# CX32L003

# ARM® Cortex® -M0+ 32-bit Microcontroller User Reference Manual

Table of contents

## graph index

# table index

# 1 Introduction

CX32L003 is an ultra-low power consumption, Low Pin Count and wide voltage operating range embedded 32-bit ARM® Cortex® -M0+ core (2.5V ~ 5.5V) microcontroller, the highest can run at 24MHz, built-in 32K/64K bytes of embedded Flash, 4K bytes of SRAM, integrated 12-bit 1Msps high-precision SAR ADC, RTC, comparator, multi-channel UART, SPI, I2C and PWM, etc. The rich peripheral interface has the characteristics of high integration, high anti-interference and high reliability.

CX32L003 series has wide voltage operating range, low power consumption, low standby current, highly integrated peripherals, high operating efficiency, fast wake-up Wake-up and cost-effective advantages, widely applicable to the following applications:

Small household appliances, chargers, remote controls, electronic cigarettes, gas alarms, digital displays, thermostats, recorders, motor drives, smart doors locks, smart sensors, smart homes, and smart cities.

| | CX32L003F6 | CX32L003F8 |
|---|---|---|
| pin count | 20 | |
| GPIO general purpose pin | 16 | |
| External Interrupt | 16 | |
| Advanced Timer (TIM1) | 1 | |
| General purpose timer (TIM2) | 1 | |
| Timer Array (PCA) | 1 | |
| TIM10/11 | 2 | |
| Number of A/D channels | 7 | |
| Flash (K bytes) | 32 | 64 |
| SRAM (K bytes) | 4 | |
| UART | 2 | |
| LPUART | 1 | |
| SPI | 1 | |
| I2C | 1 | |
| IWDG | 1 | |
| WWDG | 1 | |
| 1-WIRE | 1 | |
| CRC16 | 1 | |
| buzzer | 1 | |
| AWK | 1 | |
| RTC | 1 | |
| LVD/VC | support | |
| CPU frequency | ARM® Cortex® -M0+ 24MHz(Maximum) | |
| voltage range | 2.5~5.5V | |
| temperature range | -40~85ÿ | |
| flash protection | support | |
| encapsulation | TSSOP20, QFN20 | |

# 2 Product Features

ÿ Kernel

ÿ ARM® Cortex® -M0+ core, running up to 24MHz

ÿOne 24-bit system timer

ÿSupport low power sleep mode

ÿSingle -cycle 32-bit hardware multiplier

ÿMemory _

ÿ 32K/64K bytes embedded Flash, with erase and write protection function

ÿ 4K bytes SRAM

ÿ Clock and Power

ÿ 4 selectable clock sources

**-External** 4MHz~24MHz high-speed crystal oscillator

**-External** 32.768KHz crystal oscillator

**-Internal** 4MHz~24MHz high speed clock

**-Internal** low speed 38.4KHz/32.768KHz clock

**-** Support hardware clock monitoring

ÿPower Management

**-Two** low power consumption modes: Sleep, Deep Sleep Mode

**-** Low voltage detection, configurable as interrupt or reset

ÿ interrupt

ÿ Nested Vectored Interrupt Controller (NVIC) for controlling 32 interrupts source, each interrupt source can be set to 4 priority levels

ÿSupport Serial Debug (SWD) with 2 watchpoints/4 breakpoints

ÿGeneral purpose I/O pins

ÿ16 I/Os in 20-Pin package

ÿCommunication interface

ÿ UART0-UART1 standard communication interface

ÿUltra- low-power UART supporting low-speed clock

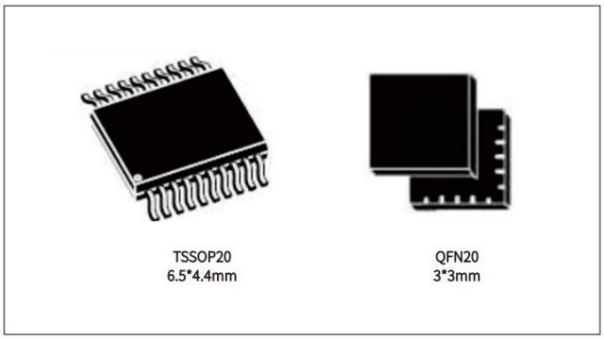
ÿ SPI standard communication interface, up to 8Mbps

ÿ I2C standard communication interface, master mode supports up to 1Mbps, slave The mode supports up to 800Kbps

ÿ One-Wire communication interface

ÿ Buzzer Frequency Generator

ÿCan generate a 1KHz, 2KHz, 4KHz buzzer signal



ÿTimer /Counter

ÿ 1x16-bit advanced control timer: with 4-channel PWM output/input capture, supports 3 complementary outputs, and dead-time generation and emergency stop stop function

ÿ 1x16-bit general-purpose timer, supports 4 comparison output/input capture Acquisition, PWM output

