# 1 Introduction

This technical reference manual serves as a supplement to the SC32F10T/10G datasheet (SC32F10T_10G_Datasheet vx.x), providing the necessary information for applications, especially software development. For details regarding the functional features, ordering information, as well as mechanical and electrical characteristics of specific SC32F10T/10G devices, please refer to their respective datasheets.

# Content

# 2 Document Conventions

## 2.1 Glossary

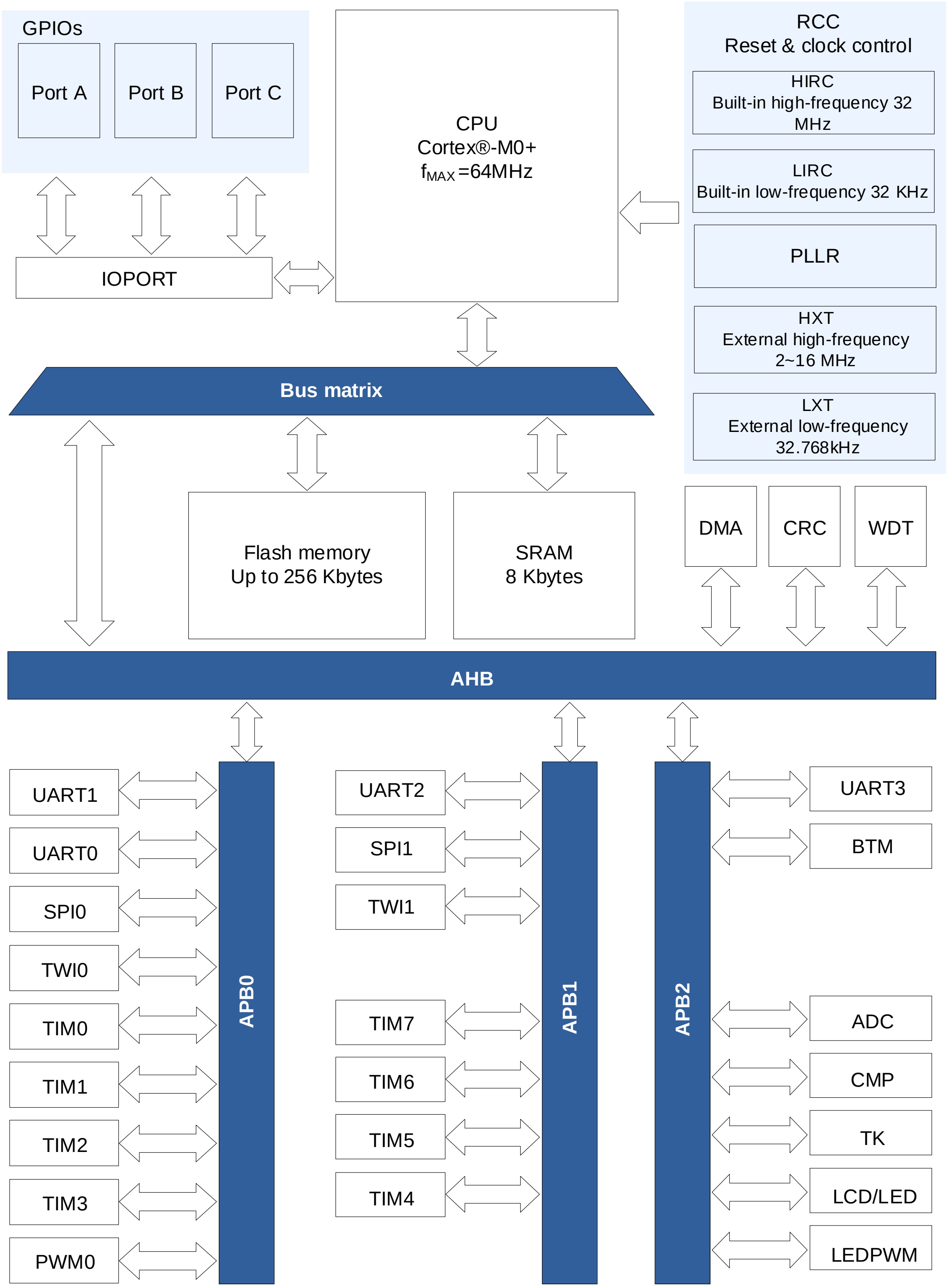
This section primarily explains the definitions of abbreviations and acronyms used in this document:

- Word: 32-bit data
- Half-word: 16-bit data
- Byte: 8-bit data
- Double word: 64-bit data
- IAP (In-Application Programming): IAP refers to the ability to reprogram the microcontroller's Flash during the execution of user programs.
- ICP (In-Circuit Programming): ICP refers to the ability to program the microcontroller's Flash when the device is installed on a user's circuit board, using the JTAG protocol, SWD protocol, or bootloader.
- ISP (In-System Programming): ISP refers to programming using a bootloader in conjunction with peripheral interfaces such as UART/SPI for programming
- JTAG protocol: JTAG protocol is an international standard testing protocol primarily used for internal chip testing.
- SWD protocol: SWD protocol, designed by ARM, represents Serial Wire Debug and is used for programming and debugging ARM microcontrollers.
- Option Byte: Configuration bits stored in Flash.
- AHB: Advanced High-Performance Bus
- APB: Advanced Peripheral Bus

## 2.2 Availability of peripherals

For information on the availability and quantity of peripherals for various product models, please refer to the latest data sheets in the product peripheral resource table section.

# 3 Resource Diagram

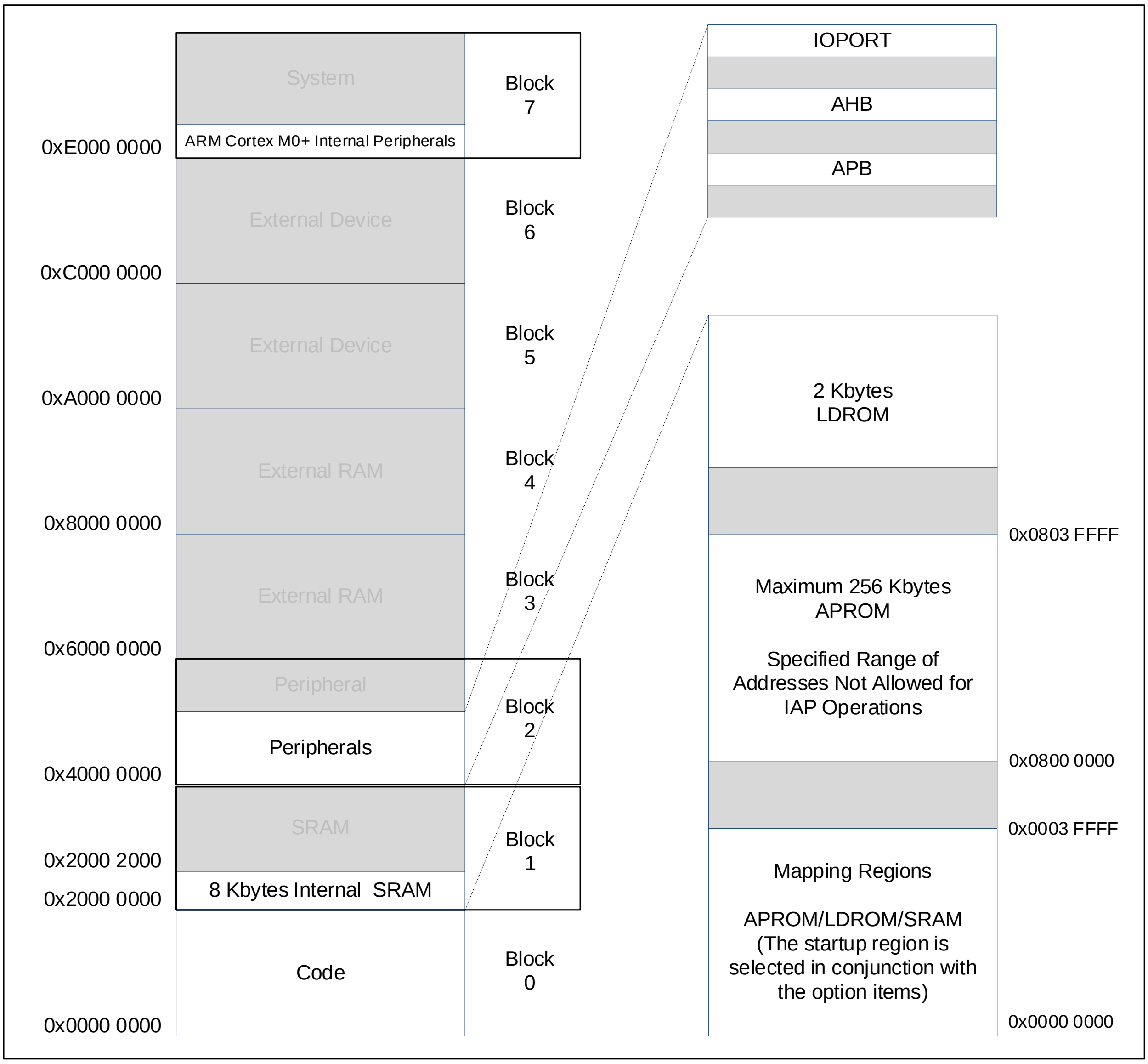


# 4 Flash

## 4.1 Description

The program memory, data memory, and registers are arranged in the same linear 4 GB address space. Each byte is encoded in little-endian format in memory. The byte with the lowest index within a word is considered the least significant byte, while the byte with the highest index is considered the most significant byte. The addressable memory space is divided into 8 main blocks, with each block being 512 MB.

## 4.2 Storage Block Diagram



SC32F10T/10G Series Memory Mapping Diagram

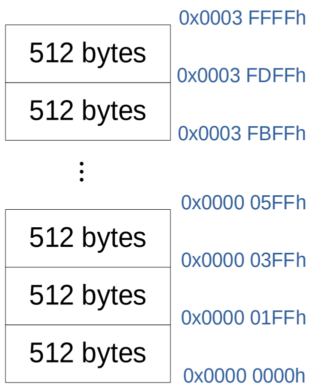
## 4.3 Feature

- 32-bit wide flash memory, and it can be rewritten up to 100,000 times
- Data retention time is over 100 years at room temperature
- The structure of the Flash includes:
  - Maximum 256 Kbytes APROM
  - 2 Kbytes LDROM
  - 8 Kbytes Internal SRAM
  - 96 bits Unique ID

## 4.4 APROM

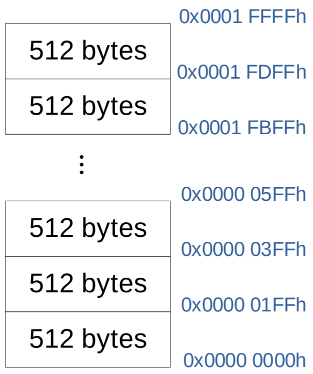
- APROM of SC32F10xx8 series has 256Kbytes
- APROM of SC32F10xx7 series has 128Kbytes
- Sector Size: 512 bytes
- Supports: Read/Write/Sector Erase/Chip Erase/Blank Check
- The CPU (Cortex®-M0+) accesses Flash through the AHB bus
- The program defaults to booting from APROM, and users can select programs to boot from other areas such as SRAM/LDROM using the customer option OP_BL[1:0].
- Read Protection: After enabling read protection, only a program that runs from APROM can read information from APROM. Other areas or third-party tools cannot access information from APROM.
- Write Protection: Provides two hardware read protection regions where IAP operations are prohibited. Users can set the range of the two read protection regions in units of sectors based on actual needs.

The 256 Kbytes of APROM is divided into 512 sectors, with each sector being 512 bytes. During programming, the sector to which the target address belongs will be forcibly erased by the programmer before writing data. During user write operations, the sector must be erased first before writing data.



SC32F10xx8 serise 256 Kbytes APROM Sector Partition Illustration

128 Kbytes of APROM is divided into 256 sectors, with each sector being 512 bytes. During programming, the sector corresponding to the target address is forcibly erased by the programmer before writing data. For user write operations, erasure must precede data writing.



SC32F10xx7 series 128 Kbytes APROM Sector Partition Illustration

## 4.5 LDROM

- 2 Kbytes of system storage area, factory-programmed with BootLoader program, Users cannot modify or access this area.
- Embedded Bootloader Program: The fixed ISP program is publicly available, allowing reprogramming of Flash via UART. The program waits for upgrade commands, and if no update command is received within 500 milliseconds, it jumps to APROM for execution (0X8000 0000).

### 4.5.1 BootLoader

Supports two Bootloader modes:

- Software Approach: Directly partition BootLoader and APP areas in software. Easy sharing interrupts of BootLoader and APP by modifying VTOR. Flexible adjustment of the size of each area；
- Hardware Approach: 2 Kbytes fixed "LDROM" as a dedicated BootLoader area that users cannot read or write
  - LDROM serves as a fixed BootLoader space with factory-programmed program, and users cannot read or write
  - Embedded Bootloader Program: The embedded bootloader program resides in LDROM and is programmed during the production stage. The fixed ISP program is publicly available, allowing reprogramming of Flash via UART.

## 4.6 SRAM

- Internal SRAM:8 Kbytes，address 0x2000 0000 ~ 0x2000 1FFF
- Users can choose to start the program from SRAM by configuring the customer option OP_BL[1:0].
- It supports byte, half-word (16-bit), or word (32-bit) access at the maximum system clock frequency, with no waiting states. Therefore, it can be accessed by both the CPU and DMA

## 4.7 Boot Area Selection

After a reset, users can independently configure the desired boot mode.

After exiting the standby mode, the startup mode configuration can be resampled. Once this startup delay has ended, the CPU will fetch the stack top value from address 0x00000000 and then begin executing code from the boot memory starting at 0x00000004.

There are three options for boot area selection: Main Flash Memory Area, System Flash Memory Area and SRAM, described in detail as follows:

### 4.7.1 Boot from APROM

APROM is aliased in the boot memory space (0x00000000) but can also be accessed from its original memory space (0x08000000). In other words, the program can start accessing from either address 0x00000000 or 0x08000000.

### 4.7.2 Boot from LDROM

- 2 Kbytes LDROM serves as a fixed BootLoader space with factory-programmed program, Users cannot modify or access this area.
- Embedded Bootloader Program: The embedded bootloader program resides in LDROM and is programmed during the production stage. The fixed ISP program is publicly available, allowing reprogramming of Flash via UART.

### 4.7.3 Boot from SRAM

SRAM has an alias in the boot memory space (0x0000 0000) but can also be accessed from its original memory space (0x2000 0000).

### 4.7.4 Boot mode config

The boot modes can be controlled by the register bits BTLD[1:0] in conjunction with the software reset (RST) control bit, both protected by the IAP_KEY::

① Set BTLD[1:0]=0x00: the chip boots from APROM after a software reset

② Set BTLD[1:0]=0x01: the chip boots from LDROM after a software reset

③ Set BTLD[1:0]=0x10: the chip boots from SRAM after a software reset

The initial boot region selection during power-up can be configured by customer option bits OP_BL[1:0]:

① Set OP_BL[1:0]=0x00: the chip boots from APROM after a software reset

② Set OP_BL[1:0]=0x01: the chip boots from LDROM after a software reset

③ Set OP_BL[1:0]=0x10: the chip boots from SRAM after a software reset

## 4.8 96 bits Unique ID

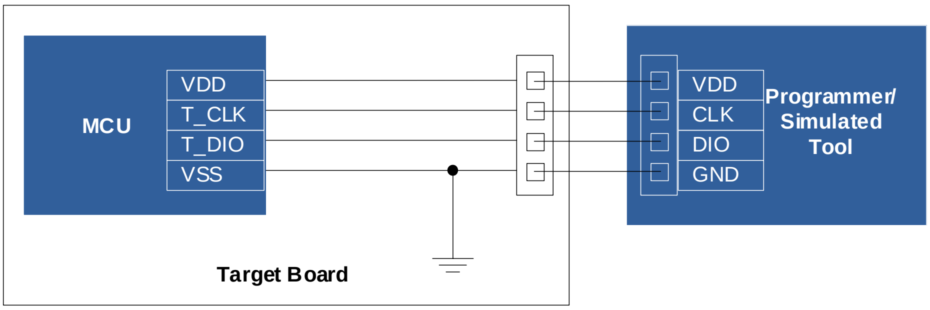
The SC32F10T/10G provides an independent Unique ID area. A 96-bit unique code can be pre-programmed before leaving the factory to ensure the uniqueness of the chip. The only way for the user to obtain the serial number is to read through the IAP instruction.

## 4.9 User ID Area

User ID area, where user-costomized ID is pre-programmed when leaving the factory. Users can read the User ID area, but cannot write the User ID area.

## 4.10 Programming

The SC32F10T/10G's Flash can be programmed through T_DIO, T_CLK, VDD, VSS, the specific connection relationship is as follows:



ICP mode Flash Writer programming connection diagram

T_DIO、T_CLK is a 2-wire JTAG programming and emulation signal line. Users can configure the mode of these two ports through the Customer Option when programming.

### 4.10.1 JTAG Specific Mode

T_DIO,T_CLK are specific port for programming and emulation, and other functions multiplexed with it are not available. This mode is generally used in the online debugging stage, which is convenient for users to simulate and debug. After the JTAG special mode takes effect, the chip can directly enter the programming or emulation mode without powering on and off again.

### 4.10.2 Normal Mode (JTAG specific port is invalid)

The JTAG function is not available, and other functions multiplexed with it can be used normally. This mode can prevent the programming port from occupying the MCU pins, which is convenient for users to maximize the use of MCU resources.

**Note: When the invalid configuration setting of the JTAG dedicated port is successful, the chip must be completely powered off and then on again to enter the programming or emulation mode, which will affect** the **programming and emulation in the live mode. SinOne recommends that users select the invalid configuration of the JTAG dedicated port during mass production and programming, and select the JTAG mode during the development and debugging phase.**

## 4.11 Security Encryption

The SC32F10T/10G series mainly involves encrypting the APROM for read protection. Users can configure the read protection encryption feature during programming through the customer option in the dedicated programming host; enable flash read protection can enter encryption mode:

- The chip defaults to a non-encrypted state while leaving the factory
- The read protection encryption feature has no mapped registers. Users can only modify it after config the customer option in the dedicated programming host and programming.
- Encryption Disabled: Operations such as reading, programming, and erasing can be performed on APROM. These operations can be also performed on Bytes and backup registers.
- Encryption Enabled:
  - Enable from APROM: Code executed in user mode (booting from user APROM) can perform all operations on APROM.
  - Debug, enable from SRAM and LDROM: In debug mode or when code is booted from SRAM or LDROM, APROM is completely inaccessible.
- Disabling encryption requires a full erase operation on APROM.

### 4.11.1 Security Encryption Access Rights

| Boot Area/Tools | Encryption Disabled Status | | | | | Read Protection Encryption Status | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Read | Write | Block Earse | Full Earse | Operate Write-Protection Region | Read | Write | Block Earse | Full Earse | Operate Write-Protection Region |
| Boot from APROM | √ | √ | √ | \ | Forbid | √ | √ | √ | \ | Forbid |
| Debug/Boot from SRAM | √ | √ | √ | √ | Forbid | Forbid | Forbid | Forbid | Forbid | Forbid |
| Boot from LDROM | √ | √ | √ | √ | √ | Forbid | Forbid | Forbid | √ | Forbid |

## 4.12 In Application Programming (IAP)

The IAP (In Application Programming) area in the APROM of SC32F10T/10G allows users to perform remote program updates through IAP operations. Users can also retrieve information from the Unique ID or User ID areas by IAP read operations. Before performing IAP write operations, users must carry out sector erasure for the target address sector.

The chip allows global IAP operations in the APROM by default while leaving the factory. Internally, the chip provides two sets of flash write protection regions. These regions are set based on sector units, and the protected areas are restricted from IAP operations. The rules for setting these regions are as follows:

| IAPPORx Register Value(x=A or B) | IAPPOR Protection Area |
|---|---|
| IAPPORx_ST = IAPPORx_ED | Sector IAPPORx |
| IAPPORx_ST > IAPPORx_ED | No protection |
| IAPPORx_ST < IAPPORx_ED | Sectors from IAPPORx_ST to IAPPORx_ED |

User can config these APROM’s write protection area through ”Customer Option” while programming.

### 4.12.1 IAP Control Register

To perform IAP operations on APROM outside the write protection regions, it can be achieved using the following registers:

#### 4.12.1.1 Data Protect Register (IAP_KEY)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| IAP_KEY | R/W | Data Protect Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | IAPKEY[31:0] | Data Protection Key<br>To prevent accidental operations on Flash due to electrical interference, IAP_CON Register requires unlocking through IAPKEY before performing a write operation. The unlocking sequence is as follows:<br>1. Write KEY1 = 0x1234_5678<br>2. Write KEY2 = 0xA05F_05FA<br>The IAP_CON Register will be locked until the next system reset if the sequence of operations is incorrect. |

#### 4.12.1.2 IAP Sector Number Setting Register (IAP_SNB)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| IAP_SNB | R/W | IAP Sector Number Setting Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|

| | | |
|---|---|---|
| 31~24 | IAPADE[7:0] | IAP Operation Area Extended Address<br>By writing different values to IAPADE, the IAP operations can be directed to different operation areas:<br>0x00: Invalid<br>0x4C: APROM<br>Others: Reserved |
| 8~0 | IAPSNB[8:0] | IAP Operation Sector Number Setting for Sector/Page Erase:<br>The actual starting address of the operated sector = Flash Base Address + [ IAPSNB[8:0] * 0x200 ] |
| 23~9 | - | Reserved |

### 4.12.1.3 IAP Control Register (IAP_CON)(Write Protection)

*This register is write-protected and can only be modified by manipulating the data protection register IAP_KEY**.**

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| IAP_CON | R/W | IAP Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31 | LOCK | The IAP_CON will be locked after setting this bit to 1.<br>When the unlock sequence is detected, this bit will be cleared by hardware, if the unlock operation fails, this bit will remain 1 until the next system reset. |
| 10~9 | BTLD[1:0] | Boot Area Selections Bit After Software Reset:<br>00: Boot from APROM after software reset<br>01: Boot from LDROM after software reset<br>10: Boot from embedded SRAM after software reset<br>11: Reserved |
| 8 | RST | Software Reset Control Bit:<br>0: Program running normally<br>1: System will reset immediately after setting this bit to 1 |
| 7 | ERASE | All Erase Control Bit<br>0:No earse operation |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 1:Setting 1 to this bit and configure CMD[1:0]=10 will initiate a full erase operation on APROM. |
| 5 | SERASE | Sector Erase Control Bit:0:No earse operation<br>1: Setting 1 to this bit and configure CMD[1:0]=10 will initiate a sector erase operation on APROM, and the selected sector will be earsed. |
| 4 | PRG | Program Control Bit:<br>0:Disable Flash Programming<br>1:Enable Flash Programming |
| 1~0 | CMD[1:0] | IAP Command Enable Control Bit:<br>10: Execute the erase operation command<br>Others: Reserved<br>**Note:**<br>**1. The corresponding operation will execute only when CMD[1:0] set to 10 after setting any earse control bit to 1.**<br>**2. Only one IAP operation can be executed at a time, so the ERASE/SERASE bit can only be set to 1 at a time** |
| 30~11<br>6<br>3~2 | - | Reserved |

### 4.12.1.4 IAP Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| IAP Base Address:0x4000_03C0 | | | | |
| IAP_KEY | 0x00 | R/W | Data protect Register | 0x0000_0000 |
| IAP_SNB | 0x04 | R/W | IAP Sector Number Setting Register | 0x0000_0000 |
| IAP_CON | 0x0C | R/W | IAP Control Register | 0x0000_0000 |

## 4.13 Customer Option

SC32F10T/10G has a separate Flash area dedicated to storing customer-defined power-up default settings, this area is called Customer Option area. Users can configure Customer Option through host, and the configured values are written into the Customer Option area during the programming process. IC will use the Customer Option data as the initial settings during the reset initialization phase.

It is also possible to temporarily modify Customer Option by operating mapping registers. However, it's important to note that modifying the mapping registers only achieves temporary adjustments and does not affect the settings in the Customer Option area. The initialization will still be based on the Customer Option parameters during programming after reset.

The operation method of Customer Option related mapping Register is as follows:

The Customer Option related SFR R/W operations are controlled by OPINX and OPREG registers, the specific location of each Customer Option SFR is determined by OPINX, as shown in the following table:

| Register | Address | Description | Reset Value |
|---|---|---|---|
| OPINX | 0x4000_03F8 | Customer Option Pointer | 0x0000_0000 |
| OPREG | 0x4000_03FC | Customer Option Register | 0x0000_0000 |
| OPT_CON0 | 0XC1 @ OPINX | Customer OptionMapping Register0 | 0x0000_0000 |
| OPT_CON1 | 0XC2 @ OPINX | Customer OptionMapping Register1 | 0x0000_0000 |

## 4.13.1 Customer Option Mapping Register

Before rewrite IFB mapping register by OPINX and OPREG,it is necessary to enable AHB_CFG.IFBEN, the clock enable switch of Customer Option Register.

### 4.13.1.1 AHB Bus Peripheral Clock Enable Register (AHB_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| AHB_CFG | R/W | AHB Bus Peripheral Clock Enable Register | 0x0020_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | IFBEN | Customer Option Mapping Register Clock Enable Bit<br>Before rewrite IFB mapping register by OPINX and OPREG,it is necessary to enable IFBEN.<br>0: Disable<br>1: Enable |
| 31~23<br>19~3 | - | Reserved |

### 4.13.1.2 Customer Option Mapping Register0 (OPT_CON0)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| OPT_CON0 | R/W | Customer Option Mapping Register0 | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | DISLVR | LVRS [1:0] | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | DISLVR | LVR Switch<br>0:LVR Enable<br>1:LVR Disable |
| 1~0 | LVRS [1:0] | LVR Voltage Select Control<br>11:4.3V Reset<br>10:3.7V Reset<br>01:2.9V Reset<br>00:1.9V Reset |
| 7~3 | - | Reserved |

### 4.13.1.3 Customer Option Mapping Register1 (OPT_CON1)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| OPT_CON1 | R/W | Customer Option Mapping Register1 | 0x0000_0000 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ENWDT | DISJTG | DISRST | - | - | - | OP_BL[1:0] | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | ENWDT | WDT Switch<br>0: WDT disable<br>1: WDT enable |
| 6 | DISJTG | JTAG Switch Control Bit<br>0: JTAG Mode Enable,corresponding pin can only work as T_CLK/T_DIO<br>1: Normal Mode Enable,JTAG function disable |
| 5 | DISRST | Reset Pin Switch Control Bit<br>**This bit is read only.Read Only**<br>0: RST corresponding pin is used as reset pin<br>1: RST corresponding pin is used as normal GPIO pin |
| 1~0 | OP_BL[1:0] | Boot area selection after reset<br>**This bit is read onlyRead Only**<br>00: Boot from APROM after reset<br>01: Boot from LDROM after reset<br>10: Boot from embedded SRAM after reset<br>11: Reserved |
| 4~3 | - | Reserved |

# 5 Power,Reset And System Clock(RCC)

## 5.1 Power-on Reset

After the SC32F10T/10G power-on, the processes carried out before execution of client software are as follows:

- Reset stage
- Loading information stage
- Normal operation stage

### 5.1.1 Reset Stage

The SC32F10T/10G will always be reset until the voltage supplied to SC32F10T/10G is higher than a certain voltage, and the internal Clock starts to be effective. The duration of reset stage is related to rising speed of external power. Once the external supply voltage is up to built-in POR voltage, the reset stage would be completed.

### 5.1.2 Loading Information Stage

There is a warm-up counter inside The SC32F10T/10G. During the reset stage, the warm-up counter is cleared to 0 until the voltage exceeds the POR voltage, the built-in HIRC oscillator starts to oscillate, and the warm-up counter starts counting. When the internal warm-up counter counts to a certain number, every certain number of HIRC clocks will read a byte of data from the IFB (including Customer Option) in the Flash ROM and store it in the internal system register. This reset signal will not end until the warm-up is completed.

### 5.1.3 Normal Operation Stage

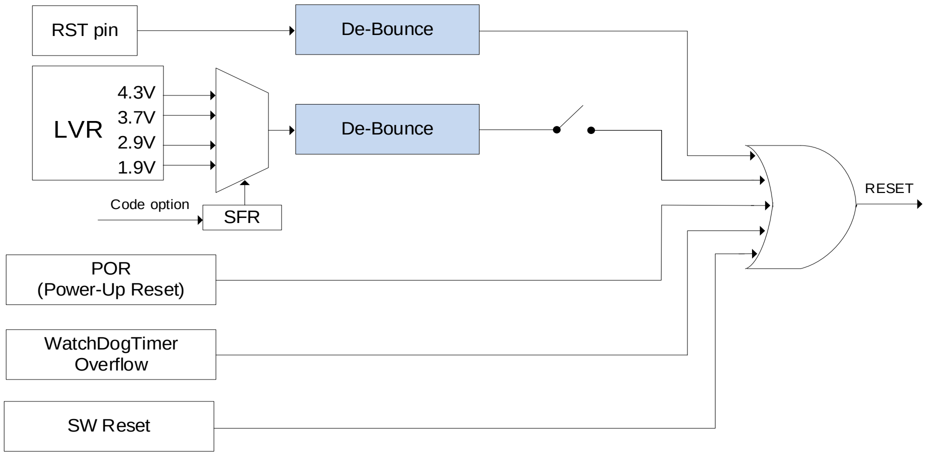
After finishing the Loading Information stage, The SC32F10T/10G starts to read the instruction code from Flash and enters the normal operation stage. The LVR voltage is the set value of Customer Option written by the user.

## 5.2 Reset Modes

The SC32F10T/10G has 5 reset methods, the first four are hardware reset:

1. External reset
2. Low-voltage reset LVR
3. Power-on reset POR
4. Watchdog WDT reset
5. Software reset.

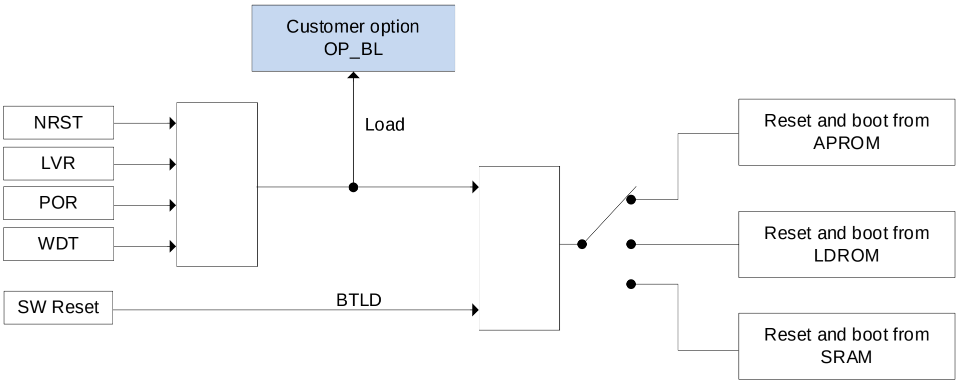
The circuit diagram of the reset part of the SC32F10T/10G is as follows:



SC32F10T/10G Reset Circuit Diagram

### 5.2.1 Boot area after the reset

After hardware reset through external RST, low voltage reset (LVR), power-on reset (POR), or watchdog reset (WDT), the chip boots from the startup area (APROM / LDROM / SRAM) set by the user in OP_BL. After the software reset, the chip boots from the startup area (APROM / LDROM / SRAM) set by BTLD[1:0].



SC32F10T/10G Boot Area Switching diagram after reset

### 5.2.2 External RST

External reset is a low-level reset pulse signal of a certain width given to SC32F10T/10G from external RST pin to realize the reset of SC32F10T/10G. User can configure the PC1/NRST pin as RST (reset pin) using the programming host software by Customer Option before programming.

### 5.2.3 Low-voltage Reset LVR

The SC32F10T/10G provides a low-voltage reset circuit. There are 4-level LVR voltage options: 4.3V, 3.7V, 2.9V, 1.9V. The default value is the Customer Option value written by the user. A reset occurs when the VDD voltage is less than the threshold voltage for low-voltage reset and the duration is greater than $T_{LVR}$. Among them, $T_{LVR}$ is the buffeting time of LVR, about 30μs.

### 5.2.4 Power-on Reset(POR)

The SC32F10T/10G has a power-on reset circuit inside. When the power supply voltage $V_{DD}$ reaches the POR reset voltage, the system automatically resets.

### 5.2.5 Watchdog Reset(WDT)

The SC32F10T/10G has a WDT, the clock source of which is the built-in 32 kHz oscillator. The user can choose whether to enable the watchdog reset function by Customer Option.

### 5.2.6 Software Reset

Enable RST(IAP_CON.8) will immediately reset the system.

### 5.2.7 Initial Reset State

When SC32F10T/10G is in the reset state, most registers return to their initial state. The watchdog (WDT) is in the disabled state. 'Hot-start' resets (such as WDT, LVR, software reset, etc.) do not affect SRAM, and SRAM values remain the same as before the reset. Loss of SRAM content occurs when the power supply voltage drops to a level where RAM cannot retain data.

## 5.3 Clock

### 5.3.1 System Clock Source

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

- Built-in high-frequency 32MHz oscillator (HIRC), default clock at power-up
- External high-frequency crystal oscillator (HXT)
- Built-in low-frequency 32KHz oscillator (LIRC)
- External low-frequency crystal oscillator (LXT)
- PLL, with a maximum frequency of 64MHz

**Note:**

1. **The default system clock source at power-up is HIRC. Users can switch the clock source through software during normal operation after power-up. Before switching, ensure that the selected clock source is in a stable operating state.**
2. **Regardless of the chosen clock source to switch to, the system clock source must first be switched to HIRC before transitioning to the target clock source.**

### 5.3.2 Bus

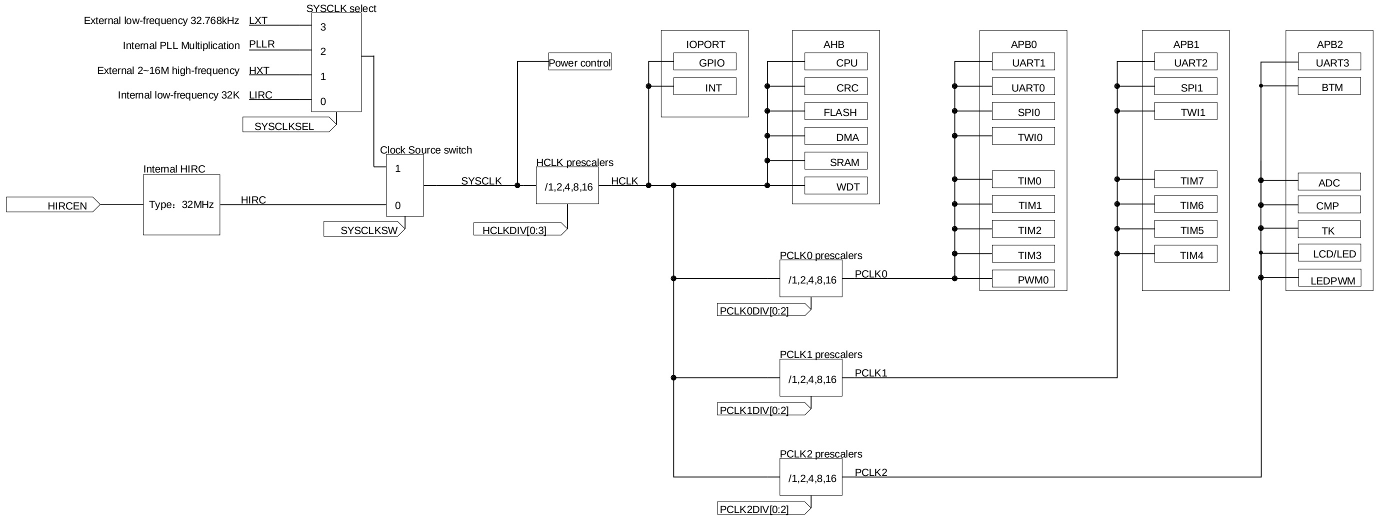
Users can configure the frequencies of the AHB, APB0, APB1, and APB2 domains through multiple prescalers.

