mode is the default communication mode unless the LCMOD or SCMOD bit in CAN_BT register is set.

### 25.3.3. Data transmission

#### Transmission register

Three transmit mailboxes are transparent to the application. You can use transmit mailboxes through four registers: CAN_TMIx, CAN_TMPx, CAN_TMDATA0x and CAN_TMDATA1x. As shown in figure below.

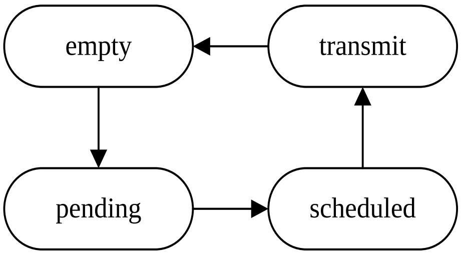
**Figure 25-2. Transmission register**



#### Transmit mailbox state

A transmit mailbox can be used when it is free: **empty** state. If the data is filled in the mailbox, setting TEN bit in CAN_TMIx register to prepare for starting the transmission: **pending** state. If more than one mailbox is in the pending state, they need schedule the transmission: **scheduled** state. A mailbox with priority enter **transmit** state and start transmitting the message. After the message has been sent, the mailbox is free: **empty** state. As shown in figure below.

Figure 25-3. State of transmission mailbox



### Transmit status and error

The CAN_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmits finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmits finished and no error. MTFNERR is set when the frame in the transmission mailbox has been sent.
- MAL: mailbox arbitration lost. MAL is set while the frame transmission is failed because of the arbitration lost.
- MTE: mailbox transmits error. MTE is set while the frame transmission is failed because of the detection error of CAN bus.

### Steps of sending a frame

To send a frame through the CAN:

Step 1: Select one free transmit mailbox.

Step 2: Fill four registers with the application’s acquirement.

Step 3: Set TEN bit in CAN_TMIx register.

Step 4: Check the transmit status. Typically, MTF and MTFNERR are set if transmission is successful.

### Transmission options

#### Abort

MST bit in CAN_TSTAT register can abort the transmission.

If the transmission mailbox’s state is **pending** or **scheduled**, the abort of transmission can be done immediately.

In the state of **transmit** the abort of transmission has two results. In case of transmission successful, the MTFNERR and MTF in CAN_TSTAT are set and state changes to **empty**. In case of transmission failed, the state changes to be **scheduled** and then the abort of transmission can be done immediately.

**Priority**

When more than one transmit mailbox is pending, the transmission order is given by the TFO bit in CAN_CTL register.

In case TFO is 1, the three transmit mailboxes work as FIFO.

In case TFO is 0, the transmit mailbox with lowest identifier has the highest priority of transmission. If the identifiers are equal, the lower mailbox number will be scheduled first.

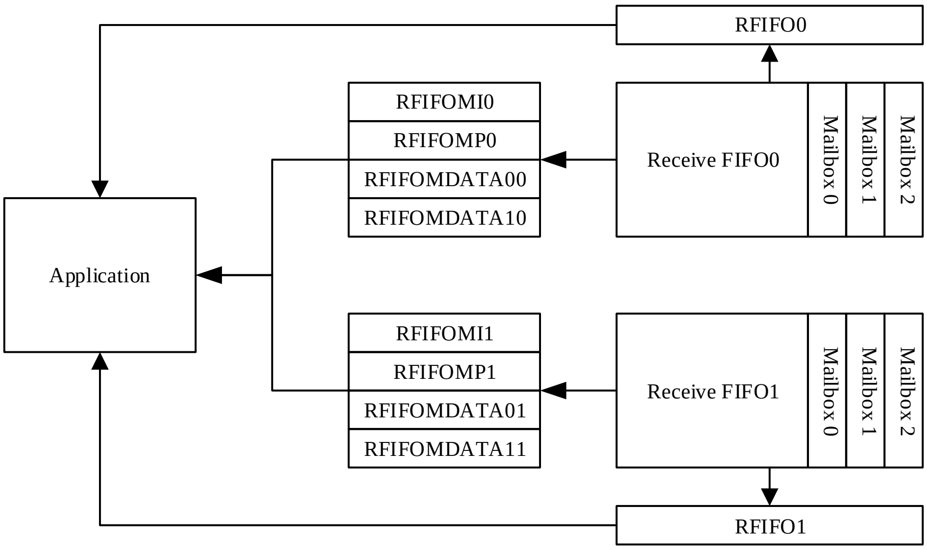
### 25.3.4. Data reception

#### Reception register

Two receive FIFOs are transparent to the application. You can use receive FIFOs through five registers: CAN_RFIFOx, CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x. FIFO’s status and operation can be handled by CAN_RFIFOx register. Reception frame data can be achieved through the registers: CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x.

Each FIFO consists of three receive mailboxes. As shown in figure below.

**Figure 25-4. Reception register**



#### Receive FIFO

Receive FIFO has three mailboxes. The reception frames are stored in the mailbox ordered by the arriving sequence of the frames. First arrived frame can be accessed by application firstly.

The number of frames in the receive FIFO and the status can be accessed by the register CAN_RFIFO0 and CAN_RFIFO1.

If at least one frame has been stored in the receive FIFO0, set RFD bit in CAN_RFIFO0 to read one frame from receive FIFO. The frame data is placed in the registers (CAN_RFIFOMI0, CAN_RFIFOMP0, CAN_RFIFOMDATA00, CAN_RFIFOMDATA10) until the RFD bit in CAN_RFIFO0 register is cleared.

### Receive FIFO status

RFL bit in CAN_RFIFOx register: receive FIFO length. It is 0 when no frame is stored in the reception FIFO and 3 when full.

RFF bit in CAN_RFIFOx register: the FIFO holds three frames.

RFO bit in CAN_RFIFOx register: one new frame arrived while the FIFO has hold three frames.

### Steps of receiving a message

Step 1: Check the number of frames in the receive FIFO.

Step 2: Set the RFD bit in CAN_RFIFOx register.

Step 3: Wait for reading CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x until the RFD bit is cleared.

## 25.3.5. Filtering Function

The CAN would receive frames from the CAN bus. If the frame is passed through the filter, it is copied into the receive FIFOs. Otherwise, the frame will be discarded without intervention by the software.