ÿ 1x16-bit programmable timer array, supports 5-way input capture/comparison Comparing output, PWM output

ÿ 2x16/32-bit basic timer/counter

ÿ 1x16-bit low-power timer

ÿAuto wake-up timer

ÿSystem window watchdog and independent watchdog timer

ÿRTC _

ÿSupport RTC counting (second/minute/hour) and perpetual calendar function (day/month/year)

ÿSupport alarm function register (second/minute/hour/day/month/year)

ÿSupport RTC to wake up the system from Deep Sleep mode

ÿ ADC

ÿ 7-channel 12-bit 1Msps sampling rate, 12-bit SAR ADC

ÿVoltage Comparator (VC) / Low Voltage Detector (LVD)

ÿHardware CRC-16 Module

ÿWorking Conditions

ÿWide voltage operating range 2.5V to 5.5V

ÿWide operating frequency up to 24MHz

ÿOperating temperature: -40ÿ to +85ÿ

ÿ 16-byte chip unique ID (UID) ÿ
Development tools

ÿFull -featured embedded debugging solution

ÿIn -system programming (ISP programming) program

ÿ Package type: TSSOP20, QFN20

# 3 Overview of Product Functions

In the following chapters, a brief overview of the functions and peripheral features of the CX32L003 series will be given.

## 3.1 32- bit Cortex® -M0+ core

ARM® Cortex® -M0+ processor is the latest generation of embedded 32-bit RISC processor, which has a small number of pins and low power consumption, and can

Provides a low-cost platform that meets the needs of MCU implementations while providing superior computing performance and advanced interrupt system response.

The Cortex® -M0+ processor fully supports Keil, IAR and other debuggers, including a hardware debugging circuit that supports 2-wire SWD debugging

Test interface.

Cortex® -M0+ features:

| instruction | Thumb / Thumb-2 2-stage |
|---|---|
| set pipeline | pipeline |
| CoreMark/MHz | 2.46 |
| DMIPS/MHz | 0.95 |
| Interrupt | 32 interrupt sources |
| Interrupt Priority | Configurable 4-level interrupt priority |
| Enhanced Instructions | Single-cycle 32-bit multiplier |
| debug interface | Support SWD 2-wire debug interface, support 4 hard interrupts (break point) and 2 watch points (watch point) |

## 3.2 Memory

### 3.2.1 Embedded flash memory (Flash)

Embedded flash memory for storing programs and data. Built-in fully integrated Flash controller, no need for external high voltage input, fully integrated

The built-in circuit generates high voltage for programming and supports ISP function.

ÿ CX32L003F8 series supports up to 64K bytes

ÿ CX32L003F6 series supports a maximum of 32K bytes

### 3.2.2 Built-in SRAM

4K bytes of built-in SRAM.

## 3.3 Clock system

An external high-speed crystal oscillator HXT with a frequency of 4M~24MHz.

An external low-speed crystal oscillator LXT with a frequency of 32.768KHz.

An external high-speed crystal oscillator HIRC with a frequency of 4M~24MHz.

An internal low-speed clock LIRC with a frequency of 32.768KHz/38.4KHz.

## 3.4 Working mode

CX32L003 supports 3 working modes:

1. Running mode Active: CPU running, peripheral function modules running.
2. Sleep mode: The CPU stops running, and the peripheral function modules run.
3. Deep sleep mode Deep Sleep: The CPU stops running, the main system clock is turned off, and the low-power function modules run.

Which working mode to run can be selected by software. In sleep mode, the CPU clock is turned off, and other parts can still work. The CPU can be woken up by an interrupt. In deep sleep mode, the main clock of the system is turned off, most of the modules stop working, and the system works On the built-in 38.4KHz/32.768KHz built-in low-speed clock, it can be woken up by RTC interrupt, AWK interrupt or external interrupt chip. In normal working mode, you can choose to work in frequency division mode or stop the clock of some unused modules to achieve power consumption and flexible switching between performance.

## 3.5 Interrupt Controller (NVIC)

The Cortex®-M0+ processor has a built-in nested vectored interrupt controller (NVIC), which supports up to 32 interrupt request (IRQ) inputs, with four

Each interrupt priority can handle complex logic, capable of real-time control and interrupt processing.

For details, please refer to "ARM® Cortex®-M0+ Technical Reference Manual" and "ARM® v6-M Architecture Reference Manual".