- HCLK: The main clock of the AHB domain, with a maximum frequency of 64MHz. It drives components such as the Cortex®-M0+ core, memory, and DMA.
- PCLK0: The main clock of the APB0 domain, with a maximum frequency equal to the HCLK frequency. Peripheral devices on the APB0 bus are driven by PCLK0.

- PCLK1: The main clock of the APB1 domain, with a maximum frequency equal to the HCLK frequency. Peripheral devices on the APB1 bus are driven by PCLK1.
- PCLK2: The main clock of the APB2 domain, with a maximum frequency equal to the HCLK frequency. Peripheral devices on the APB2 bus are driven by PCLK2.

The RCC divides the AHB clock (HCLK) by 8 to serve as the external clock for SysTick. By setting the control and status registers of SysTick, you can choose either the above-mentioned clock or the core clock as the SysTick clock source.

### 5.3.3 Clock and Bus Allocation Block Diagram



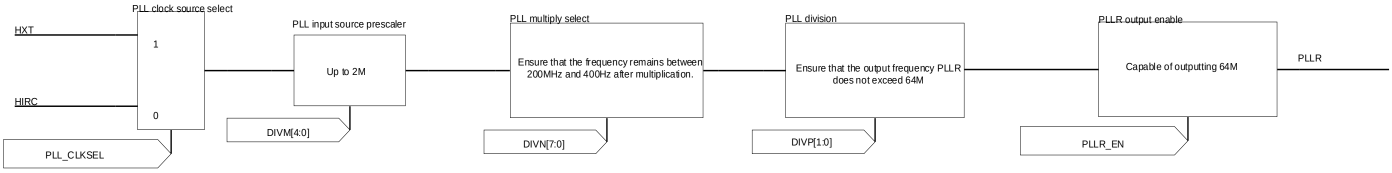
Clock and Bus Allocation Block Diagram

## 5.4 RCC Interrupt

In coordination with the stop oscillation detection mechanism, SC32F10T/10G's clock source provides a user-configurable RCC interrupt: when the system clock source is LXT/HXT/PLL, if an abnormality is detected in the clock source, the stop oscillation detection interrupt flag will be set. If the corresponding interrupt is enabled at this point, a stop oscillation detection interrupt will be generated.

## 5.5 PLL

- The system operating clock can be multiplied to 64MHz through PLL.
- Users can configure the desired frequency using the formula:
  - PLL input clock frequency fPLL_IN
  - PLL output frequency $f_{PLLR}$ = [ ( $f_{PLL\_IN}$ / DIVM ) * DIVN ] /  2 ^ ( DIVP +1 )
- The PLL circuit diagram is as follows:


PLL clock source select
HXT
HIRC
1
0
PLL_CLKSEL
PLL input source prescaler
Up to 2M
DIVM[4:0]
PLL multiply select
Ensure that the frequency remains between 200MHz and 400Hz after multiplication.
DIVN[7:0]
PLL division
Ensure that the output frequency PLLR does not exceed 64M
DIVP[1:0]
PLLR output enable
Capable of outputting 64M
PLLR_EN
PLLR

## 5.6 Built-in high-frequency 32MHz oscillator (HIRC)

- Can be selected as the system operating clock
- Can be selected as the PLL clock source
- Frequency error: Within ±1% @ -40 ~ 105℃ @ 2.0V~ 5.5V
- The system clock can be automatically calibrated by 32.768 kHz external crystal oscillator, after calibration HIRC accuracy can be infinitely close to the accuracy of external 32.768 kHz crystal oscillator

## 5.7 External High-Frequency Crystal Oscillator Circuit (HXT)

- Can be selected as the system operating clock
- Can be selected as the PLL clock source
- Can be externally connected to a 2~16MHz high-frequency oscillator

## 5.8 Built-in Low-Frequency 32kHz Oscillator

- Can be selected as the system operating clock
- Can be selected as the LCD/LED clock source
- Can be selected as the Base Timer and WDT clock source
- Frequency error: Within ±4% @ -20 ~ 85℃ @ 4.0V~ 5.5V, after register correction

## 5.9 External Low-Frequency Oscillator Circuit (LXT)

- Can be selected as the system operating clock
- Can be selected as the Base Timer clock source
- Can be selected as the LCD/LED clock source
- Allows for an external 32.768kHz low-frequency oscillator
- Automatic calibration of HIRC can be performed using LXT

## 5.10 RCC Register

### 5.10.1 RCC Protect Register (RCC_KEY)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| RCC_KEY | R/W | RCC Protect Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | RCCKEY[7:0] | The operation enable switch and timing limit settings for RCC_CFG0,RCC_CFG1,PLL_CFG registers.<br>Write a value “n” greater than or equal to 0x40 means:<br>1. Enable the write operation function for RCC_CFG0,RCC_CFG1,PLL_CFG registers.<br>2. If no register write command is received after “n” system clock,the RCC rewrite function will be disabled again. |
| 31~8 | - | Reserved |

### 5.10.2 System Clock Source Selection Register (RCC_CFG0) (Write Protection)

***This register is write-protected and can only be modified by manipulating the RCCprotection register RCC_KEY.**

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| RCC_CFG0 | R/W | System Clock Source Selection Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INTEN | - | - | - | HPLDO_DP | - | SYSCLKSEL[1:0] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYSCLKSW | HIRCEN | HXTEN | CRY_HF | - | - | LIRCEN | LXTEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |

| | | |
|---|---|---|
| 11 | HPLDO_DP | Low Frequency System Clock Power Consumption Adjust Bit<br>0: The recommended setting for the system clock source when not using LIRC<br>1: The recommended setting for the system clock source when using LIRC, when the system clock is set to LIRC, writing this bit to 1 can reduce power consumption |
| 9~8 | SYSCLKSEL[1:0] | System Clock Source Selection Bit<br>00: System clock source is from LIRC<br>01: System clock source is from HXT<br>10: System clock source is from PLLR<br>11: System clock source is from LXT<br>**Note:**<br>**1. The default system clock source after power-up is HIRC. Users can switch the clock source through software during normal operation after power-up. Before switching, ensure that the selected clock source is in a stable operating state.**<br>**2. Regardless of the chosen clock source for switching, the system clock source must first be switched to HIRC before switching to the target clock source.** |
| 7 | SYSCLKSW | The system clock source switching bit, when enabled, allows the system clock source to switch from HIRC to the clock selected by SYSCLKSEL:<br>0: System clock source is HIRC<br>1: System clock source is the option set by SYSCLKSEL<br>After rewriting this bit, the internal circuit must successfully switch for the updated value to take effect; otherwise, the read value will remain the status before rewriting. Users could determine whether the clock source has successfully switched by reading this bit.<br>This bit will be automatically cleared after a reset/wake-up, meaning that HIRC provides the system clock after a reset/wake-up.<br>**Note:**<br>**1. The default system clock source after power-up is HIRC. Users can switch the clock source through software during normal operation after power-up. Before switching, ensure that the selected clock source is in a stable operating state.**<br>**2. Regardless of the chosen clock source for switching, the system clock source must first be switched to HIRC before switching to the target clock source.** |

| | | |
|---|---|---|
| 6 | HIRCEN | Built-In High-Frequency 32MHz Oscillator Enable Bit<br>0: Disable<br>1: Enable<br>When SYSCLKSW = 0, and HIRC is selected as the system clock, this bit cannot be written.<br>This bit will be set to 1 by hardware after a reset/wake-up, meaning that HIRC provides the system clock after a reset/wake-up. |
| 5 | HXTEN | External High-Frequency HXT Crystal Oscillator Enable Bit<br>0: Disable<br>1: Enable<br>This bit will be automatically cleared after a reset/wake-up. |
| 4 | CRY_HF | External High-Frequency HXT Crystal Oscillator Frequency Rage Selection Bit<br>0: External crystal oscillator frequency<12M<br>1: External crystal oscillator frequency≥12M |
| 1 | LIRCEN | Built-In Low-Frequency LIRC Oscillator Enable Bit<br>0: Disable<br>1: Enable |
| 0 | LXTEN | External Low-Frequency LXT Crystal Oscillator Enable Bit<br>0: Disable<br>1: Enable |
| 31~16<br>14~12<br>10<br>3~2 | - | Reserved |

### 5.10.3 Peripheral Clock Source Selection Register (RCC_CFG1)(Write Protection)

***This register is write-protected and can only be modified by manipulating the RCCprotection register RCC_KEY.**

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| RCC_CFG1 | R/W | Peripheral Clock Source Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~5 | STCLKSEL[2:0] | SysTick Clock Source Selection Bit<br>000: Clock source is from HCLK/8<br>001: Clock source is from HIRC/2<br>010: Clock source is from HXT/2<br>011: Clock source is from LIRC<br>100: Clock source is from LXT<br>**Note:When cofiguring clock source, users should note that if SysTick source is not from HCLK, the clock source frequency of SysTick must equal to or fewer than $f_{HCLK}$/2.** |
| 1 | LCDCLKSEL | LCD Clock Source Selection Bit<br>0: Clock source is from LIRC<br>1: Clock source is from LXT<br>After rewriting this bit, the internal circuit must successfully switch for the updated value to take effect; otherwise, the read value will remain the status before rewriting. Users could determine whether the clock source has successfully switched by reading this bit. |
| 0 | BTMCLKSEL | BTM Clock Source Selection Bit<br>0: Clock source is from LIRC<br>1: Clock source is from LXT<br>After rewriting this bit, the internal circuit must successfully switch for the updated value to take effect; otherwise, the read value will remain the status before rewriting. Users could determine whether the clock source has successfully switched by reading this bit. |
| 31~8<br>4~2 | - | Reserved |

### 5.10.4 PLL Configuration Register (PLL_CFG)(Write Protection)

*****This register is write-protected and can only be modified by manipulating the RCCprotection register RCC_KEY**.**

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PLL_CFG | R/W | PLL Configuration Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 23 | PLLCLKSEL | PLL Clock Source Selection Bit<br>0: The clock source of PLL is from built-in high-frequency oscillator HIRC, the input clock frequency of PLL $f_{PLL\_IN}=f_{HIRC}$.<br>1: The clock source of PLL is from external high-frequency crystal oscillator HXT, the input clock frequency of PLL $f_{PLL\_IN}=f_{HXT}$.<br>**Note: Users could only write this bit when PLL is disable.** |
| 20~16 | DIVM[4:0] | PLL Clock Source Prescaler<br>Valid value:0x01~0x1F<br>**Note: Users need to ensure that the PLL input frequency is divided to 2MHz.** |
| 15~8 | DIVN[7:0] | PLL Multiplication Factor<br>The value of DIVN should be set between 2 and 192(include 2 and 192);other values are invalid.Formula:<br>$f_{PLLR} = [ ( f_{PLL\_IN} / DIVM ) * DIVN ] / 2 \wedge ( DIVP +1 )$<br>**Note:**<br>**Users need to ensure that the frequency after multiplication stays between 200MHz and 400MHz. Afterward, users need to configure DIVP division to ensure that the output frequency PLLR does not exceed 64MHz.**<br>**PLLR can serve as the system clock source.** |
| 7 | PLLON | PLL Enable Bit<br>0: PLL disable<br>1: PLL enable<br>This bit will be cleared by hardware when entering stop,standby,or shutdown mode.<br>**Note:This bit could not be 0 when PLL clock is served as the system clock.** |
| 6 | PLLREN | PLLRCLK Clock Output Enable Bit<br>0: Disable<br>1: Enable<br>This bit cannot be written when PLLRCLK is served as the system clock. |
| 1~0 | DIVP[1:0] | PLLRCLK Clock Output Division Factor<br>00: Division by 2<br>01: Division by 4<br>10: Division by 8<br>11: Division by 16<br>These bits can only be written when PLL is disable.<br>PLLR can serve as the system clock source. |
| 31~24<br>22~21<br>5~2 | - | Reserved |

### 5.10.5 Clock Status Register (RCC_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| RCC_STS | R/W | Clock Status Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | PLLRDY | PLL Clock Ready Flag<br>This bit is read-only and is set by hardware to indicate that PLL is lockedRead Only<br>0: PLL is not locked<br>1: PLL is locked |
| 1 | LOCKERR | PLL Loss of Lock Record Bit<br>This bit is read-only and is set by hardware to indicate that whether PLL has experienced a loss of lockRead Only<br>0: PLL has not experienced a loss of lock<br>1: PLL has experienced a loss of lock |
| 0 | CLKFIF | Clock Source Exception Flag<br>For the case when the system clock source is external crystal or PLL.<br>0: No exception in the current clock source<br>1: Exception in the current clock source, and the system clock source has automatically switched to HIRC. If RCC interrupt is enable (RCC_CFG0.INTEN=1), an interrupt will be generated.<br>The CLKFIF flag can be cleared after reset. |
| 31~3 | - | Reserved |

### 5.10.6 SysTick Calibration Parameter Register (SYST_CALIB)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SYST_CALIB | R/W | SysTick Calibration Parameter Register | 0x0000_2327 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | CALIB[15:0] | Calibration Register Default Value:<br>If the default clock after power-up is $f_{HCLK}$/n (MHz), (n is the default division factor after power-up, and HIRC is the default clock source after power-up).<br>Then the SysTick calibration initial value is set to 1000*($f_{HCLK}$/n), this ensures that a default 1ms time base can be generated. |
| 31~16 | - | Reserved |

### 5.10.7 AHB Bus Peripheral Clock Enable Register (AHB_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| AHB_CFG | R/W | AHB Bus Peripheral Clock Enable Register | 0x0020_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 22~20 | CLKDIV[2:0] | AHB Clock Division Configure Bit<br>The division factor of $f_{SYSCLK}$ to generate $f_{HCLK:}$<br>000: $f_{HCLK}$= $f_{SYSCLK}$<br>001: $f_{HCLK}$= $f_{SYSCLK}$ / 2<br>010: $f_{HCLK}$= $f_{SYSCLK}$ / 4 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 011: $f_{HCLK}= f_{SYSCLK}$ / 8<br>100: $f_{HCLK}= f_{SYSCLK}$ / 16<br>Others: Reserved |
| 2 | IFBEN | Customer Option Mapping Register Clock Enable Bit<br>Before rewrite IFB mapping register by OPINX and OPREG,it is necessary to enable IFBEN.<br>0: Disable<br>1: Enable |
| 1 | CRCEN | CRC Module Clock Enable Bit<br>0: Disable<br>1: Enable |
| 0 | DMAEN | DMAClock Enable Bit<br>0: Disable<br>1: Enable |
| 31~23<br>19~3 | - | Reserved |

### 5.10.8 APB0 Bus Peripheral Clock Enable Register (APB0_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB0_CFG | R/W | APB0 Bus Peripheral Clock Enable Register | 0x0000_0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ENAPB | CLKDIV[2:0] | | | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | PWM0EN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART1EN | UART0EN | SPI0EN | TWI0EN | TIM3EN | TIM2EN | TIM1EN | TIM0EN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 23 | ENAPB | APB0 Bus Clock Control Bit<br>0: Disable<br>1: Enable |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 22~20 | CLKDIV[2:0] | APB0 Clock Division Configure Bit<br>The division factor of $f_{HCLK}$ to generate $f_{PCLK0}$:<br>000: $f_{PCLK0} = f_{HCLK}$<br>001: $f_{PCLK0} = f_{HCLK} / 2$<br>010: $f_{PCLK0} = f_{HCLK} / 4$<br>011: $f_{PCLK0} = f_{HCLK} / 8$<br>100: $f_{PCLK0} = f_{HCLK} / 16$<br>101: $f_{PCLK0} = f_{HCLK} / 32$<br>110: $f_{PCLK0} = f_{HCLK} / 64$<br>111: $f_{PCLK0} = f_{HCLK} / 128$ |
| 8 | PWM0EN | PWM0 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 7 | UART1EN | UART1 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 6 | UART0EN | UART0 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 5 | SPI0EN | SPI0 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 4 | TWI0EN | TWI0 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 3 | TIM3EN | Timer3 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 2 | TIM2EN | Timer2 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 1 | TIM1EN | Timer1 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 0 | TIM0EN | Timer0 Clock Enable Bit<br>0: Disable<br>1: Enable |
| 31~24<br>19~9 | - | Reserved |

### 5.10.9 APB1 Bus Peripheral Clock Enable Register (APB1_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB1_CFG | R/W | APB1 Bus Peripheral Clock Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ENAPB | CLKDIV[2:0] | | | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART2EN | - | - | TWI1EN | TIM7EN | TIM6EN | TIM5EN | TIM4EN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 23 | ENAPB | APB1 Bus Clock Control Bit<br>0: Disable<br>1: Enable |
| 22~20 | CLKDIV[2:0] | APB1 Clock Division Configure Bit<br>The division factor of $f_{HCLK}$ to generate $f_{PCLK1}$:<br>000: $f_{PCLK1}= f_{HCLK}$<br>001: $f_{PCLK1}= f_{HCLK} / 2$<br>010: $f_{PCLK1}= f_{HCLK} / 4$<br>011: $f_{PCLK1}= f_{HCLK} / 8$<br>100: $f_{PCLK1}= f_{HCLK} / 16$<br>101: $f_{PCLK1}= f_{HCLK} / 32$<br>110: $f_{PCLK1}= f_{HCLK} / 64$<br>111: $f_{PCLK1}= f_{HCLK} / 128$ |
| 7 | UART2EN | UART2Clock Enable Bit<br>0: Disable<br>1: Enable |
| 4 | TWI1EN | TWI1Clock Enable Bit<br>0: Disable<br>1: Enable |
| 3 | TIM7EN | Timer7Clock Enable Bit<br>0: Disable<br>1: Enable |
| 2 | TIM6EN | Timer6Clock Enable Bit<br>0: Disable<br>1: Enable |
| 1 | TIM5EN | Timer5Clock Enable Bit<br>0: Disable<br>1: Enable |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | TIM4EN | Timer4Clock Enable Bit<br>0: Disable<br>1: Enable |
| 31~24<br>19~8<br>6~5 | - | Reserved |

### 5.10.10 APB2 Bus Peripheral Clock Enable Register (APB2_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB2_CFG | R/W | APB2 Bus Peripheral Clock Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ENAPB | CLKDIV[2:0] | | | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | UART3EN | LCDEN | LEDPWMEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 23 | ENAPB | APB2 Bus Clock Control Bit<br>0: Disable<br>1: Enable |
| 22~20 | CLKDIV[2:0] | APB2 Clock Division Configure Bit<br>The division factor of $f_{HCLK}$ to generate $f_{PCLK2}$:<br>000: $f_{PCLK2}$= $f_{HCLK}$<br>001: $f_{PCLK2}$= $f_{HCLK}$ / 2<br>010: $f_{PCLK2}$= $f_{HCLK}$ / 4<br>011: $f_{PCLK2}$= $f_{HCLK}$ / 8<br>100: $f_{PCLK2}$= $f_{HCLK}$ / 16<br>101: $f_{PCLK2}$= $f_{HCLK}$ / 32<br>110: $f_{PCLK2}$= $f_{HCLK}$ / 64<br>111: $f_{PCLK2}$= $f_{HCLK}$ / 128 |
| 2 | UART3EN | UART3Clock Enable Bit<br>0: Disable<br>1: Enable |
| 1 | LCDEN | LCD/LED Module Clock Enable Bit<br>0: LCD/LED Clock Disable |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 1: LCD/LED Clock Enable, it is recommanded to enable LEDPWMEN together; otherwise, operating SEGn to display RAM may not be possible. |
| 0 | LEDPWMEN | LEDPWM Clock Enable Bit And LCD/LED RAM Switch<br>0: LEDPWM clock disable, SEGn display RAM disable<br>1: LEDPWM clock enable, SEGn display RAM enable |
| 31~24<br>19~3 | - | Reserved |

### 5.10.11 AHB Bus Peripheral Reset Control Register (AHB_RST)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| AHB_RST | R/W | AHB Bus Peripheral Reset Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 1 | CRCRST | CRC Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset RCC |
| 0 | DMARST | DMA Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset DMA |
| 31~2 | - | Reserved |

### 5.10.12 APB0 Bus Peripheral Reset Control Register (APB0_RST)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB0_RST | R/W | APB0 Bus Peripheral Reset Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | PWM0RST |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART1RST | UART0RST | SPI0RST | TWI0RST | TIM3RST | TIM2RST | TIM1RST | TIM0RST |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 8 | PWM0RST | PWM0 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset PWM0 |
| 7 | UART1RST | UART1 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset UART1 |
| 6 | UART0RST | UART0 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset UART0 |
| 5 | SPI0RST | SPI0 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset SPI0 |
| 4 | TWI0RST | TWI0 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset TWI0 |
| 3 | TIM3RST | Timer3 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer3 |
| 2 | TIM2RST | Timer2 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer2 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 1 | TIM1RST | Timer1 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer1 |
| 0 | TIM0RST | Timer0 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer0 |
| 31~9 | - | Reserved |

### 5.10.13 APB1 Bus Peripheral Reset Control Register (APB1_RST)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB1_RST | R/W | APB1 Bus Peripheral Reset Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART2RST | - | - | TWI1RST | TIM7RST | TIM6RST | TIM5RST | TIM4RST |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | UART2RST | UART2 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset UART2 |
| 4 | TWI1RST | TWI1 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset TWI1 |
| 3 | TIM7RST | Timer7 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer7 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | TIM6RST | Timer6 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer6 |
| 1 | TIM5RST | Timer5 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer5 |
| 0 | TIM4RST | Timer4 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset Timer4 |
| 31~8<br>6~5 | - | Reserved |

### 5.10.14 APB2 Bus Peripheral Reset Control Register (APB2_RST)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| APB2_RST | R/W | APB2 Bus Peripheral Reset Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | UART3RST | LCDRST | LEDPWMRST |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | UART3RST | UART3 Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset UART3 |
| 1 | LCDRST | LCD/LED Module Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset LCD |
| 0 | LEDPWMRST | LEDPWM Reset Control Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset LEDPWM |
| 31~3 | - | Reserved |

### 5.10.15 NMI Interrupt Configuration Register (NMI_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| NMI_CFG | R/W | Non-Maskable Interrupt(NMI) Interrupt Configuration Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY[15:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | INT0EN | CMPEN | CSSEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~16 | KEY[15:0] | NMI_CFG Register Write Protection Switch<br>Writing 0xA05F to KEY[15:0] is required to unlock the lower bits of the current register for modification |
| 2 | INT0EN | External Interrupt INT0 NMI Enable Bit<br>0: INT0 NMI disable<br>1: INT0 NMI enable |

| | | |
|---|---|---|
| | | When enabled, both rising and falling edge interrupts on the INT0 pin will trigger NMI. The corresponding flag must be manually cleared to exit the NMI interrupt.<br>**Note: If INT0 interrupt is enabled, NMI will still be given priority.** |
| 1 | CMPEN | CMP NMI Enable Bit<br>0: CMP NMI disable<br>1: CMP NMI enable<br>When enabled, CMPIF flag set will trigger NMI, and the NMI interrupt can only be exited after manually clearing the CMPIF flag.<br>**Note: If CMP interrupt is enable (CMP_CFG.CMPIM[1:0]=1), NMI will still be given priority.** |
| 0 | CSSEN | CSS NMI Enable Bit<br>0: CSS NMI disable<br>1: CSS NMI enable<br>When enabled, CLKFIF flag set will trigger NMI, and the CLKFIF flag will be cleared after reset.<br>**Note: If RCC interrupt is enable (RCC_CFG.INTEN=1), NMI will still be given priority.** |
| 15~3 | - | Reserved |

### 5.10.16 RCC Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| RCC Base Address:0x4000_3000 | | | | |
| AHB_CFG | 0x00 | R/W | AHB Bus Peripheral Clock Enable Register | 0x0020_0000 |
| AHB_RST | 0x04 | R/W | AHB Bus Peripheral Reset Control Register | 0x0000_0000 |
| RCC_KEY | 0x0C | R/W | RCC Protect Register | 0x0000_0000 |
| RCC_CFG0 | 0x14 | R/W | System Clock Source Selection Register | 0x0000_0000 |
| RCC_CFG1 | 0x18 | R/W | Peripheral Clock Source Selection Register | 0x0000_0000 |
| PLL_CFG | 0x1C | R/W | PLL Configuration Register | 0x0000_0000 |
| RCC_STS | 0x20 | R/W | Clock Status Register | 0x0000_0000 |
| SYST_CALIB | 0x28 | R/W | SysTick Calibration Parameter Register | 0x0000_0000 |
| NMI_CFG | 0x2C | R/W | NMI Interrupt Configuration Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| RCC Base Address:0x4002_0000 | | | | |
| APB0_CFG | 0x4002_0000 | R/W | APB0 Bus Peripheral Clock Enable Register | 0x0000_0000 |
| APB0_RST | 0x4002_0004 | R/W | APB0 Bus Peripheral Reset Control Register | 0x0000_0000 |
| APB1_CFG | 0x4002_1000 | R/W | APB1 Bus Peripheral Clock Enable Register | 0x0000_0000 |
| APB1_RST | 0x4002_1004 | R/W | APB1 Bus Peripheral Reset Control Register | 0x0000_0000 |
| APB2_CFG | 0x4002_2000 | R/W | APB2 Bus Peripheral Clock Enable Register | 0x0000_0000 |
| APB2_RST | 0x4002_2004 | R/W | APB2 Bus Peripheral Reset Control Register | 0x0000_0000 |

# 6 Interrupts

- M0+ core could provide a maximum of 32 interrupt sources, numbered from 0 to 31, while SC32F10T/10G series has 27 interrupt sources.
- Four-level interrupt priorities can be configured, and the interrupt priorities are set through the Interrupt Priority Registers in the core registers.

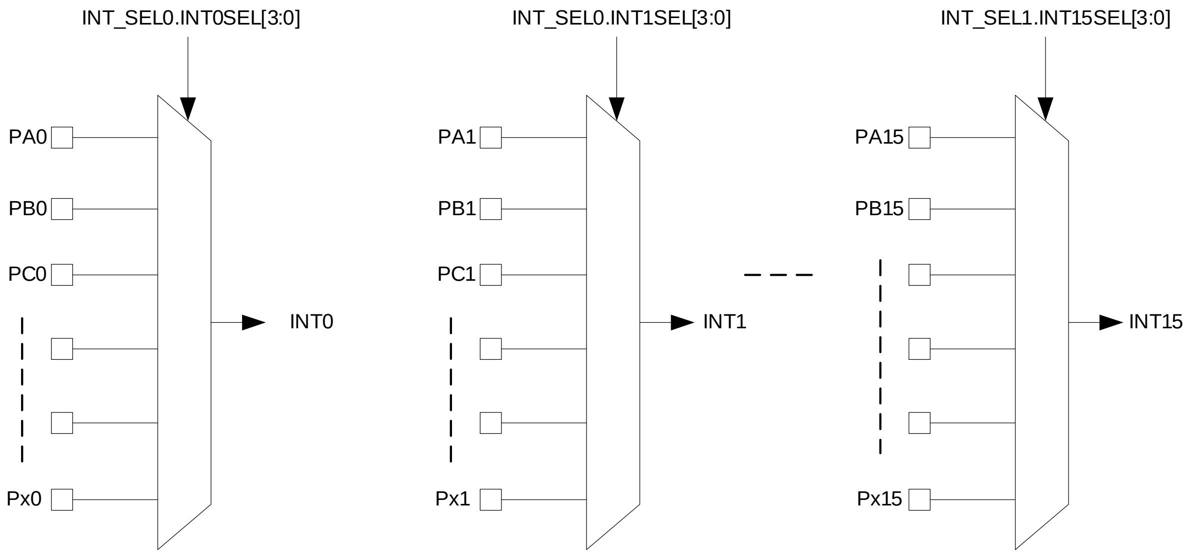
## 6.1 External interrupts INT0~15

External interrupts comprise 16 interrupt sources, occupying a total of 4 interrupt vectors. All 16 external interrupt sources can be configured to respond to rising edges, falling edges, or both edges. Once configured, these interrupts can cover all GPIO pins. When the corresponding event occurs, software sets the corresponding interrupt flag (RIF/FIF to 1), triggering entry into the corresponding interrupt service.

The external interrupt features of the SC32F10T/10G series are as follows:

- 16 INT interrupt sources, occupying 4 interrupt vectors in total.
- After configuration, INT can cover all GPIO pins.
- All INT sources can be configured for rising edge, falling edge, or both edge interrupts, each having independent corresponding interrupt flag.

**Note: When using INT functions, users need to manually set the GPIO port corresponding to INTn (n=0~15) to pull-up input mode. External interrupts cannot be detected in output mode.**



External Interrupt Port Multiplexer

## 6.2 Interrupt and Events

- When NVIC is disabled, interrupt request masks are enabled, events can be generated, but interrupt cannot be generated.
- When NVIC is enabled, interrupt request masks act as internal master interrupt control bit in the module.