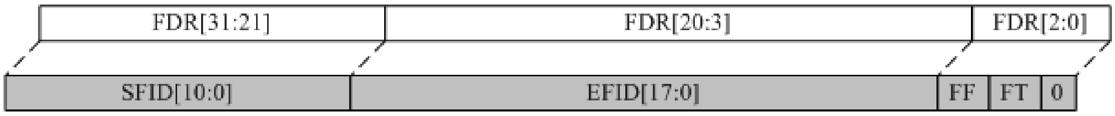
The identifier of frame from the CAN bus takes part in the matching of the filter.

### Scale

Each filter bank can be configured 32-bit or 16-bit.

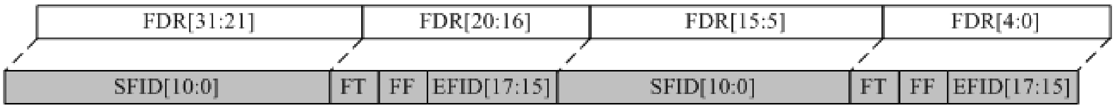
32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As shown in figure below.

**Figure 25-5. 32-bit filter**



16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As shown in figure below.

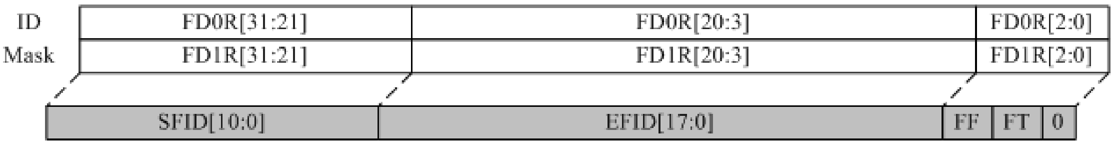
**Figure 25-6. 16-bit filter**



### Mask mode

In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”. 32-bit mask mode example is shown in figure below.

**Figure 25-7. 32-bit mask mode filter**

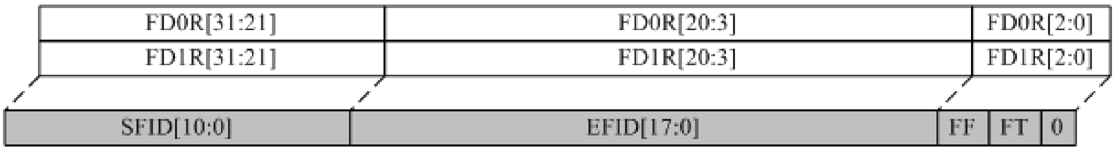


### List mode

The filter consists of frame identifiers. The filter can decide whether a frame will be discarded or not. When one frame arrived, the filter will check which member can match the identifier of the frame.

32-bit list mode example is shown in figure below.

**Figure 25-8. 32-bit list mode filter**

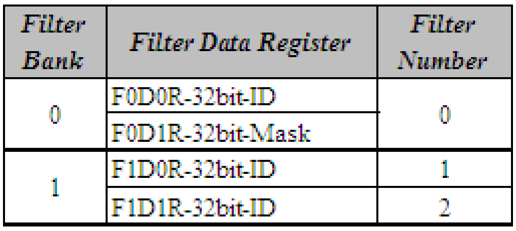


### Filter number

Each filter within a filter bank is numbered from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

For example, there are two filter banks. Bank 0 is configured as 32-bit mask mode. Bank 1 is configured as 16-bit list mode. The filter number is shown in figure below.

**Figure 25-9. 32-bit filter number**



| *Filter Bank* | *Filter Data Register* | *Filter Number* |
|---|---|---|
| 0 | F0D0R-32bit-ID | 0 |
|  | F0D1R-32bit-Mask |  |
| 1 | F1D0R-32bit-ID | 1 |
|  | F1D1R-32bit-ID | 2 |

### Associated FIFO

28 banks can be associated with FIFO0 or FIFO1. If the bank is associated with FIFO 0, the frames passed through the bank will fill the FIFO0.

### Active

The filter bank needs to be configured activation if the application wants the bank working and while filters not used by the application should be left deactivated.

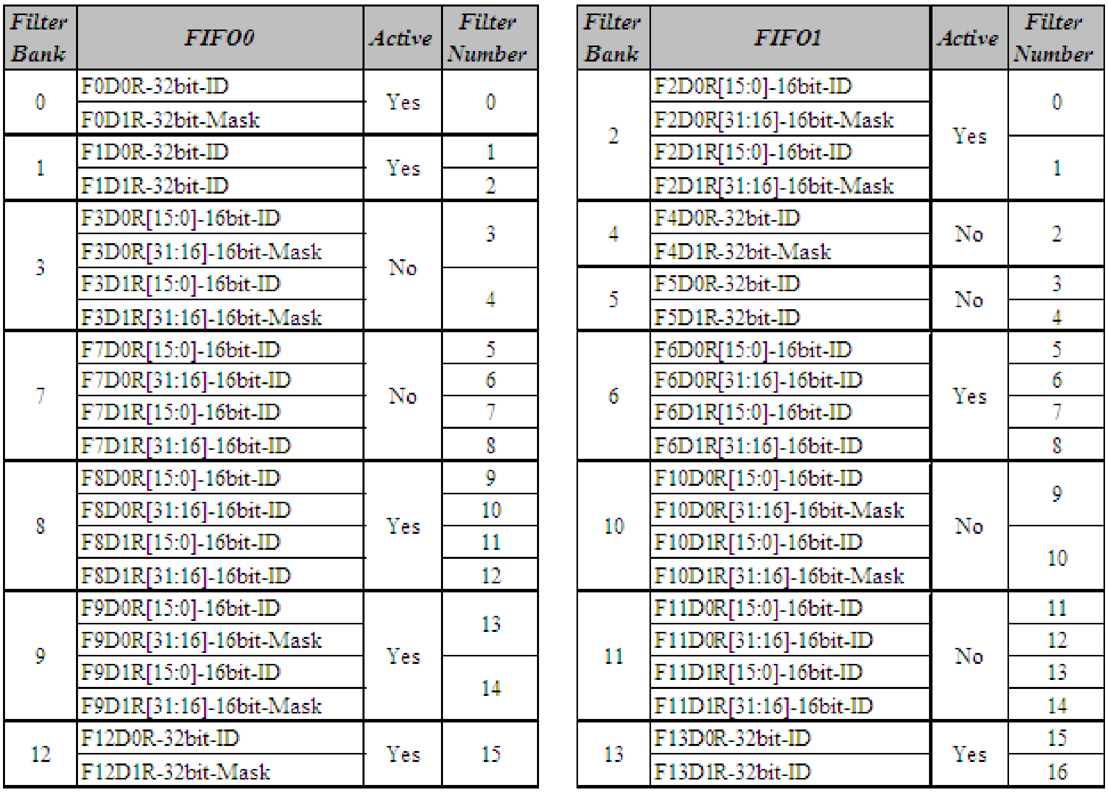
### Filtering index

Each filter number corresponds to a filtering rule. When the frame from the CAN bus passes the filters, a filter number must associate with the frame. The filter number is called filtering index. It stores in the FI bit in CAN_RFIFOMPx when the frame is read by the application.