32 interrupt sources, as shown in Table 3-1 interrupt sources:

Table 3-1 Interrupt sources

| external interrupt number (IRQ#) 0 | interrupt source | Introduction | Sleep mode wake up | Wake up from Deep Sleep mode | vector address |
|---|---|---|---|---|---|
| | GPIO_PA | GPIOA interrupt | Y | Y | 0x0000 0040 |
| 1 | GPIO_PB | GPIOB interrupt | Y | Y | 0x0000 0044 |
| 2 | GPIO_PC | GPIOC interrupt | Y | Y | 0x0000 0048 |
| 3 | GPIO_PD | GPIOD interrupt | Y | Y | 0x0000 004C |
| 4 | Flash | Flash interrupt | N | N | 0x0000 0050 |
| 5 | reserve | - | - | - | 0x0000 0054 |
| 6 | UART0 | UART0 interrupt | Y | N | 0x0000 0058 |
| 7 | UART1 | UART1 interrupt | Y | N | 0x0000 005C |
| 8 | LPUART | LPUART interrupt | Y | Y | 0x0000 0060 |
| 9 | reserve | - | - | - | 0x0000 0064 |
| 10 | SPI | SPI interrupt | Y | N | 0x0000 0068 |
| 11 | reserve | - | - | - | 0x0000 006C |
| 12 | I2C | I2C interrupt | Y | N | 0x0000 0070 |
| 13 | reserve | - | - | - | 0x0000 006C |
| 14 | TIM10 | TIM10 interrupt | Y | N | 0x0000 0078 |
| 15 | TIM11 | TIM11 interrupt | Y | N | 0x0000 007C |
| 16 | LPTIM | LPTIM interrupt | Y | Y | 0x0000 0080 |
| 17 | reserve | - | - | - | 0x0000 007C |
| 18 | TIM1 | TIM1 interrupt | Y | N | 0x0000 0088 |
| 19 | TIM2 | TIM2 interrupt | Y | N | 0x0000 008C |
| 20 | reserve | - | - | - | 0x0000 0088 |
| twenty one | PCA | PCA interrupt | Y | N | 0x0000 0094 |
| twenty two | WWDG | WWDG Interruption | Y | N | 0x0000 0098 |
| twenty three | IWDG | IWDG Interruption | Y | Y | 0x0000 009C |
| twenty four | ADC | ADC interrupt | Y | N | 0x0000 00A0 |
| 25 | LVD | LVD interrupt | Y | Y | 0x0000 00A4 |
| 26 | VC | VC interrupt | Y | Y | 0x0000 00A8 |
| 27 | reserve | - | - | - | 0x0000 00A4 |
| 28 | AWK | AWK interrupt | Y | Y | 0x0000 00B0 |
| 29 | OWIRE | 1-WIRE interrupt | Y | N | 0x0000 00B4 |
| 30 | RTC | RTC interrupt | Y | Y | 0x0000 00B8 |
| 31 | CLKTRIM | CLKTRIM Interrupt Y | | Y Note | 0x0000 00BC |

Note: It can only wake up when the internal low-speed monitoring external low-speed clock function is selected

## 3.6 Reset Controller

This product has 9 reset signal sources, each reset signal can make the CPU run again, most of the registers will be reset bit, the program counter PC will be reset to point to the reset address (0x0000 0000).

| serial number | Interrupt |
|---|---|
| 1 | source power-on/power-down reset |
| 2 | External Reset Pin Reset |
| 3 | IWDG reset |
| 4 | WWDG reset |
| 5 | System software reset |
| 6 | Brown-out (LVD) reset |
| 7 | LOCKUP reset |
| 8 | Register CPURST Reset |
| 9 | Register MCURST reset |

## 3.7 General-purpose I/O port (GPIO)

Up to 16 GPIO ports can be provided, some of which are multiplexed with analog ports. Each port consists of independent control register bits to control. Supports edge-triggered interrupts and level-triggered interrupts, which can wake up the MCU from various power consumption modes to work mode. support Push-Pull CMOS push-pull output, Open-Drain open-drain output. Built-in pull-up resistor, pull-down resistor, with Schmitt trigger input filter function. The output drive capability is configurable and supports a maximum current drive capability of 12mA. 16 general-purpose IOs can support external Department asynchronous interrupt.

## 3.8 Timers and Watchdog

CX32L003 products include 1 advanced control timer, 1 general-purpose timer, 1 programmable counter array, 2 basic timing timer, 1 low power base timer, 1 system window watchdog timer, 1 independent watchdog timer and 1 system tick (SysTick) timer.