## 6.3 Interrupt Source and Vector

| Interrupt Vector | Interrupt Number | Priority | Interrupt Vector Address | Interrupt Source | Core/NVIC Enable Bit | Interrupt Request Mask Bit | Interrupt Subroutine Control Bit | Interrupt Flag | Capability of Waking up STOP |
|---|---|---|---|---|---|---|---|---|---|
| 0 | - | - | 0x0000_0000 | - | | - | \ | \ | YES |
| 1 | - | Fixed | 0x0000_0004 | RESET | PRIMASK | SCB | \ | \ | YES |
| 2 | - | Fixed | 0x0000_0008 | NMI_Handler | | SCB | \ | \ | YES |
| 3 | - | Fixed | 0x0000_000C | HardFault_Handler | PRIMASK | SCB | \ | \ | YES |
| 4~10 | - | - | 0x0000_0010<br>-<br>0x0000_0028 | - | | - | \ | \ | YES |
| 11 | - | Settable | | SVC_Handler | PRIMASK | SCB | \ | \ | YES |
| 12~13 | - | - | 0x0000_0030<br>0x0000_0034 | - | | - | \ | \ | YES |
| 14 | - | Settable | 0x0000_0038 | PendSV_Handler | PRIMASK | SCB | \ | \ | YES |
| 15 | - | Settable | 0x0000_003C | SysTick_Handler | PRIMASK | SysTick_CTRL | \ | \ | YES |
| 16 | 0 | Settable | 0x0000_0040 | INT0 | NVIC->ISER[0].0 | INTF_IE->ENFx，x=0<br>INTR_IE->ENRx | \ | INTF_STS->FIFx<br>INTR_STS->RIFx | YES |
| 17 | 1 | Settable | 0x0000_0044 | INT1-7 | NVIC->ISER[0].1 | INTF_IE->ENFx，<br>x=1~7<br>INTR_IE->ENRx | \ | INTF_STS->FIFx<br>INTR_STS->RIFx | YES |
| 18 | 2 | Settable | 0x0000_0048 | INT8-11 | NVIC->ISER[0].2 | INTF_IE->ENFx，<br>x=8~11<br>INTR_IE->ENRx | \ | INTF_STS->FIFx<br>INTR_STS->RIFx | YES |
| 19 | 3 | Settable | 0x0000_004C | INT12-15 | NVIC->ISER[0].3 | INTF_IE->ENFx，<br>x=12~15<br>INTR_IE->ENRx | \ | INTF_STS->FIFx<br>INTR_STS->RIFx | YES |
| 20 | 4 | Settable | 0x0000_0050 | RCC Oscillation Stop Detection | NVIC->ISER[0].4 | RCC_CFG->INTEN | \ | RCC_STS->CLKFIF | YES |
| 21 | 5 | Reserved | 0x0000_0054 | \ | \ | \ | \ | \ | |
| 22 | 6 | Settable | 0x0000_0058 | BTM | NVIC->ISER[0].6 | BTM_CON->INTEN | \ | BTM_STS->BTMIF | YES |
| 23 | 7 | Settable | 0x0000_005C | UART0 | NVIC->ISER[0].7 | UART0_IDE->INTEN | UART0_IDE->TXIE<br>UART0_IDE->RXIE | UART0_STS->TXIF<br>UART0_STS->RXIF | NO |
| | | | | UART2 | \ | UART2_IDE->INTEN | UART2_IDE->TXIE<br>UART2_IDE->RXIE | UART2_STS->TXIF<br>UART2_STS->RXIF | |
| 24 | 8 | Settable | 0x0000_0060 | UART1 | NVIC->ISER[0].8 | UART1_IDE->INTEN | UART1_IDE->TXIE<br>UART1_IDE->RXIE | UART1_STS->TXIF<br>UART1_STS->RXIF | NO |
| | | | | UART3 | \ | UART3_IDE->INTEN | UART3_IDE->TXIE<br>UART3_IDE->RXIE | UART3_STS->TXIF<br>UART3_STS->RXIF | |
| 25 | 9 | Settable | 0x0000_0064 | SPI0 | NVIC->ISER[0].9 | SPI0_IDE->INTEN | SPI0_IDE->RXNEIE<br>SPI0_IDE->TBIE<br>SPI0_IDE->RXIE | SPI0_STS->SPIF<br>SPI0_STS->RXNEIF<br>SPI0_STS->TXEIF | NO |

| Interrupt Vector | Interrupt Number | Priority | Interrupt Vector Address | Interrupt Source | Core/NVIC Enable Bit | Interrupt Request Mask Bit | Interrupt Subroutine Control Bit | Interrupt Flag | Capability of Waking up STOP |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  | SPI0_IDE->RXHIE<br>SPI0_IDE->TXHIE | SPI0_STS->RXFIF<br>SPI0_STS->RXHIF<br>SPI0_STS->TXHIF |  |
| 26 | 10 | Settable | 0x0000_0068 | SPI1 | NVIC->ISER[0].10 | SPI1_IDE->INTEN | \ | SPI1_STS->TXHIF | NO |
| 27 | 11 | Settable | 0x0000_006C | DMA0 | NVIC->ISER[0].11 | DMA0_CFG->INTEN | DMA0_CFG->TCIE<br>DMA0_CFG->HTIE<br>DMA0_CFG->TEIE | DMA0_STS->GIF<br>DMA0_STS->TCIF<br>DMA0_STS->HTIF<br>DMA0_STS->TEIF | NO |
| 28 | 12 | Settable | 0x0000_0070 | DMA1 | NVIC->ISER[0].12 | DMA1_CFG->INTEN | DMA1_CFG->TCIE<br>DMA1_CFG->HTIE<br>DMA1_CFG->TEIE | DMA1_STS->GIF<br>DMA1_STS->TCIF<br>DMA1_STS->HTIF<br>DMA1_STS->TEIF | NO |
| 29 | 13 | Settable | 0x0000_0074 | DMA2 | NVIC->ISER[0].13 | DMA2_CFG->INTEN | DMA2_CFG->TCIE<br>DMA2_CFG->HTIE<br>DMA2_CFG->TEIE | DMA2_STS->GIF<br>DMA2_STS->TCIF<br>DMA2_STS->HTIF<br>DMA2_STS->TEIF | NO |
| 30 | 14 | Settable | 0x0000_0078 | DMA3 | NVIC->ISER[0].14 | DMA3_CFG->INTEN | DMA3_CFG->TCIE<br>DMA3_CFG->HTIE<br>DMA3_CFG->TEIE | DMA3_STS->GIF<br>DMA3_STS->TCIF<br>DMA3_STS->HTIF<br>DMA3_STS->TEIF | NO |
| 31 | 15 | Settable | 0x0000_007C | TIM0 | NVIC->ISER[0].15 | TIM0_IDE->INTEN | TIM0_IDE->TIE<br>TIM0_IDE->EXFIE<br>TIM0_IDE->EXRIE | TIM0_STS->TIF<br>TIM0_STS->EXIF<br>TIM0_STS->EXIR | NO |
| 32 | 16 | Settable | 0x0000_0080 | TIM1 | NVIC->ISER[0].16 | TIM1_IDE->INTEN | TIM1_IDE->TIE<br>TIM1_IDE->EXFIE<br>TIM1_IDE->EXRIE | TIM1_STS->TIF<br>TIM1_STS->EXIF<br>TIM1_STS->EXIR | NO |
| 33 | 17 | Settable | 0x0000_0084 | TIM2 | NVIC->ISER[0].17 | TIM2_IDE->INTEN | TIM2_IDE->TIE<br>TIM2_IDE->EXFIE<br>TIM2_IDE->EXRIE | TIM2_STS->TIF<br>TIM2_STS->EXIF<br>TIM2_STS->EXIR | NO |
| 34 | 18 | Settable | 0x0000_0088 | TIM3 | NVIC->ISER[0].18 | TIM3_IDE->INTEN | TIM3_IDE->TIE<br>TIM3_IDE->EXFIE<br>TIM3_IDE->EXRIE | TIM3_STS->TIF<br>TIM3_STS->EXIF<br>TIM3_STS->EXIR | NO |
| 35 | 19 | Settable | 0x0000_008C | TIM4 | NVIC->ISER[0].19 | TIM4_IDE->INTEN | TIM4_IDE->TIE<br>TIM4_IDE->EXFIE<br>TIM4_IDE->EXRIE | TIM4_STS->TIF<br>TIM4_STS->EXIF<br>TIM4_STS->EXIR | NO |
|  |  |  |  | TIM5 | \ | TIM5_IDE->INTEN | TIM5_IDE->TIE<br>TIM5_IDE->EXFIE<br>TIM5_IDE->EXRIE | TIM5_STS->TIF<br>TIM5_STS->EXIF<br>TIM5_STS->EXIR | NO |
| 36 | 20 | Settable | 0x0000_0090 | TIM6 | NVIC->ISER[0].20 | TIM6_IDE->INTEN | TIM6_IDE->TIE<br>TIM6_IDE->EXFIE<br>TIM6_IDE->EXRIE | TIM6_STS->TIF<br>TIM6_STS->EXIF<br>TIM6_STS->EXIR | NO |

| Interrupt Vector | Interrupt Number | Priority | Interrupt Vector Address | Interrupt Source | Core/NVIC Enable Bit | Interrupt Request Mask Bit | Interrupt Subroutine Control Bit | Interrupt Flag | Capability of Waking up STOP |
|---|---|---|---|---|---|---|---|---|---|
| | | | | TIM7 | \ | TIM7_IDE->INTEN | TIM7_IDE->TIE<br>TIM7_IDE->EXFIE<br>TIM7_IDE->EXRIE | TIM7_STS->TIF<br>TIM7_STS->EXIF<br>TIM7_STS->EXIR | NO |
| 37 | 21 | Settable | 0x0000_0094 | PWM0 | NVIC->ISER[0].21 | PWM0_CON->INTEN | \ | PWM0_STS->PWMIF | NO |
| 38 | 22 | Settable | 0x0000_0098 | LEDPWM | NVIC->ISER[0].22 | LEDPWM_CON->INTEN | \ | LEDPWM_STS->PWMIF | NO |
| 39 | 23 | Settable | 0x0000_009C | TWI0 | NVIC->ISER[0].23 | TWI0_IDE->INTEN | \ | TWI0_STS->TWIF | NO |
| 40 | 24 | Settable | 0x0000_00A0 | TWI1 | NVIC->ISER[0].24 | TWI1_IDE->INTEN | \ | TWI1_STS->TWIF | NO |
| 41 | 25 | Reserved | 0x0000_00A4 | \ | \ | \ | \ | \ | |
| 42 | 26 | Reserved | 0x0000_00A8 | \ | \ | \ | \ | \ | |
| 43 | 27 | Reserved | 0x0000_00AC | \ | \ | \ | \ | \ | |
| 44 | 28 | Reserved | 0x0000_00B0 | \ | \ | \ | \ | \ | |
| 45 | 29 | Settable | 0x0000_00B4 | ADC | NVIC->ISER[0].29 | ADC_CON->INTEN | \ | ADC_STS->ADCIF | NO |
| 46 | 30 | Settable | 0x0000_00B8 | CMP | NVIC->ISER[0].30 | \ | CMPCFG->CMPIM[1:0] | CMP_STS->CMPIF | YES |
| 47 | 31 | Settable | 0x0000_00BC | TK | NVIC->ISER[0].31 | TKCON->INTEN | \ | TKIF | YES |

## 6.4 External Interrupt Register

### 6.4.1 External Interrupt Falling Edge Interrupt Enable Register (INTF_IE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTF_IE | R/W | INT Falling Edge Enable Register | 0x0000_0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ENF15 | ENF14 | ENF13 | ENF12 | ENF11 | ENF10 | ENF9 | ENF8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENF7 | ENF6 | ENF5 | ENF4 | ENF3 | ENF2 | ENF1 | ENF0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | ENFx<br>(x=0~15) | INTx Falling Edge Enable Control Bit(x=0~15)<br>0: Disable<br>1: Enable |
| 31~16 | - | Reserved |

### 6.4.2 External Interrupt Rising Edge Interrupt Enable Register (INTR_IE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTR_IE | R/W | INT Rising Edge Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ENR15 | ENR14 | ENR13 | ENR12 | ENR11 | ENR10 | ENR9 | ENR8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENR7 | ENR6 | ENR5 | ENR4 | ENR3 | ENR2 | ENR1 | ENR0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | ENRx<br>(x=0~15) | INTx Rising Edge Enable Control Bit(x=0~15)<br>0: Disable<br>1: Enable |
| 31~16 | - | Reserved |

### 6.4.3 External Interrupt Port Selection Register0 (INT_SEL0)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INT_SEL0 | R/W | External Interrupt Port Selection Register0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| INT7SEL[3:0] | | | | INT6SEL[3:0] | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INT5SEL[3:0] | | | | INT4SEL[3:0] | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INT3SEL[3:0] | | | | INT2SEL[3:0] | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT1SEL[3:0] | | | | INT0SEL[3:0] | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | INTxSEL[3:0]<br>(x=0~7) | External Interrupt INTx Port Selection Bit(x=0~7)<br>0000: Select PAx Port<br>0001: Select PBx Port<br>0010: Select PCx Port<br>Others: Reserved |

### 6.4.4 External Interrupt Port Selection Register1 (INT_SEL1)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INT_SEL1 | R/W | External Interrupt Port Selection Register1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| INT15SEL[3:0] | | | | INT14SEL[3:0] | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INT13SEL[3:0] | | | | INT12SEL[3:0] | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INT11SEL[3:0] | | | | INT10SEL[3:0] | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT9SEL[3:0] | | | | INT8SEL[3:0] | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | INTxSEL[3:0]<br>(x=8~15) | External Interrupt INTx Port Selection Bit(x=8~15)<br>0000: Select PAx Port<br>0001: Select PBx Port<br>0010: Select PCx Port<br>Others: Reserved |

### 6.4.5 External Interrupt Falling Edge Control Register (INTF_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTF_CON | R/W | External Interrupt Falling Edge Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | FTx<br>(x=0~15) | INTx Falling Edge Detection Enable Bit(x=0~15)<br>0: Disable<br>1: Enable |
| 31~16 | - | Reserved |

### 6.4.6 External Interrupt Rising Edge Control Register (INTR_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTR_CON | R/W | External Interrupt Rising Edge Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | RTx<br>(x=0~15) | INTx Rising Edge Detection Enable Bit(x=0~15)<br>0: Disable<br>1: Enable |
| 31~16 | - | Reserved |

### 6.4.7 External Interrupt Falling Edge Flag Register (INTF_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTF_STS | R/W | External Interrupt Falling Edge Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FIF15 | FIF14 | FIF13 | FIF12 | FIF11 | FIF10 | FIF9 | FIF8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIF7 | FIF6 | FIF5 | FIF4 | FIF3 | FIF2 | FIF1 | FIF0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | FIFx<br>(x=0~15) | INTx Falling Edge Capture Flag(x=0~15)<br>This bit will be set to 1 by hardware when a falling edge is detected, and can be cleared by software.<br>It is possible to trigger a falling edge capture interrupt by setting this bit to 1 by software. |
| 31~16 | - | Reserved |

### 6.4.8 External Interrupt Rising Edge Flag Register (INTR_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| INTR_STS | R/W | External Interrupt Rising Edge Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RIF15 | RIF14 | RIF13 | RIF12 | RIF11 | RIF10 | RIF9 | RIF8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RIF7 | RIF6 | RIF5 | RIF4 | RIF3 | RIF2 | RIF1 | RIF0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | RIFx<br>(x=0~15) | INTx Rising Edge Capture Flag(x=0~15)<br>This bit will be set to 1 by hardware when a rising edge is detected, and can be cleared by software.<br>It is possible to trigger a rising edge capture interrupt by setting this bit to 1 by software. |
| 31~16 | - | Reserved |

### 6.4.9 External Interrupt Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| External Interrupt Base Address:0x4001_1300 | | | | |
| INTF_IE | 0x00 | R/W | External Interrupt Falling Edge Interrupt Enable Register | 0x0000_0000 |
| INTR_IE | 0x20 | R/W | External Interrupt Rising Edge Interrupt Enable Register | 0x0000_0000 |
| INT_SEL0 | 0x40 | R/W | External Interrupt Port Selection Register0 | 0x0000_0000 |
| INT_SEL1 | 0x60 | R/W | External Interrupt Port Selection Register1 | 0x0000_0000 |
| INTF_CON | 0x80 | R/W | External Interrupt Falling Edge Control Register | 0x0000_0000 |
| INTR_CON | 0xA0 | R/W | External Interrupt Rising Edge Control Register | 0x0000_0000 |
| INTF_STS | 0xC0 | R/W | External Interrupt Falling Edge Flag Register | 0x0000_0000 |
| INTR_STS | 0xE0 | R/W | External Interrupt Rising Edge Flag Register | 0x0000_0000 |

# 7 Power Saving Mode

Upon initial power-up, the system runs in Normal Mode. Additionally, three power-saving modes are available:

- Low-Speed Mode: The system clock source can be LIRC, and the CPU can operate at 32KHz.
- IDLE Mode: The system can be awakened by any interrupt.
- STOP Mode: The system can be awakened by INT0~15, Base Timer, TK, and CMP.

# 8 GPIO

## 8.1 Clock Source

M0+ core can achieve single-cycle access to GPIO through the IOPORT bus, resulting in highly efficient data transfer. The IOPORT bus clock is derived from HCLK.

## 8.2 Feature

The GPIO port features of the SC32F10T/10G series are as follows:

- A maximum of 46 bidirectional independently controlled GPIOs
- CPU can access GPIO ports through the IOPORT bus in a single cycle
- Independent setting of pull-up resistors
- All ports have four levels of source driving capability
- All I/Os have high sink current driving capability (50mA)
- 16 I/Os in one group
- Whether input mode or output mode, reading from the port data register retrieves the actual status value of the port

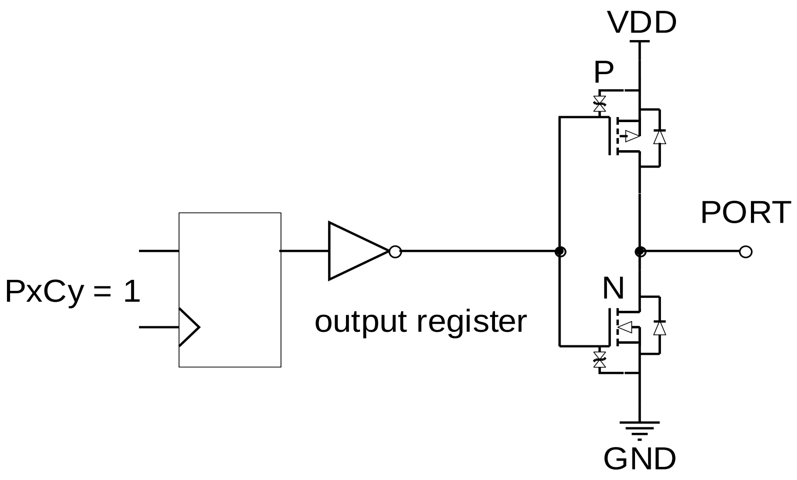
**Note: Unused and non-exported ports should be set to strong push-pull output mode**

## 8.3 GPIO Structure Diagram

### Strong Push-pull Output Mode

In the strong push-pull output mode, it can provide continuous high-current drive: an output greater than 10mA is high, and an output greater than 50mA is low.

The schematic diagram of the port structure of the strong push-pull output mode is as follows:

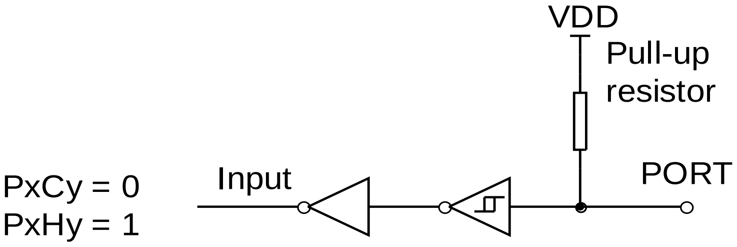


Strong push-pull output mode

**Pull-up Input Mode**

In the pull-up input mode, a pull-up resistor is constantly connected to the input port. Only when the input port is pulled low, the low-level signal is detected.

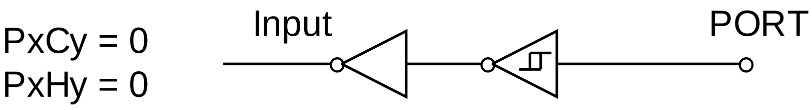
The schematic diagram of the port structure with pull-up input mode is as follows:



Input mode with pull-up resistor

**High Impedance Input Mode (Input only)**

The schematic diagram of the port structure of the high impedance input mode is as follows:



High impedance input mode

## 8.4 GPIO Register

### 8.4.1 Port PX Data Register (PX)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PX<br>X=A,B,C,D | R/W | Port PX Data Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PD15 | PD14 | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PDn<br>(n=0~15) | Port PXn Data Register，X=A,B,C,D，n=0~15<br>Port latch register data, value read from port data register is the actual state value of the port. |
| 31~16 | - | Reserved |

### 8.4.2 Port PX Data Register (PXn_BIT)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PXn_BIT<br>X=A,B,C,D n=0~15 | R/W | Port PX Data Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | BSRn | Port PXn Bit Assignment Control, n=0~15<br>Used for individual assignment of the PXn port bit. |
| 31~1 | - | Reserved |

### 8.4.3 Port PX Data Register (PXn_XR)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PXn_XR<br>X=A,B,C,D n=0~15 | R/W | Toggle PXn | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | XRn | Port PXn Bit Toggle Control, n=0~15 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 0: Invalid<br>1: Toggle the output of PXn |
| 31~1 | - | Reserved |

### 8.4.4 Port PX Input/Output Control Register (PXCON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PXCON<br>X=A,B,C,D | R/W | Port PX Input/Output Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | MODEn<br>(n=0~15) | Port PXn Strong Push-Pull Mode Enable Bit, n=0~15<br>0: PXn in input mode(default at power-up)<br>1: PXn in strong push-pull mode |
| 31~16 | - | Reserved |

### 8.4.5 Port PX Pull-up Resister Control Register (PXPH)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PXPH<br>X=A,B,C,D | R/W | Port PX Pull-Up Resister Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PUPDn<br>(n=0~15) | Port PXn Pull-Up Resister Enable Bit, n=0~15 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 0: PXn in high-impedance input mode(default at power-up), pull-up resister disable<br>1: PXn pull-up resistor enable |
| 31~16 | - | Reserved |

### 8.4.6 GPIO Drive Level Register (PXLEV)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PXLEV<br>X=A,B,C,D | R/W | GPIO Drive Level Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LEV15[1:0] | | LEV14[1:0] | | LEV13[1:0] | | LEV12[1:0] | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LEV11[1:0] | | LEV10[1:0] | | LEV9[1:0] | | LEV8[1:0] | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LEV7[1:0] | | LEV6[1:0] | | LEV5[1:0] | | LEV4[1:0] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LEV3[1:0] | | LEV2[1:0] | | LEV1[1:0] | | LEV0[1:0] | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | LEVn[1:0]<br>(n=0~15) | Port PXn Level Control Bit, n=0~15<br>Used for configuring the $I_{OH}$ level of Port PXn<br>00: Level 0(Maximum)<br>01: Level 1<br>10: Level 2<br>11: Level 3(Minimum) |

### 8.4.7 GPIO Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PA Base Address:0x4001_1000 | | | | |
| PA | 0x00 | R/W | Port PA Data Register Register | 0x0000_0000 |
| PACON | 0x20 | R/W | Port PA Input/Output Control Register | 0x0000_0000 |
| PAPH | 0x40 | R/W | Port PA Pull-Up Resister Control Register | 0x0000_0000 |
| PALEV | 0x60 | R/W | GPIO Drive Level Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PB Base Address: 0x4001_1100 | | | | |
| PB | 0x00 | R/W | Port PB Data Register Register | 0x0000_0000 |
| PBCON | 0x20 | R/W | Port PB Input/Output Control Register | 0x0000_0000 |
| PBPH | 0x40 | R/W | Port PB Pull-Up Resister Control Register | 0x0000_0000 |
| PBLEV | 0x60 | R/W | GPIO Drive Level Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PC Base Address:0x4001_1200 | | | | |
| PC | 0x00 | R/W | Port PC Data Register Register | 0x0000_0000 |
| PCCON | 0x20 | R/W | Port PC Input/Output Control Register | 0x0000_0000 |
| PCPH | 0x40 | R/W | Port PC Pull-Up Resister Control Register | 0x0000_0000 |
| PCLEV | 0x60 | R/W | GPIO Drive Level Register | 0x0000_0000 |

# 9 Analog-to-Digital Converter ADC

## 9.1 Overview

The SC32F10T/10G series features a 14-bit successive approximation type analog-to-digital converter (ADC). It supports up to 13 multiplexed channels and can measure signals from 13 external sources and 1 internal source. The A/D conversion for each channel can be performed in single-shot or continuous sampling modes. The results of the ADC are stored in a 32-bit data register.

## 9.2 Clock source

- The SC32F10T/10G series ADC has only one clock source, which is derived from PCLK
- Fixed conversion time of 950ns

## 9.3 Feature

- Precision:14 bits
- Maximum Channels: Supports up to 13 channels, including 12 external ADC sampling channels and other functions multiplexed with I/O ports. Additionally, one internal ADC channel can directly measure the VDD voltage
- Built-in Reference Voltages: 2.4V,2.048V, and 1.024V
- Reference Voltage Selection: VDD,2.4V,2.048V and 1.024V
- Direct Measurement of VDD: The internal ADC can directly measure the VDD voltage
- ADC Input Channel Selection: Can be configured through the ADCIS[4:0] bits.
- Software-Triggered Conversion: The conversion process can be initiated by software
- Interrupt Support: Configurable ADC conversion completion interrupt
- Conversion Time: Sampling to completion time as low as 2μs
- DMA Transfer Support: ADC conversion completion can generate a DMA request
- Single-Channel Continuous Conversion Mode Support: Allows continuous conversion in single-channel mode
- Overflow Flag: The ADC conversion result supports an overflow flag, and the OVERRUN flag is in the same register (ADCV), allowing the user to read both at once

## 9.4 Conversion Modes

The SC32F10T/10G series ADC has two conversion modes:

### 9.4.1 Single Conversion Mode (CONT=0)

Single conversion mode is commonly used for software-triggered sampling. In this mode, if a software or hardware trigger event occurs, ADC will perform a single conversion on the selected sampling channel. This mode is selected when ADC_CON.CONT=0. After the conversion is complete:

- The converted data will be stored in the ADCV Register

- The ADCIF (conversion complete) flag will be set to 1
- An interrupt will be generated when ADC_CON.INTEN=1

Afterward, the ADC will stop working until the ADCS bit is set to 1 again.

### 9.4.2 Continuous Conversion (CONT=1)

Continuous conversion mode is commonly used in conjunction with DMA. In this mode, if a software or hardware trigger event occurs, the ADC performs continuous conversions on the selected sampling channel. This mode is selected when ADC_CON.CONT=1. After each conversion is complete:

- The converted data will be stored in the ADCV Register
- The ADCIF (conversion complete) flag will be set to 1
- An interrupt will be generated when ADC_CON.INTEN=1

Afterward, the ADC continues to repeat the conversions on the selected sampling channel.

## 9.5 ADC Overflow

If the converted data is not read promptly by the CPU or DMA before new data is generated, an overflow flag (OVERRUN) will indicate an overflow event.

When an overflow occurs, the ADC will remain in working state and can continue with conversions. However, the OVERRUN flag will be set to 1 by hardware, and the value of ADCV will be overwritten by the latest conversion result and any previously unread data will be lost.

The OVERRUN flag is set to 1 by hardware when an overflow occurs, and it is automatically cleared to 0 after reading ADCV.

## 9.6 ADC and DMA Controller Collaboration

By selecting one of the DMA channels with REQSRC[5:0]=59 (indicating the DMA channel's request source is ADC) and setting ADC_CON.DMAEN=1, a DMA request will be generated after every ADC conversion. After enabling DMA and ADC, DMA can transfer the converted data from the ADCV Register to the target location selected by the software.

If DMA cannot process the DMA transfer request promptly, an overflow (OVERRUN=1) will be generated by the ADC. However, this does not affect the DMA transfer request. Users can read the ADCV value from the RAM area and check if the most significant bit is 1 to determine whether an overflow has occurred.

## 9.7 ADC Conversion Steps

The actual operation steps required for the user to perform ADC conversion are as follows:

① Set the ADC input pin; (set the bit corresponding to AINx as ADC input, usually the ADC pin will be fixed in advance);

② Set ADC reference voltage Vref, set the frequency used for ADC conversion;

③ Set ADCEN to enable the ADC module power supply;

④ Select ADC input channel; (set ADCIS bit, select ADC input channel);

⑤ Start ADCS and start conversion;

⑥ Wait for EOC/ADCIF=1. If the ADC interrupt is enabled, the ADC interrupt will be generated. The user needs to clear the EOC/ADCIF flag by software;

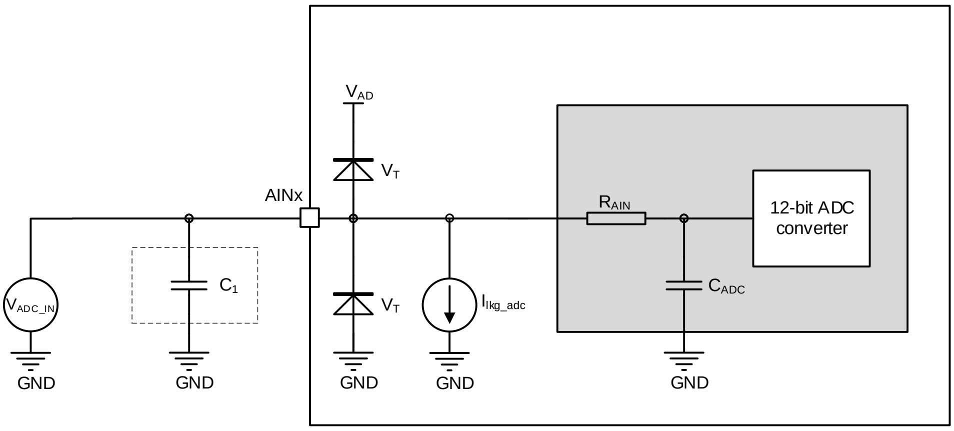
⑦ Get 14-bit data from ADCV, then one conversion is completed;

⑧ If the input channel is not changed, continuous conversion mode can be set by setting CONT to 1 through software. The conversion will continue until this bit is cleared to 0;

⑨ When the ADC conversion result overflows, the OVERRUN flag will set to 1;

⑩ Conversion data can be transferred using DMA;

## 9.8 ADC Connection Circuit Diagram



**Note: If users need to further enhance ADC performance, it is** recommended **to externally connect a C1 capacitor to the AINx channel, with a capacitance value of 0.01uF.**

## 9.9 ADC Interrupt

After the SC32F10T/10G series ADC conversion complete, the ADCIF flag will be set, and if ADC_CON.INTEN=1, an interrupt will be generated.

| Interrupt Event | Event Flag | Interrupt Enable Control Bit |
|---|---|---|
| ADC conversion completion interrupt request | ADCIF | ADC_CON->INTEN |

## 9.10 ADC Register

### 9.10.1 ADC Control Register (ADC_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| ADC_CON | R/W | ADC Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | VREFS[1:0] | | - | LOWSP[2:0] | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ADCEN | - | - | DMAEN | CONT | - | - | INTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCS | - | - | ADCIS[4:0] | | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 21~20 | VREFS[1:0] | Reference Voltage Selection Control Bit<br>00: Select VDD as $V_{REF}$ of ADC<br>01: Select internally accurate 2.048V as $V_{REF}$ of ADC<br>10: Select internally accurate 1.024V as $V_{REF}$ of ADC<br>11: Select internally accurate 2.4V as $V_{REF}$ of ADC |
| 18~16 | LOWSP[2:0] | ADC Sampling Period Selection Control Bit<br>100: Sampling tim is 3 system clock(about 100ns @ $f_{PCLK2}$=32MHz)<br>101: Sampling tim is 6 system clock(about 200ns @ $f_{PCLK2}$=32MHz)<br>110: Sampling tim is 16 system clock(about 500ns @ $f_{PCLK2}$=32MHz)<br>111: Sampling tim is 32 system clock(about 1000ns @ $f_{PCLK2}$=32MHz)<br>Others: Reserved<br>Description: The total time for ADC from sampling to completing the conversion is calculated as follows:<br>$T_{ADC}$ = Sampling time + Conversion time<br>ADC conversion time is fixed at 950ns |
| 15 | ADCEN | ADC Module Power Startup Control Bit<br>0: Disable ADC module power<br>1: Enable ADC module power |
| 12 | DMAEN | DMA Request Enable Control Bit<br>This bit is used to enable the generation of DMA requests. Setting to 1 allows the DMA controller to automatically manage the data from ADC conversions.<br>0: Disable DMA request<br>1: Enable DMA request<br>**Note: When performing a write operation on this bit through software, please ensure no conversions are currently ongoing.** |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 11 | CONT | Single/Continuous Conversion Mode Select Bit<br>This bit can be set to 1 or cleared by software. When this bit is set to 1, conversion will continue until this bit is cleared<br>0: Single Conversion Mode<br>1: Continuous Conversion Mode |
| 8 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 7 | ADCS | ADC Conversion Trigger Control Bit<br>This bit serves as the trigger signal for ADC conversion, set to 1 by software, and cleared to 0 by hardware. Writing 1 to this bit triggers a single ADC conversion.<br>**Note: After setting ADCS to 1, refrain from writing to the ADC_CON Register until the interrupt flag ADCIF is set.** |
| 4~0 | ADCIS[4:0] | ADC Input Channel Selection Bit<br>00000: Select AIN0 as the input of ADC<br>00001: Select AIN0 as the input of ADC<br>00111: Select AIN7 as the input of ADC<br>01000: Select AIN8 as the input of ADC<br>01001: Select AIN9 as the input of ADC<br>01010: Select AIN10 as the input of ADC<br>01011: Select AIN11 as the input of ADC<br>01100: Select AIN12 as the input of ADC<br>01101: Select AIN13 as the input of ADC<br>01110: Select AIN14 as the input of ADC<br>01111: Select AIN15 as the input of ADC<br>10000: Select AIN16 as the input of ADC<br>10001~11110: Select AIN16 as the input of ADC<br>11111: Select 1/4VDD as the input of ADC, and can be used to measure the supply voltage |
| 31~22<br>19<br>14~13<br>10~9<br>6~5 | - | Reserved |

### 9.10.2 ADC Flag Register (ADC_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| ADC_STS | R/W | ADC Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | ADCIF |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | ADCIF | ADC Interrupt Request Flag<br>This bit is set to 1 by hardware, and is cleared by writing to 1 through software.<br>This bit will be set to 1 by hardware after the ADC conversion is complete, and if ADC_CON.INTEN=1, an interrupt will be generated. |
| 31~1 | - | Reserved |

### 9.10.3 ADC Conversion Value Register (ADCV)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| ADCV | Read Only | ADC Conversion Value Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31 | OVERRUN | Flowout Flag(Read Only)<br>If ADC conversion request is not handled promptly by the CPU or DMA, this bit will be set by hardware. This bit will be automatically cleared after reading ADCV.<br>**Note: When overflow occurs, the value of ADCV will be overwritten by the latest conversion result and any previously unread data will be lost.** |
| 13~0 | ADCV[13:0] | 14 bits ADC conversion results |
| 30~14 | - | Reserved |

### 9.10.4 ADC Port Configuration Register (ADC_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| ADC_CFG | R/W | ADC Port Configuration Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | AIN16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AIN15 | AIN14 | AIN13 | AIN12 | AIN11 | AIN10 | AIN9 | AIN8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AIN7 | - | - | - | - | - | AIN1 | AIN0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 16~7<br>1~0 | AINx<br>(x=0~1,7~16) | ADC Port Configuration Register<br>0: Not selected as AINx, the pin corresponding to AINx functions as GPIO or another multiplexed function<br>1: Selected as AINx for ADC input, and automatically removes the pull-up resistor on the pin associated with AINx |
| 31~17<br>6~2 | - | Reserved |

### 9.10.5 ADC Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| ADC Base Address:0x4002_2110 | | | | |
| ADC_CON | 0x00 | R/W | ADC Control Register | 0x0000_0000 |
| ADC_STS | 0x04 | R/W | ADC Flag Register | 0x0000_0000 |
| ADCV | 0x08 | R/W | ADC Conversion Value Register | 0x0000_0000 |
| ADC_CFG | 0x0C | R/W | ADC Port Configuration Register | 0x0000_0000 |

# 10 Analog Comparator CMP

The SC32F10T/10G series features a built-in analog comparator (CMP), and CMP interrupt can wake up the STOP Mode. It can be used for applications such as alarm circuits, power supply voltage monitoring circuits, zero-crossing detection circuits, etc.

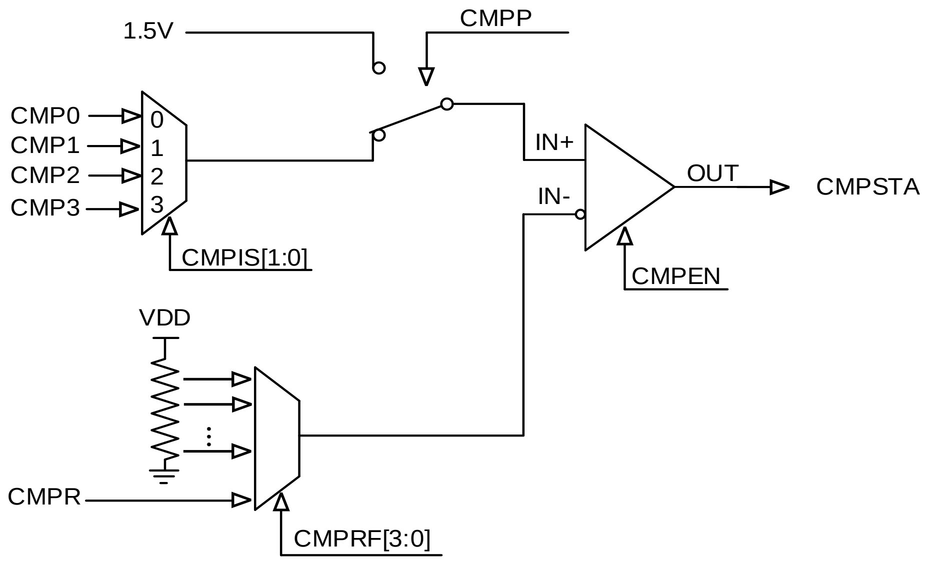
The comparator has four analog signal positive input terminals: CMP0~3, which can be selected through CMPIS [1:0]. The negative input terminal voltage can be switched through CMPRF[3:0] to an external voltage on the CMPR pin or one of the 16 reference voltages internally.

The interrupt mode of the comparator can be conveniently set using CMPIM[1:0]. When the interrupt condition set by CMPIM[1:0] occurs, the comparator interrupt flag CMPIF will set to 1. This interrupt flag needs to be cleared by software.