Each FIFO numbers the filters within the banks associated with the FIFO itself whatever the bank is active or not.

The example about filtering index is shown in figure below.

**Figure 25-10. Filtering index**



| *Filter Bank* | *FIFO0* | *Active* | *Filter Number* |
|---|---|---|---|
| 0 | F0D0R-32bit-ID | Yes | 0 |
| | F0D1R-32bit-Mask | | |
| 1 | F1D0R-32bit-ID | Yes | 1 |
| | F1D1R-32bit-ID | | 2 |
| 3 | F3D0R[15:0]-16bit-ID | No | 3 |
| | F3D0R[31:16]-16bit-Mask | | |
| | F3D1R[15:0]-16bit-ID | | 4 |
| | F3D1R[31:16]-16bit-Mask | | |
| 7 | F7D0R[15:0]-16bit-ID | No | 5 |
| | F7D0R[31:16]-16bit-ID | | 6 |
| | F7D1R[15:0]-16bit-ID | | 7 |
| | F7D1R[31:16]-16bit-ID | | 8 |
| 8 | F8D0R[15:0]-16bit-ID | Yes | 9 |
| | F8D0R[31:16]-16bit-ID | | 10 |
| | F8D1R[15:0]-16bit-ID | | 11 |
| | F8D1R[31:16]-16bit-ID | | 12 |
| 9 | F9D0R[15:0]-16bit-ID | Yes | 13 |
| | F9D0R[31:16]-16bit-Mask | | |
| | F9D1R[15:0]-16bit-ID | | 14 |
| | F9D1R[31:16]-16bit-Mask | | |
| 12 | F12D0R-32bit-ID | Yes | 15 |
| | F12D1R-32bit-Mask | | |

| *Filter Bank* | *FIFO1* | *Active* | *Filter Number* |
|---|---|---|---|
| 2 | F2D0R[15:0]-16bit-ID | Yes | 0 |
| | F2D0R[31:16]-16bit-Mask | | |
| | F2D1R[15:0]-16bit-ID | | 1 |
| | F2D1R[31:16]-16bit-Mask | | |
| 4 | F4D0R-32bit-ID | No | 2 |
| | F4D1R-32bit-Mask | | |
| 5 | F5D0R-32bit-ID | No | 3 |
| | F5D1R-32bit-ID | | 4 |
| 6 | F6D0R[15:0]-16bit-ID | Yes | 5 |
| | F6D0R[31:16]-16bit-ID | | 6 |
| | F6D1R[15:0]-16bit-ID | | 7 |
| | F6D1R[31:16]-16bit-ID | | 8 |
| 10 | F10D0R[15:0]-16bit-ID | No | 9 |
| | F10D0R[31:16]-16bit-Mask | | |
| | F10D1R[15:0]-16bit-ID | | 10 |
| | F10D1R[31:16]-16bit-Mask | | |
| 11 | F11D0R[15:0]-16bit-ID | No | 11 |
| | F11D0R[31:16]-16bit-ID | | 12 |
| | F11D1R[15:0]-16bit-ID | | 13 |
| | F11D1R[31:16]-16bit-ID | | 14 |
| 13 | F13D0R-32bit-ID | Yes | 15 |
| | F13D1R-32bit-ID | | 16 |

### Priority

The filters have the priority:

1. 32-bit mode is higher than 16-bit mode.

2. List mode is higher than mask mode.
3. Smaller filter index value has the higher priority.

### 25.3.6. Time-triggered communication

The time-triggered CAN protocol is a higher layer protocol on top of the CAN (Controller Area Network) data link layer. Time-triggered communication means that activities are triggered by the elapsing of time segments. In a time-triggered communication system all points of time of message transmission are defined during the development of a system. A time-triggered communication system is ideal for applications in which the data traffic is of a periodic nature.

In this mode, the 16-bit internal counter of the CAN hardware is activated and used to generate the time stamp value stored in the CAN_RFIFOMPx and CAN_TMPx registers for reception and transmission respectively. The internal counter is incremented each CAN bit time. The internal counter is captured on the sample point of the SOF (Start of Frame) bit in both reception and transmission.

The automatic retransmission is disabled in the time-triggered CAN communication.

### 25.3.7. Communication parameters

#### Nonautomatic retransmission mode

This mode has been implemented in order to fulfill the requirement of the time-triggered communication option of the CAN standard. To configure the hardware in this mode the ARD bit in the CAN_CTL register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the frame transmission.

At the end of the first transmission attempt, the hardware considers the request as finished and sets the MTF bit in the CAN_TSTAT register. The result of the transmission is indicated in the CAN_TSTAT register by the MTFNERR, MAL and MTE bits.

#### Bit time

On the bit-level the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also means that a sophisticated method of bit synchronization is required. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception of the start bit available with each character, a synchronous transmission protocol there is just one start bit available at the beginning of a frame. To ensure the receiver to correctly read the messages, continuous resynchronization is required. Phase buffer segments are therefore inserted before and after the nominal sample point within a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagation from sender to receiver and back to the sender must be completed within one bit-time. For synchronization purposes a further time segment, the propagation delay segment, is needed in addition to the time reserved for synchronization, the phase buffer segments. The propagation delay segment takes into account the signal propagation on the bus as well as signal delays caused by transmitting and receiving nodes.

The normal bit time simplified by the CAN from the CAN protocol has three segments as follows:

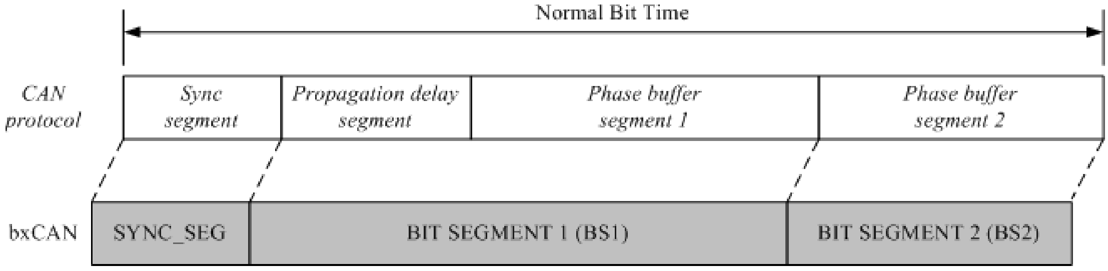
**Synchronization segment (SYNC_SEG)**: a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).