The following table compares the functionality of advanced control timers, general-purpose timers, and basic timers:

Table 3-2 Timer characteristics table

| Timer class<br>type | name | counter bit<br>Width | Prescaler coefficient | Counting direction | PWM<br>output | capture/<br>compare<br>channel | complementary<br>output |
|---|---|---|---|---|---|---|---|
| advanced | TIM1 | 16 bits | 1/2/4/8/16/64/256/1024 | increment, decrement, descend<br>Increment/ | have | 4 | 3 pairs |
| universal | TIM2 | 16-bit | 1/2/4/8/16/64/256/1024 | Decrement Increment, Decrement, Decrement<br>increase/decrease | have | 4 | none |
| programmable meter<br>PCA Low Power | LPTIM | 16-bit | 2/4/8/16/32 Counter Array | increment | have | 5 | none |
| 16-bit None | | | | increment | no | no no no | |
| Base | TIM10 | 16/32 bit | 1/2/4/8/16/32/64/128 increment | | no | no no no | |
| | TIM11 | 16/32 bit | 1/2/4/8/16/32/64/128 increment | | no | no no no | |

### 3.8.1 Advanced Control Timer (TIM1)

1 high-level control timer (TIM1) can be regarded as a three-phase PWM generator distributed to 6 channels, it has a dead-band insertion Complementary PWM output, can also be regarded as a complete general-purpose timer. Four independent channels can be used for:

ÿ Input Capture

ÿ output compare

ÿ Generate PWM (edge or center aligned mode)

ÿWhen the one-shot pulse output is configured as a 16-bit standard timer, it has the same function as the TIMx timer. Configured as 16-bit

When used as a PWM generator, it has full modulation capability (0~100%).

In debug mode, the counters can be frozen while the PWM outputs are disabled, cutting off the switches controlled by these outputs.

Many functions are the same as the general TIM timer, and the internal structure is also the same, so the advanced control timer can be linked through the timer The function cooperates with other TIM timers to provide synchronization or event chaining functions.

### 3.8.2 General purpose timer (TIM2)

The general-purpose timer (TIM2) has a 16-bit autoload up/down counter, a 16-bit prescaler, and 4 independent channel channels, each channel can be used for input capture, output compare, PWM and single pulse mode output, they can also be linked by timer Functions work in conjunction with advanced control timers to provide synchronization or event chaining capabilities. In debug mode, the counters can be frozen. Either Standard timers can be used to generate PWM outputs.

### 3.8.3 Programmable Counter Array (PCA)

PCA (Programmable Counter Array Programmable Counter Array) supports up to five 16-bit capture/comparison modules. It should be The hour/counter can be used as a general-purpose clock count/event counter with capture/compare functions. Each channel of the PCA can be Row independent programming, providing input capture/output compare or pulse width modulation.

### 3.8.4 Low Power Timer (LPTIM)

The low-power timer is an asynchronous 16-bit selectable timer. After the system clock is turned off, it can still pass through the internal low-speed LIRC or External low-speed crystal oscillator for timing/counting. The system can be woken up in low power mode by interrupt.

### 3.8.5 Basic timer (TIM10/TIM11)

The base timer consists of two 16/32-bit selectable timers TIM10/TIM11. The functions of TIM10/TIM11 are exactly the same, both are synchronous Hour/counter, you can choose to work in reload mode and non-reload mode. TIM10/TIM11 can count external pulses or Implement system timing.

### 3.8.6 Independent Watchdog (IWDG)

The independent watchdog is a 20-bit down counter. It is clocked by an internal independent LIRC; since the internal LIRC is independent of the main clock so it works in shutdown and standby modes. It can be used either as a watchdog to reset the device if a problem occurs, or as a Used as a free-running timer to provide timeout management for applications. Through the option byte, it can be configured in hardware or software place. In debug mode, the counters can be frozen.

### 3.8.7 System Window Watchdog (WWDG)

The system window watchdog is based on an 8-bit down-counter and supports 20-bit prescaler, which is provided by the APB clock (PCLK) when operating bell. It can be used as a watchdog to reset the device in the event of a system problem, it has an early warning interrupt function, and the counter Can be frozen in debug mode.

### 3.8.8 SysTick timer (SYST)

This timer is dedicated to real-time operating systems, but can also be used as a standard down counter. It has the following properties:

ÿ 24-bit down counter

ÿAutomatic reload function

ÿWhen the counter counts to 0, a maskable system interrupt is generated

ÿProgrammable clock source (HCLK or HCLK/4)

## 3.9 Real Time Clock (RTC)

ÿSupport RTC counting (seconds, minutes, hours) and perpetual calendar function (day, month, year)

ÿ Support alarm register (second, minute, hour, day, month, year)

ÿ RTC can wake up the system from Sleep mode

## 3.10 Universal Asynchronous Transceiver (UART0/UART1)

2-way Universal Asynchronous Receiver/Transmitter

## 3.11 Low Power Universal Asynchronous Transceiver (LPUART)

1 Low Power Universal Asynchronous Transceiver (Low Power Universal Asynchronous Receiver/Transmitter)

## 3.12 Serial Peripheral Interface (SPI)

1 serial peripheral interface (Serial Peripheral Interface), supports master-slave mode.

## 3.13 I2C interface (I2C)

1-way I2C interface, support master-slave mode. Using serial synchronous clock, it can realize data transmission between devices at different rates, serial The maximum speed of 8-bit bidirectional data transmission can reach 1Mbps.

## 3.14 One-Wire Interface (OWIRE)

Support One-Wire bus protocol.