## 10.1 Feature

- Four analog signal positive input terminals: CMP0~CMP3
- Negative input voltage can be selected from CMPR handover or one of the 16 comparison voltages derived from the internal VDD division
- CMP interrupt can wake up the STOP Mode

## 10.2 Analog Comparator Structure Diagram



Analog Comparator Structure Diagram

## 10.3 CMP Register

### 10.3.1 CMP Status Register (CMP_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CMP_STS | R/W | CMP Status Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 1 | CMPSTA | CMP Output Status Bit<br>0: CMP positive terminal voltage is less than negetive terminal voltage<br>1: CMP positive terminal voltage is greater than negetive terminal voltage |
| 0 | CMPIF | CMP Interrupt Flag<br>This bit is set to 1 by hardware, and is cleared by writing to 1 through software.<br>0: CMP interrupt has not been inturrput<br>1: This bit will be set to 1 by hardware if CMP meets the interrupt trigger condition. And CMP interrupt will be generated if CMPIM[1:0] is not 00. |
| 31~2 | - | Reserved |

### 10.3.2 CMP Configuration Register (CMP_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CMP_CFG | R/W | CMP Configuration Register | 0x0000_0000 |

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

| | | | | | |
|---|---|---|---|---|---|
| CMPEN | CMPIM[1:0] | CMPP | - | - | CMPIS[1:0] |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 11~8 | CMPRF[3:0] | CMP Negetive Terminal Voltage Selection Bit<br>CMP negetive terminal voltage settings is as follow:<br>0000: CMPR；<br>0001: 1/16VDD<br>0010: 2/16VDD<br>0011: 3/16VDD<br>0100: 4/16VDD<br>0101: 5/16VDD<br>0110: 6/16VDD<br>0111: 7/16VDD<br>1000: 8/16VDD<br>1001: 9/16VDD<br>1010: 10/16VDD<br>1011: 11/16VDD<br>1100: 12/16VDD<br>1101: 13/16VDD<br>1110: 14/16VDD<br>1111: 15/16VDD |
| 7 | CMPEN | CMP Enable Bit<br>0: Disable CMP<br>1: Enable CMP |
| 6~5 | CMPIM[1:0] | CMP Interrupt Mode Selection Bit<br>00: No interrupt generated<br>01: Rising edge interrupt: Interrupt will be generated when IN+ transitions from being less than IN- to being greater than IN-<br>10: Falling edge interrupt: Interrupt will be generated when IN+ transitions from being greater than IN- to being less than IN-<br>11: Both edge interrupt: Interrupt will be generated when IN+ transitions from being less than IN- to being greater than IN- or from being greater than IN- to being less than IN- |
| 4 | CMPP | CMP Positive Terminal Input Selection:<br>0: The positive input of CMP is one of CMP0~3, as set by CMPIS[1:0]<br>1: The positive input of CMP is the internal 1.5V reference voltage |
| 1~0 | CMPIS[1:0] | CMP Positive Terminal Input Channel Selection Bit<br>This bit is invalid when CMPP=1:<br>00: Select CMP0 as the input of CMP positive terminal<br>01: Select CMP1 as the input of CMP positive terminal<br>10: Select CMP2 as the input of CMP positive terminal<br>11: Select CMP3 as the input of CMP positive terminal |
| 31~12<br>3~2 | - | Reserved |

### 10.3.3 CMP Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| CMP Base Address:0x4002_2110 | | | | |
| CMP_STS | 0x00 | R/W | CMP Status Register | 0x0000_0000 |
| CMP_CFG | 0x04 | R/W | CMP Configuration Register | 0x0000_0000 |

# 11 UART0~3

## 11.1 Clock Source

The SC32F10T/10G series UART has only one clock source, which is derived from PCLK

## 11.2 Feature

- Four UARTs，UART0~3
  - UART2 can be mapped to another set of ports
- Each UART has three communication modes to choose from:
  - Mode 0: 8-bit half-duplex synchronous communication mode, serial data is transmitted and received on the RX pin. The TX pin is used as the transmit shift clock. Each frame transmits or receives 8 bits, with the low bit transmitted or received first
  - Mode 1: 10-bit full-duplex asynchronous communication, consisting of 1 start bit, 8 data bits, and 1 stop bit. The communication baud rate is variable
  - Mode 3: 11-bit full-duplex asynchronous communication, consisting of 1 start bit, 8 data bits, 1 programmable 9th bit and 1 stop bit. The communication baud rate is variable
- Interrupts will be generated and corresponding flags TXIF and RXIF will be set when transmission and reception are complete. Interrupt flags need to be cleared by software
- UART0 and UART1 can generate DMA requests
- UART2 and UART3 cannot generate DMA requests
- Independent baud rate generator
- Supports waking up from STOP Mode:
  - The falling edge of the START bit can wake up STOP Mode
  - Provides corresponding wake-up interrupt enable bit WKIE and wake-up interrupt flag WKIF

**Note: If users intend to perform UART wake-up in STOP mode, please refer to the <SC32F10 Series Special Function Firmware Library User Manual>.**

## 11.3 UART Interrupt

For UARTn, n=0~3, interrupts will be generated upon “wake-up” or “data transmission/reception completion”. Separate interrupt enable bits can be used to enhance flexibility.

| Interrupt Event | Interrupt Request Control Bit | Event Flag | Interrupt Enable Sub-Switch |
|---|---|---|---|
| Uart wake up from STOP mode | UARTn_IDE ->INTEN | WKIF | WKIE |
| Data transmission completion | | TXIF | TXIE |
| Data reception completion | | RXIF | RXIE |

## 11.4 UART Register

### 11.4.1 UART Control Register (UARTn_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| UARTn_CON<br>(n=0~3) | R/W | UART Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SPOS[1:0] | | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXEN | RXEN | - | PRESCALER | - | SM2 | SM1 | SM0 |

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>15~14</td><td>SPOS[1:0]<br>(Only valid when UARTn_CON, n=2)</td><td>This bit is only valid when UARTn_CON, n=2:<br>● UART2 Port Mapping Control Bit@UART2_CON<br>
<table>
<tr><th>Port<br>SPOS value</th><th>RX2</th><th>TX2</th></tr>
<tr><td>SPOS[1:0]=00</td><td>PB0</td><td>PA15</td></tr>
<tr><td>SPOS[1:0]=01</td><td>PA11</td><td>PA10</td></tr>
</table></td></tr>
<tr><td>7</td><td>TXEN</td><td>UART Transmission Enable Control Bit<br>0: Disallow data transmission, and the TXD signal no longer affects the state of the associated pin. If the user program restricts the sending function and only utilizes reception, other functions multiplexed with the TX pin will not be affected<br>1: Allow data transmission, and the pin associated with TXD switches to the TXD signal mode</td></tr>
<tr><td>6</td><td>RXEN</td><td>UART Reception Enable Control Bit<br>0: Disallow data reception<br>1: Allow data reception</td></tr>
<tr><td>4</td><td>PRESCALER</td><td>Baud Rate Multiplier Setting Bit<br>This bit has different definitions in different modes of UART:<br>● When SM0~1=01(UART mode 1) or SM0~1=11(UART mode 3):<br>■ 0: Serial port runs at 1/1 frequency of the system clock<br>■ 1: Serial port runs at 1/16 frequency of the system clock<br>● When SM0~1=00(UART mode 0):</td></tr>
</table>

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | ■ 0: Serial port runs at 1/12 frequency of the system clock<br>■ 1: Serial port runs at 1/4 frequency of the system clock |
| 2 | SM2 | RB8 Set Interrupt Enable Bit<br>This bit is only valid in mode 3<br>0: Set RI interrupt request upon receiving each complete data frame<br>1: Set RI interrupt request only when RB8=1 upon receiving a complete data frame |
| 1~0 | SM[1:0] | UART Communication Mode Control Bits<br>00: Mode 0, 8-bit half-duplex synchronous communication mode. Serial data is transmitted and received on the RX pin. The TX pin is used as the transmit shift clock. Each frame transmits and receives 8 bits, with the low bit first. Enabling the RXEN bit in this mode will cause the UART to generate a complete frame clock, and set RXIF to 1<br>01: Mode 1, 10-bit full-duplex asynchronous communication, consisting of 1 start bit, 8 data bits, and 1 stop bit. Communication baud rate is variable<br>10: Reserved<br>11: Mode 3, 11-bit full-duplex asynchronous communication, consisting of 1 start bit, 8 data bits, a programmable 9th bit, and 1 stop bit. Communication baud rate is variable |
| 31~16<br>13~8<br>5,3 | - | Reserved |

### 11.4.2 UART Flag Register (UARTn_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| UARTn_STS<br>(n=0~3) | R/W | UART Flag Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 4 | WKIF | UART Wake Up Flag<br>This bit will be set to 1 after UART wake up from STOP mode, and an interrupt will be generated if WKIE=1.<br>This bit is cleared by writing to 1 through software. |
| 1 | TXIF | Transmission Interrupt Flag<br>This bit will be set to 1 upon data transmission complete, and an interrupt will be generated if TXIE=1.<br>This bit is cleared by writing to 1 through software.<br>**Note: In DMA mode, after DMA writes to the transmit buffer, this bit is cleared by the DMA module, users do not need to clear it by software.** |
| 0 | RXIF | Reception Interrupt Flag<br>This bit will be set to 1 upon data reception complete, and an interrupt will be generated if RXIE=1.<br>This bit is cleared by writing to 1 through software.<br>**Note: In DMA mode, after DMA writes to the receive buffer, this bit is cleared by the DMA module, users do not need to clear it by software.** |
| 31~5<br>3~2 | - | Reserved |

**Note: If users intend to perform UART wake-up in STOP mode, please refer to the <SC32F10 Series Special Function Firmware Library User Manual>.**

### 11.4.3 UART Baud Configuration Register (UARTn_BAUD)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| UARTn_BAUD<br>(n=0~3) | R/W | UART Baud Configuration Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | BAUD[15:0] | UART Baud Configuration Bit<br>After writing to BAUD[15:0], the UART baud rate will be configured according to the following formula:<br>$BaudRate=f_{UART} / BAUD[15:0]$ |

| | | |
|---|---|---|
| | | $f_{UART}$ is the final frequency of the UART clock source after prescaling, as described in the PRESCALER bit description.<br>**Note:BAUD[15:0] must be greater than 0x0010。** |
| 31~16 | - | Reserved |

### 11.4.4 UART Data Register (UARTn_DATA)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| UARTn_DATA (n=0~3) | R/W | UART Data Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | SBUF8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SBUF[7:0] | | | | | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 8 | SBUF8 | The 9th bit of UART transmission/reception<br>This bit is only valid in mode 3 |
| 7~0 | SBUF[7:0] | UART Data Buffer<br>Read operation: Returns the content of the receive buffer<br>Write operation: The data in SBUF will be sent to the transmit shift register, initiating the transmission process. |
| 31~9 | - | Reserved |

### 11.4.5 UART Interrupt Enable And DMA Control Register (UARTn_IDE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| UARTn_IDE (n=0~3) | R/W | UART Interrupt Enable And DMA Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDMAEN | RXDMAEN | - | WKIE | - | TXIE | RXIE | INTEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TXDMAEN | DMA Transmission Channel Enable Bit<br>0: Disable DMA transmission function<br>1: Enable DMA transmission function<br>The set of TXIF can trigger DMA channel sending request after enabling this bit.<br>**Note:**<br>**1. UART0 and UART1 can generate DMA request**<br>**2. UART2 and UART3 cannot generate DMA request** |
| 6 | RXDMAEN | DMA Reception Channel Enable Bit<br>0: Disable DMA reception function<br>1: Enable DMA reception function<br>The set of RXIF can trigger DMA channel receving request after enabling this bit.<br>**Note:**<br>**1. UART0 and UART1 can generate DMA request**<br>**2. UART2 and UART3 cannot generate DMA request** |
| 4 | WKIE | UART Wake Up Interrupt Enable Bit<br>0: An interrupt will not be generated after WKIF is set<br>1: An interrupt will be generated after WKIF is set |
| 2 | TXIE | UART Transmission Interrupt Enable Bit<br>0: An interrupt will not be generated after TXIF is set<br>1: An interrupt will be generated after TXIF is set |
| 1 | RXIE | UART Receving Interrupt Enable Bit<br>0: An interrupt will not be generated after RXIF is set<br>1: An interrupt will be generated after RXIF is set |
| 0 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 31~8<br>5,3 | - | Reserved |

### 11.4.6 UART Register Mapping

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| UART0 Base Address:0x4002_0020 | | | | | |
| UART0_CON | 0x00 | R/W | UART0 Control Register | 0x0000_0000 | - |
| UART0_STS | 0x04 | R/W | UART0 Flag Register | 0x0000_0000 | - |
| UART0_BAUD | 0x08 | R/W | UART0 Baud Configuration Register | 0x0000_0000 | - |

| | | | | | |
|---|---|---|---|---|---|
| UART0_DATA | 0x0C | R/W | UART0 Data Register | 0x0000_0000 | Do not support byte/half word access |
| UART0_IDE | 0x10 | R/W | UART0 Interrupt Enable and DMA Control Register | 0x0000_0000 | - |

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| UART1 Base Address:0x4002_0080 | | | | | |
| UART1_CON | 0x00 | R/W | UART1 Control Register | 0x0000_0000 | - |
| UART1_STS | 0x04 | R/W | UART1 Flag Register | 0x0000_0000 | - |
| UART1_BAUD | 0x08 | R/W | UART1 Baud Configuration Register | 0x0000_0000 | - |
| UART1_DATA | 0x0C | R/W | UART1 Data Register | 0x0000_0000 | Do not support byte/half word access |
| UART1_IDE | 0x10 | R/W | UART1 Interrupt Enable and DMA Control Register | 0x0000_0000 | - |

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| UART2 Base Address:0x4002_1020 | | | | | |
| UART2_CON | 0x00 | R/W | UART2 Control Register | 0x0000_0000 | - |
| UART2_STS | 0x04 | R/W | UART2 Flag Register | 0x0000_0000 | - |
| UART2_BAUD | 0x08 | R/W | UART2 Baud Configuration Register | 0x0000_0000 | - |
| UART2_DATA | 0x0C | R/W | UART2 Data Register | 0x0000_0000 | Do not support byte/half word access |
| UART2_IDE | 0x10 | R/W | UART2 Interrupt Enable and DMA Control Register | 0x0000_0000 | - |

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| UART3 Base Address:0x4002_2020 | | | | | |
| UART3_CON | 0x00 | R/W | UART3 Control Register | 0x0000_0000 | - |
| UART3_STS | 0x04 | R/W | UART3 Flag Register | 0x0000_0000 | - |
| UART3_BAUD | 0x08 | R/W | UART3 Baud Configuration Register | 0x0000_0000 | - |

| | | | | | |
|---|---|---|---|---|---|
| UART3_DATA | 0x0C | R/W | UART3 Data Register | 0x0000_0000 | Do not support byte/half word access |
| UART3_IDE | 0x10 | R/W | UART3 Interrupt Enable and DMA Control Register | 0x0000_0000 | - |

# 12 SPI0~1

## 12.1 Clock Source

The SC32F10T/10G series SPI has only one clock source, which is derived from PCLK.

## 12.2 SPI0 Feature

- Supports 11-stage SPI clock pre-scaling, allowing users to set to lower frequencies
- Signal ports can be mapped to two additional sets of ports
- Features a 16-bit 8-level FIFO with independent transmission and reception
    - SPI0's FIFO function allows continuous writing of 8 or fewer 8-bit or 16-bit transmit data to the SPI send buffer (SPI0_DATA). During SPI transmission, the data written into the FIFO first is also sent first. When the data written by the user to the FIFO is sent, the FIFO empty flag TXEIF will be set; if the FIFO is full, the write conflict flag WCOL will be set, and the user cannot write data to the FIFO until the data in the FIFO is sent out and the FIFO is not full. The interrupt flag SPIF will be set only when all the data in the FIFO has been sent
    - Continuously read 8 or fewer 8-bit or 16-bit receive data from the SPI receive buffer (SPI0_DATA), with the first received data being the first to be read
    - FIFO data transfer half-interrupt and corresponding flags for timely reading/writing of data:
        - Provides an interrupt and corresponding flag TXHIF when there is less than half of the valid data in the transmit FIFO
        - Provides an interrupt and corresponding flag RXHIF when there is more than half of the data in the receive FIFO
    - Support recieve buffer overflow interrupt and corresponding flag to promptly notify exceptions
- Support DMA
    - Enable TXDMAEN, and the DMA request can be triggered after the transmit buffer empty flag TXEIF is set. After DMA writes to the transmit buffer, the TXEIF flag is automatically cleared
    - Enable RXDMAEN, and the DMA request can be triggered after the receive buffer not empty status flag RXNEIF is set. After DMA reads from the receive buffer, the RXEIF flag is automatically cleared

## 12.3 SPI1 Feature

- Supports 11-stage SPI clock pre-scaling, allowing users to set to lower frequencies
- Signal ports can be mapped to one additional sets of ports
- No FIFO
- Supports DMA: A request is uniformly set at the end of a frame

## 12.4 SPI0 and SPI1 Comparison

| Comparison<br>BIT | SPI0 | SPI1 |
|---|---|---|
| WCOL | When the send FIFO is full, attempting to write to the FIFO will fail, and WCOL will be set, indicating a buffer write conflict | When one frame is sending, attempting to write to the FIFO will fail, and WCOL will be set, indicating a buffer write conflict |
| SPIF | This position being set indicates the completion of receiving/sending one frame of data | This position being set indicates the completion of receiving/sending one frame of data |
| RXHIE | Interrupt enable bit for the valid data in the recieve FIFO is more than half | None |
| TXHIE | Interrupt enable bit for the valid data in the transmit FIFO is less than half | None |
| RXIE | Interrupt enable bit for the receive FIFO full | None |
| TBIE | Interrupt enable bit for the transmit FIFO empty | Interrupt enable bit for the transmit FIFO empty |
| RXNEIE | Interrupt enable bit for the receive FIFO not empty | None |
| RXHIF | Set when the valid data in the receive FIFO is more than half | None |
| TXHIF | Set when the valid data in the receive FIFO is less than half | None |
| RXFIF | Set when the receive FIFO is full | None |
| TXEIF | Set when the receive FIFO is empty | Set when the receive FIFO is empty |
| RXNEIF | Recieve FIFO not empty flag | None |
| DMA | Triggering DMA requests through the TXEIF flag and the RXNEIF flag | A request is uniformly set at the end of a frame |

## 12.5 Signal Description

**Master Output Slave Input (MOSI):**

This signal connects the master device to a slave device. Data is transmitted serially from the master device to the slave device through MOSI, which is an output from the master and an input to the slave.

**Master Input Slave Output (MISO):**

This signal connects the slave device to the master device. Data is transmitted serially from the slave device to the master device through MISO, which is an output from the slave and an input to the master. When the SPI is configured as a slave and not selected, the MISO pin of the slave device will be in a high-impedance state.

**SPI Serial Clock (SCK):**

The SCK signal is used to control the synchronous movement of input and output data on the MOSI and MISO lines. One byte is transferred on the line every 8 clock cycles. If a slave device is not selected, the SCK signal will be ignored by that slave device.

## 12.6 Working Mode

SPI can be configured in either master or slave mode. The configuration and initialization of the SPI module are accomplished by setting SPI control registers (SPIn_CON, n=0~1) and SPI interrupt enable and DMA control register (SPIn_IDE, n=0~1). Once configured, data transmission will be achieved by setting the SPI data register (SPIn_DATA, n=0~1) during SPI communication.

During SPI communication, data is serially shifted in and out in a synchronous manner. The serial clock line (SCK) synchronizes the movement and sampling of data on the two serial data lines (MOSI and MISO). If a slave device is not selected, it will not participate in activities on the SPI bus.

When the SPI master device transmits data to the slave device through the MOSI line, the slave device responds by sending data to the master device through the MISO line. This achieves synchronous full-duplex transmission of data at the same clock. The transmit shift register and receive shift register share the same special function register address. Writing to the SPI data register (SPD) will write to the transmit shift register, and reading from SPD will retrieve data from the receive shift register.

Some devices with SPI interfaces may have an SS pin (slave select pin, active low). When communicating with SC32F10T/10G through the SPI, the connection of the SS pins of other devices on the SPI bus should be configured according to the different communication modes. The table below outlines the connection methods for the SS pins of other devices on the SPI bus in different communication modes for SC32F10T/10G:

| SC32F10T/10G SPI | Other Devices On SPI Bus | Mode | Slave SS |
|---|---|---|---|
| Master | Slave | One Master One Slave | Pull low |
| | | One Master Multiple Slave | SC32F10T/10G has multiple I/O pins, each connected to the SS pin of different slave devices. Before data transmission, the SS pin of the specific slave device must be pulled low. |
| Slave | Master | One Master One Slave | Pull high |

**Master Mode**

- **Mode Active:**

The SPI master device controls the initiation of all data transfers on the SPI bus. When SPIn_CON.MSTR=1 (n=0~1), the SPI operates in master mode, and only one master device can initiate the transfer.

- **Transmission:**

In SPI master mode, users can perform the following operation on SPD: write a byte of data to SPD[7:0] in 8-bit mode or write a 16-bit data to SPD[15:0], then the data will be written to the transmit shift buffer. If there is already data in the transmit shift register, the master SPI will generate a WCOL signal to indicate that the write is too fast. However, the data in the transmit shift register will not be affected, and the transmission will not be interrupted. Additionally, if the transmit shift register is not empty, the master device immediately serially shifts out the data from the transmit shift register to MOSI at the SPI clock frequency on SCK. When the transfer is complete, the SPIF bit in the SPI status register SPIn_STS (n=0~1)will be set to 1. If SPI interrupts are enabled, an interrupt will also be generated when SPIF is set to 1.

- **Reception:**

When the master device sends data to the slave device via MOSI, the corresponding data will be simultaneously transmitted by the slave device via MISO to the receive shift register of the master device, achieving full-duplex operation. Therefore, when the SPIF flag is set to 1, it indicates that the transmission is complete and the reception of data is also complete. The received data from the slave device is stored in the receive shift register of the master device according to the MSB or LSB priority transmission direction. When a byte of data is fully moved into the receive register, processor can obtain the data by reading SPD.

**Slave Mode**

- **Mode Active:**

When SPIn_CON.MSTR (n=0~1) is cleared, the SPI operates in slave mode.

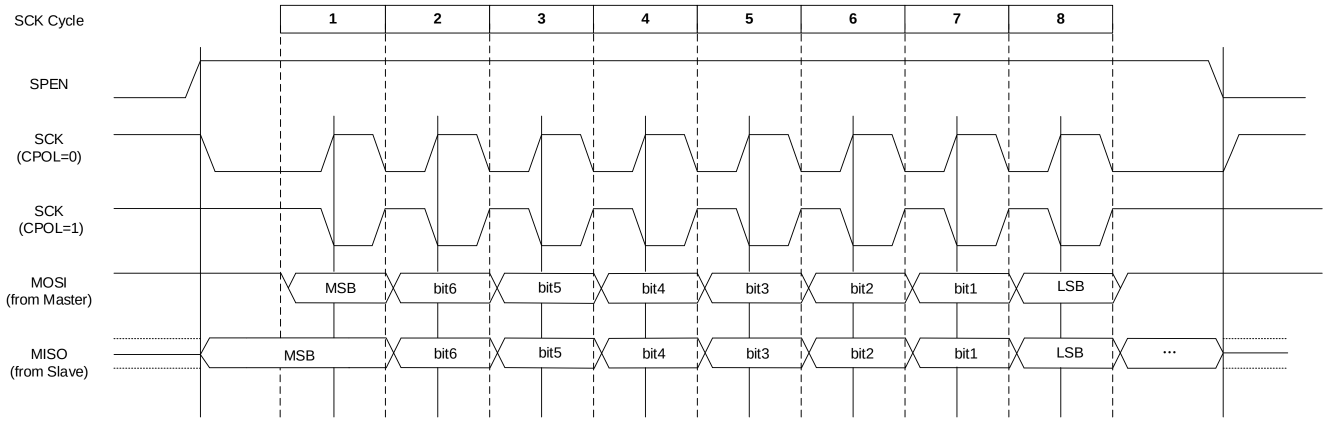
- **Transmission And Reception:**

In slave mode, data will be input through MOSI and output through MISO according to the SCK signal controlled by the master device. A bit counter records the number of SCK edges. When the receive shift register moves in 8 bits of data (one byte) while the transmit shift register moves out 8 bits of data (one byte), the SPIF flag will be set to 1. The data can be obtained by reading the SPD register. If SPI interrupts are enabled, an interrupt will be generated when SPIF is set to 1. At this time, the receive shift register retains the original data and SPIF is set to 1, indicating that the SPI slave device will not receive any data until SPIF is cleared. The SPI slave device must write the data to be transmitted into the transmit shift register before the master device starts a new transmission. If no data is written before starting transmission, the slave device will send the "0x00" to the master device. If a write to SPD occurs during the transmission process, the WCOL flag of the SPI slave device will be set to 1, indicating a write SPD conflict. However, the data in the shift register is not affected, and the transmission will not be interrupted.

## 12.7 Transmission format

Setting the CPOL (Clock Polarity) and CPHA (Clock Phase) bits in the SPI control register SPIn_CON (n=0~1) by software, users can choose from four combinations of SPI clock polarity and phase. CPOL determines the clock polarity, indicating the electrical level of idle state. It has minimal impact on the SPI transfer format. CPHA defines the clock phase, determining the clock edge at which data is sampled and shifted. In a communication link between a master and a slave device, the settings of clock polarity and phase should be consistent.

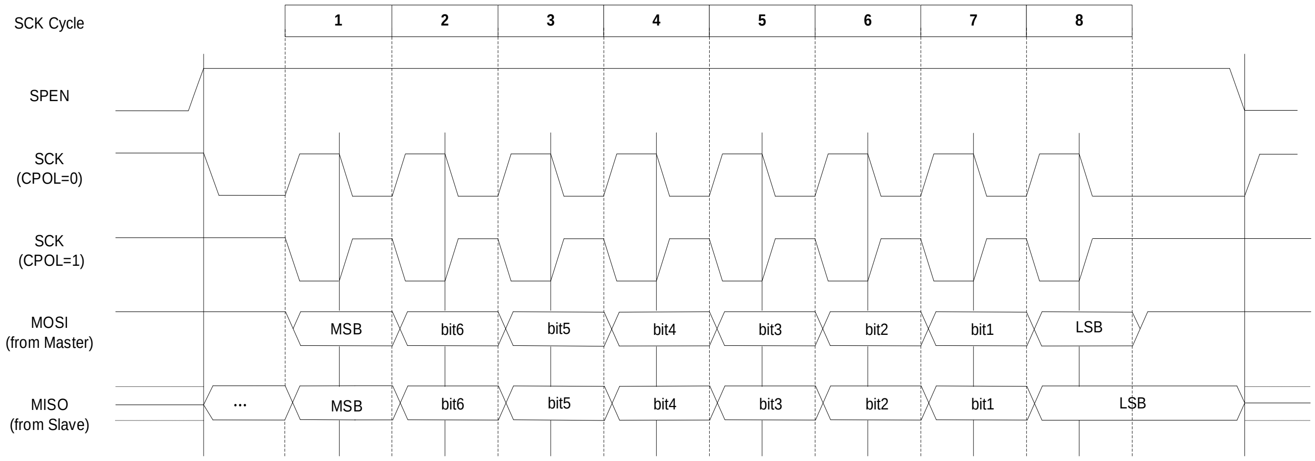
When CPHA = 0, data will be captured at the first edge of SCK. and the data must be prepared by the slave device before the first edge of SCK.



CPHA = 0 Data Transfer Diagram

When CPHA = 1, the master device outputs data to MOSI on the first edge of SCK, and the slave device treats the first edge of SCK as the start of the transmission. The second edge of SCK is used to capture the data, so users must complete the write operation to SPD register within the first two edges of SCK.

This data transmission format is a preferred mode for communication between one master device and one slave device in the SPI protocol.



CPHA = 1 Data Transfer Diagram

## 12.8 Error Detection

Writing to SPD during the transmission of a data sequence will lead to a write collision, resulting in the setting of the WCOL bit. This will not trigger an interrupt, and the transmission will not stop, and the WCOL bit needs to be cleared by software.

## 12.9 SPI Interrupt

As for SPI0, interrupts will be generated when “transmission complete”, “FIFO half transmit”, or “transmit buffer empty”. Separate interrupt enable bits can be used to enhance flexibility.

| Interrupt Event | Event Flag | Interrupt Request Control Bit | Sub-Event Flag | Interrupt Enable Sub-Switch |
|---|---|---|---|---|
| The valid data in the recieve FIFO is more than half | SPIF | SPI0_IDE ->INTEN | RXHIF | RXHIE |
| The valid data in the transmit FIFO is less than half | | | TXHIF | TXHIE |
| The receive FIFO is full | | | RXFIF | RXIE |
| The transmit FIFO is empty | | | TXEIF | TBIE |
| The receive FIFO is not full | | | RXNEIF | RXNEIE |

As for SPI1, interrupts will be generated when “transmission complete”, or “transmit buffer empty”. Separate interrupt enable bits can be used to enhance flexibility.

| Interrupt Event | Event Flag | Interrupt Request Control Bit | Sub-Event Flag | Interrupt Enable Sub-Switch |
|---|---|---|---|---|
| One frame transmit/receive complete | SPIF | SPI1_IDE ->INTEN | \ | \ |
| The transmit buffer is empty | | | TXEIF | TBIE |

## 12.10 SPI0 Register

### 12.10.1 SPI0 Control Register (SPI0_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI0_CON | R/W | SPI0 Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SPOS[1:0] | | - | - | SPR[3:0] | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPEN | - | - | CPOL | CPHA | DORD | SPMD | MSTR |

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>15~14</td><td>SPOS[1:0]</td><td>SPI Port Mapping Control Bit
<table>
<tr><th>Port<br>SPOS Value</th><th>MISO0</th><th>MOSI0</th><th>SCK0</th></tr>
<tr><td>SPOS[1:0]=00</td><td>PA15</td><td>PB0</td><td>PB1</td></tr>
<tr><td>SPOS[1:0]=01</td><td>PA3</td><td>PA2</td><td>PA1</td></tr>
<tr><td>SPOS[1:0]=10</td><td>PA9</td><td>PA8</td><td>PA7</td></tr>
</table></td></tr>
<tr><td>11~8</td><td>SPR[3:0]</td><td>SPI Clock Presclar Control Bit<br>0000: $f_{PCLK0}$<br>0001: $f_{PCLK0}$ /2<br>0010: $f_{PCLK0}$ /4<br>0011: $f_{PCLK0}$ /8<br>0100: $f_{PCLK0}$ /16<br>0101: $f_{PCLK0}$ /32<br>0110: $f_{PCLK0}$ /64<br>0111: $f_{PCLK0}$ /128<br>1000: $f_{PCLK0}$ /256<br>1001: $f_{PCLK0}$ /512<br>1010: $f_{PCLK0}$ /1024<br>Others: $f_{PCLK0}$ /1024</td></tr>
<tr><td>7</td><td>SPEN</td><td>SPI Enable Control Bit<br>0: Disable SPI0<br>1: Enable SPI0</td></tr>
<tr><td>4</td><td>CPOL</td><td>SPI Clock Polarity Control Bit<br>0: SCK is at low level in the idle state<br>1: SCK is at high level in the idle state</td></tr>
<tr><td>3</td><td>CPHA</td><td>SPI Clock Phase Control Bit<br>0: Capture data at the first edge of SCK<br>1: Capture data at the second edge of SCK</td></tr>
<tr><td>2</td><td>DORD</td><td>SPI Transmission Direction Selection Bit<br>0: MSB sending priority<br>1: LSB sending priority</td></tr>
<tr><td>1</td><td>SPMD</td><td>SPI Transmission Mode Selection Bit<br>0: 8-bit mode<br>1: 16-bit mode</td></tr>
<tr><td>0</td><td>MSTR</td><td>SPI Master/Slave Selection Bit<br>0: SPI0 is slave device<br>1: SPI0 is master device</td></tr>
<tr><td>31~16<br>13~12<br>6~5</td><td>-</td><td>Reserved</td></tr>
</table>

### 12.10.2 SPI0 Flag Register (SPI0_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI0_STS | R/W | SPI0 Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WCOL | - | TXHIF | RXHIF | RXFIF | TXEIF | RXNEIF | SPIF |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | WCOL | Write Conflict Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether a write conflict has occured:<br>0: No writing conflict detected<br>1: Writing conflict detected |
| 5 | TXHIF | Valid Data In Transmit FIFO Is Less Than Half Flag<br>This bit is read only, and can be set or cleared by hardware, indicating the current status of the transmit FIFO:<br>0: The valid data in the transmit FIFO is not less than half<br>1: The valid data in the transmit FIFO is less than half, an interrupt will be generated if TXHIE=1 |
| 4 | RXHIF | Valid Data In Recieve FIFO Is More Than Half Flag<br>This bit is read only, and can be set or cleared by hardware, indicating the current status of the receive FIFO:<br>0: The valid data in the recieve FIFO is not more than half flag<br>1: The valid data in the recieve FIFO is more than half flag, an interrupt will be generated if RXHIE=1 |
| 3 | RXFIF | Receive FIFO Is Full Fag<br>This bit is read only, and can be set or cleared by hardware, indicating whether current recieve FIFO is full:<br>0: Receive FIFO is not full<br>1: Receive FIFO is full |
| 2 | TXEIF | Transmit FIFO Is Empty Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether current transmit FIFO is empty:<br>0: Transmit FIFO is not empty<br>1: Transmit FIFO is empty |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | **Note: In DMA mode, after DMA writes to the transmit buffer, this bit is cleared by the DMA module, and users do not need to clear it through software.** |
| 1 | RXNEIF | Receive FIFO Is Not Full Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether current receive FIFO is empty:<br>0: Receive FIFO is empty<br>1: Receive FIFO is not empty<br>**Note: In DMA mode, after DMA writes to the transmit buffer, this bit is cleared by the DMA module, and users do not need to clear it through software.** |
| 0 | SPIF | SPI Data Transmission Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether current SPI transmission is complete:<br>0: Data transmission is ongoing<br>1: Data transmission is complete |
| 31~8<br>6 | - | Reserved |

### 12.10.3 SPI0 Data Register (SPI0_DATA)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI0_DATA | R/W | SPI0 Data Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | SPD[15:0] | SPI Data Buffer<br>Read operation: Read the received data from the SPI0 receive FIFO<br>Write operation: write data to the SPI0 transmit FIFO |
| 31~16 | - | Reserved |