**Bit segment 1 (BS1)**: defines the location of the sample point. It includes the *Propagation delay segment* and *Phase buffer segment 1* of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.

**Bit segment 2 (BS2)**: defines the location of the transmit point. It represents the *Phase buffer segment 2* of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the figure below.

**Figure 25-11. The bit time**



The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

### Baud Rate

The CAN's clock derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$BaudRate = \frac{1}{Normal\ Bit\ Time}$$

$$Normal\ Bit\ Time = t_{SYNC\_SEG} + t_{BS1} + t_{BS2}$$

with:

$t_{SYNC\_SEG} = 1 \times t_q$

$t_{BS1} = (1 + BTR.TS1) \times t_q$

$t_{BS2} = (1 + BTR.TS2) \times t_q$

$t_q = (1 + BTR.BRP) \times t_{PCLK1}$

### 25.3.8. Error flags

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TECNT value, in CAN_ERR register) and a Receive Error Counter (RECNT value, in the CAN_ERR register), which get incremented or decremented according to the error condition. For detailed information about TECNT and RECNT management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network.

Furthermore, the CAN hardware provides detailed information on the current error status in CAN_ERR register. By means of the CAN_INTEN register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

#### Bus-Off recovery

The Bus-Off state is reached when TECNT is greater than 255. This state is indicated by BOERR bit in CAN_ERR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages.

Depending on the ABOR bit in the CAN_CTL register, CAN will recover from Bus-Off (becomes error active again) either automatically or on software request. But in both cases the CAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

If ABOR is set, the CAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOR is cleared, the software must initiate the recovering sequence by requesting CAN to enter and to leave initialization mode.

### 25.3.9. CAN interrupts

Four interrupt vectors are dedicated to CAN. Each interrupt source can be independently enabled or disabled seting or reseting related bits in CAN_INTEN.

The interrupt sources can be classified into:

- transmit interrupt
- FIFO0 interrupt
- FIFO1 interrupt
- error and status change interrupt

#### Transmit interrupt

The transmit interrupt can be generated by the following events:

- Transmit mailbox 0 becomes empty, MTF0 bit in the CAN_TSTAT register set.
- Transmit mailbox 1 becomes empty, MTF1 bit in the CAN_TSTAT register set.
- Transmit mailbox 2 becomes empty, MTF2 bit in the CAN_TSTAT register set.

#### FIFO0 interrupt

The FIFO0 interrupt can be generated by the following events:

- Reception of a new message, RFL0 bits in the CAN_RFIFO0 register are not ‘00’.
- FIFO0 full condition, RFF0 bit in the CAN_RFIFO0 register set.
- FIFO0 overrun condition, RFO0 bit in the CAN_RFIFO0 register set.

#### FIFO1 interrupt

The FIFO1 interrupt can be generated by the following events:

- Reception of a new message, RFL1 bits in the CAN_RFIFO1 register are not ‘00’.
- FIFO1 full condition, RFF1 bit in the CAN_RFIFO1 register set.
- FIFO1 overrun condition, RFO1 bit in the CAN_RFIFO1 register set.

#### Error and status change interrupt

The error and status change interrupt can be generated by the following events:

- Error condition, for more details on error conditions please refer to the CAN Error register (CAN_ERR).
- Wakeup condition, SOF monitored on the CAN RX signal.

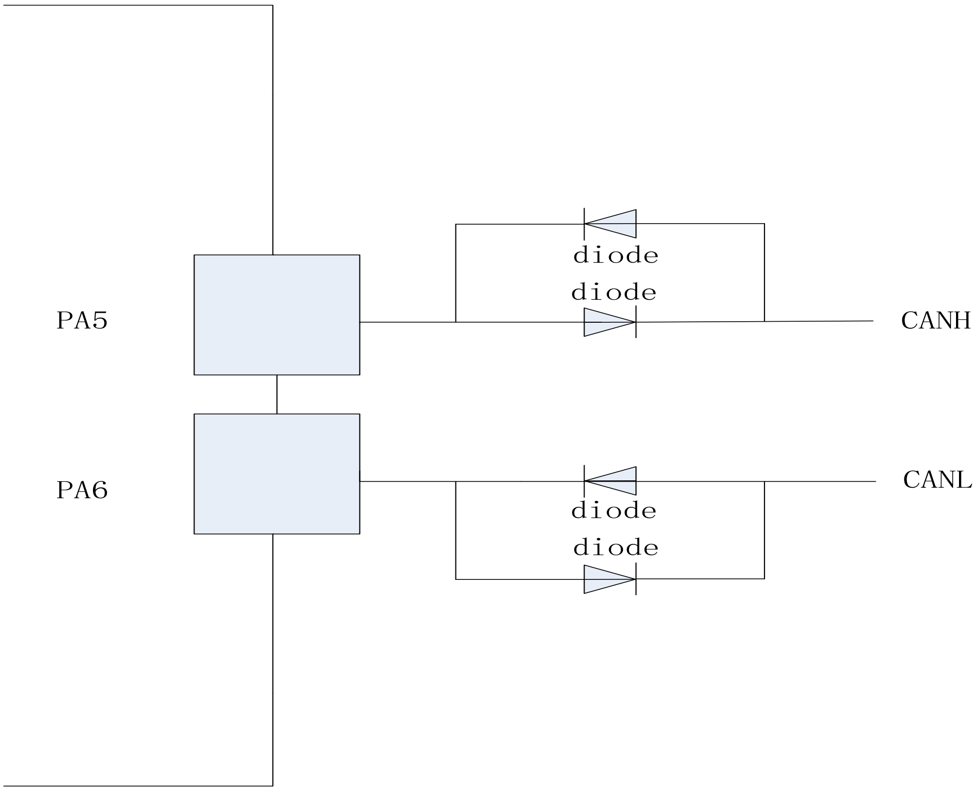
－ Enter into sleep mode.

### 25.3.10. CAN PHY mode

If set PHYEN bit in CAN_PHYCTL register, integrated CAN PHY is enabled. At this time the internal transceiver will begin to work. This mode only used for CAN0.