## 3.15 Buzzer (BEEP)

The buzzer module can generate a 1KHz, 2KHz, 4KHz buzzer signal on the BEEP pin to drive an external buzzer device.

2 basic timers TIM10/TIM11 and 1 LPTIM can be multiplexed and output to provide programmable driving frequency for Buzzer.

Complementary output can be supported without additional triode.

## 3.16 Self-wake-up timer (AWK)

AWK is used to provide an internal wake-up time reference when the MCU enters a low-power mode. The time base clock is determined by the internal The low-speed RC oscillator clock (LIRC) or the prescaled HXT crystal clock is provided.

## 3.17 Clock Trim/Monitor Module (CLKTRIM)

The built-in clock calibration circuit can calibrate the internal RC clock through an external precise crystal oscillator clock, or use the internal RC clock to check Check whether the external crystal oscillator clock is working properly.

## 3.18 Unique ID number (UID)

Each chip has a unique 16-byte device identification number when it leaves the factory, including waferlot information and chip coordinate information. ID Address 0x180000F0-0x180000FF.

## 3.19 Cyclic Redundancy Check Calculation Unit (CRC)

Complies with the polynomial $F(x)=X^{16} + X^{12} + X^{5} + 1$ given in ISO/IEC13239 .

## 3.20 Analog/Digital Converter (ADC)

Monotonic non-missing code 12-bit successive approximation analog-to-digital converter, when working at 16MHz ADC clock, the sampling rate reaches 1Msps. ginseng The test voltage can choose the power supply voltage. 7 external channels, can realize single, scan, cycle conversion. In scan/cycle mode, automatic Perform conversions on a selected set of analog inputs.

ÿInput voltage range: 0 to VDD

ÿ Conversion cycle: 16/20 clock cycles

ÿ ADC sampling can be triggered from external terminals, internal TIM1, TIM2, TIM10/TIM11, VC and other modules

ÿSample complete (EOC) interrupt

## 3.21 Low Voltage Detector (LVD)

Detect the chip power supply voltage or chip pin voltage. 8 levels of voltage monitoring value (2.5-4.4V). Can be generated according to rising/falling edge Asynchronous interrupt or reset. With hardware hysteresis circuit and configurable software anti-shake function.

## 3.22 Voltage comparator (VC)

Chip pin voltage monitoring/comparison circuit. 3 configurable positive/negative external input channels; 1 internal BGR 2.5V reference voltage. VC output can be used for timer TIM1, TIM10/TIM11, LPTimer and programmable count array PCA capture, gating, external counting Number used. Asynchronous interrupt can be generated based on rising/falling edge to wake up MCU from low power mode. Configurable software anti-shake.

## 3.23 Embedded Debugging System

The embedded debugging solution provides a full-featured real-time debugger, and cooperates with standard mature Keil/IAR and other debugging and development software. support 4 hard breakpoints and multiple soft breakpoints.

## 3.24 Encrypted Embedded Debug Support (DBG)

The encrypted embedded debugging solution provides a full-featured real-time debugger, see the relevant chapters of the user manual for details.

# 4 System and Memory Overview 4.1

## System Architecture Diagram

Main system composition:

ÿ 1 AHB bus system Master:

- Cortex® -M0+ core

ÿ 6 AHB bus slaves

-Internal SRAM

-Internal Flash

- AHB to APB Bridge, including all peripherals of APB interface

- GPIO interface

- RCC module

- AHB interface modules such as CRC

The module block diagram of the system is shown in Figure 4-1:

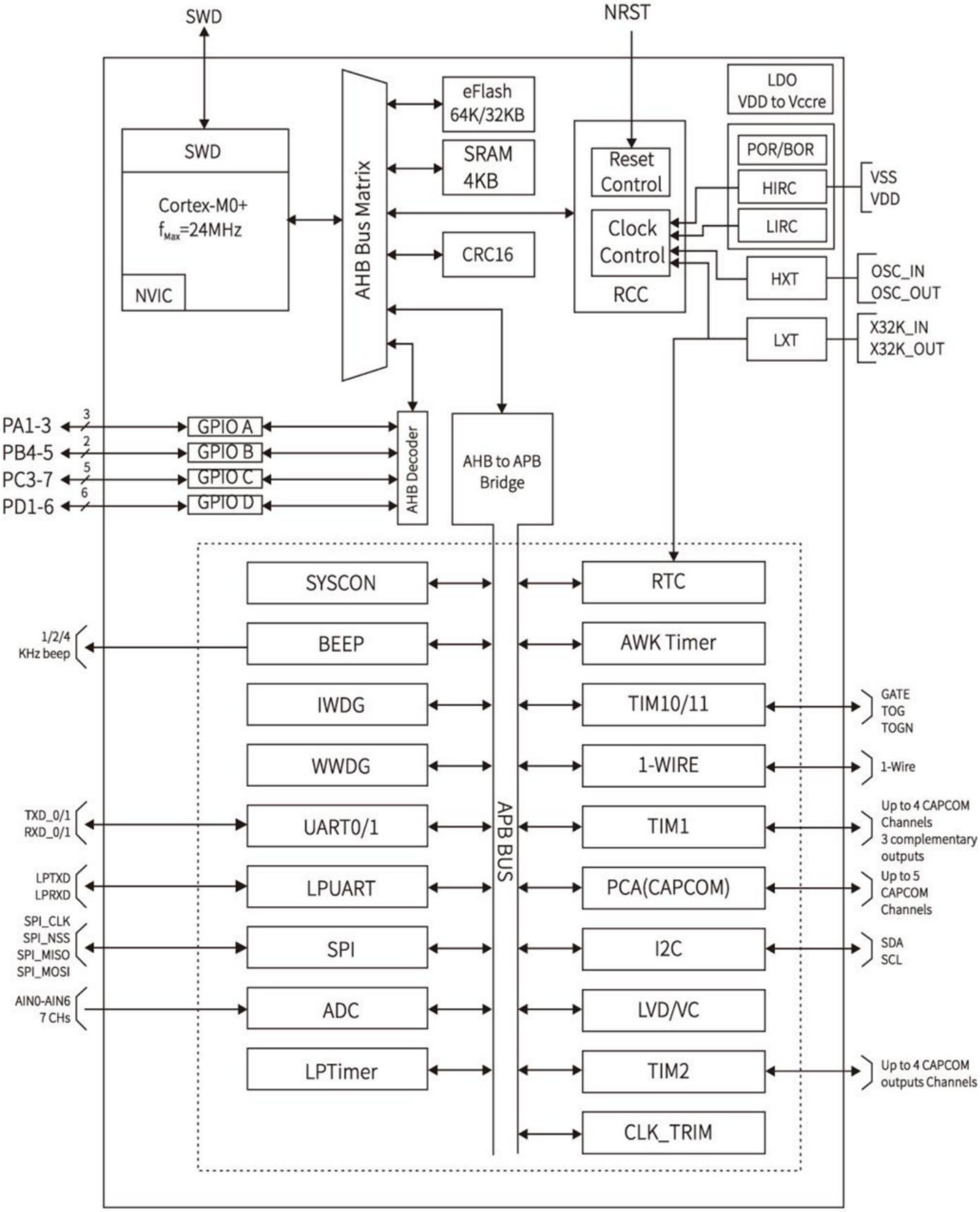


Figure 4-1 System block diagram

## 4.2 Memory Mapping

The total address space of the system is 4GB, including program storage space, data storage space, peripheral module registers, I/O ports, etc.

The data uses the small end point format, that is, the high byte of the data is stored in the high address of the memory, and the low byte of the data is stored in the low address of the memory address. The division of the entire system address space is shown in Figure 4-2:

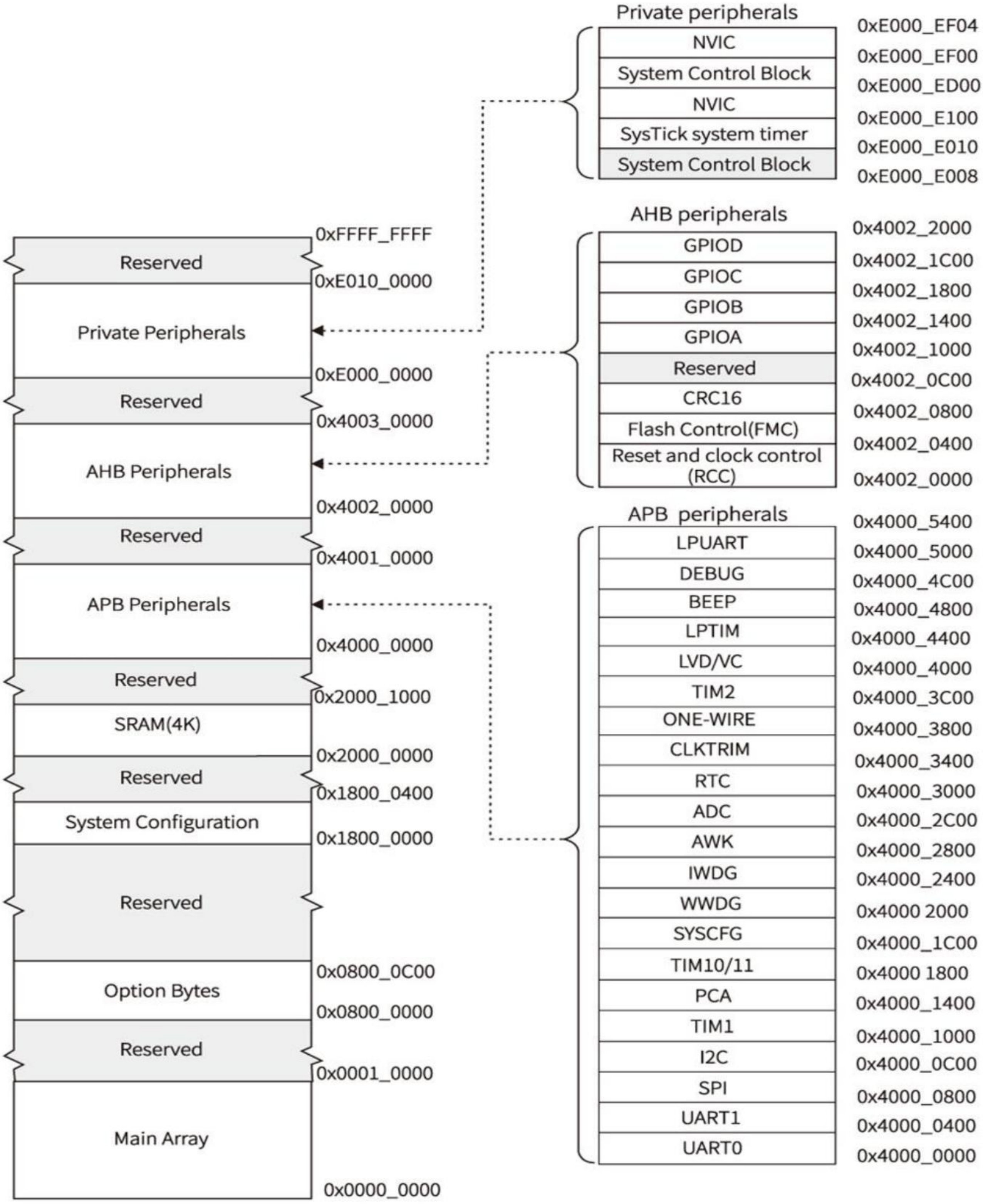


Figure 4-2 Memory Map

## 4.3 Storage space and module address

Table 4-1 below shows the address space and boundary information of each module contained in the CX32L003 device.

Table 4-1 CX32L003 memory map and peripheral register addressing

| bus | boundary address | Space size (Bytes) | module |
|---|---|---|---|
| | 0xE000_0000 - 0xE00F_FFFF | 1M | Coretex-M0+ peripheral |
| | 0x4003_0000 - 0xDFFF_FFFF | | reserve |
| AHB | 0x4002_1000 - 0x4002_1FFF | 1K | GPIOD |
| | 0x4002_1000 - 0x4002_1BFF | 1K | GPIOC |
| | 0x4002_1000 - 0x4002_17FF | 1K | GPIOB |
| | 0x4002_1000 - 0x4002_13FF | 1K | GPIOA |
| | 0x4002_0C00 - 0x4002_0FFF | 1K | reserve |
| | 0x4002_0800 - 0x4002_0BFF | 1K | CRC16 |
| | 0x4002_0400 - 0x4002_07FF | 1K | FMC |
| | 0x4002_0000 - 0x4002_03FF | 1K | RCC |
| | 0x4000_5400 - 0x4001_FFFF | | reserve |
| APB | 0x4000_5000 - 0x4000_53FF | 1K | LPUART |
| | 0x4000_4C00 - 0x4000_4FFF | 1K | DEBUG |
| | 0x4000_4800 - 0x4000_4BFF | 1K | BEEP |
| | 0x4000_4400 - 0x4000_47FF | 1K | LPTIM |
| | 0x4000_4000 - 0x4000_43FF | 1K | LVD/VC |
| | 0x4000_3C00 - 0x4000_3FFF | 1K | TIM2 |
| | 0x4000_3800 - 0x4000_3BFF | 1K | OWIER |
| | 0x4000_3400 - 0x4000_37FF | 1K | CLKTRIM |
| | 0x4000_3000 - 0x4000_33FF | 1K | RTC |
| | 0x4000_2C00 - 0x4000_2FFF | 1K | ADC |
| | 0x4000_2800 - 0x4000_2BFF | 1K | AWK |
| | 0x4000_2400 - 0x4000_27FF | 1K | IWDT |
| | 0x4000_2000 - 0x4000_23FF | 1K | WWDT |
| | 0x4000_1C00 - 0x4000_1FFF | 1K | SYSCON |
| | 0x4000_1800 - 0x4000_1BFF | 1K | TIM10/11 |
| | 0x4000_1400 - 0x4000_17FF | 1K | PCA |
| | 0x4000_1000 - 0x4000_13FF | 1K | TIM1 |
| | 0x4000_0C00 - 0x4000_0FFF | 1K | I2C |
| | 0x4000_0800 - 0x4000_0BFF | 1K | SPI |
| | 0x4000_0400 - 0x4000_07FF | 1K | UART1 |
| | 0x4000_0000 - 0x4000_03FF | 1K | UART0 |
| AHB | 0x2000_1000 - 0x3FFF_FFFF | | reserve |
| | 0x2000_0000 - 0x2000_0FFF | 4K | SRAM |
| | 0x1800_0100 - 0x1FFF_FFFF | | reserve |
| | 0x1800_0000 - 0x1800_00FF | 256 | System Configuration |
| | 0x0800_0200 - 0x17FF_FFFF | | reserve |
| | 0x0800_0000 - 0x0800_01FF | 512 | Option Bytes |
| | 0x0001_0000 - 0x07FF_FFFF | | reserve |
| | 0x0000_0000 - 0x0000_FFFF | 64K | Main Array (Flash) |