### 12.10.4 SPI0 Interrupt Enable And DMA Control Register (SPI0_IDE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI0_IDE | R/W | SPI0 Interrupt Enable And DMA Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXDMAEN | RXDMAEN | TXHIE | RXHIE | RXIE | TBIE | RXNEIE | INTEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TXDMAEN | DMA Transmit Channel Enable Bit<br>0: Disable DMA transmit function<br>1: Enable DMA transmit function<br>The set of TXEIF can trigger DMA channel transmit request after enabling this bit. |
| 6 | RXDMAEN | DMA Recieve Channel Enable Bit<br>0: Disable DMA receive function<br>1: Enable DMA receive function<br>The set of RXNEIF can trigger DMA channel receive request after enabling this bit. |
| 5 | TXHIE | Valid Data In Transmit FIFO Is Less Than Half Interrupt Enable Bit<br>0: An interrupt will not be generated when TXHIF is set<br>1: An interrupt will be generated when TXHIF is set |
| 4 | RXHIE | Valid Data In Recieve FIFO Is More Than Half Interrupt Enable Bit<br>0: An interrupt will not be generated when RXHIF is set<br>1: An interrupt will be generated when RXHIF is set |
| 3 | RXIE | Receive FIFO Is Full Interrupt Enable Bit<br>0: An interrupt will not be generated when RXFIF is set<br>1: An interrupt will be generated when RXFIF is set |
| 2 | TBIE | Transmit FIFO Is Empty Interrupt Enable Bit<br>0: An interrupt will be not generated when TXEIF is set<br>1: An interrupt will be generated when TXEIF is set |
| 1 | RXNEIE | Receive FIFO Is Not Full Interrupt Enable Bit<br>0: An interrupt will not be generated when RXNEIF is set<br>1: An interrupt will be generated when RXNEIF is set |
| 0 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~8 | - | Reserved |

### 12.10.5 SPI0 Register Mapping

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| SPI0 Base Address:0x4002_0040 | | | | | |
| SPI0_CON | 0x00 | R/W | SPI0 Control Register | 0x0000_0000 | - |
| SPI0_STS | 0x04 | R/W | SPI0 Flag Register | 0x0000_0000 | - |
| SPI0_DATA | 0x0C | R/W | SPI0 Data Register | 0x0000_0000 | Do not support byte/half word access |
| SPI0_IDE | 0x10 | R/W | SPI0 Interrupt Enable And DMA Control Register | 0x0000_0000 | - |

## 12.11 SPI1 Register

### 12.11.1 SPI1 Control Register (SPI1_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI1_CON | R/W | SPI1 Control Register | 0x0000_0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SPOS[1:0] | | - | - | SPR[3:0] | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPEN | - | - | CPOL | CPHA | DORD | SPMD | MSTR |

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>15~14</td><td>SPOS[1:0]</td><td>SPI Port Mapping Control Bit
<table>
<tr><th>Port<br>SPOS Value</th><th>MISO1</th><th>MOSI1</th><th>SCK1</th></tr>
<tr><td>SPOS[1:0]=00</td><td>PA14</td><td>PA13</td><td>PA12</td></tr>
<tr><td>SPOS[1:0]=01</td><td>PA4</td><td>PA5</td><td>PA6</td></tr>
</table></td></tr>
</table>

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 11~8 | SPR[3:0] | SPI Clock Presclar Control Bit<br>0000: $f_{PCLK1}$<br>0001: $f_{PCLK1}$ /2<br>0010: $f_{PCLK1}$ /4<br>0011: $f_{PCLK1}$ /8<br>0100: $f_{PCLK1}$ /16<br>0101: $f_{PCLK1}$ /32<br>0110: $f_{PCLK1}$ /64<br>0111: $f_{PCLK1}$ /128<br>1000: $f_{PCLK1}$ /256<br>1001: $f_{PCLK1}$ /512<br>1010: $f_{PCLK1}$ /1024<br>Others: $f_{PCLK1}$ /1024 |
| 7 | SPEN | SPI Enable Control Bit<br>0: Disable SPI1<br>1: Enable SPI1 |
| 4 | CPOL | SPI Clock Polarity Control Bit<br>0: SCK is at low level in the idle state<br>1: SCK is at high level in the idle state |
| 3 | CPHA | SPI Clock Phase Control Bit<br>0: Capture data at the first edge of SCK<br>1: Capture data at the second edge of SCK |
| 2 | DORD | SPI Transmission Direction Selection Bit<br>0: MSB sending priority<br>1: LSB sending priority |
| 1 | SPMD | SPI Transmission Mode Selection Bit<br>0: 8-bit mode<br>1: 16-bit mode |
| 0 | MSTR | SPI Master/Slave Selection Bit<br>0: SPI1 is slave device<br>1: SPI1 is master device |
| 31~16<br>13~12<br>6~5 | - | Reserved |

### 12.11.2 SPI1 Flag Register (SPI1_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI1_STS | R/W | SPI1 Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WCOL | - | - | - | - | TXEIF | - | SPIF |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | WCOL | Write Conflict Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether a write conflict has occured:<br>0: No writing conflict detected<br>1: Writing conflict detected |
| 2 | TXEIF | Transmit Buffer Is Empty Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether current transmit Buffer is empty:<br>0: Transmit Buffer is not empty<br>1: Transmit Buffer is empty<br>**Note: In DMA mode, after DMA writes to the transmit buffer, this bit is cleared by the DMA module, and users do not need to clear it through software.** |
| 0 | SPIF | SPI Data Transmission Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software, indicating whether current SPI transmission is complete:<br>0: Data transmission is ongoing<br>1: Data transmission is complete |
| 31~8<br>6~3,1 | - | Reserved |

### 12.11.3 SPI1 Data Register (SPI1_DATA)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI1_DATA | R/W | SPI1 Data Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | SPD[15:0] | SPI Data Buffer<br>Read operation: Read the received data from the SPI1 receive buffer<br>Write operation: write data to the SPI1 transmit buffer |
| 31~16 | - | Reserved |

### 12.11.4 SPI1 Interrupt Enable And DMA Control Register (SPI1_IDE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SPI1_IDE | R/W | SPI1 Interrupt Enable And DMA Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TXDMAEN | DMA Transmit Channel Enable Bit<br>0: Disable DMA transmit function<br>1: Enable DMA transmit function<br>The set of TXEIF can trigger DMA channel transmit request after enabling this bit. |
| 6 | RXDMAEN | DMA Recieve Channel Enable Bit<br>0: Disable DMA receive function<br>1: Enable DMA receive function<br>The set of SPIF can trigger DMA channel receive request after enabling this bit. |
| 2 | TBIE | Transmit Buffer Is Empty Interrupt Enable Bit<br>0: An interrupt will be not generated when TXEIF is set<br>1: An interrupt will be generated when TXEIF is set |
| 0 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 31~8<br>5~3,1 | - | Reserved |

### 12.11.5 SPI1 Register Mapping

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| SPI1 Base Address:0x4002_1040 | | | | | |
| SPI1_CON | 0x00 | R/W | SPI1 Control Register | 0x0000_0000 | - |
| SPI1_STS | 0x04 | R/W | SPI1 Flag Register | 0x0000_0000 | - |
| SPI1_DATA | 0x0C | R/W | SPI1 Data Register | 0x0000_0000 | Do not support byte/half word access |
| SPI1_IDE | 0x10 | R/W | SPI1 Interrupt Enable And DMA Control Register | 0x0000_0000 | - |

# 13 TWI0~1

## 13.1 Clock Source

The SC32F10T/10G series TWI has only one clock source, which is derived from PCLK

## 13.2 Feature

- Support 2 sets of TWI interfaces: TWI0 and TWI1
- Support TWI signal mapping
    - TWI0 can be mapped to three other groups of IO
    - TWI1 can be mapped to two other groups of IO
- Support master/slave mode
- Bidirectional data transmission between master and slave
- Communication speed can reach up to 1 Mbps
- Optional clock extension
- Support DMA
    - TWI0 can generate DMA requests
    - TWI1 cannot generate DMA requests

## 13.3 TWI Signal Description

On the TWI bus, data is synchronously transmitted between the master and slave devices using the clock line (SCL) and the data line (SDA). Each data byte has a length of 8 bits, and one data bit is transferred with each SCL clock pulse. The data is transmitted starting from the most significant bit (MSB), and after each byte, an acknowledgment bit follows. Each bit is sampled when SCL is high. Therefore, the SDA line may change when SCL is low, but it must remain stable when SCL is high. When SCL is high, any transition on the SDA line is considered a command (START or STOP)

**TWI Clock Signal Line(SCL):**

The clock signal is generated by the master and is connected to all the slaves. It transmits one byte of data every 9 clock cycles. The first 8 cycles are used for data transmission, and the last one is used as the acknowledgment clock for receiver. It should be pulled up by the pull-up resistor on the SDA line when idle.

**TWI Data Signal Line(SDA):**

SDA is a bidirectional signal line and should be pulled up by the pull-up resistor on the SDA line when idle.

## 13.4 Slave Operating Mode

- **Mode Initiation:**

When TWEN = 1 and the slave receives the start signal sent by the master, the mode will be initiated.

The slave from idle state (STATE[2:0] = 000) change to first frame address reception state (STATE[2:0] = 001), waiting for the master's first frame of data. The first frame of data is sent by the master and includes 7 address bits and 1 read/write bit. All slaves on the TWI bus can receive the master's first frame of data. The SDA signal line will be released after transmiting the first frame of data. If the address sent by the master matches the value in the slave's own address register, the selected slave will be selected and will check the 8th bit on the bus, which is the data read/write bit (1 for read command; 0 for write command). The selected slave then holds the SDA signal line, gives a low-level acknowledgment signal to the master on the 9th clock cycle, and then releases the bus. After being selected, the slave will enter different states depending on the read/write bit:

- **Non-general call address response, slave reception mode:**

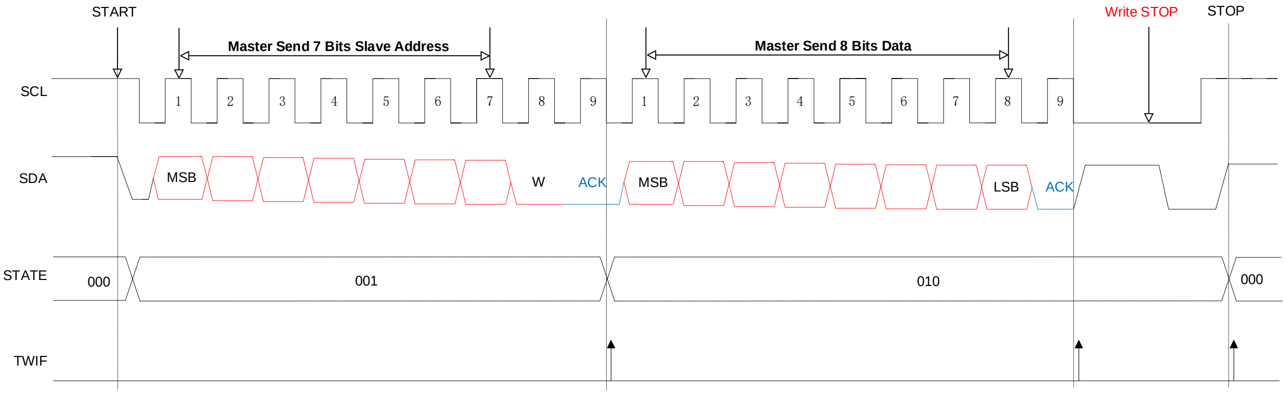
If the read/write bit received in the first frame is a write (0), the slave will enter the slave receive state (STATE[2:0] = 010) to wait for the master to transmit data. The bus will be released every 8 bits the master transmits, and the slave awaits the 9th clock cycle for the acknowledgment signal.

If the slave's acknowledgment signal is low, the master will have the following three actions:

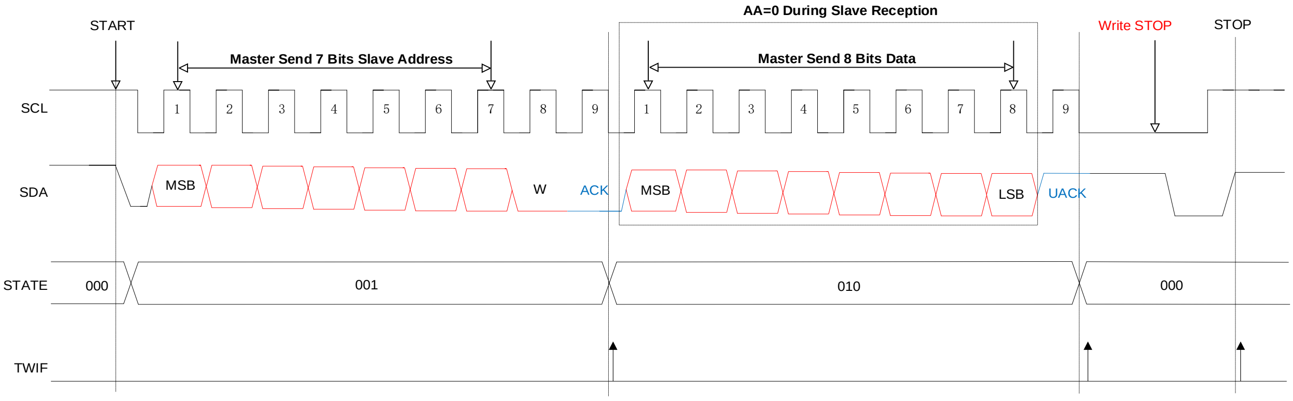
① Continue transmiting data

② Resend the start signal, at which point the slave re-enters the reception of the first address frame state (STATE[2:0] = 001).

③ Transmit a stop signal, indicating the end of this transmission. The slave will return to the idle state and wait for the master's next start signal



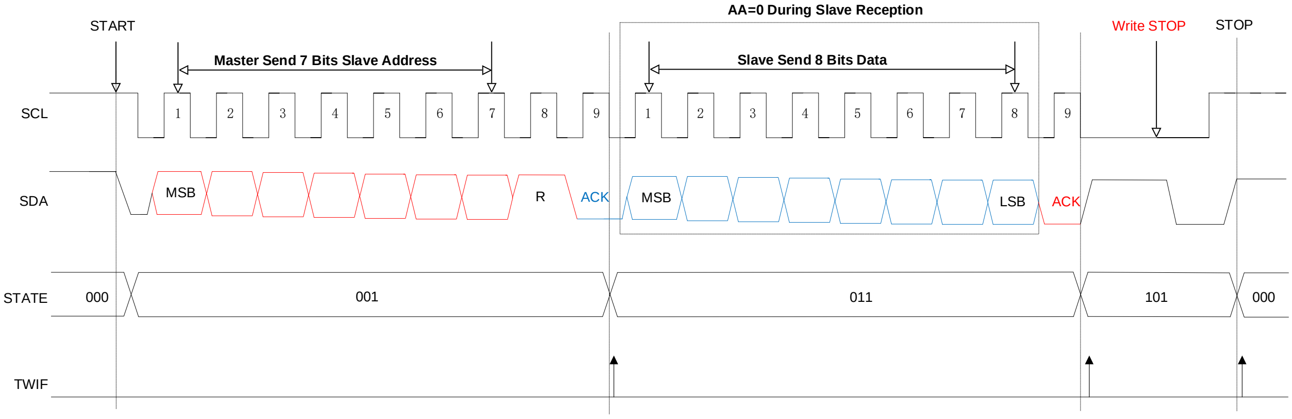
If the slave responds with a high level (during the reception process, the value of AA in the slave's register will be rewritten to 0), it indicates that after the current byte transmission is complete, the slave will actively terminate this transmission, returning to the idle state (STATE[2:0] = 000), and will no longer receive data sent by the master



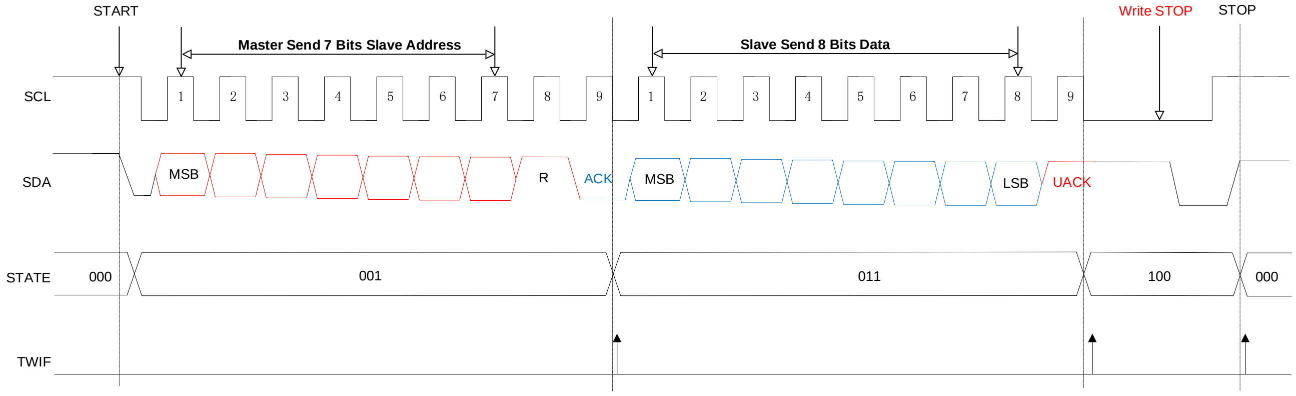
- **Non-general call address response, slave transmission mode:**

If the read/write bit received in the first frame is read, the slave will occupy the bus and transmit data to the master. After transmiting each 8 bits of data, the slave will release the bus and wait for the master's acknowledgment:

If the master responds with a low level, the slave will continue to transmit data. During the transmission, if the value of AA in the slave register is modified to 0, the slave will actively terminate the transmission and release the bus after completing the current byte transmission. It then waits for the master's stop signal or a restart signal (STATE[2:0] = 101).



If the master responds with a high level, the slave’s STATE[2:0] = 100. It then waits for the master's stop signal or a restart signal.



- **General call address response:**

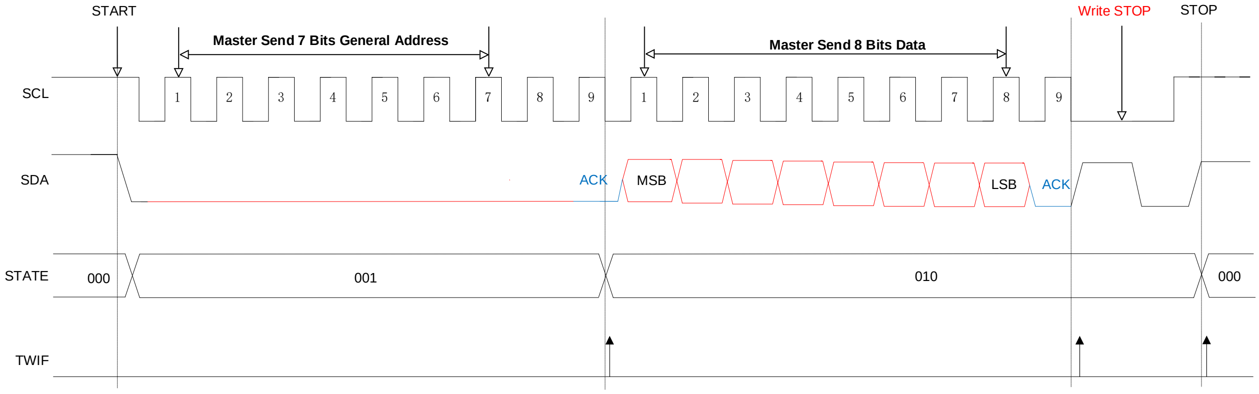
When GC is set to 1, general call address will be allowed. the slave enters the state of receiving the first address frame (STATE[2:0] = 001). If the address bits of first frame is 0x00, all slave will respond to the master. The master transmits a read/write bit, which must be set to write. All slaves then enter the state of receiving data (STATE[2:0] = 010). The master will release the SDA line every 8 data transmissions and read the status on the SDA line:

If there is a slave acknowledgment, the master will have the following three actions:

① Continue transmiting data

② Restart the communication

③ Transmit a stop signal, indicating the end of this transmission



If there is no acknowledgment from any slave, SDA line will be in idle state

**Note: In one master multiple slaves mode using a general call address, the read/write bit sent by the master must not be set to read. Otherwise, all devices on the bus will respond, except the one transmiting data.**

## 13.5 Slave Mode Operation Steps

1. Configure TWI control register (TWIn_CON.TWEN = 1) to enable TWI
2. Configure TWI control register (TWIn_CON)
3. Configure TWI address register (TWIn_ADD)
4. If slave recieve data, wait for the interrupt flag TWIF in the TWI status register (TWIn_STS) to be set to 1. The TWIF flag will be set each time the slave receives 8 bits of data, and TWIF need to be cleared manually.
5. If slave transmit data, write the data to be transmitted into the TWI data register (TWIDAT), than TWI will automatically transmit the data, and the TWIF flag will be set for every 8 bits transmitted.

## 13.6 Master Operating Mode

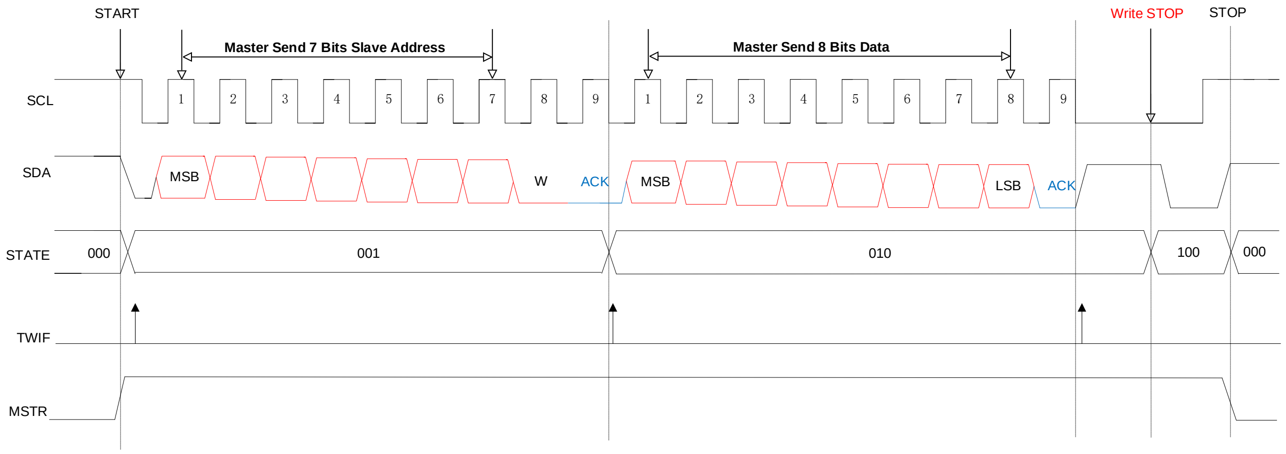
- **Mode Initiation:**

When the TWI interface transmits a start condition to the bus, it automatically switches to master mode, and the hardware will set the MSTR bit to 1. The master status bits STATE[2:0] change from 000 to 001, and simultaneously, the interrupt condition TWIF is set to 1.

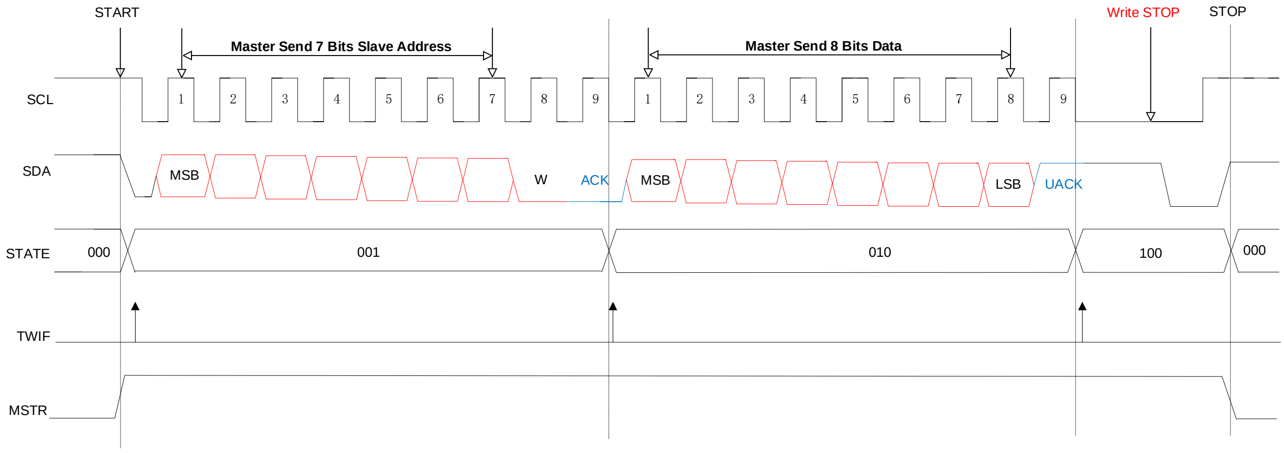
- **TWI Master Transmission Mode:**

In the master transmission mode, the first frame of data sent by the master includes 7 bits of address (the address of the selected slave) and 1 bit of read/write indicator (0 for write command). All slaves on the TWI bus will receive this first frame of data from the master. After transmiting the first frame, master will release the SDA signal line. The selected slave, upon receiving the first frame, responds to the master with an acknowledgment signal on the 9th clock cycle of the SCL. Afterward, the slave releases the bus and enters the slave receive state to await the reception of data from the master. The master will release the bus after transmiting each 8 bits, then wait for the acknowledgment signal from the slave on the 9th cycle.

If the slave responds with a low level, the master can continue transmiting data. It can also resend the start signal:



If the slave responds with a high level, it indicates that the current byte transmission is complete, and the slave will actively terminate the current transmission. The slave will no longer receive data from the master, and the master's STATE[2:0] will change from 010 to 100:

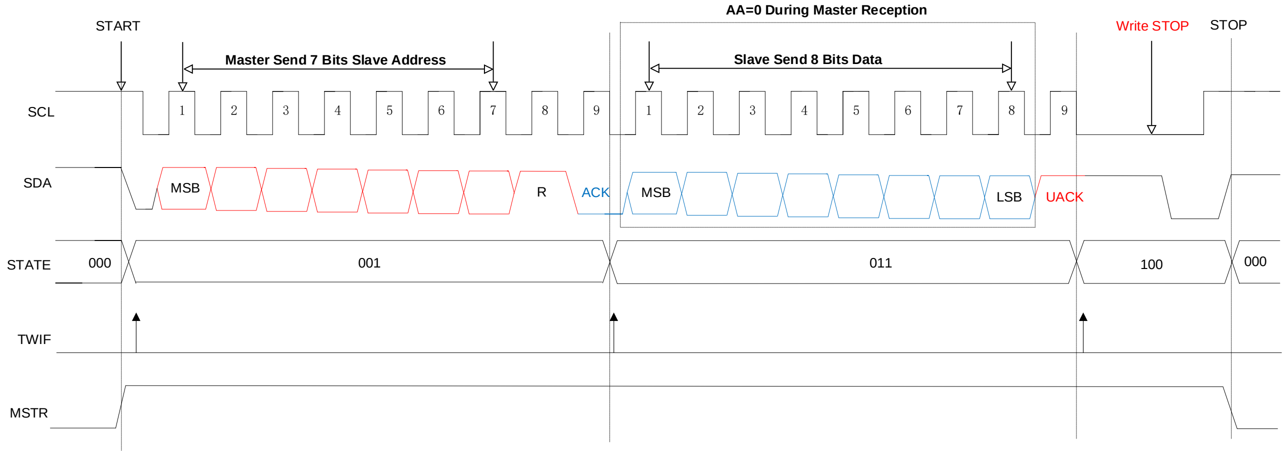


- **TWI Master Reception Mode:**

In master transmission mode, the first frame of data sent by the master includes 7 address bits (the address of the selected slave) and 1 read/write bit (1 for read command). All slaves on the TWI bus will receive this first frame of data from the master. After transmiting the first frame of data, the master will release the SDA signal line. The selected slave will respond to the master with an acknowledgment signal on the 9th clock cycle of SCL. Subsequently, the slave will occupy the bus and transmit data to the master. After transmiting 8 bits of data, the slave will release the bus and wait for the master's acknowledgment. Upon receiving a successful acknowledgment (ACK) from the slave after matching address, the master will begin to receive data from the slave (STATE=011):

1. If the master acknowledgment bit is enabled (AA=1), the master will respond with an acknowledgment signal (ACK) after receiving each byte of data, and TWIF will be set.
2. Before receiving the last byte of data, if the acknowledgment enable bit is disabled (AA=0), the master will respond with a unacknowledge (UACK) after receiving the last byte of data. Then, the master can transmit a stop signal.

In master receiving mode, the method for actively releasing the bus is as follows:



## 13.7 Master Mode Operation Steps

1. Configure TWI control register (TWIn_CON.TWEN = 1) to enable TWI
2. Configure TWI control register (TWIn_CON): confiugure TWI communication rate bit(TWCK[3:0]) and set start bit STA to “1”
3. Configure the TWI address register (TWIn_ADD): Write the slave address and read/write bit into TWIDAT to transmit address frame on the bus
4. If the master is receiving data, wait for the interrupt flag TWIF in the TWIn_STS to be set to 1. The interrupt flag will be set to 1 for every 8 bits of data received and need to be cleared manually
5. If the master is sending data, write the data to be transmitted into TWIDAT. TWI will automatically transmit the data. The interrupt flag TWIF will be set to 1 for every 8 bits transmitted
6. Once data reception or transmission is complete, the master can transmit a stop signal (STO=1), and the master's state transitions to 000. Alternatively, the master can transmit a repeated start signal to begin a new round of data transmission

**Note: The master's TWIF flag will not be set after generating a stop condition!**

## 13.8 TWI Interrupt

For TWI0 and TWI1, the following events can trigger an interrupt. All TWI events share a common interrupt flag

| Interrupt Event | Event Flag | Interrupt Request Control Bit |
|---|---|---|
| Master mode: start signal transmission complete | TWIF | TWIn_IDE ->INTEN |
| Master mode: address frame transmission complete | | |
| Master mode: data frame reception or transmission complete | | |
| Slave mode: first frame address successfully match | | |
| Slave mode: successfully receive or transmit 8 bits data | | |
| Slave mode: receive restart signal | | |
| Slave mode: receive stop signal | | |

## 13.9 TWI Register

### 13.9.1 TWI Control Register (TWIn_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TWIn_CON<br>(n=0~1) | R/W | TWI Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SPOS[1:0] | | - | - | TWCK[3:0] | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TWEN | - | STA | STO | - | - | AA | STRETCH |

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>15~14</td><td>SPOS[1:0]</td><td>
● TWI0 Port Mapping Control Bit@TWI0_CON
<table>
<tr><td>Port<br>SPOS Value</td><td>SCL0</td><td>SDA0</td></tr>
<tr><td>SPOS[1:0]=00</td><td>PB1</td><td>PB0</td></tr>
<tr><td>SPOS[1:0]=01</td><td>PC10</td><td>PC11</td></tr>
<tr><td>SPOS[1:0]=10</td><td>PA6</td><td>PA5</td></tr>
<tr><td>SPOS[1:0]=11</td><td>PA13</td><td>PA14</td></tr>
</table>
● TWI1 Port Mapping Control Bit@TWI1_CON
<table>
<tr><td>Port<br>SPOS Value</td><td>SCL1</td><td>SDA1</td></tr>
<tr><td>SPOS[1:0]=00</td><td>PA12</td><td>PA13</td></tr>
<tr><td>SPOS[1:0]=01</td><td>PA1</td><td>PA2</td></tr>
<tr><td>SPOS[1:0]=10</td><td>PA7</td><td>PA8</td></tr>
</table>
</td></tr>
<tr><td>11~8</td><td>TWCK[3:0]</td><td>TWI Master Mode Clock Presclar Control Bit:<br>0000: $f_{PCLK}$ /4096<br>0001: $f_{PCLK}$ /2048<br>0010: $f_{PCLK}$ /1024<br>0011: $f_{PCLK}$ /512<br>0100: $f_{PCLK}$ /256<br>0101: $f_{PCLK}$ /128<br>0110: $f_{PCLK}$ /64<br>0111: $f_{PCLK}$ /32<br>1000: $f_{PCLK}$ /16<br>1001: $f_{PCLK}$ /8<br>1010: $f_{PCLK}$ /4<br>Others: $f_{PCLK}$ /4<br>$f_{PCLK}$ = $f_{PCLK0}$ while choose TWI0 as master.<br>$f_{PCLK}$ = $f_{PCLK1}$ while choose TWI1 as master.</td></tr>
<tr><td>7</td><td>TWEN</td><td>TWI Enable Control Bit<br>0: Disable TWI<br>1: Enable TWI</td></tr>
<tr><td>5</td><td>STA</td><td>TWI Initial Position Trigger Switch<br>Start condition will be generated when this bit is set to 1, and the TWI will switch to master mode.<br>Software can set or clear this bit, or it can be cleared by hardware after the start condition is issued.</td></tr>
<tr><td>4</td><td>STO</td><td>TWI Stop Bit Trigger Switch<br>In master mode, writing 1 to this bit will generate a stop condition after the current byte transmission or the start condition is issued.</td></tr>
</table>

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | Software can set or clear this bit, or it can be cleared by hardware when a stop condition is detected. |
| 1 | AA | TWI Acknowledge Enable Bit<br>0: No acknowledgment, returns NACK (acknowledge bit is high level).<br>1: Returns an acknowledgment (ACK) after receiving a matching address or data |
| 0 | STRETCH | TWI Clock Stretching Enable Bit<br>This bit is only valid in slave mode.<br>0: Disable clock stretching.<br>1: Enable clock stretching, and the master needs to support clock stretching.<br>Description: Clock stretching will occur after data transmission is complete and ACK is 0. |
| 31~16<br>13~12<br>6,3~2 | - | Reserved |