1. The connection outside of MCU chip is refer to the figure as follows:

**Figure 25-12. CAN PHY connection**



2. Configure the PA5/PA6 to analog input mode.

3. Enable the CAN clock.

4. Set PHYEN bit and PDMODE if needed in CAN_PHYCTL register.

5. Configure CAN registers as usual, same as not in CAN PHY mode.

## 25.4. CAN registers

### 25.4.1. Control register (CAN_CTL)

Address offset: 0x00

Reset value: 0x0001 0002

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 | | | | | | | | | | | | | | | DFZ |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWRST | Reserved | | | | | | | TTC | ABOR | AWU | ARD | RFOD | TFO | SLPWMOD | IWMOD |
| rs | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |
| 16 | DFZ | Debug freeze<br>0: CAN reception and transmission working during debug<br>1: CAN reception and transmission stop working during debug |
| 15 | SWRST | Software reset<br>0: Normal operation.<br>1: Reset CAN with working mode of sleep. This bit is automatically reset to 0. |
| 14:8 | Reserved | Must be kept at reset value |
| 7 | TTC | Time-triggered communication<br>0: Disable time-triggered communication<br>1: Enable time-triggered communication |
| 6 | ABOR | Automatic bus-off recovery<br>0: The bus-off state is left manually by software<br>1: The bus-off state is left automatically by hardware |
| 5 | AWU | Automatic wakeup<br>0: The sleeping working mode is left manually by software<br>1: The sleeping working mode is left automatically by hardware |
| 4 | ARD | Automatic retransmission disable<br>0: Enable Automatic retransmission<br>1: Disable Automatic retransmission |
| 3 | RFOD | Receive FIFO overwrite disable<br>0: Enable receive FIFO overwrite when receive FIFO is full and overwrite the FIFO with the incoming frame<br>1: Disable receive FIFO overwrite when receive FIFO is full and discard the incoming frame |
| 2 | TFO | Transmit FIFO order<br>0: Order with the identifier of the frame |

| | | |
|---|---|---|
| | | 1: Order with first in and first out |
| 1 | SLPWMOD | Sleep working mode<br>0: Disable sleep working mode<br>1: Enable sleep working mode |
| 0 | IWMOD | Initial working mode<br>0: Disable initial working mode<br>1: Enable initial working mode |

### 25.4.2. Status register (CAN_STAT)

Address offset: 0x04
Reset value: 0x0000 0C02

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 | | | | RXL | LASTRX | RS | TS | Reserved | | | SLPIF | WUIF | ERRIF | SLPWS | IWS |
| | | | | r | r | r | r | | | | rc_w1 | rc_w1 | rc_w1 | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11 | RXL | RX level |
| 10 | LASTRX | Last sample value of Rx pin |
| 9 | RS | Receiving state<br>0: CAN is not working in the receiving state<br>1: CAN is working in the receiving state |
| 8 | TS | Transmitting state<br>0: CAN is not working in the transmitting state<br>1: CAN is working in the transmitting state |
| 7:5 | Reserved | Must be kept at reset value |
| 4 | SLPIF | Status change interrupt flag of sleep working mode entering<br>0: CAN is not entering the sleep working mode<br>1: CAN is entering the sleep working mode. This bit is set while the interrupt is enabling. |
| 3 | WUIF | Status change interrupt flag of wakeup from sleep working mode<br>0: Wakeup event is not coming |

| | | |
|---|---|---|
| | | 1: Wakeup event is coming. This bit is set while the interrupt is enabling. |
| 2 | ERRIF | Error interrupt flag<br>0: No error interrupt<br>1: Any error interrupt has happened while the interrupt is enabling |
| 1 | SLPWS | Sleep working state<br>0: CAN is not the state of sleep working mode<br>1: CAN is the state of sleep working mode |
| 0 | IWS | Initial working state<br>0: CAN is not the state of initial working mode<br>1: CAN is the state of initial working mode |

### 25.4.3. Transmit status register (CAN_TSTAT)

Address offset: 0x08

Reset value: 0x1C00 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TMLS2 | TMLS1 | TMLS0 | TME2 | TME1 | TME0 | NUM[1:0] | MST2 | Reserved | MTE2 | MAL2 | MTFNERR2 | MTF2 |
| r | r | r | r | r | r | r | rs | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| 15 | 14 13 12 | 11 | 10 | 9 | 8 | 7 | 6 5 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MST1 | Reserved | MTE1 | MAL1 | MTFNERR1 | MTF1 | MST0 | Reserved | MTE0 | MAL0 | MTFNERR0 | MTF0 |
| rs | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rs | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TMLS2 | Transmit mailbox 2 last sending in transmit FIFO<br>This bit is set by hardware when transmit mailbox 2 has the last sending order in the transmit FIFO with at least two frame are pending. |
| 30 | TMLS1 | Transmit mailbox 1 last sending in transmit FIFO<br>This bit is set by hardware when transmit mailbox 1 has the last sending order in the transmit FIFO with at least two frame are pending. |
| 29 | TMLS0 | Transmit mailbox 0 last sending in transmit FIFO<br>This bit is set by hardware when transmit mailbox 0 has the last sending order in the transmit FIFO with at least two frame are pending. |
| 28 | TME2 | Transmit mailbox 2 empty<br>0: Transmit mailbox 2 not empty<br>1: Transmit mailbox 2 empty |
| 27 | TME1 | Transmit mailbox 1 empty<br>0: Transmit mailbox 1 not empty |

| | | |
|---|---|---|
| | | 1: Transmit mailbox 1 empty |
| 26 | TME0 | Transmit mailbox 0 empty<br>0: Transmit mailbox 0 not empty<br>1: Transmit mailbox 0 empty |
| 25:24 | NUM[1:0] | These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty.<br>These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted lastly if all mailboxes are full. |
| 23 | MST2 | Mailbox 2 stop transmitting<br>This bit is set by the software to stop mailbox 2 transmitting.<br>This bit is reset by the hardware while the mailbox 2 is empty. |
| 22:20 | Reserved | Must be kept at reset value |
| 19 | MTE2 | Mailbox 2 transmit error<br>This bit is set while the transmit error is occurred. |
| 18 | MAL2 | Mailbox 2 arbitration lost<br>This bit is set while the arbitration lost is occurred. |
| 17 | MTFNERR2 | Mailbox 2 transmit finished and no error<br>0: Mailbox 2 transmit finished with error<br>1: Mailbox 2 transmit finished and no error |
| 16 | MTF2 | Mailbox 2 transmit finished<br>0: Mailbox 2 transmit is progressing<br>1: Mailbox 2 transmit finished |
| 15 | MST1 | Mailbox 1 stop transmitting<br>This bit is set by the software to stop mailbox 1 transmitting.<br>This bit is reset by the hardware while the mailbox 1 is empty. |
| 14:12 | Reserved | Must be kept at reset value |
| 11 | MTE1 | Mailbox 1 transmit error<br>This bit is set while the transmit error is occurred. |
| 10 | MAL1 | Mailbox 1 arbitration lost<br>This bit is set while the arbitration lost is occurred. |
| 9 | MTFNERR1 | Mailbox 1 transmit finished and no error<br>0: Mailbox 1 transmit finished with error<br>1: Mailbox 1 transmit finished and no error |
| 8 | MTF1 | Mailbox 1 transmit finished<br>0: Mailbox 1 transmit is progressing<br>1: Mailbox 1 transmit finished |

| | | |
|---|---|---|
| 7 | MST0 | Mailbox 0 stop transmitting<br>This bit is set by the software to stop mailbox 0 transmitting.<br>This bit is reset by the hardware while the mailbox 0 is empty. |
| 6:4 | Reserved | Must be kept at reset value |
| 3 | MTE0 | Mailbox 0 transmit error<br>This bit is set while the transmit error is occurred. |
| 2 | MAL0 | Mailbox 0 arbitration lost<br>This bit is set while the arbitration lost is occurred. |
| 1 | MTFNERR0 | Mailbox 0 transmit finished and no error<br>0: Mailbox 0 transmit finished with error<br>1: Mailbox 0 transmit finished and no error |
| 0 | MTF0 | Mailbox 0 transmit finished<br>0: Mailbox 0 transmit is progressing<br>1: Mailbox 0 transmit finished |

### 25.4.4. Receive message FIFO0 register (CAN_RFIFO0)

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 | | | | | | | | | | RFD0 | RFO0 | RFF0 | Reserved | RFL0[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value |
| 5 | RFD0 | Receive FIFO 0 dequeue<br>This bit is set by the software to start dequeuing a frame from receive FIFO 0.<br>This bit is reset by the hardware while the dequeuing is done. |
| 4 | RFO0 | Receive FIFO 0 overfull<br>0: The receive FIFO 0 is not overfull<br>1: The receive FIFO 0 is overfull |
| 3 | RFF0 | Receive FIFO 0 full<br>0: The receive FIFO 0 is not full |

| | | |
|---|---|---|
| | | 1: The receive FIFO 0 is full |
| 2 | Reserved | Must be kept at reset value |
| 1:0 | RFL0[1:0] | Receive FIFO 0 length<br>These bits are the length of the receive FIFO0. |

### 25.4.5. Receive message FIFO1 register (CAN_RFIFO1)

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 | | | | | | | | | | RFD1 | RFO1 | RFF1 | Reserved | RFL1[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value |
| 5 | RFD1 | Receive FIFO1 dequeue<br>This bit is set by the software to start dequeuing a frame from receive FIFO1.<br>This bit is reset by the hardware while the dequeuing is done. |
| 4 | RFO1 | Receive FIFO1 overfull<br>0: The receive FIFO1 is not overfull<br>1: The receive FIFO1 is overfull |
| 3 | RFF1 | Receive FIFO1 full<br>0: The receive FIFO1 is not full<br>1: The receive FIFO1 is full |
| 2 | Reserved | Must be kept at reset value |
| 1:0 | RFL1[1:0] | Receive FIFO1 length<br>These bits are the length of the receive FIFO1. |

### 25.4.6. Interrupt enable register (CAN_INTEN)

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 | | | | | | | | | | | | | | SLPWIE | WIE |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRIE | Reserved | | | ERRNIE | BOIE | PERRIE | WERRIE | Reserved | RFOIE1 | RFFIE1 | RFNEIE1 | RFOIE0 | RFFIE0 | RFNEIE0 | TMEIE |
| rw | | | | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value |
| 17 | SLPWIE | Sleep working interrupt enable<br>0: Sleep working interrupt disable<br>1: Sleep working interrupt enable |
| 16 | WIE | Wakeup interrupt enable<br>0: Wakeup interrupt disable<br>1: Wakeup interrupt enable |
| 15 | ERRIE | Error interrupt enable<br>0: Error interrupt disable<br>1: Error interrupt enable |
| 14:12 | Reserved | Must be kept at reset value |
| 11 | ERRNIE | Error number interrupt enable<br>0: Error number interrupt disable<br>1: Error number interrupt enable |
| 10 | BOIE | Bus-off interrupt enable<br>0: Bus-off interrupt disable<br>1: Bus-off interrupt enable |
| 9 | PERRIE | Passive error interrupt enable<br>0: Passive error interrupt disable<br>1: Passive error interrupt enable |
| 8 | WERRIE | Warning error interrupt enable<br>0: Warning error interrupt disable<br>1: Warning error interrupt enable |
| 7 | Reserved | Must be kept at reset value |
| 6 | RFOIE1 | Receive FIFO1 overfull interrupt enable<br>0: Receive FIFO1 overfull interrupt disable<br>1: Receive FIFO1 overfull interrupt enable |

| | | |
|---|---|---|
| 5 | RFFIE1 | Receive FIFO1 full interrupt enable<br>0: Receive FIFO1 full interrupt disable<br>1: Receive FIFO1 full interrupt enable |
| 4 | RFNEIE1 | Receive FIFO1 not empty interrupt enable<br>0: Receive FIFO1 not empty interrupt disable<br>1: Receive FIFO1 not empty interrupt enable |
| 3 | RFOIE0 | Receive FIFO0 overfull interrupt enable<br>0: Receive FIFO0 overfull interrupt disable<br>1: Receive FIFO0 overfull interrupt enable |
| 2 | RFFIE0 | Receive FIFO0 full interrupt enable<br>0: Receive FIFO0 full interrupt disable<br>1: Receive FIFO0 full interrupt enable |
| 1 | RFNEIE0 | Receive FIFO0 not empty interrupt enable<br>0: Receive FIFO0 not empty interrupt disable<br>1: Receive FIFO0 not empty interrupt enable |
| 0 | TMEIE | Transmit mailbox empty interrupt enable<br>0: Transmit mailbox empty interrupt disable<br>1: Transmit mailbox empty interrupt enable |