### 13.9.2 TWI Status Flag Register (TWIn_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TWIn_STS<br>(n=0~1) | R/W | TWI Status Flag Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| NBYTES[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | STATE[2:0] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | MSTR | GCA | TXnE/RXnE | TWIF |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 23~16 | NBYTES[7:0] | Transmission/Reception Buffer Number Setting Bit<br>Used to set the number of bytes to be transmitted/received.<br>For each successful transmission/reception, NBYTES will automatically decrease by 1. When NBYTES reaches 0, the TC flag will be set.<br>**Note: Modification is not allowed when STA is set to 1.** |
| 10~8 | STATE[2:0] | TWI Status Bits<br>Used to indicate the TWI status, with different meanings in master/slave mode<br>● Slave mode: |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 000: Slave is in idle state, waiting for TWEN to be set, detecting the TWI start signal. The slave will transit to this state after receiving a stop condition<br>001: Slave is receiving the first frame of address and read/write bit (the 8th bit is the read/write bit, 1 for read, 0 for write). The slave will transit to this state after receiving the start condition<br>010: Slave is in the data reception state<br>011: Slave is in the data transmission state<br>100: Slave will transit to this state when the master responds with NACK in the data transmission state, waiting for a restart signal or stop signal<br>101: Slave will transit to this state when writing AA to 0 in transmission state, waiting for a restart signal or stop signal<br>110: Slave will transit to this state if the slave's address does not match the address sent by the master, waiting for a new start condition or stop condition<br>● Master mode:<br>000: State machine is in idle state<br>001: Master is transmitting the start condition or the address of slave device<br>010: Master is transmitting data<br>011: Master is receiving data<br>100: Master has transmitted the stop condition or received the NACK signal from the slave |
| 3 | MSTR | TWI Master/Slave Mode Flag Bit<br>0: Slave mode<br>1: Master mode<br>Description:<br>1. When the TWI interface transmit a start condition to the bus, it automatically switches to master mode, and the hardware will set this bit.<br>2. When a stop condition is detected on the bus, the hardware will clear this bit. |
| 2 | GCA | TWI General Call Address Response Flag Bit<br>0: Non-response to general call address<br>1: When GC is set to 1 and there is a match with the general call address, this bit will set to 1 by the hardware and then automatically cleared |
| 1 | TXnE/RXnE | TWI Transmission Complete Flag Bit<br>TXnE/RXnE will be set to 1 by the hardware in the following cases<br>● Master mode:<br>■ Master transmits an address frame (write), and receives ACK from the slave |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | ■ Master finishes sending data and receives ACK from the slave<br>■ Master receives data and responds with ACK to the slave<br>● Slave mode:<br>■ Slave receives an address frame (read), and the received address matches the slave address (TWA)<br>■ Slave receives data and responds with ACK to the master<br>■ Slave finishes sending data and receives ACK from the master (AA=1)<br>After a read/write operation on TWIDAT, this bit will be cleared by the hardware |
| 0 | TWIF | TWI Interrupt Flag Bit<br>This bit is set to 1 by the hardware and can be cleared by writing 1 through software<br>● Master mode:<br>■ Transmit start signal<br>■ Finish transmitting the address frame<br>■ Receive or finish transmitting a data frame<br>● Slave mode:<br>■ Successful match of the first address frame<br>■ Successfully receive or transmit 8 bits of data<br>■ Receive a repeated start condition<br>■ Slave receives a stop signal |
| 31~24<br>15~11<br>7~4 | - | Reserved |

### 13.9.3 TWI Address Register (TWIn_ADD)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TWIn_ADD<br>(n=0~1) | R/W | TWI Address Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~1 | TWA[6:0] | TWI Address Register<br>TWA[6:0] cannot be written as all 0; 00H is reserved for general call address.<br>This bit is not valid in master mode |
| 0 | GC | TWI General Call Address Response Enable Bit<br>0: Disable response to general call address 00H<br>1: Enable response to general call address 00H |
| 31~8 | - | Reserved |

### 13.9.4 TWI Data Register (TWIn_DATA)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TWIn_DATA<br>(n=0~1) | R/W | TWI Data Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | TWIDAT[7:0] | TWI Data Buffer<br>Read operation: Read the received data from the TWI reception buffer.<br>Write operation: Write the data to be transmitted into the TWI transmission buffer. |
| 31~8 | - | Reserved |

### 13.9.5 TWI Interrupt Enable And DMA Control Register (TWIn_IDE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TWIn_IDE<br>(n=0~1) | R/W | TWI Interrupt Enable And DMA Control Register | 0x0000_0000 |

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

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TXDMAEN | DMA Transmission Channel Enable Bit<br>0: Disable DMA transmission function<br>1: Enable DMA transmission function<br>When this bit is enabled, setting TXnE can trigger DMA channel transmit requests.<br>**Note:**<br>**1. TWI0 can generate DMA requests**<br>**2. TWI1 cannot generate DMA requests** |
| 6 | RXDMAEN | DMA Receive Channel Enable Bit<br>0: Disable DMA receive function<br>1: Enable DMA receive function<br>When this bit is enabled, setting RXnE can trigger DMA channel transmit requests.<br>**Note:**<br>**1. TWI0 can generate DMA requests**<br>**2. TWI1 cannot generate DMA requests** |
| 0 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 31~8<br>5~1 | - | Reserved |

### 13.9.6 TWI Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TWI0 Base Address:0x4002_0060 | | | | |
| TWI0_CON | 0x00 | R/W | TWI0 Control Register | 0x0000_0000 |
| TWI0_STS | 0x04 | R/W | TWI0 Status Flag Register | 0x0000_0000 |
| TWI0_ADD | 0x08 | R/W | TWI0 Address Register | 0x0000_0000 |
| TWI0_DATA | 0x0C | R/W | TWI0 Data Register | 0x0000_0000 |
| TWI0_IDE | 0x10 | R/W | TWI0 Interrupt Enable and DMA Control Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TWI1 Base Address:0x4002_1060 | | | | |
| TWI1_CON | 0x00 | R/W | TWI1 Control Register | 0x0000_0000 |
| TWI1_STS | 0x04 | R/W | TWI1 Status Flag Register | 0x0000_0000 |
| TWI1_ADD | 0x08 | R/W | TWI1 Address Register | 0x0000_0000 |
| TWI1_DATA | 0x0C | R/W | TWI1 Data Register | 0x0000_0000 |
| TWI1_IDE | 0x10 | R/W | TWI1 Interrupt Enable and DMA Control Register | 0x0000_0000 |

# 14 Hardware Watchdog WDT

## 14.1 Overview

The SC32F10T/10G series features a built-in hardware watchdog (WDT) with an internal 32kHz oscillator as its clock source. Users can choose to enable the watchdog reset function by setting the ENWDT control bit in the Code Option through a programmer.

Hardware watchdog (WDT) features high security, accurate timing, and flexibility in use. This watchdog peripheral can detect and resolve faults caused by software errors, triggering a system reset when the counter reaches a predefined overflow time.

The WDT is driven by its internal low-frequency oscillator, ensuring it remains operational even in the event of a failure in the main clock.

## 14.2 Clock Source

The SC32F10T/10G series WDT is fixed to LIRC. Once the WDT is enabled, LIRC will automatically start, and it will remain oscillating throughout the operation of the WDT and users cannot turn off LIRC while the WDT is active.

## 14.3 WDT Register

### 14.3.1 WDT Control Register (WDTCON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| WDTCON | R/W | WDT Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | CLRWDT | WDT Counter Clear Bit<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0:None effect |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 1: WDT counter count from 0 |
| 31~1 | - | Reserved |

### 14.3.2 WDT Configuration Register (WDTCFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| WDTCFG | R/W | WDT Configuration Register | 0x0000_0000 |

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

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>2~0</td><td>WDTCKS[2:0]</td><td>Watchdog Clock Selection:
<table>
<tr><th>WDTCKS[2:0]</th><th>WDT Overflow Time</th></tr>
<tr><td>000</td><td>500ms</td></tr>
<tr><td>001</td><td>250ms</td></tr>
<tr><td>010</td><td>125ms</td></tr>
<tr><td>011</td><td>62.5ms</td></tr>
<tr><td>100</td><td>31.5ms</td></tr>
<tr><td>101</td><td>15.75ms</td></tr>
<tr><td>110</td><td>7.88ms</td></tr>
<tr><td>111</td><td>3.94ms</td></tr>
</table></td></tr>
<tr><td>31~3</td><td>-</td><td>Reserved</td></tr>
</table>

### 14.3.3 WDT Register Mapping

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| WDT Base Address:0x4000_0330 | | | | | |
| WDTCON | 0x0C | R/W | WDT Control Register | 0x0000_0000 | Do not support byte/half word access |
| WDTCFG | 0x10 | R/W | WDT Configuration Register | 0x0000_0000 | Do not support byte/half word access |

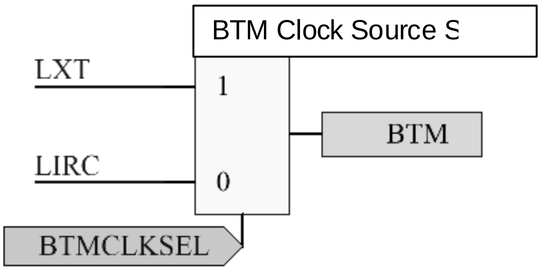
# 15 Base Timer(BTM)

## 15.1 Overview

The SC32F10T/10G series features a Base Timer (BTM) that can generate interrupts at intervals ranging from 15.625ms to 32s. The BTM can use either 32kHz LIRC or external 32.768kHz crystal oscillator (LXT) as its clock source. The interrupts generated by the BTM can wake up the CPU from STOP mode.

## 15.2 Clock Sourse

SC32F10T/10G series BTM can choose LXT or LIRC as its clock sourse



## 15.3 Feature

- Selectable interrupt frequency intervals from 15.625ms to 32s
- Can wake up from STOP Mode

## 15.4 BTM Interrupt

When the SC32F10T/10G series BTM counter reaches the conditions set by BTMFS, the BTMIF will be set. If BTM_CON.INTEN = 1, an interrupt will be generated.

| Interrupt Event | Event Flag | Interrupt Enable Control Bit |
|---|---|---|
| BTM interrupt request | BTMIF | BTM_CON->INTEN |

## 15.5 BTM Register

### 15.5.1 BTM Control Register (BTM_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| BTM_CON | R/W | BTM Control Register | 0x0000_0000 |

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

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | ENBTM | Base Timer Enable Control Bit<br>0:Base Timer disable<br>1:Base Timer enable |
| 6 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 3~0 | BTMFS[3:0] | BTM Interrupt Frequency<br>0000: generate an interrupt every 16.625ms<br>0001: generate an interrupt every 31.25ms<br>0010: generate an interrupt every 62.5ms<br>0011: generate an interrupt every 125ms<br>0100: generate an interrupt every 0.25s<br>0101: generate an interrupt every 0.5s<br>0110: generate an interrupt every 1s<br>0111: generate an interrupt every 2s<br>1000: generate an interrupt every 4s<br>1001: generate an interrupt every 8s<br>1010: generate an interrupt every 16s<br>1011: generate an interrupt every 32s<br>1100~1111: Reserved |
| 31~8<br>5~4 | - | Reserved |

### 15.5.2 BTM Flag Register (BTM_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| BTM_STS | R/W | BTM Flag Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | BTMIF | BTM Interrupt Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software.<br>BTMIF will be set when BTM counter meets the conditions set by BFMFS. |
| 31~1 | - | Reserved |

### 15.5.3 BTM Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| BTM Base Address:0x4002_2100 | | | | |
| BTM_CON | 0x00 | R/W | BTM Control Register | 0x0000_0000 |
| BTM_STS | 0x04 | R/W | BTM Flag Register | 0x0000_0000 |

# 16 Built-in CRC Module

## 16.1 Overview

The SC32F10T/10G series has a built-in CRC (Cyclic Redundancy Check) module that utilizes a polynomial generator to generate CRC codes from an 8-bit/16-bit/32-bit data word. In numerous applications, CRC-based techniques are commonly used to verify the integrity of data transmission or storage. According to the functional safety standards, these techniques offer a means to verify the integrity of Flash. The CRC calculation unit helps compute the software signature during runtime, and this signature is then compared with the reference signature generated at link time and stored in a designated storage unit.

## 16.2 Clock Source

The SC32F10T/10G series CRC has only one clock source, which is derived from PCLK.

## 16.3 Feature

- 1 built-in hardware CRC module
- Configurable initial value, with a default of 0xFFFFFFFF
- Supports 8-bit/16-bit/32-bit data units
- Programmable polynomial, with a default of 0x4C11DB7
- Only supports software-driven data computation mode
- Supports DMA: CRC_DR can serve as the DMA destination address or be accessed directly via registers
- Calculating CRC for a single byte requires 1 system clock

| CRC algorithm | CRC-32/MPEG-2 |
|---|---|
| Polynomial Formula | $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$ |
| Data Width | 32bit |
| Inital Value | 0xFFFFFFFF |
| Result XOR Value | 0x00000000 |
| Input Value Reversal | false |
| Output Value Reversal | false |
| LSB/MSB | MSB |

**Note: The written and read data in CRCDR cannot be the same.**

## 16.4 CRC Register

### 16.4.1 CRC Data Register (CRC_DR)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CRC_DR | R/W | CRC Data Register (calculation result) | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | CRCDR[31:0] | CRC Data Register<br>This register is used to write new data to the CRC calculator. When reading the register, the previous CRC calculation result can be obtained. If the data size is less than 32 bits, the least significant bits can be used to write/read the correct value. The operation requirements for this register are as follows:<br>1. First, CRC_CON.CRCRST need to be set to 1 to reset CRCDR<br>2. When "CRCREG" is written, the hardware automatically calculates the CRC result and continues to store it in CRCDR<br>When needed, read out the CRC calculation result instantly. |

### 16.4.2 CRC Control Register (CRC_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CRC_CON | R/W | CRC Control Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~6 | POLYSIZE[1:0] | CRC Polynomial Size Setting Bits00:32 bits polynomial<br>01: 16 bits polynomial<br>10: 8 bits polynomial<br>11: 7 bits polynomial |
| 0 | CRCRST | CRCDR Register Reset Bit(Q31~Q0)<br>This bit is set to 1 by software, and is automatically cleared by hardware.<br>0: None effect<br>1: Reset CRCDR, and the reset value is the value of CRC_INIT register user write in. |
| 31~8<br>5~1 | - | Reserved |

### 16.4.3 CRC Initial Value Register (CRC_INT)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CRC_INT | R/W | CRC Initial Value Register | 0xFFFF_FFFF |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | CRC_INIT[31:0] | Programmable CRC intial value, reset value:0xFFFF FFFF<br>This register is used for users to write in CRC initial value. |

### 16.4.4 CRC Polynomial Setting Register (CRC_POL)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| CRC_POL | R/W | CRC Polynomial Setting Register | 0x04C1_1DB7 |

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

| POL[7:0] |
|---|

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | POL[31:0] | Programmable polynomial, reset value:0x04C1_1DB7<br>This register is used to write the coefficients of the polynomial to be used for CRC calculation. If the polynomial size is less than 32 bits, the least significant bits must be used to program the correct values. |

### 16.4.5 CRC Register Mapping

| Register | Offset Address | R/W | Description | Reset Value | Access Restriction |
|---|---|---|---|---|---|
| CRC Base Address:0x4000_2000 | | | | | |
| CRC_DR | 0x00 | R/W | CRC Data Register | 0x0000_0000 | - |
| CRC_CON | 0x04 | R/W | CRC Control Register | 0x0000_0000 | - |
| CRC_INT | 0x08 | R/W | CRC Initial Value Register | 0x0000_0000 | Do not support byte/half word access |
| CRC_POL | 0x0C | R/W | CRC Polynomial Setting Register | 0x0000_0000 | Do not support byte/half word access |

# 17 PWM0: 8 Channels of 16-bit Multifunctional PWM

## 17.1 Overview

The PWM0 of the SC32F10T/10G series is an 8-channel 16-bit shared-cycle multifunctional PWM. PWM0 has rich functionalities, including support for adjusting the cycle and duty cycle, the option to choose between center-aligned or edge-aligned output waveforms, selectable independent or complementary output modes, support for dead-time functionality, and a fault detection mechanism. The Register PWM0_CON and PWM0_STS control the state and cycle of the PWM. Each channel of PWM can be individually adjusted for enabling, output waveform, waveform inversion, and duty cycle.

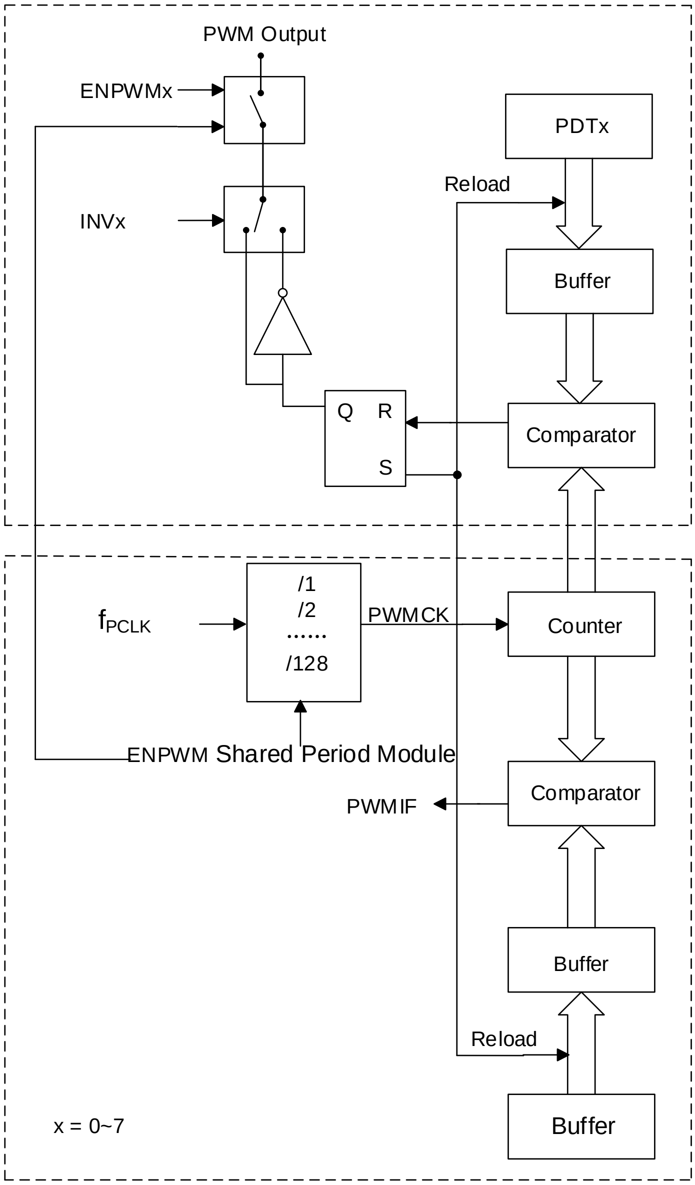
## 17.2 Clock Source

- The SC32F10T/10G series PWM0 has only one clock source, which is derived from PCLK
- PWM0 output frequency is at its maximum the frequency of the selected clock source
- PWM0 clock pre-scaler can select from 1 to 128

## 17.3 Feature

- 8 channels of 16-bit shared-period multifunctional PWM
- The output waveform can be inverted
- Waveform types: can be set as center-aligned or edge-aligned
- PWM modes: can be set as independent mode or complementary mode:
  - In independent mode, all 8 PWM channels share the same period, but the duty cycle of each PWM channel can be adjusted independently
  - In complementary mode, four pairs of complementary PWM waveforms with dead time can be generated simultaneously
- Provides one PWM overflow interrupt
- Supports fault detection
- Has independent interrupt request flags

## 17.4 PWM0 Structure Diagram



PWM0 Structure Diagram

## 17.5 PWM0 General Configuration

### 17.5.1 Output Mode

- In independent mode, all 8 PWM channels share the same period, but the duty cycle of each PWM channel can be adjusted independently
- In complementary mode, four pairs of complementary PWM waveforms with dead time can be generated simultaneously

## 17.5.2 Alignment Type

### 17.5.2.1 Edge-aligned

The PWM counter counts up from 0, and when the count matches the value set for the duty cycle in PDT0x [15:0], the PWM output waveform switches between high and low levels. Then, the PWM counter continues to count up until it matches the value of the period set in PWMPD[15:0] + 1 (one PWM cycle completes). The PWM counter will then reset to 0, and if PWM interrupt is enabled, a PWM interrupt will be generated at this point. The PWM output waveform is in the left-aligned mode.

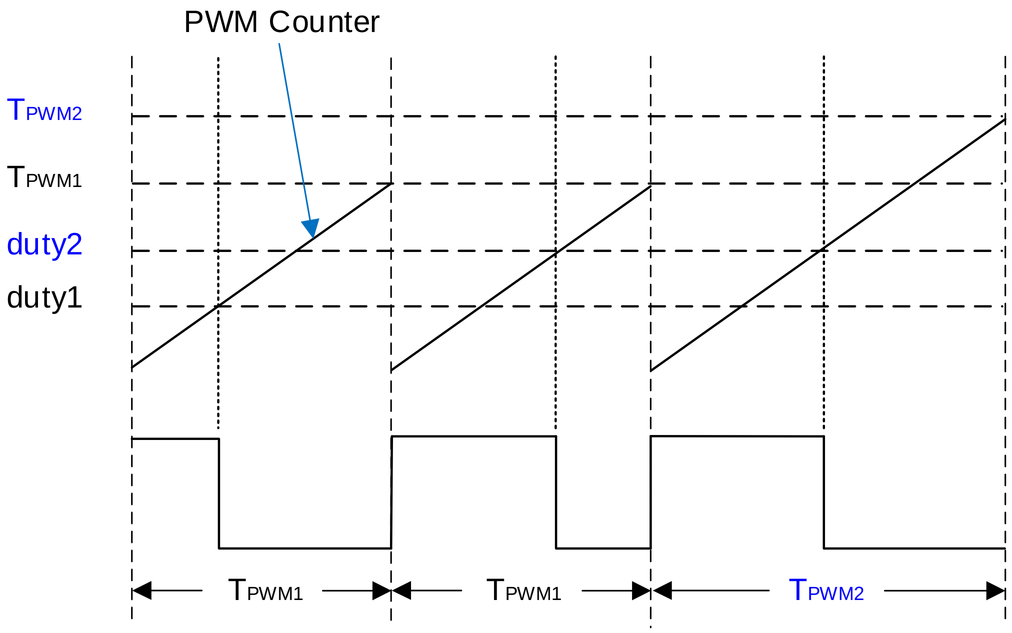
The formula for the edge-aligned period TPWM is calculated as follows:

$$\mathrm{Tpwm} = \frac{\mathrm{PWMPD}[15:0] + 1}{\mathrm{PWM\ Clock\ Frequency}}$$

Duty cycle (duty) calculation formula for edge-aligned mode:

$$\mathrm{duty} = \frac{\mathrm{PDT0x}\ [15:0]}{\mathrm{PWMPD}[15:0] + 1}$$

Edge-aligned waveform diagram is as follows:



Edge-Aligned PWM Waveform Diagram

### 17.5.2.2 Center-aligned

The PWM counter counts up from 0, and when the count matches the value set for the duty cycle in PDT0x [15:0], the PWM output waveform switches between high and low levels. Then, the PWM counter continues to count up until it matches the value of the period set in PWMPD[15:0] + 1 (the midpoint of the PWM period), it automatically starts counting downwards. When the count value matches PDT0x [15:0] again, the PWM output waveform switches again, and the PWM counter continues counting downwards until overflow (the end of a PWM period). If PWM interrupt is enabled, a PWM interrupt will be generated at this point.

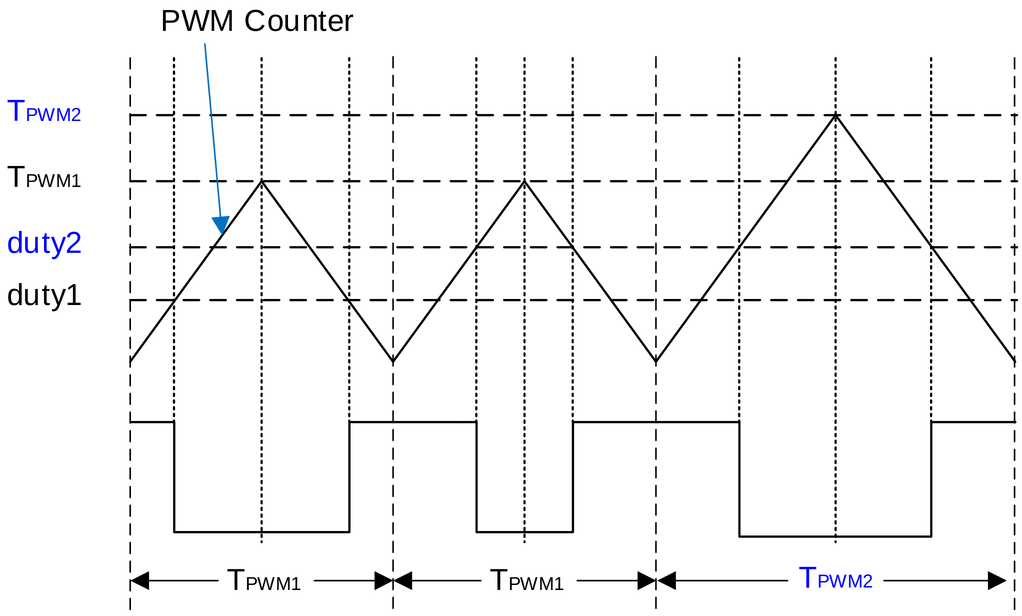
The formula for the center-aligned period TPWM is calculated as follows:

$$\text{Tpwm} = 2 * \frac{\text{PWMPD}[15:0] + 1}{\text{PWM Clock Frequency}}$$

Duty cycle (duty) calculation formula for center-aligned mode:

$$\text{duty} = \frac{\text{PDT0x}\ [15:0]}{\text{PWMPD}[15:0] + 1}$$

center-aligned waveform diagram is as follows:



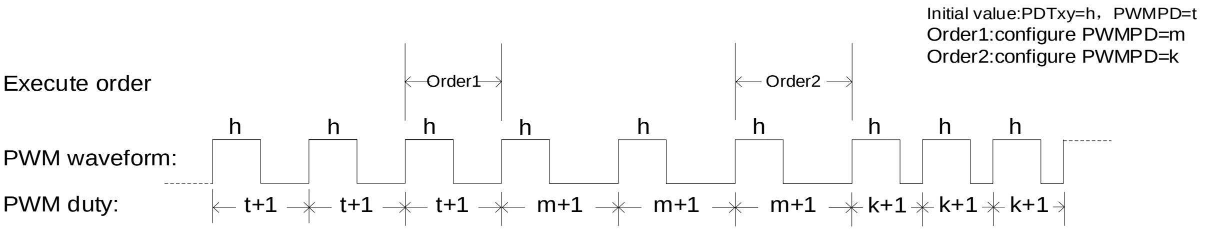
Center-Aligned PWM Waveform Diagram

### 17.5.3 Duty Cycle Change Characteristics

When generating the PWM0n output waveform, if it is necessary to change the duty cycle, it can be achieved by modifying the high-level setting register (PDT0x). However, it is important to note that changing the value of PDT0x will not immediately alter the duty cycle. Instead, the change takes place when the PWM counter counts to 0 or counts up to a value matching the setting in the period setting item PWMPD[15:0] +1.

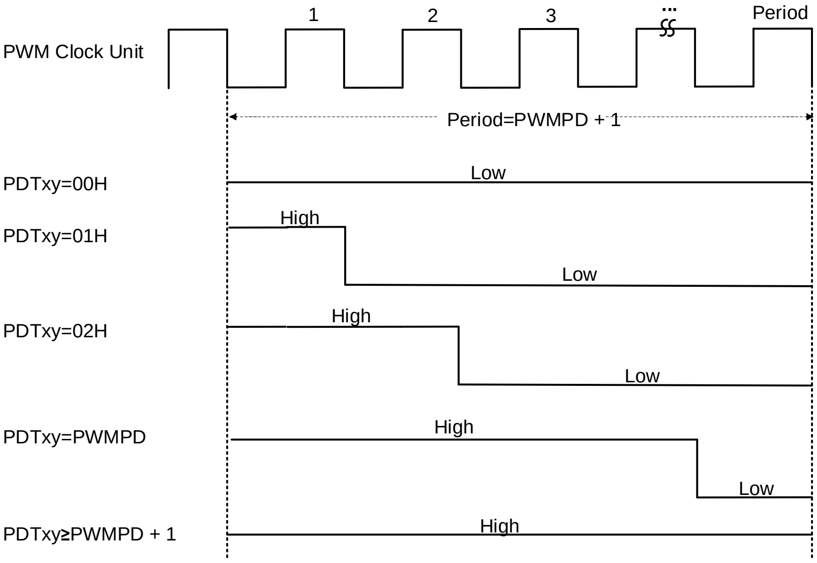
### 17.5.4 Period Change Characteristics

When generating PWM output waveforms, if it is necessary to change the period, it can be achieved by modifying the period setting register PWMPD. Similar to the duty cycle, changing the value of PWMPD will not immediately alter the period. The change takes place when the PWM counter counts to 0 or counts up to a value matching the setting in the period setting item PWMPD[15:0] +1.The reference diagram is as follows:



## 17.5.5 Relationship Between Period and Duty Cycle

The assumption for this result is that the initial value of PWM output inversion control (INVx, x=0~7) is 0. If the opposite result is desired, INVx should be set to 1. The relationship between period and duty cycle is as follows:



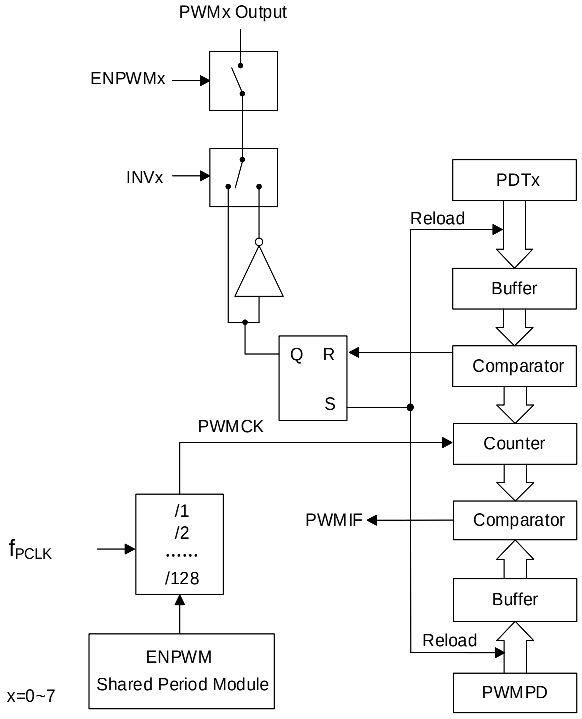
Period and Duty Cycle Relationship Diagram

## 17.6 PWM0 Fault Detection Mechanism Configuration

The fault detection function is often applied to the protection of motor systems. When the fault detection function is enabled and FLTEN (PWM0_FLT.7) is set to 1, the fault detection signal input pin (FLT) becomes effective. When the signal on the FLT pin meets the fault conditions, the FLTSTA flag will be set to 1 by hardware, and PWM output will stop, but the PWM counter continues counting, and the PWM interrupt will not be affected. The fault detection mode has latched mode and immediate mode. In immediate mode, when the fault signal on the FLT pin meets the disable condition, the FLTSTA flag will be cleared by hardware until the PWM counter returns to zero, and the PWM resumes output. In latched mode, when the fault signal on the FLT pin meets the disable condition, the FLTSTA flag will remain unchanged. Users can clear it through software. Once the FLTSTA status is cleared, the PWM counter will resume counting until it returns to zero, then the PWM will resume output.

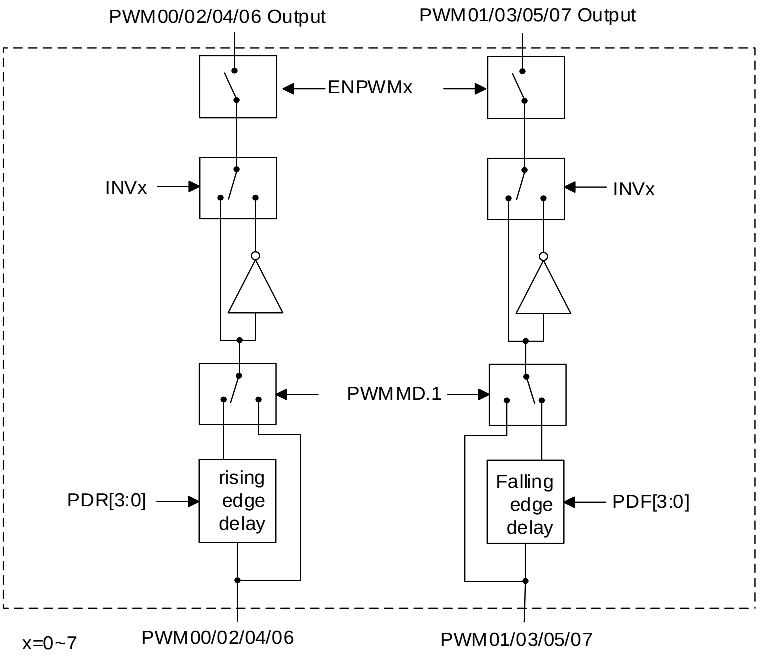
In independent mode (PWM0_CON.5 = 0), the duty cycle of each 8 PWM channels can be independently set. After configuring the output state and period of PWM, the PWM waveform can be output with a fixed duty cycle by configuring the duty cycle register of the corresponding PWM channel.

## 17.7 PWM0 Independent Mode



SC32F10XX Series PWM0 Independent Mode Diagram

## 17.8 PWM0 Complementary Mode



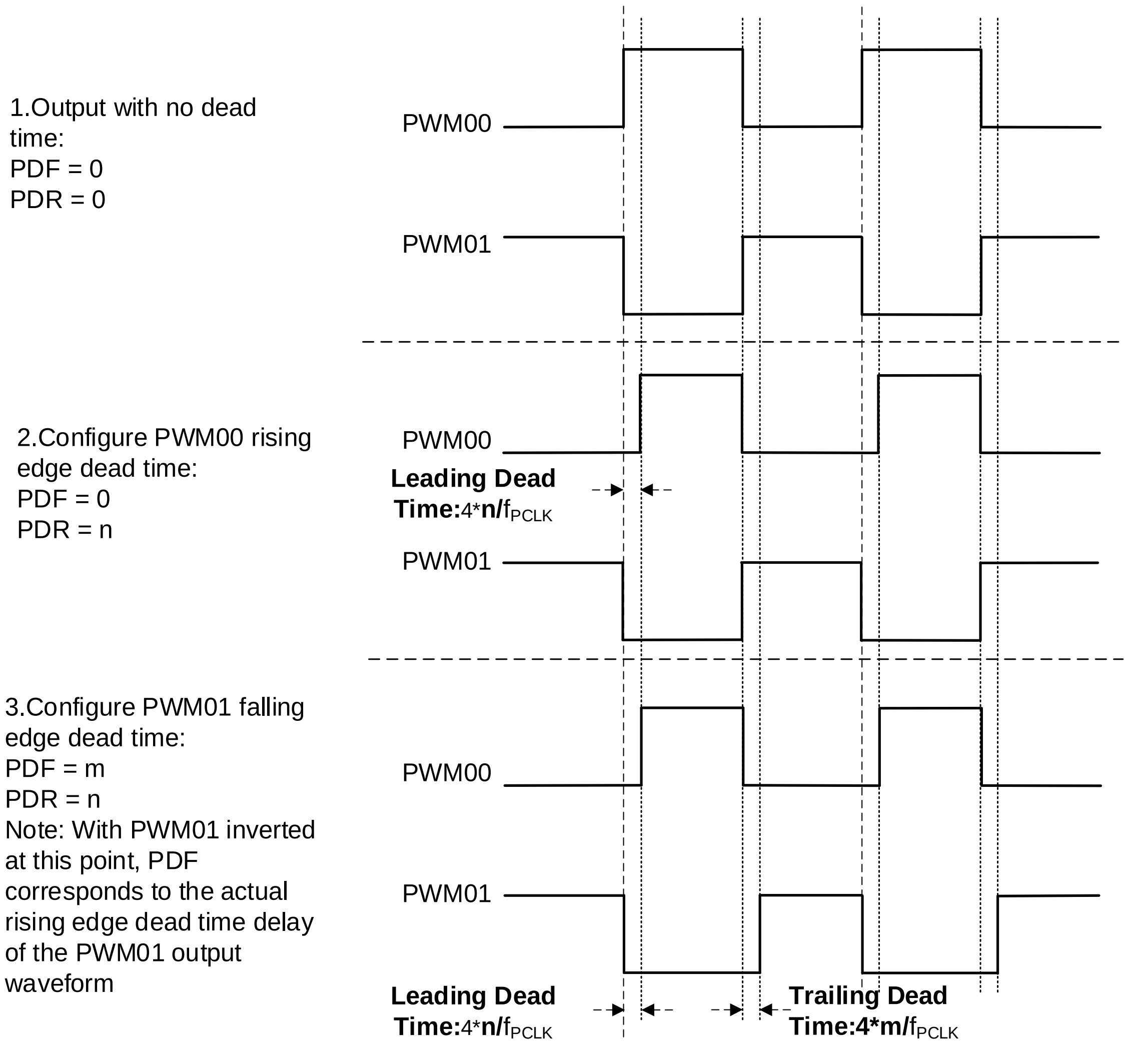
SC32F10XX Series PWM0 Complementary Diagram

### 17.8.1 PWM0 Complementary Mode Dead Time Configuration

When the SC32F10XX series PWM0 operates in complementary mode, the dead time control module can prevent the two complementary PWM signals from overlapping in the effective region, ensuring that the pair of complementary power switching transistors driven by PWM signals do not conduct simultaneously in practical applications.