## 25.4.7. Error register (CAN_ERR)

Address offset: 0x18

Reset value: 0x0000 0000

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RECNT[7:0]</td><td colspan="8">TECNT[7:0]</td></tr>
<tr><td colspan="8">r</td><td colspan="8">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="9">Reserved</td><td colspan="3">ERRN[2:0]</td><td>Reserved</td><td>BOERR</td><td>PERR</td><td>WERR</td></tr>
<tr><td colspan="9"></td><td colspan="3">rw</td><td></td><td>r</td><td>r</td><td>r</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | RECNT[7:0] | Receive Error Count |
| 23:16 | TECNT[7:0] | Transmit Error Count |
| 15:7 | Reserved | Must be kept at reset value |
| 6:4 | ERRN[2:0] | Error number<br>These bits indicate the error status of bit transformation. They are updated by the hardware. While the bit transformation is successful, they are equal to 0. Software |

| | | |
|---|---|---|
| | | can set these bits to 0b111.<br>000: No Error<br>001: Stuff Error<br>010: Form Error<br>011: Acknowledgment Error<br>100: Bit recessive Error<br>101: Bit dominant Error<br>110: CRC Error<br>111: Set by software |
| 3 | Reserved | Must be kept at reset value |
| 2 | BOERR | Bus-off error<br>Whenever the CAN enters buf-off state, the bit will be set by the hardware. |
| 1 | PERR | Passive error<br>Whenever the TECNT or RECNT is greater than 127, the bit will be set by the hardware. |
| 0 | WERR | Warning error<br>Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by the hardware. |

## 25.4.8. Bit timing register (CAN_BT)

Address offset: 0x1C

Reset value: 0x0123 0000

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>SCMOD</td><td>LCMOD</td><td colspan="4">Reserved</td><td colspan="2">SJW[1:0]</td><td>Reserved</td><td colspan="3">BS2[2:0]</td><td colspan="4">BS1[3:0]</td></tr>
<tr><td>rw</td><td>rw</td><td colspan="4"></td><td colspan="2">rw</td><td></td><td colspan="3">rw</td><td colspan="4">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">BAUDPSC[9:0]</td></tr>
<tr><td colspan="6"></td><td colspan="10">rw</td></tr>
</table>

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | SCMOD | Silent communication mode<br>0: Silent communication disable<br>1: Silent communication enable |
| 30 | LCMOD | Loopback communication mode<br>0: Loopback communication disable<br>1: Loopback communication enable |

| | | |
|---|---|---|
| 29:26 | Reserved | Must be kept at reset value |
| 25:24 | SJW[1:0] | Resynchronization jump width<br>Resynchronization jump width time quantum= SJW[1:0]+1 |
| 23 | Reserved | Must be kept at reset value |
| 22:20 | BS2[2:0] | Bit segment 2<br>Bit segment 2 time quantum=BS2[2:0]+1 |
| 19:16 | BS1[3:0] | Bit segment 1<br>Bit segment 1 time quantum=BS1[3:0]+1 |
| 15:10 | Reserved | Must be kept at reset value |
| 9:0 | BAUDPSC[9:0] | Baud rate prescaler<br>The CAN baud rate prescaler |

### 25.4.9. Transmit mailbox identifier register (CAN_TMIx) (x=0..2)

Address offset: 0x180, 0x190, 0x1A0

Reset value: 0xXXXX XXXX (bit0=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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SFID[10:0]/EFID[28:18] | | | | | | | | | | | EFID[17:13] | | | | |
| rw | | | | | | | | | | | rw | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | SFID[10:0]/EFID[28:18] | The frame identifier<br>SFID[10:0]: Standard format frame identifier<br>EFID[28:18]: Extended format frame identifier |
| 20:16 | EFID[17:13] | The frame identifier<br>EFID[17:13]: Extended format frame identifier |
| 15:3 | EFID[12:0] | The frame identifier<br>EFID[12:0]: Extended format frame identifier |
| 2 | FF | Frame format<br>0: Standard format frame<br>1: Extended format frame |
| 1 | FT | Frame type<br>0: Data frame |

| | | |
|---|---|---|
| | | 1: Remote frame |
| 0 | TEN | Transmit enable<br>0: Transmit disable<br>1: Transmit enable<br>This bit is set by the software when one frame will be transmitted and reset by the hardware when the transmit mailbox is empty. |

### 25.4.10. Transmit mailbox property register (CAN_TMPx) (x=0..2)

Address offset: 0x184, 0x194, 0x1A4

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TS[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | TS[15:0] | Time stamp<br>The time stamp of frame in transmit mailbox. |
| 15:9 | Reserved | Must be kept at reset value |
| 8 | TSEN | Time stamp enable<br>0: Time stamp disable<br>1: Time stamp enable. The TS[15:0] will be transmitted in the DATA6 and DATA7 in DL.<br>This bit is available while the TTC bit in CAN_CTL is set. |
| 7:4 | Reserved | Must be kept at reset value |
| 3:0 | DLENC[3:0] | Data length code<br>DLENC[3:0] is the number of bytes in a frame. |

### 25.4.11. Transmit mailbox data0 register (CAN_TMDATA0x) (x=0..2)

Address offset: 0x188, 0x198, 0x1A8

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB3[7:0] | | | | | | | | DB2[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB1[7:0] | | | | | | | | DB0[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB3[7:0] | Data byte 3 |
| 23:16 | DB2[7:0] | Data byte 2 |
| 15:8 | DB1[7:0] | Data byte 1 |
| 7:0 | DB0[7:0] | Data byte 0 |

### 25.4.12. Transmit mailbox data1 register (CAN_TMDATA1x) (x=0..2)

Address offset: 0x18C, 0x19C, 0x1AC
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB7[7:0] | | | | | | | | DB6[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB5[7:0] | | | | | | | | DB4[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB7[7:0] | Data byte 7 |
| 23:16 | DB6[7:0] | Data byte 6 |
| 15:8 | DB5[7:0] | Data byte 5 |
| 7:0 | DB4[7:0] | Data byte 4 |

### 25.4.13. Receive FIFO mailbox identifier register (CAN_RFIFOMIx) (x=0..1)

Address offset: 0x1B0, 0x1C0
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SFID[10:0]/EFID[28:18] | | | | | | | | | | | EFID[17:13] | | | | |
| r | | | | | | | | | | | r | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | SFID[10:0]/EFID[28:18] | The frame identifier<br>SFID[10:0]: Standard format frame identifier<br>EFID[28:18]: Extended format frame identifier |
| 20:16 | EFID[17:13] | The frame identifier<br>EFID[17:13]: Extended format frame identifier |
| 15:3 | EFID[12:0] | The frame identifier<br>EFID[12:0]: Extended format frame identifier |
| 2 | FF | Frame format<br>0: Standard format frame<br>1: Extended format frame |
| 1 | FT | Frame type<br>0: Data frame<br>1: Remote frame |
| 0 | Reserved | Must be kept at reset value |

### 25.4.14. Receive FIFO mailbox property register (CAN_RFIFOMPx) (x=0..1)

Address offset: 0x1B4, 0x1C4
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TS[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

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

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | TS[15:0] | Time stamp<br>The time stamp of frame in transmit mailbox. |

| | | |
|---|---|---|
| 15:8 | FI[7:0] | Filtering index<br>The index of the filter by which the frame is passed. |
| 7:4 | Reserved | Must be kept at reset value |
| 3:0 | DLENC[3:0] | Data length code<br>DLENC[3:0] is the number of bytes in a frame. |

### 25.4.15. Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x=0..1)

Address offset: 0x1B8, 0x1C8
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB3[7:0] | | | | | | | | DB2[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB1[7:0] | | | | | | | | DB0[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB3[7:0] | Data byte 3 |
| 23:16 | DB2[7:0] | Data byte 2 |
| 15:8 | DB1[7:0] | Data byte 1 |
| 7:0 | DB0[7:0] | Data byte 0 |

### 25.4.16. Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x=0..1)

Address offset: 0x1BC, 0x1CC
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB7[7:0] | | | | | | | | DB6[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB5[7:0] | | | | | | | | DB4[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

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

| | | |
|---|---|---|
| 31:24 | DB7[7:0] | Data byte 7 |
| 23:16 | DB6[7:0] | Data byte 6 |
| 15:8 | DB5[7:0] | Data byte 5 |
| 7:0 | DB4[7:0] | Data byte 4 |

## 25.4.17. Filter control register (CAN_FCTL)

Address offset: 0x200

Reset value: 0x2A1C 0E01

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 | | HBC1F[5:0] | | | | | | Reserved | | | | | | | FLD |
| | | rw | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value |
| 13:8 | HBC1F[5:0] | Header bank of CAN1 filter |
| 7:1 | Reserved | Must be kept at reset value |
| 0 | FLD | Filter lock disable<br>0: Filter lock enable<br>1: Filter lock disable |

## 25.4.18. Filter mode configuration register (CAN_FMCFG)

Address offset: 0x204

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 | | | | FMOD27 | FMOD26 | FMOD25 | FMOD24 | FMOD23 | FMOD22 | FMOD21 | FMOD20 | FMOD19 | FMOD18 | FMOD17 | FMOD16 |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMOD15 | FMOD14 | FMOD13 | FMOD12 | FMOD11 | FMOD10 | FMOD9 | FMOD8 | FMOD7 | FMOD6 | FMOD5 | FMOD4 | FMOD3 | FMOD2 | FMOD1 | FMOD0 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | FMODx | Filter mode<br>0: Filter x with Mask mode<br>1: Filter x with List mode |

## 25.4.19. Filter scale configuration register (CAN_FSCFG)

Address offset: 0x20C

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 | | | | FS27 | FS26 | FS25 | FS24 | FS23 | FS22 | FS21 | FS20 | FS19 | FS18 | FS17 | FS16 |
| | | | | 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 |
| FS15 | FS14 | FS13 | FS12 | FS11 | FS10 | FS9 | FS8 | FS7 | FS6 | FS5 | FS4 | FS3 | FS2 | FS1 | FS0 |
| 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 | FSx | Filter scale<br>0: Filter x with 16-bit scale<br>1: Filter x with 32-bit scale |

## 25.4.20. Filter associated FIFO register (CAN_FAFIFO)

Address offset: 0x214

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 | | | | FAF27 | FAF26 | FAF25 | FAF24 | FAF23 | FAF22 | FAF21 | FAF20 | FAF19 | FAF18 | FAF17 | FAF16 |
| | | | | 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 |
| FAF15 | FAF14 | FAF13 | FAF12 | FAF11 | FAF10 | FAF9 | FAF8 | FAF7 | FAF6 | FAF5 | FAF4 | FAF3 | FAF2 | FAF1 | FAF0 |
| 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 | FAFx | Filter associated FIFO<br>0: Filter x associated with FIFO0<br>1: Filter x associated with FIFO1 |

## 25.4.21. Filter working register (CAN_FW)

Address offset: 0x21C

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 | | | | FW27 | FW26 | FW25 | FW24 | FW23 | FW22 | FW21 | FW20 | FW19 | FW18 | FW17 | FW16 |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FW15 | FW14 | FW13 | FW12 | FW11 | FW10 | FW9 | FW8 | FW7 | FW6 | FW5 | FW4 | FW3 | FW2 | FW1 | FW0 |
| 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 | FWx | Filter working<br>0: Filter x working disable<br>1: Filter x working enable |

## 25.4.22. Filter x data y register (CAN_FxDATAy) (x=0..27, y=0..1)

Address offset: 0x240+8*x+4*y, (x=0..27,y=0,1)

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FD31 | FD30 | FD29 | FD28 | FD27 | FD26 | FD25 | FD24 | FD23 | FD22 | FD21 | FD20 | FD19 | FD18 | FD17 | FD16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FD15 | FD14 | FD13 | FD12 | FD11 | FD10 | FD9 | FD8 | FD7 | FD6 | FD5 | FD4 | FD3 | FD2 | FD1 | FD0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FDx | Filter data<br>Mask mode<br>0: Mask match disable<br>1: Mask match enable<br><br>List mode<br>0: List identifier bit is 0<br>1: List identifier bit is 1 |

### 25.4.23. PHY control register (CAN_PHYCTL)

Address offset: 0x3FC

Reset value: 0x0000 0300

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 | | | | | | POMOD | | Reserved | | | | | | | PHYEN |
| | | | | | | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:8 | POMOD | CAN PHY output driver control register<br>This bit set and cleared by software.<br>00: Low Slope Mode<br>01: Middle Slope Mode<br>10: High Slope Mode<br>11: High Speed Mode(Default) |
| 7:1 | Reserved | Must be kept at reset value |
| 0 | PHYEN | PHY enable bit<br>This bit set and cleared by software. This bit can use CAN PHY for CAN0 or not.<br>0: No CAN PHY mode<br>1: CAN PHY enable for CAN0 |

# 26. Revision history

**Table 26-1. Revision history**

| Revision No. | Description | Date |
|---|---|---|
| 1.0 | Initial Release | Mar.18, 2014 |
| 2.0 | Add GD32F170/190 Products | Jan.15, 2016 |
| 3.0 | Adapt To New Name Convention | Jun.24, 2016 |
|  |  |  |