### 17.8.2 PWM0 Dead Time Output Waveform

The following diagram shows the waveform of dead time adjustment with PWM00 and PWM01 in complementary mode. For clarity, PWM01 has been inverted (INV1=1).


1.Output with no dead time:
PDF = 0
PDR = 0
PWM00
PWM01
2.Configure PWM00 rising edge dead time:
PDF = 0
PDR = n
PWM00
Leading Dead Time:4*n/$f_{PCLK}$
PWM01
3.Configure PWM01 falling edge dead time:
PDF = m
PDR = n
Note: With PWM01 inverted at this point, PDF corresponds to the actual rising edge dead time delay of the PWM01 output waveform
PWM00
PWM01
Leading Dead Time:4*n/$f_{PCLK}$
Trailing Dead Time:4*m/$f_{PCLK}$

PWM0 Dead Time Output Wave

## 17.9 PWM0 Interrupt

When the SC32F10T/10G series PWM complete one cycle of output, the PWMIF will be set. If PWM0_CON.INTEN = 1, an interrupt will be generated.

| Interrupt Event | Event Flag | Interrupt Enable Control Bit |
|---|---|---|
| PWM0 interrupt request | PWMIF | PWM0_CON->INTEN |

## 17.10 PWM0 Register

### 17.10.1 PWM0 Control Register (PWM0_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_CON | R/W | PWM0 Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | INTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENPWM | PWMMD0 | PWMMD1 | - | - | PWMCK[2:0] | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 8 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 7 | ENPWM | PWM Module Switch Control Bit<br>0: PWM unit stops working, PWM counter clears to 0, and all PWM output ports are set to GPIO status<br>1: Allow the Clock to enter the PWM unit, and the PWM will be in working state. The state of PWM output ports are controlled by PWM_CHN.ENPWMx (x=0~7) |
| 6 | PWMMD0 | PWM Waveform Alignment Type Selection Bit<br>0: Edge-aligned<br>1: Center-aligned |
| 5 | PWMMD1 | PWM Waveform Complementary Mode Selection Bit<br>0: Independent mode<br>1: Complementary mode |
| 2~1 | PWMCK[2:0] | PWM Clock Frequency Control Bits<br>Used for control PWM clock frequency $f_{PWM0}$<br>000: $f_{PCLK0}$/1<br>001: $f_{PCLK0}$/2<br>010: $f_{PCLK0}$/4<br>011: $f_{PCLK0}$/8<br>100: $f_{PCLK0}$/16<br>101: $f_{PCLK0}$/32<br>110: $f_{PCLK0}$/64<br>111: $f_{PCLK0}$/128 |
| 31~9<br>4~3 | - | Reserved |

### 17.10.2 PWM0 Channel Configuration Register (PWM0_CHN)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_CHN | R/W | PWM0 Channel Configuration Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENPWM7 | ENPWM6 | ENPWM5 | ENPWM4 | ENPWM3 | ENPWM2 | ENPWM1 | ENPWM0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | ENPWMx<br>(x=0~7) | PWM0x Waveform Output Seletion<br>0: PWM0x output is turned off and functions as GPIO<br>1: When ENPWM=1, the pin associated with PWM0x serves as a waveform output port<br>**Note: If ENPWM is set to 1, the PWM module will be enabled, but if ENPWMx is set to 0, the PWM output will be turned off and function as a GPIO port. In this case, the PWM module can still be used as a 16-bit timer, and if PWM0_CON.INTEN = 1, an interrupt will be generated by PWM** |
| 31~8 | - | Reserved |

### 17.10.3 PWM0 Status Flag Register (PWM0_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_STS | R/W | PWM0 Status Flag Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 1 | FLTSTA | PWM Fault Detection Status Flag<br>0: PWM is in normal output state<br>1: Fault detection is active, and PWM output is in a high-impedance state. If PWM works in latch mode, this bit can be cleared by software |
| 0 | PWMIF | PWM Interrupt Request Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software. |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | When the PWM counter overflows (the count value exceeds PWMPD), this bit will be set by hardware. If PWM0_CON.INTEN = 1 at this time, a PWM0 interrupt will be generated |
| 31~2 | - | Reserved |

### 17.10.4 PWM0 Waveform Inversion Output Control Register (PWM0_INV)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_INV | R/W | PWM0 Waveform Inversion Output Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INV7 | INV6 | INV5 | INV4 | INV3 | INV2 | INV1 | INV0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | INVx<br>(x=0~7) | PWM0x Waveform Inversion Output Control Bit<br>0: PWM0x waveform output not inverted<br>1: PWM0x waveform output inverted |
| 31~8 | - | Reserved |

### 17.10.5 PWM0 Dead Time Configuration Register (PWM0_DFR)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_DFR | R/W | PWM0 Dead Time Configuration Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 11~8 | PDF[3:0] | Falling Edge Dead Time Configuration Bit<br>This bit is only valid in complementary mode:<br>PWM falling edge dead time= 4*PDF[3:0] / $f_{PCLK}$ |
| 3~0 | PDR[3:0] | Rising Edge Dead Time Configuration Bit<br>This bit is only valid in complementary mode:<br>PWM rising edge dead time = 4*PDR[3:0] / $f_{PCLK}$ |
| 31~12<br>7~4 | - | Reserved |

### 17.10.6 PWM0 Fault Detection Configuration Register (PWM0_FLT)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_FLT | R/W | PWM0 Fault Detection Configuration Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7 | FLTEN | PWM Fault Detection Function Control Bit<br>0: Fault detection function disable<br>1: Fault detection function enable |
| 5 | FLTMD | PWM Fault Detection Mode Configuration Bit<br>0: Latch mode, when fault input detected, the fault detection status flag FLTSTA will be set to 1 by hardware, and PWM output will stop; FLTSTA status will remain unchanged when the fault input is not detected<br>1: Immediate mode, when fault input detected, the fault detection status flag FLTSTA will be set to 1 by hardware, and PWM output will stop; FLTSTA will be cleared by hardware immediately when the fault input is not detected, and PWM output will resume output when PWM counter counts to 0 |
| 4 | FLTLV | PWM Fault Detection Level Selection Bit<br>0: Fault detection is valid when low level<br>1: Fault detection is valid when high level |

| | | |
|---|---|---|
| 1~0 | FLTDT[1:0] | PWM Fault Detection Input Signal Filter Time Configuration<br>00: Filter time is 0<br>01: Filter time is 1us<br>10: Filter time is 4us<br>11: Filter time is 16us |
| 31~8<br>6，3~2 | - | Reserved |

### 17.10.7 PWM0 Cycle Register (PWM0_CYCLE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_CYCLE | R/W | PWM0 Cycle Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PWMPD[15:0] | PWM0 Cycle Configuration Bits<br>This value represents the (period – 1) of the PWM output waveform; that means, the period of the PWM output is (PWMPD[15:0] + 1) * $f_{PWM0}$ |
| 31~16 | - | Reserved |

### 17.10.8 PWM0 Channel Duty Cycle Adjustment Register (PWM0_DTx)(x = 0~7)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| PWM0_DTx<br>(x = 0~7) | R/W | PWM0 Channel Duty Cycle Adjustment Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PDT[15:0] | PWM0x Duty Cycle Length Configuration, x = 0~7<br>● Independent mode: The high-level width of the PWM0x waveform is PDTx [15:0] PWM clocks<br>● Complementary mode: For complementary channels PWM0x and PWM0y (y = x + 1), the high-level width of the PWM0x and PWM0y waveforms is PDTx [15:0] PWM clocks |
| 31~16 | - | Reserved |

### 17.10.9 PWM0 Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| PWM0 Base Address:0x4002_0200 | | | | |
| PWM0_CON | 0x00 | R/W | PWM0 Control Register | 0x0000_0000 |
| PWM0_CHN | 0x04 | R/W | PWM0 Channel Configuration Register | 0x0000_0000 |
| PWM0_STS | 0x08 | R/W | PWM0 Status Flag Register | 0x0000_0000 |
| PWM0_INV | 0x0C | R/W | PWM0 Waveform Inversion Output Control Register | 0x0000_0000 |
| PWM0_DFR | 0x10 | R/W | PWM0 Dead Time Configuration Register | 0x0000_0000 |
| PWM0_FLT | 0x14 | R/W | PWM0 Fault Detection Configuration Register | 0x0000_0000 |
| PWM0_CYCLE | 0x18 | R/W | PWM0 Cycle Register | 0x0000_0000 |
| PWM0_DTx(x = 0~7) Base Address:0x4002_0230 | | | | |
| PWM0_DT0 | 0x00 | R/W | PWM0 Channel 0 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT1 | 0x04 | R/W | PWM0 Channel 1 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT2 | 0x08 | R/W | PWM0 Channel 2 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT3 | 0x0C | R/W | PWM0 Channel 3 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT4 | 0x10 | R/W | PWM0 Channel 4 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT5 | 0x14 | R/W | PWM0 Channel 5 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT6 | 0x18 | R/W | PWM0 Channel 6 Duty Cycle Adjustment Register | 0x0000_0000 |
| PWM0_DT7 | 0x1C | R/W | PWM0 Channel 7 Duty Cycle Adjustment Register | 0x0000_0000 |

# 18 LEDPWM: 8 Channels of 32-bit LEDPWM

## 18.1 Clock Source

The SC32F10T/10G series LEDPWM has only one clock source, which is derived from PCLK2.

## 18.2 Feature

- Shared period and independently adjustable duty cycle
- Support center-aligned mode for driving LEDs conveniently
- Duty cycle register shares with 28 SEG registers, serving as an alternative to LED circuits, generating LED driving waveforms
- The highest pre-scaling option is /256, with each step being 2^n
- Support independent interrupt request flags
- Achieve grayscale adjustment through center-aligned LEDPWM:
  - In grayscale adjustment, one COM corresponds to a maximum of 28 duty values, offering options like 8 X 24, 6 X 26, 5 X 27, 4 X 28
  - During LEDPWM interrupts, switch COM and write corresponding duty value into DUTY register of LEDPWM can achieve the adjustment of each SEG's grayscale.

## 18.3 LEDPWM Interrupt

When the SC32F10T/10G series PWM complete one cycle of output, the PWMIF will be set. If LEDPWM_CON.INTEN = 1, an interrupt will be generated.

| Interrupt Event | Event Flag | Interrupt Enable Control Bit |
|---|---|---|
| LEDPWM Interrupt Request | PWMIF | LEDPWM_CON->INTEN |

## 18.4 LEDPWM Register

### 18.4.1 LEDPWM Control Register (LEDPWM_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_CON | R/W | LEDPWM Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | INTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENPWM | PWMMD0 | - | - | PWMCK[3:0] | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 8 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 7 | ENPWM | LEDPWM Module Switch Control Bit<br>0: PWM unit stops working, PWM counter clears to 0, and all PWM output ports are set to GPIO status<br>1: Allow the Clock to enter the PWM unit, and the PWM will be in working state. The state of PWM output ports are controlled by PWM_CHN.ENPWMx (x=0~7) |
| 6 | PWMMD0 | LEDPWM Waveform Alignment Type Selection Bit<br>0: Edge-aligned<br>1: Center-aligned |
| 3~0 | PWMCK[3:0] | LEDPWM Clock Frequency Control Bits<br>Used for control LEDPWM clock frequency $f_{LEDPWM}$<br>000: $f_{PCLK2}$/1<br>001: $f_{PCLK2}$/2<br>010: $f_{PCLK2}$/4<br>011: $f_{PCLK2}$/8<br>100: $f_{PCLK2}$/16<br>101: $f_{PCLK2}$/32<br>110: $f_{PCLK2}$/64<br>111: $f_{PCLK2}$/128 |
| 31~9<br>5~4 | - | Reserved |

### 18.4.2 LEDPWM Channel Configuration Register (LEDPWM_CHN)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_CHN | R/W | LEDPWM Channel Configuration Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ENPWM31 | ENPWM30 | ENPWM29 | ENPWM28 | ENPWM27 | ENPWM26 | ENPWM25 | ENPWM24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ENPWM23 | ENPWM22 | ENPWM21 | ENPWM20 | ENPWM19 | ENPWM18 | ENPWM17 | ENPWM16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ENPWM15 | ENPWM14 | ENPWM13 | ENPWM12 | ENPWM11 | ENPWM10 | ENPWM9 | ENPWM8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ENPWM7 | ENPWM6 | ENPWM5 | ENPWM4 | ENPWM3 | ENPWM2 | ENPWM1 | ENPWM0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | ENPWMx<br>x=31~0 | LEDPWM Waveform Output Seletion<br>0: LEDPWMx output is turned off and functions as GPIO<br>1: When ENPWM=1, the pin associated with LEDPWMx serves as a waveform output port<br>**Note: If ENPWM is set to 1, the PWM module will be enabled, but if ENPWMx is set to 0, the PWM output will be turned off and function as a GPIO port. In this case, the PWM module can still be used as a 16-bit timer, and if LEDPWM_CON.INTEN = 1, an interrupt will be generated by PWM** |

### 18.4.3 LEDPWM Status Flag Register (LEDPWM_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_STS | R/W | LEDPWM Status Flag Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 0 | PWMIF | LEDPWM Interrupt Request Flag<br>This bit is set to 1 by hardware, and cleared by writing 1 through software.<br>When the PWM counter overflows (the count value exceeds PWMPD), this bit will be set by hardware. If LEDPWM_CON.INTEN = 1 at this time, a LEDPWM interrupt will be generated |
| 31~1 | - | Reserved |

### 18.4.4 LEDPWM Waveform Inversion Output Control Register (LEDPWM_INV)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_INV | R/W | LEDPWM Waveform Inversion Output Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| INV31 | INV30 | INV29 | INV28 | INV27 | INV26 | INV25 | INV24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| INV23 | INV22 | INV21 | INV20 | INV19 | INV18 | INV17 | INV16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INV15 | INV14 | INV13 | INV12 | INV11 | INV10 | INV9 | INV8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INV7 | INV6 | INV5 | INV4 | INV3 | INV2 | INV1 | INV0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | INVx<br>X=31~0 | LEDPWMx Waveform Inversion Output Control Bit<br>0: LEDPWMx waveform output not inverted<br>1: LEDPWMx waveform output inverted |

### 18.4.5 LEDPWM Cycle Register (LEDPWM_CYCLE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_CYCLE | R/W | LEDPWM Cycle Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | PWMPD[7:0] | LEDPWM Cycle Configuration Bits<br>This value represents the (period – 1) of the PWM output waveform; that means, the period of the PWM output is (PWMPD[7:0] + 1) * $f_{LEDPWM}$ |
| 31~8 | - | Reserved |

### 18.4.6 LEDPWM Channel Duty Cycle Adjustment Register (LEDPWM_DTn)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| LEDPWM_DTn<br>n = 0~31 | R/W | LEDPWM Channel Duty Cycle Adjustment Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWMPD [7:0] | | | | | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | PWMPD[7:0] | LEDPWMn Duty Cycle Length Configuration<br>The high-level width of the LEDPWMx waveform is PDTx [7:0] PWM clocks |
| 31~8 | - | Reserved |

### 18.4.7 LEDPWM Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| LEDPWM Base Address:0x4002_2300 | | | | |
| LEDPWM_CON | 0x00 | R/W | LEDPWM Control Register | 0x0000_0000 |
| LEDPWM_CHN | 0x04 | R/W | LEDPWM Channel Configuration Register | 0x0000_0000 |
| LEDPWM_STS | 0x08 | R/W | LEDPWM Status Flag Register | 0x0000_0000 |
| LEDPWM_INV | 0x0C | R/W | LEDPWM Waveform Inversion Output Control Register | 0x0000_0000 |
| LEDPWM_CYCLE | 0x18 | R/W | LEDPWM Cycle Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| LEDPWM _DTn(n = 0~31) Base Address:0x4002_2330 | | | | |
| LEDPWM_DT0 | 0x00 | R/W | LEDPWM Channel 0 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT1 | 0x04 | R/W | LEDPWM Channel 1 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT2 | 0x08 | R/W | LEDPWM Channel 2 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT3 | 0x0C | R/W | LEDPWM Channel 3 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT4 | 0x10 | R/W | LEDPWM Channel 4 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT5 | 0x14 | R/W | LEDPWM Channel 5 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT6 | 0x18 | R/W | LEDPWM Channel 6 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT7 | 0x1C | R/W | LEDPWM Channel 7 Duty Cycle Adjustment Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| LEDPWM_DT8 | 0x20 | R/W | LEDPWM Channel 8 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT9 | 0x24 | R/W | LEDPWM Channel 9 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT10 | 0x28 | R/W | LEDPWM Channel 10 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT11 | 0x2C | R/W | LEDPWM Channel 11 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT12 | 0x30 | R/W | LEDPWM Channel 12 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT13 | 0x34 | R/W | LEDPWM Channel 13 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT14 | 0x38 | R/W | LEDPWM Channel 14 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT15 | 0x3C | R/W | LEDPWM Channel 15 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT16 | 0x40 | R/W | LEDPWM Channel 16 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT17 | 0x44 | R/W | LEDPWM Channel 17 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT18 | 0x48 | R/W | LEDPWM Channel 18 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT19 | 0x4C | R/W | LEDPWM Channel 19 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT20 | 0x50 | R/W | LEDPWM Channel 20 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT21 | 0x54 | R/W | LEDPWM Channel 21 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT22 | 0x58 | R/W | LEDPWM Channel 22 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT23 | 0x5C | R/W | LEDPWM Channel 23 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT24 | 0x60 | R/W | LEDPWM Channel 24 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT25 | 0x64 | R/W | LEDPWM Channel 25 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT26 | 0x68 | R/W | LEDPWM Channel 26 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT27 | 0x6C | R/W | LEDPWM Channel 27 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT28 | 0x70 | R/W | LEDPWM Channel 28 Duty Cycle Adjustment Register | 0x0000_0000 |

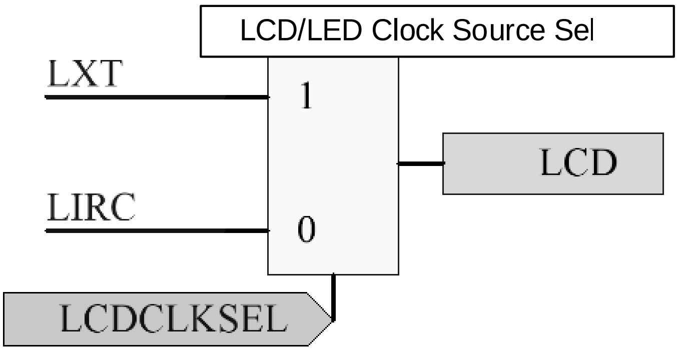
| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| LEDPWM_DT29 | 0x74 | R/W | LEDPWM Channel 29 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT30 | 0x78 | R/W | LEDPWM Channel 30 Duty Cycle Adjustment Register | 0x0000_0000 |
| LEDPWM_DT31 | 0x7C | R/W | LEDPWM Channel 31 Duty Cycle Adjustment Register | 0x0000_0000 |

# 19 LCD/LED Driver

LCD/LED option, sharing registers and I/O ports.

## 19.1 Clock Source

SC32F10T/10G series LCD/LED can choose LXT or LIRC as its clock sourse



## 19.2 Built-in 8 COM x 24 SEG LED Driver

- 1/1~1/8 duty voltage driving mode
- LED segment source driving capability with four-level control

## 19.3 Built-in 8 COM x 24 SEG LCD Driver

- Type A / Type B waveform selectable
- 8 X 24、6 X 26、5 X 27、4 X 28
- Optional voltage division resistor for LCD voltage output port
- LCD display driver bias voltage
  - 1/4 bias voltage
  - 1/3 bias voltage
- Three selectable frame rates:
  - Type A mode 32/64/128Hz
  - Type B mode 64/128/256Hz

## 19.4 LCD/LED Register

### 19.4.1 Display Driver Control Register (DDR_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DDR_CON | R/W | Display Driver Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TRIMODE | TRICOM | - | - | - | - | DDRCK[1:0] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDRON | - | - | TPYE | VOIRSIF | - | BIAS | DMOD |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15 | TRIMODE | Custom Frame Frequency Mode Control Bit<br>0: Disable custom frame rate mode<br>1: Enable custom frame rate mode<br><br>Used in conjunction with TIM interrupt to control the frame rate |
| 14 | TRICOM | When TRIMODE is set to 1, enabling the custom frame frequency mode, each '1' written to this bit will trigger a switch of the starting scanning COM port.<br>The following illustrates one scanning cycle for different duty cycle configurations:<br>● 1/8 duty cycle: Starts scanning from COM0 and ends at COM7, completing one scanning cycle.<br>● 1/6 duty cycle: Starts scanning from COM2 and ends at COM7, completing one scanning cycle.<br>● 1/5 duty cycle: Starts scanning from COM3 and ends at COM7, completing one scanning cycle.<br>● 1/4 duty cycle @ SCS=0: Starts scanning from COM4 and ends at COM7, completing one scanning cycle.<br>● 1/4 duty cycle @ SCS=1: Starts scanning from COM0 and ends at COM3, completing one scanning cycle. |
| 9~8 | DDRCK[1:0] | LCD/LED Frame Rate Prescaler Setting Bits<br>00: B waveform frame frequency 64Hz, A waveform frame frequency 32Hz<br>01: B waveform frame frequency 128Hz, A waveform frame frequency 64Hz<br>10: B waveform frame frequency 256Hz, A waveform frame frequency 128Hz<br>11: Reserved |
| 7 | DDRON | LCD/LED Display Driver Enable Control Bit<br>0: Display driver scan disable<br>1: Display driver scan enable |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 4 | TPYE | LCD Driver Waveform Selection Bit<br>0: B Waveform<br>1: A Waveform<br>**Note: In LED mode, modifying this bit will also affect the waveform frequency of the LED.** |
| 3 | VOIRSIF | LCD Fast Charging Enable Bit<br>0: Disable fast charging<br>1: Enable fast charging, select a 33k resistor for fast charging for 5 cycles, and then switch to the resistor value selected by VOIRS |
| 1 | BIAS | LCD Display Driver Bias Voltage Setting Bit<br>0: 1/4 bias voltage<br>1: 1/3 bias voltage |
| 0 | DMOD | LCD/LED Display Driver Mode Selection Bit<br>0: LCD mode<br>1: LED mode |
| 31~16<br>13~10<br>6~5<br>2 | - | Reserved |

### 19.4.2 Display Driver Configuration Register (DDR_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DDR_CFG | R/W | Display Driver Configuration Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 11~8 | VLCD[3:0] | LCD Voltage Adjustment Setting Bit<br>LCD output voltage: $V_{LCD}$ = VDD*(17+VLCD[3:0])/32 |
| 7 | SCS | LCD/LED Segment/Common Multiplexing Pin Selection Bit<br>0: When setting a 1/4 duty cycle,S0~S27 are segment,C4~C7 are common<br>1: When setting a 1/4 duty cycle,S4~S27 are segment,C0~C3 are common |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 5~4 | DUTY[1:0] | LCD/LED Display Duty Cycle Setting Bit<br>00: 1/8 duty cycle,S4~S27 are segment,C0~C7 are common<br>01: 1/6 duty cycle,S2~S27 are segment,C2~C7 are common<br>10: 1/5 duty cycle,S1~S27 are segment,C3~C7 are common<br>11: 1/4 duty cycle,S0~S27 are segment,C4~C7 are common or S4~S27 are segment,C0~C3 are common |
| 1~0 | VOIRS[1:0] | LCD Voltage Output Port Voltage Divider Resistor Selection:<br>00: Set the total resistance value of the internal divider resistor to 33k<br>01: Set the total resistance value of the internal divider resistor to 100k<br>10: Set the total resistance value of the internal divider resistor to 300k<br>11: Set the total resistance value of the internal divider resistor to 800k |
| 31~12<br>6<br>3~2 | - | Reserved |

### 19.4.3 SEG Enable Register (SEG_EN)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SEG_EN | R/W | SEG Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | SEG27 | SEG26 | SEG25 | SEG24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SEG23 | SEG22 | SEG21 | SEG20 | SEG19 | SEG18 | SEG17 | SEG16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SEG15 | SEG14 | SEG13 | SEG12 | SEG11 | SEG10 | SEG9 | SEG8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEG7 | SEG6 | SEG5 | SEG4 | SEG3 | SEG2 | SEG1 | SEG0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 27~0 | SEGx<br>(x=0~27) | SEGx Display Driver Output Control Bit, x= 0~27<br>0: Disable SEGx display driver output function<br>1: Enable SEGx display driver output function |
| 31~28 | - | Reserved |

### 19.4.4 COM Enable Register (COM_EN)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| COM_EN | R/W | COM Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COM7 | COM6 | COM5 | COM4 | COM3 | COM2 | COM1 | COM0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | COMx<br>(x=0~7) | COMx Display Driver Output Control Bit,x= 0~7<br>0: Disable COMx display driver output function<br>1: Enable COMx display driver output function |
| 31~8 | - | Reserved |

### 19.4.5 SEGn Display Register SEGRn

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| SEGRn<br>(n=0~27) | R/W | SEGn Display Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COM7 | COM6 | COM5 | COM4 | COM3 | COM2 | COM1 | COM0 |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 7~0 | COMx<br>(x=0~7) | SEGn Display Drive Output Control Bit for COMm, n= 0~27, m=0~7. Used to configure the SEGn display drive output for the corresponding COMm.<br>0: Disable<br>1: Enable |
| 31~8 | - | Reserved |

### 19.4.6 LCD/LED Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| LCD/LED Base Address:0x4002_2280 | | | | |
| DDR_CON | 0x00 | R/W | Display Driver Control Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DDR_CFG | 0x04 | R/W | Display Driver Configuration Register | 0x0000_0000 |
| SEG_EN | 0x08 | R/W | SEG Enable Register | 0x0000_0000 |
| COM_EN | 0x10 | R/W | COM Enable Register | 0x0000_0000 |
| SEGR Base Address:0x4002_2330 | | | | |
| SEGR0 | 0x00 | R/W | SEG0 Display Register | 0x0000_0000 |
| SEGR1 | 0x04 | R/W | SEG1 Display Register | 0x0000_0000 |
| SEGR2 | 0x08 | R/W | SEG2 Display Register | 0x0000_0000 |
| SEGR3 | 0x0C | R/W | SEG3 Display Register | 0x0000_0000 |
| SEGR4 | 0x10 | R/W | SEG4 Display Register | 0x0000_0000 |
| SEGR5 | 0x14 | R/W | SEG5 Display Register | 0x0000_0000 |
| SEGR6 | 0x18 | R/W | SEG6 Display Register | 0x0000_0000 |
| SEGR7 | 0x1C | R/W | SEG7 Display Register | 0x0000_0000 |
| SEGR8 | 0x20 | R/W | SEG8 Display Register | 0x0000_0000 |
| SEGR9 | 0x24 | R/W | SEG9 Display Register | 0x0000_0000 |
| SEGR10 | 0x28 | R/W | SEG10 Display Register | 0x0000_0000 |
| SEGR11 | 0x2C | R/W | SEG11 Display Register | 0x0000_0000 |
| SEGR12 | 0x30 | R/W | SEG12 Display Register | 0x0000_0000 |
| SEGR13 | 0x34 | R/W | SEG13 Display Register | 0x0000_0000 |
| SEGR14 | 0x38 | R/W | SEG14 Display Register | 0x0000_0000 |
| SEGR15 | 0x3C | R/W | SEG15 Display Register | 0x0000_0000 |
| SEGR16 | 0x40 | R/W | SEG16 Display Register | 0x0000_0000 |
| SEGR17 | 0x44 | R/W | SEG17 Display Register | 0x0000_0000 |
| SEGR18 | 0x48 | R/W | SEG18 Display Register | 0x0000_0000 |
| SEGR19 | 0x4C | R/W | SEG19 Display Register | 0x0000_0000 |
| SEGR20 | 0x50 | R/W | SEG20 Display Register | 0x0000_0000 |
| SEGR21 | 0x54 | R/W | SEG21 Display Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| SEGR22 | 0x58 | R/W | SEG22 Display Register | 0x0000_0000 |
| SEGR23 | 0x5C | R/W | SEG23 Display Register | 0x0000_0000 |
| SEGR24 | 0x60 | R/W | SEG24 Display Register | 0x0000_0000 |
| SEGR25 | 0x64 | R/W | SEG25 Display Register | 0x0000_0000 |
| SEGR26 | 0x68 | R/W | SEG26 Display Register | 0x0000_0000 |
| SEGR27 | 0x6C | R/W | SEG27 Display Register | 0x0000_0000 |

# 20 32-Channel High-Sensitivity Touch Key Circuit (TK)

- High-sensitivity mode
- Suitable for touch applications with high sensitivity requirements, such as proximity sensing and touch keys
- Channels can be scanned in parallel
- Support self-capacitance mode and mutual-capacitance mode
- Support low-power mode
- Comprehensive development support: Highly flexible touch software library, intelligent debugging software

**Note: Exclusive to the SC32F10T series**

# 21 16-bit Timers (Timer0~Timer7)

## 21.1 Clock Source

- In timer mode/PWM output mode, the TIM clock source is derived from PCLK
- In counter mode, the Tn pin serves as the counting source input

## 21.2 Feature

- 8 independent 16-bit auto-reload counters: Timer0 to Timer7
- 16-bit incremental, decremental, and increment/decrement auto-reload counters
- Support rising/falling edge capture, enabling PWM duty and period capture
- 16-bit programmable prescaler, allowing division of the counter clock frequency by any number between 1 and 65535
- Overflow and capture events of TIM1/2/6 can generate DMA requests
- TIM2/3/7 pins can be remapped:
  - TIM2
    - 0: T2CAP/T2 pin is PA10
    - 1: T2CAP/T2 pin is PA12
  - TIM3
    - 0: T3CAP/T3 pin is PA0
    - 1: T3CAP/T3 pin is PA13
  - TIM7
    - 0: T7CAP/T7 pin is PC1
    - 1: T7CAP/T7 pin is PB1

## 21.3 Counting method

### 21.3.1 Counting Method in Timer Mode

- Upward Counting: Counts from the set value upwards to overflow at 0xFFFF
- Downward Counting: Counts from 0xFFFF downwards to the set value

### 21.3.2 Counting Method in PWM Mode

Only upward counting mode is available in PWM output mode: The counter starts from 0 and counts up until PDT, then PWM output waveform will switch between the high and low levels. The counting will then continue up to RLD, causing an overflow and the counter reset to 0.

The formula of TPWM is shown as follows:

$$T_{PWM} = \frac{RLD[15:0] + 1}{PCLK}$$

The formula of duty is shown as follows:

$$\text{duty} = \frac{\text{PDT}[15:0]}{\text{RLD}[15:0] + 1}$$

## 21.4 Timer Signal Port

- Tn, n=1~7
  - Clock input/output
  - Both rising and falling edges can be captured
- TnEX, n=0
  - In reload mode, the external event input (falling edge) on the TnEX pin is used for reload enable/disable control
  - In capture mode, when FSEL = 1, it serves as a falling edge capture signal input. Detection of a falling edge on the TnEX pin generates a capture, sets EXIF, and captures the value of the TnCNT register into the FCAP register
- TnPWM, n=0~7
  - TIM1~7 can provide PWM with independently adjustable duty cycle through the Tn port: TnPWMA
  - TIM0 can provide PWM with independently adjustable duty cycle through the TnEX port: TnPWMB
  - Optional clock source follows TIM
  - Note: TIM's PWM capture function and PWM output function cannot be enabled simultaneously

## 21.5 Interrupts and Corresponding Flags for TIM:

- Overflow/underflow of the counter share the interrupt flag TIF
- Capture status flags:
  - EXIF: Flag indicating detection of a falling edge on the external event input
  - EXIR: Flag indicating detection of a rising edge on the external event input
- Interrupt and priority configuration control bits are merged into the NVIC module

## 21.6 TIM Interrupt

In timed or counting mode, when the count value of the CNT counter reaches the TIMn count value, TIF (Timer Interrupt Flag) will be set, and an interrupt will be generated if TIMn_IDE.INTEN = 1.

In external event input mode, when a valid edge transition is detected, EXIR/EXIF will be set, and an interrupt will be generated if TIMn_IDE.INTEN = 1.

| Interrupt Event | Event Flag | Interrupt Enable Control Bit | Interrupt Enable Sub-Switch |
|---|---|---|---|
| Timer overflow | TIF | TIMn_IDE->INTEN<br>(n=0~7) | TIMn_IDE->TIE |
| External event input rising edge interrupt | EXIR | | TIMn_IDE->EXRIE |
| External event input falling edge interrupt | EXIF | | TIMn_IDE->EXFIE |

## 21.7 TIM Register

### 21.7.1 Timer Control Register (TIMn_CON)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_CON (n=0~7) | R/W | Timer Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | SPOS | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXOE | EPWMNA | EPWMNB | INVNA | INVNB | TIMCK[2:0] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TR | DEC | EXENX | FSEL | EXENF | EXENR | CTSEL | CPRL |

<table>
<tr><th>Bit number</th><th>Bit Mnemonic</th><th>Description</th></tr>
<tr><td>19</td><td>SPOS</td><td>TIM2 Pin Mapping Control Bit@ TIM2_CON
<table>
<tr><th>Port<br>SPOS value</th><th>T2CAP/T2</th></tr>
<tr><td>SPOS=0</td><td>PA10</td></tr>
<tr><td>SPOS=1</td><td>PA12</td></tr>
</table>
TIM3 Pin Mapping Control Bit @ TIM3_CON
<table>
<tr><th>Port<br>SPOS value</th><th>T3CAP/T3</th></tr>
<tr><td>SPOS=0</td><td>PA0</td></tr>
<tr><td>SPOS=1</td><td>PA13</td></tr>
</table>
TIM7 Pin Mapping Control Bit @ TIM7_CON
<table>
<tr><th>Port<br>SPOS value</th><th>T7CAP/T7</th></tr>
<tr><td>SPOS=0</td><td>PC1</td></tr>
<tr><td>SPOS=1</td><td>PB1</td></tr>
</table></td></tr>
<tr><td>15</td><td>TXOE</td><td>Tn Pin Signal Direction Control Bit<br>0: Tn is used as clock input or I/O<br>1: Tn is used as programmable clock output</td></tr>
<tr><td>14</td><td>EPWMNA</td><td>Tn_PWMA Pin PWM Waveform Output Enable Bit<br>0: Disable<br>1: Enable</td></tr>
<tr><td>13</td><td>EPWMNB</td><td>Tn_PWMB Pin PWM Waveform Output Enable Bit<br>0: Disable<br>1: Enable</td></tr>
</table>

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 12 | INVNA | TPWMnA Waveform Output Inversion Control Bit<br>0: Normal<br>1: Waveform Output Inverted |
| 11 | INVNB | TPWMnB Waveform Output Inversion Control Bit<br>0: Normal<br>1: Waveform Output Inverted |
| 10~8 | TIMCK[2:0] | TIM Clock Frequency Prescaler Bit<br>000: $f_{TIM}/1$<br>001: $f_{TIM}/2$<br>010: $f_{TIM}/4$<br>011: $f_{TIM}/8$<br>100: $f_{TIM}/16$<br>101: $f_{TIM}/32$<br>110: $f_{TIM}/64$<br>111: $f_{TIM}/128$<br>The clock corresponding to $f_{TIM}$ may be either $f_{PCLK}$ or the input Tn. |
| 7 | TR | TIMn Start/Stop Control Bit<br>0: Stop the TIMn/TPWMn counter<br>1: Start the TIMn/TPWMn counter |
| 6 | DEC | Increment/Decrement Direction Control Bit<br>0: TIMn is an incrementing timer/counter<br>1: TIMn is an incrementing/decrementing timer/counter, and TnEX is used to select the counting direction |
| 5 | EXENX | TnEX Setting Bit, n=0<br>The function of this bit varies in different modes:<br>● Reload mode (CPRL = 0):<br>This bit controls the external event input (falling edge) on the TnEX pin for reload enable/disable control:<br>0: Ignore events on the TnEX pin.<br>1: Generate a reload when detect a falling edge on the TnEX pin.<br>● Capture mode (CPRL = 1):<br>This bit serves as the TnEX falling edge signal capture selection bit:<br>0: Ignore events on the TnEX pin.<br>1: When FSEL = 1, generate a capture when detect a falling edge on the TnEX pin, set EXIF, and capture the value of the TnCNT register into the register FCAP. |
| 4 | FSEL | Falling Edge Signal Selection Bit<br>This bit is only valid in capture mode (CPRL=1):<br>0: Generate a capture when detect a falling edge on the Tn pin,. Ignore events on the TnEX pin.<br>1: Generate a capture when detect a falling edge on the TnEX pin. Ignore events on the TnEX pin. |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 3 | EXENF | Falling Edge Signal Capture Enable Bit:<br>0: Ignore events on the Tn pin<br>1: Generate a capture when detect a falling edge on the Tn pin, set EXIF, and capture the value of the TnCNT register into the register FCAP |
| 2 | EXENR | Rising Edge Signal Capture Enable Bit:<br>0: Ignore events on the Tn pin<br>1: Generate a capture when detect a rising edge on the Tn pin, set EXIR, and capture the value of the TnCNT register into the register RCAP |
| 1 | CTSEL | Timer/Counter Selection Bit<br>0: Timer<br>1: Counter |
| 0 | CPRL | Capture/Reload Function Selection Bit<br>0: Reload function<br>1: Capture function |
| 31~20<br>18~16 | - | Reserved |

### 21.7.2 Timer Count Value Register (TIMn_CNT)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_CNT (n=0~7) | R/W | Timer Count Value Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | CNT[15:0] | TIMn count value |
| 31~16 | - | Reserved |

### 21.7.3 Timer Reload Register (TIMn_RLD)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_RLD (n=0~7) | R/W | Timer Reload Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | RLD[15:0] | A 16-bit reload can be triggered by either a timer overflow or a falling edge on the external input TnEX. When a reload is triggered, the timer automatically loads the user-programmed RLD[15:0] value into TnCNT register |
| 31~16 | - | Reserved |

### 21.7.4 Timer Flag Register (TIMn_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_STS<br>(n=0~7) | R/W | Timer Flag Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 2 | EXIF | Flag indicating the detection of a falling edge on the external event input. This bit is set by hardware and cleared by writing 1 through software.<br>0: No external event input detected<br>1: External input detected (set to 1 by hardware if EXENF = 1)<br>**Note: In capture mode, updating the TnFCAP value is not allowed before clearing this bit through software.** |
| 1 | EXIR | Flag indicating the detection of a rising edge on the external event input. This bit is set by hardware and cleared by writing 1 through software. |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 0: No external event input detected<br>1: External input detected (set to 1 by hardware if EXENF = 1)<br>**Note: In capture mode, updating the TnRCAP value is not allowed before clearing this bit through software.** |
| 0 | TIF | Timer Overflow Flag.<br>This bit is set by hardware and cleared by writing 1 through software.<br>0: No overflow (must be cleared by software).<br>1: Overflow (set to 1 by hardware if RCLK = 0 and TCLK = 0). |
| 31~3 | - | Reserved |

### 21.7.5 TnPWMA Duty Cycle Configuration Register (TIMn_PDTA)(@CPRL = 0)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_PDTA(n=1~7) | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PDT[15:0] | TPWMnA Duty Cycle Register, n=1~7.<br>The high-level width of the TPWMnA waveform is PDT[15:0] TIM clocks. |
| 31~16 | - | Reserved |

### 21.7.6 TnPWMB Duty Cycle Configuration Register (TIMn_PDTB)(@CPRL = 0)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_PDTB<br>(n=0) | R/W | TnPWMB Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |

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

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | PDT[15:0] | TPWMnB Duty Cycle Register, n=0.<br>The high-level width of the TPWMnB waveform is PDT[15:0] TIM clocks. |
| 31~16 | - | Reserved |

### 21.7.7 Rising Edge Data Capture Register (TIMn_RCAP)(@CPRL = 1)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_RCAP<br>(n=0~7) | R/W | Rising Edge Data Capture Register<br>(@CPRL = 1) | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | RCAP [15:0] | In PWM capture mode of TIMn, when the rising edge capture condition occurs, the value of the CNT counter will be saved in this register. |
| 31~16 | - | Reserved |

### 21.7.8 Falling Edge Data Capture Register (TIMn_FCAP) (@CPRL = 1)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_FCAP<br>(n=0~7) | R/W | Falling Edge Data Capture Register<br>(@CPRL = 1) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FCAP[15:8] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FCAP[7:0] | | | | | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 15~0 | FCAP [15:0] | In PWM capture mode of TIMn, when the falling edge capture condition occurs, the value of the CNT counter will be saved in this register. |
| 31~16 | - | Reserved |

### 21.7.9 TIMn Interrupt Enable And DMA Control Register (TIMn_IDE)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| TIMn_IDE<br>(n=0~7) | R/W | TIMn Interrupt Enable And DMA Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | - | - | - | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | CAPFDE | CAPRDE | TIDE | EXFIE | EXRIE | TIE | INTEN |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 6 | CAPFDE | Trigger DMA Request On Falling Edge Capture Event Enable Bit<br>0: Disable DMA request on falling edge capture event<br>1: Trigger DMA request on the occurrence of a new falling edge capture, DMA will transfer the value of the FCAP register. |
| 5 | CAPRDE | Trigger DMA Request On Rising Edge Capture Event Enable Bit<br>0: Disable DMA request on rising edge capture event<br>1: Trigger DMA request on the occurrence of a new rising edge capture, DMA will transfer the value of the RCAP register. |
| 4 | TIDE | Trigger DMA Request On Timer Overflow Event Enable Bit<br>0: Disable DMA request on timer overflow<br>1: Enable DMA request on timer overflow |
| 3 | EXFIE | External Event Input Falling Edge Interrupt Enable Bit<br>0: Disable falling edge interrupt<br>1: Enable falling edge interrupt |
| 2 | EXRIE | External Event Input Rising Edge Interrupt Enable Bit<br>0: Disable rising edge interrupt<br>1: Enable rising edge interrupt |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 1 | TIE | Timer Overflow Interrupt Enable Bit<br>0: Disable overflow interrupt<br>1: Enable overflow interrupt |
| 0 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 31~7 | - | Reserved |

### 21.7.10 TIM Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TIM0 Base Address:0x4002_0100 | | | | |
| TIM0_CON | 0x00 | R/W | Timer0 Control Register | 0x0000_0000 |
| TIM0_CNT | 0x04 | R/W | Timer0 Count Value Register | 0x0000_0000 |
| TIM0_RLD | 0x08 | R/W | Timer0 Reload Register | 0x0000_0000 |
| TIM0_STS | 0x0C | R/W | Timer0 Flag Register | 0x0000_0000 |
| TIM0_PDTB | 0x14 | R/W | TnPWMB Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM0_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM0_IDE | 0x18 | R/W | TIM0 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM1 Base Address:0x4002_0140 | | | | |
| TIM1_CON | 0x00 | R/W | Timer1 Control Register | 0x0000_0000 |
| TIM1_CNT | 0x04 | R/W | Timer1 Count Value Register | 0x0000_0000 |
| TIM1_RLD | 0x08 | R/W | Timer1 Reload Register | 0x0000_0000 |
| TIM1_STS | 0x0C | R/W | Timer1 Flag Register | 0x0000_0000 |
| TIM1_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM1_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM1_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM1_IDE | 0x18 | R/W | TIM1 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM2 Base Address:0x4002_0180 | | | | |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TIM2_CON | 0x00 | R/W | Timer2 Control Register | 0x0000_0000 |
| TIM2_CNT | 0x04 | R/W | Timer2 Count Value Register | 0x0000_0000 |
| TIM2_RLD | 0x08 | R/W | Timer2 Reload Register | 0x0000_0000 |
| TIM2_STS | 0x0C | R/W | Timer2 Flag Register | 0x0000_0000 |
| TIM2_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM2_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM2_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM2_IDE | 0x18 | R/W | TIM2 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM3 Base Address:0x4002_01C0 | | | | |
| TIM3_CON | 0x00 | R/W | Timer3 Control Register | 0x0000_0000 |
| TIM3_CNT | 0x04 | R/W | Timer3 Count Value Register | 0x0000_0000 |
| TIM3_RLD | 0x08 | R/W | Timer3 Reload Register | 0x0000_0000 |
| TIM3_STS | 0x0C | R/W | Timer3 Flag Register | 0x0000_0000 |
| TIM3_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM3_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM3_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM3_IDE | 0x18 | R/W | TIM3 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM4 Base Address:0x4002_1100 | | | | |
| TIM4_CON | 0x00 | R/W | Timer4 Control Register | 0x0000_0000 |
| TIM4_CNT | 0x04 | R/W | Timer4 Count Value Register | 0x0000_0000 |
| TIM4_RLD | 0x08 | R/W | Timer4 Reload Register | 0x0000_0000 |
| TIM4_STS | 0x0C | R/W | Timer4 Flag Register | 0x0000_0000 |
| TIM4_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM4_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM4_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TIM4_IDE | 0x18 | R/W | TIM4 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM5 Base Address:0x4002_1140 | | | | |
| TIM5_CON | 0x00 | R/W | Timer5 Control Register | 0x0000_0000 |
| TIM5_CNT | 0x04 | R/W | Timer5 Count Value Register | 0x0000_0000 |
| TIM5_RLD | 0x08 | R/W | Timer5 Reload Register | 0x0000_0000 |
| TIM5_STS | 0x0C | R/W | Timer5 Flag Register | 0x0000_0000 |
| TIM5_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM5_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM5_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM5_IDE | 0x18 | R/W | TIM5 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM6 Base Address:0x4002_1180 | | | | |
| TIM6_CON | 0x00 | R/W | Timer6 Control Register | 0x0000_0000 |
| TIM6_CNT | 0x04 | R/W | Timer6 Count Value Register | 0x0000_0000 |
| TIM6_RLD | 0x08 | R/W | Timer6 Reload Register | 0x0000_0000 |
| TIM6_STS | 0x0C | R/W | Timer6 Flag Register | 0x0000_0000 |
| TIM6_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |
| TIM6_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM6_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM6_IDE | 0x18 | R/W | TIM6 Interrupt Enable And DMA Control Register | 0x0000_0000 |
| TIM7 Base Address:0x4002_11C0 | | | | |
| TIM7_CON | 0x00 | R/W | Timer7 Control Register | 0x0000_0000 |
| TIM7_CNT | 0x04 | R/W | Timer7 Count Value Register | 0x0000_0000 |
| TIM7_RLD | 0x08 | R/W | Timer7 Reload Register | 0x0000_0000 |
| TIM7_STS | 0x0C | R/W | Timer7 Flag Register | 0x0000_0000 |
| TIM7_PDTA | 0x10 | R/W | TnPWMA Duty Cycle Configuration Register (@CPRL = 0) | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| TIM7_RCAP | 0x10 | R/W | Rising Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM7_FCAP | 0x14 | R/W | Falling Edge Data Capture Register (@CPRL = 1) | 0x0000_0000 |
| TIM7_IDE | 0x18 | R/W | TIM7 Interrupt Enable And DMA Control Register | 0x0000_0000 |

# 22 Direct Memory Access (DMA)

## 22.1 Overview

The DMA controller is designed for high-speed data transfer, allowing the movement of data from one address to another without the need for CPU intervention. Leveraging DMA for data transfer can reduce the workload on the CPU, enabling the saved CPU resources to be utilized for other applications. The DMA controller comprises 4 channels, each directly connected to dedicated hardware DMA requests. Additionally, each channel supports software triggering. The DMA controller features support for 4-level channel priority, facilitating the management of priority between DMA requests to ensure that only one DMA channel operates at any given time. It also supports both single and batch transfers, with the request source being either a software request or an interface request. Data transfer between memories is accomplished using software requests.

**Note: For a bidirectional data transfer application, two DMA channels are required to handle sending and receiving operations.**

## 22.2 Clock Source

The clock source of DMA is derived from HCLK, and the external peripheral clock of DMA is enabled through AHB_CFG.DMAEN.

## 22.3 Feature

- Support 4 independent configurable channels
- Support 4 priority levels for requests
- Support 8-bit, 16-bit, 32-bit data transfers
- Support automatic increment/decrement or fixed source and destination addresses, with data widths of byte, half-word, and word
- Support single and burst transfer modes

## 22.4 Function Description

### 22.4.1 Transmission

No transmit limitation between peripheral and memory for DMA:

| Memory-to-Memory | Memory-to-Peripheral | Peripheral-to-Memory | Peripheral-to-Periphral |
|---|---|---|---|
| No limitation | No limitation | No limitation | No limitation |

### 22.4.2 DMA Access Restriction

Users are not allowed to perform write operations on Flash or access the core through DMA. Violating these restrictions may lead to unpredictable exceptions.

### 22.4.3 Channel Priority

There are 4 priority levels can be configured through PL[1:0] registers:

- 00: Low
- 01: Medium
- 10: High
- 11: Very High

### 22.4.4 Single Transmission and Burst Transmission

The DMA controller supports single and burst data transfer types, and the request source can be a software request or an interface request while data transfer between memory is done by software requests. Single transfer means that the software or interface is ready to transfer one data (each data requires one request), while burst transfer means that the software or interface will transfer multiple data (multiple data requiring only one request).

The modes of single and burst transfer can be set through TPTYPE register (DMAn_CFG[15]).

In single transfer mode, each transfer of data requires one request. As each data is transferred, the values in the register DMAn_CNT[31:0](n=0~3) decrease by 1, the transfer of data is completed when the count in DMAn_CNT[31:0] becomes 0. In this mode, BURSIZE (DMAn_CFG[14:12]) is not used to control the size of the transferred data and its value is fixed at 1.

In burst transfer mode, DMA transfer DMAn_CNT[31:0] data with only one request. After transferring BURSIZE (DMAn_CFG[14:12]) data, the value in DMAn_CNT[31:0] is decreased by BURSIZE. The transfer of data is completed when the count in DMAn_CNT[31:0] becomes 0.

### 22.4.5 Loop Mode

The loop mode can be used to handle circular buffers and continuous data streams (such as ADC scan mode). During the loop mode transfer, the number of data to be transferred will automatically reload to the initial value set in the channel configuration phase and continue to respond to DMA requests. To stop loop transfer, the software needs to stop the generation of DMA requests by the peripheral before disabling the DMA channel (for example, exiting ADC scan mode). The software must explicitly set the DMACNT value before starting/enabling the transfer and after stopping the loop transfer.

The SC32F10T/10G series DMA controller supports normal mode and loop mode:

- When CIRC=0 (DMA channel is in non-loop mode), it will no longer accept any DMA requests after reaching the set number of data to be transferred
- When CIRC=1 (DMA channel is in loop mode), after the transfer is complete, the DMACNT of the channel will automatically reload the previously set value and wait for the next loop

Users can flexibly choose according to their actual needs.

## 22.5 DMA Interrupt

For each DMAn ( n=0~3) channel, an interrupt will be generated when "transmission complete," "half transmission," or "transmission error." Separate interrupt enable bits can be used to enhance flexibility.

| Interrupt Event | Event Flag | Interrupt Request Control Bit | Sub-Event Flag | Interrupt Enable Sub-Switch |
|---|---|---|---|---|
| DMAn transmission complete | GIF | DMAn_CFG ->INTEN | TCIF | TCIE |
| DMAn half transmission | | | HTIF | HTIE |
| DMAn transmission error | | | TEIF | TEIE |

## 22.6 DMA Register

### 22.6.1 DMAn Transmission Source Address Cache Register (DMAn_SADR)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DMAn_SADR<br>n = 0~3 | R/W | DMAn Transmission Source Address Cache Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | SADR[31:0] | DMA Transmission Source Address Cache<br>● Read:<br>■ When the channel is enabled, what is read is the internal source address working register.<br>■ When the channel is disabled, what is read is the apparent source address cache register.<br>● Update:<br>■ After each transmission, the source address working register will automatically change based on the SAINC[1:0] settings, and the width of the change is determined by TXWIDTH[1:0].<br>■ In the loop mode (SAINC = 11), the source address cache register will reload into the source address working register.<br>● Write:<br>■ The conditions for writing to the source address cache register: CHEN=1, and DMA channel has completed the transmission and stay in the IDLE state, or CHEN=0. |

### 22.6.2 DMAn Transmission Target Address Cache Register (DMAn_DADR)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DMAn_DADR<br>n = 0~3 | R/W | DMAn Transmission Target Address Cache Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | DADR[31:0] | DMA Transmission Target Address Cache<br>● Read:<br>■ When the channel is enabled, what is read is the internal target address working register.<br>■ When the channel is disabled, what is read is the apparent target address cache register.<br>● Update:<br>■ After each transmission, the target address working register will automatically change based on the DAINC[1:0] settings, and the width of the change is determined by TXWIDTH[1:0].<br>■ In the loop mode (SAINC = 11), the target address cache register will reload into the target address working register.<br>● Write:<br>■ The conditions for writing to the target address cache register: first, CHEN=1, and DMA channel has completed the transmission and stay in the IDLE state, or CHEN=0. |

### 22.6.3 DMAn Control/Configuration Register (DMAn_CFG)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DMAn_CFG<br>n = 0~3 | R/W | DMAn Control/Configuration Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | REQSRC[5:0] | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CHRQ | - | - | - | TEIE | HTIE | TCIE | INTEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TPTYPE | BURSIZE[2:0] | | | SAINC[1:0] | | DAINC[1:0] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHEN | CHRST | PAUSE | CIRC | TXWIDTH[1:0] | | PL[1:0] | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 29~24 | REQSRC[5:0] | DMA Channel Request Source Selection Bit<br>0: Disable peripheral request for the current DMA channel<br>Select the following configuration values, if the peripheral DMA request enable in the selected setting, corresponding request source will be generated:<br>2: UART0_IDE->TXDMAEN<br>3: UART0_IDE->RXDMAEN<br>4: UART1_IDE->TXDMAEN<br>5: UART1_IDE->RXDMAEN<br>12: SPI0_IDE->TXDMAEN<br>13: SPI0_IDE->RXDMAEN<br>14: SPI1_IDE->TXDMAEN<br>15: SPI1_IDE->RXDMAEN<br>20: TWI0_IDE->TXDMAEN<br>21: TWI0_IDE->RXDMAEN<br>33: TIM1_IDE->TIDE<br>34: TIM1_IDE->CAPFDE<br>35: TIM1_IDE->CAPRDE<br>36: TIM2_IDE->TIDE<br>37: TIM2_IDE->CAPFDE<br>38: TIM2_IDE->CAPRDE<br>48: TIM6_IDE->TIDE<br>49: TIM6_IDE->CAPFDE<br>50: TIM6_IDE->CAPRDE<br>59: ADCCON->DMAEN<br>60: DMA0_CFG->CHRQ<br>61: DMA1_CFG->CHRQ<br>62: DMA2_CFG->CHRQ<br>63: DMA3_CFG->CHRQ<br>Others:Disable DMA peripheral request |
| 23 | CHRQ | DMA Request Enable Bit For DMA Channel:<br>0: Disable, the current DMA channel is prohibited from serving as the request source for other DMA channels |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 1: Enable, the current DMA channel can serve as the request source for other DMA channels, meaning it can generate DMA requests. like other peripherals<br>When this bit is enabled, it allows DMA to request DMA.For example: If CHRQ =1，after DMA channell n completes data transmission, it will generate a DMA request to DMA channel m. Channel m will respond to the request and update the pre-configured parameter table to the register of channel n, thereby achieving automatic parameter updates for channel n. |
| 19 | TEIE | DMA Transmission Error Interrupt Enable Bit<br>0: Disable DMA transmission error interrupt<br>1: Enable DMA transmission error interrupt |
| 18 | HTIE | DMA Half Transmission Interrupt Enable Bit<br>0: Disable DMA half transmission interrupt<br>1: Enable DMA half transmission interrupt |
| 17 | TCIE | DMA Transmission Complete Interrupt Enable Bit<br>0: Disable DMA transmission complete interrupt<br>1: Enable DMA transmission complete interrupt |
| 16 | INTEN | Interrupt Request CPU Enable Control Bit<br>0: Disable interrupt request<br>1: Enable interrupt request |
| 15 | TPTYPE | DMA Channel Transmission Type Selection Bit<br>0: Single transmission<br>1: Burst transmission. In burst transmission mode, The DMA controller moves DMACNT data with just one request. Once the channel responds to this request, the data will be transferred in a burst mode, meaning it moves in units of BURSIZE until DMACNT decrements to 0. The data processing for a single burst transfer is considered complete only when DMACNT reaches zero. |
| 14~12 | BURSIZE[2:0] | In Burst transmission, based on the definition of Burst transmission mode, Burst size can be selected as:<br>000: 128<br>001: 64<br>010: 32<br>011: 16<br>100: 8<br>101: 4<br>110: 2<br>111: 1 |
| 11~10 | SAINC[1:0] | DMA Channel Transmission Source Address Increment/Decrement Mode Configuration Bit<br>00: No increment (Fixed address mode)<br>01: Increment mode |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 10: Decrement mode<br>11: Incremental circular mode (Refer to the DMA transmission source address cache register)<br>The values of SAINC[1:0] can be modified freely and take effect immediately when the channel is disabled. When the channel is enabled, the modified values will take effect during the reload in circular mode. |
| 9~8 | DAINC[1:0] | DMA Channel Transmission Target Address Increment/Decrement Mode Configuration Bit<br>00: No increment (Fixed address mode)<br>01: Increment mode<br>10: Decrement mode<br>11: Incremental circular mode (Refer to the DMA transmission target address cache register)<br>The values of DAINC [1:0] can be modified freely and take effect immediately when the channel is disabled. When the channel is enabled, the modified values will take effect during the reload in circular mode. |
| 7 | CHEN | DMA Channel Enable Bit<br>0: Disable DMA channel<br>1: Enable DMA channel |
| 6 | CHRST | DMA Channel Reset Control Bit<br>This bit is used to control the reset of DMA channel.<br>0: Invalid<br>1: Reset the current DMA channel. At this point, CHEN for the current DMA channel is disabled, the interrupt flag is cleared, and the values of other registers remain unchanged. |
| 5 | PAUSE | DMA Channel Transfer Pause Control Bit<br>0: Invalid<br>1: Pause the current DMA channel. At this point, CHEN for the current DMA channel is disabled, and the state machine returns to state=1 after completing the current read/write cycle. The internal register values (source/destination address register, counters) are maintained. When CHEN for the current DMA channel is enabled again, the channel will resume the previous transfer.<br>**Note: To resume the transfer after pausing, both CHEN and PAUSE need to be assigned values: CHEN=1, PAUSE=0.** |
| 4 | CIRC | DMA Channel Loop Mode Enable Bit<br>0: The channel is not in loop mode. When the set number of data to be transferred is reached, the DMACNT for that channel will remain at zero. |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| | | 1: The channel is in loop mode. After the transfer is complete, the DMACNT for that channel will automatically reload the previously set value.<br>Loop mode can be used for handling circular buffers and continuous data streams (such as ADC scan mode). During the loop mode transfer, the number of data to be transferred will automatically reload to the initial value set during the channel configuration phase, and the channel will continue to respond to DMA requests. To stop the loop transfer, software needs to stop the peripheral from generating DMA requests before disabling the DMA channel (for example, exiting ADC scan mode). Software must explicitly set the DMACNT value before starting/enabling the transfer and after stopping the loop transfer. |
| 3~2 | TXWIDTH[1:0] | DMA Channel Transmission Width Selection Bit<br>Choose the data width of the source and target addresses for each transmission of the current DMA channel:<br>00: 8bit<br>01: 16bit<br>10: 32bit<br>11: 32bit<br>The values of TXWIDTH[1:0] can be freely modified and take effect immediately when the channel is disabled. When the channel is enable, the modified values will take effect during the reload in loop mode. |
| 1~0 | PL[1:0] | DMA Channel Priority Setting Bit<br>When DMA has a channel in operation, and other channels also receive requests but are pending, priority arbitration will be initiated once the currently active channel completes its operation.<br>00: Low<br>01: Medium<br>10: High<br>11: Very High<br>**Note: For equal priority configurations, lower channel numbers have higher priority.** |
| 31~30<br>22~20 | - | Reserved |

### 22.6.4 DMAn Counter Cache Register (DMAn_CNT)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DMAn_CNT<br>n = 0~3 | R/W | DMAn Counter Cache Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| DMACNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMACNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DMACNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMACNT[7:0] | | | | | | | |

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 31~0 | DMACNT[31:0] | DMA Channel Counter Cache Register<br>● Write:<br>■ The value of DMACNT refers to the remaining transfer count for the current DMA channel.<br>■ Each DMA channel has an internal “working counter” that decrements by the TXWIDTH units after each transmission:<br>◆ When CIRC=0 (DMA channel is not in loop mode), the 'working counter' will stop accepting any further DMA requests after decrementing to 0.<br>◆ When CIRC=1 (DMA channel is in loop mode), after the “working counter” decrements to 0, it will reload the value of DMACNT into the “working counter” and wait for the next loop.<br>● Read:<br>■ When the channel is disabled, reading returns the value of DMACNT.<br>■ When the channel is enabled, reading returns the real-time data of the internal “working counter”. |

### 22.6.5 DMAn Status Register (DMAn_STS)

| Register | R/W | Description | Reset Value |
|---|---|---|---|
| DMAn_STS<br>n = 0~3 | R/W | DMAn Status Register | 0x0000_0000 |

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

| Bit number | Bit Mnemonic | Description |
|---|---|---|
| 8 | SWREQ | DMA Channel Software Request Trigger Bit<br>When this bit is written to 1, the current DMA channel will remain pending software requests until the channel responds, and this bit is automatically cleared by hardware. |
| 7~4 | STATUS[3:0] | DMA Channel Status Bit<br>0000: Idle<br>0001: Write to source address<br>0010: Read source address data and write to target address<br>0011: Write to target address data<br>0100: Reserved<br>0101: Pending (When a channel is busy, requests from other channels are suspended.)<br>0110: Pause pending (In burst transmission mode, after writing PAUSE to 1)<br>0111: Burst transmission in progress<br>1000:Burst transmission stopped( PAUSE is enabled, DMACNT counts to 0, or bursize counts to 0, will enter this state.) |
| 3 | TEIF | DMA Transmission Error Interrupt Flag<br>When DMA reads or writes to an undefined address, TEIF will be set to 1 by the hardware.<br>Writing 1 clears the bit to zero. |
| 2 | HTIF | DMA HalfTransmission Interrupt Flag<br>When the counter value of DMACNT reaches DMACNT/2, HTIF will be set to 1 by the hardware.<br>Writing 1 clears the bit to zero. |
| 1 | TCIF | DMA Transmission Complete Interrupt Flag<br>When the counter value of DMACNT reaches 0, TCIF will be set to 1 by the hardware.<br>Writing 1 clears the bit to zero. |
| 0 | GIF | DMA Channel Global Interrupt Flag<br>0: The current DMA channel has no interrupt generated.<br>1: The current DMA channel has generated an interrupt: transmission error, half-transmission, or transmission complete. |
| 31~9 | - | Reserved |

### 22.6.6 DMA Register Mapping

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMA0 Base Address:0x4001_0800 | | | | |
| DMA0_SADR | 0x00 | R/W | DMA0 Transmission Source Address Cache Register | 0x0000_0000 |

| Register | Offset Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| DMA0_DADR | 0x04 | R/W | DMA0 Transmission Target Address Cache Register | 0x0000_0000 |
| DMA0_CFG | 0x08 | R/W | DMA0 Control/Configuration Register | 0x0000_0000 |
| DMA0_CNT | 0x0C | R/W | DMA0 Counter Cache Register | 0x0000_0000 |
| DMA0_STS | 0x10 | R/W | DMA0 Status Register | 0x0000_0000 |
| DMA1 Base Address:0x4001_0840 | | | | |
| DMA1_SADR | 0x00 | R/W | DMA1 Transmission Source Address Cache Register | 0x0000_0000 |
| DMA1_DADR | 0x04 | R/W | DMA1 Transmission Target Address Cache Register | 0x0000_0000 |
| DMA1_CFG | 0x08 | R/W | DMA1 Control/Configuration Register | 0x0000_0000 |
| DMA1_CNT | 0x0C | R/W | DMA1 Counter Cache Register | 0x0000_0000 |
| DMA1_STS | 0x10 | R/W | DMA1 Status Register | 0x0000_0000 |
| DMA2 Base Address:0x4001_0880 | | | | |
| DMA2_SADR | 0x00 | R/W | DMA2 Transmission Source Address Cache Register | 0x0000_0000 |
| DMA2_DADR | 0x04 | R/W | DMA2 Transmission Target Address Cache Register | 0x0000_0000 |
| DMA2_CFG | 0x08 | R/W | DMA2 Control/Configuration Register | 0x0000_0000 |
| DMA2_CNT | 0x0C | R/W | DMA2 Counter Cache Register | 0x0000_0000 |
| DMA2_STS | 0x10 | R/W | DMA2 Status Register | 0x0000_0000 |
| DMA3 Base Address:0x4001_08C0 | | | | |
| DMA3_SADR | 0x00 | R/W | DMA3 Transmission Source Address Cache Register | 0x0000_0000 |
| DMA3_DADR | 0x04 | R/W | DMA3 Transmission Target Address Cache Register | 0x0000_0000 |
| DMA3_CFG | 0x08 | R/W | DMA3 Control/Configuration Register | 0x0000_0000 |
| DMA3_CNT | 0x0C | R/W | DMA3 Counter Cache Register | 0x0000_0000 |
| DMA3_STS | 0x10 | R/W | DMA3 Status Register | 0x0000_0000 |

# 23 SysTick

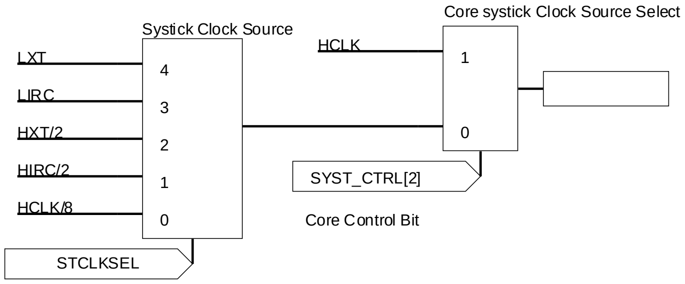
SysTick is a simple, 24-bit, writable-clear, decrementing automatic reload counter with a flexible control mechanism. This counter can be used as a tick timer for a Real-Time Operating System (RTOS) or as a simple counter.

## 23.1 Clock Source

SysTick (Cortex®-M0+ Core System Timer) has internal clock source and external clock source:

- Internal clock source: CPU Clock
- 5 external clock sources

SysTick clock sourse diagram is as follow:



## 23.2 SysTick Calibration Register Default Value

The calibration value for the SysTick Calibration Register is set as follows:

- If the default clock after power-up is $f_{HCLK}$/n (MHz), (n is the default division factor after power-up, and HIRC is the default clock source after power-up).
- Then the SysTick calibration initial value is set to 1000*($f_{HCLK}$/n), this ensures that a default 1ms time base can be generated.

# 24 Revision History

| Version | Notes | Date |
|---|---|---|
| V0.1 | Initial Release | 2024.01.25 |

# 25 Important Notice

Shenzhen SinOne Microelectronics Co., Ltd. (hereinafter referred to as SinOne) reserves the right to change, correct, enhance, modify and improve SinOne products, documents or services at any time without prior notice. SinOne considers the information provided to be accurate and reliable. The information in this document will be used in January 2024. In the actual production design, please refer to the latest data manual of each product and other relevant materials.