## 4.4 Embedded SRAM

CX32L003 has built-in 4K bytes of SRAM. It can be accessed as a byte (8 bits), halfword (16 bits) or word (32 bits).

## 4.5 Flash memory

Flash memory has two distinct memory areas:

ÿMain Array area, including application program and user data area (if needed)

ÿNon -volatile storage area (NVR), consists of three parts:

- Option Bytes field. See 31 Option Bytes.
- System Configuration area
- ISP program area

The Flash memory interface performs instruction and data access based on the AHB protocol.

### 4.5.1 Embedded bootloader (Bootloader)

The embedded bootloader is stored in the NVR area and written during production. This program can reprogram Flash through UART1 Procedure.

# 5 working modes and power management

The power management module of CX32L003 is responsible for managing the switch between various working modes of this product, and controlling the

The working status of each functional module. The working voltage (VDD) of this product is 2.5~5.5V. This product has the following working modes:

1. Active Mode: CPU is running and peripheral function modules are running.
2. Sleep Mode: The CPU stops running, and the peripheral function modules run.
3. Deep Sleep Mode: The CPU stops running, and the high-speed clock stops running.

From run mode, other low-power modes can be entered by executing a software program. From various other low-power modes, triggered by interrupts, can return

to run mode.

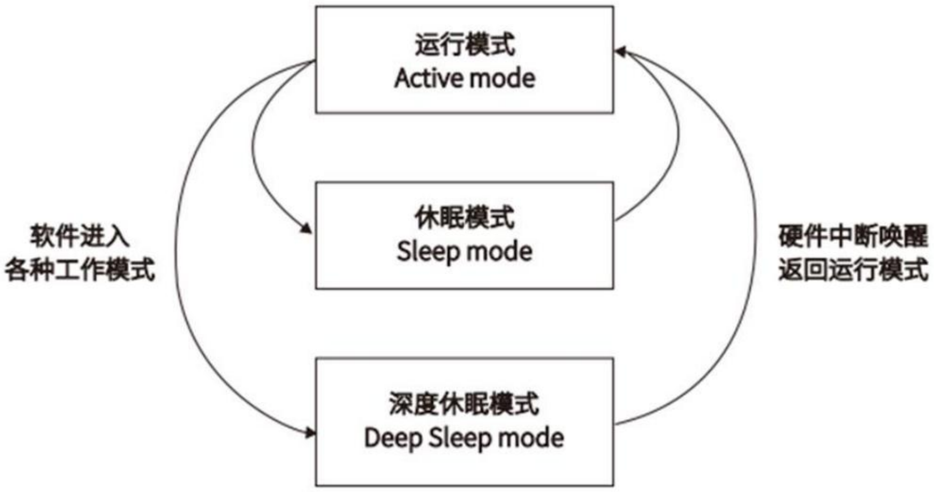


Figure 5-1 Control mode block diagram

For the types of interrupts that the CPU can respond to in various modes, please refer to Table 3-1 Interrupt Sources.

Responsive reset source types in each mode

| number | reset source | operating mode | Sleep mode wake up | Wake up from Deep Sleep mode |
|---|---|---|---|---|
| 1 | Power-on/power-off | Y | Y | Y |
| 2 | reset External Reset Pin Reset | Y | Y | Y |
| 3 | IWDG reset | Y | Y | Y |
| 4 | WWDG reset | Y | Y | N |
| 5 | System software | Y | N | N |
| 6 | reset Brown-out voltage (LVD) reset | Y | Y | Y |
| 7 | LOCKUP reset | Y | N | N |
| 8 | register CPURST reset register | Y | N | N |
| 9 | MCURST reset | Y | N | N |

## 5.1 Operating mode (Active Mode)

After the system is powered on and reset, or after waking up from various low power consumption modes, the microcontroller MCU is in the running state, and the operation of each module

The status is shown in Figure 5-2 List of Runnable Modules in Running Mode. When the CPU does not need to continue to run, you can take advantage of various low-power modes

way to save energy. The user needs to select an optimal low-power

consumption mode.

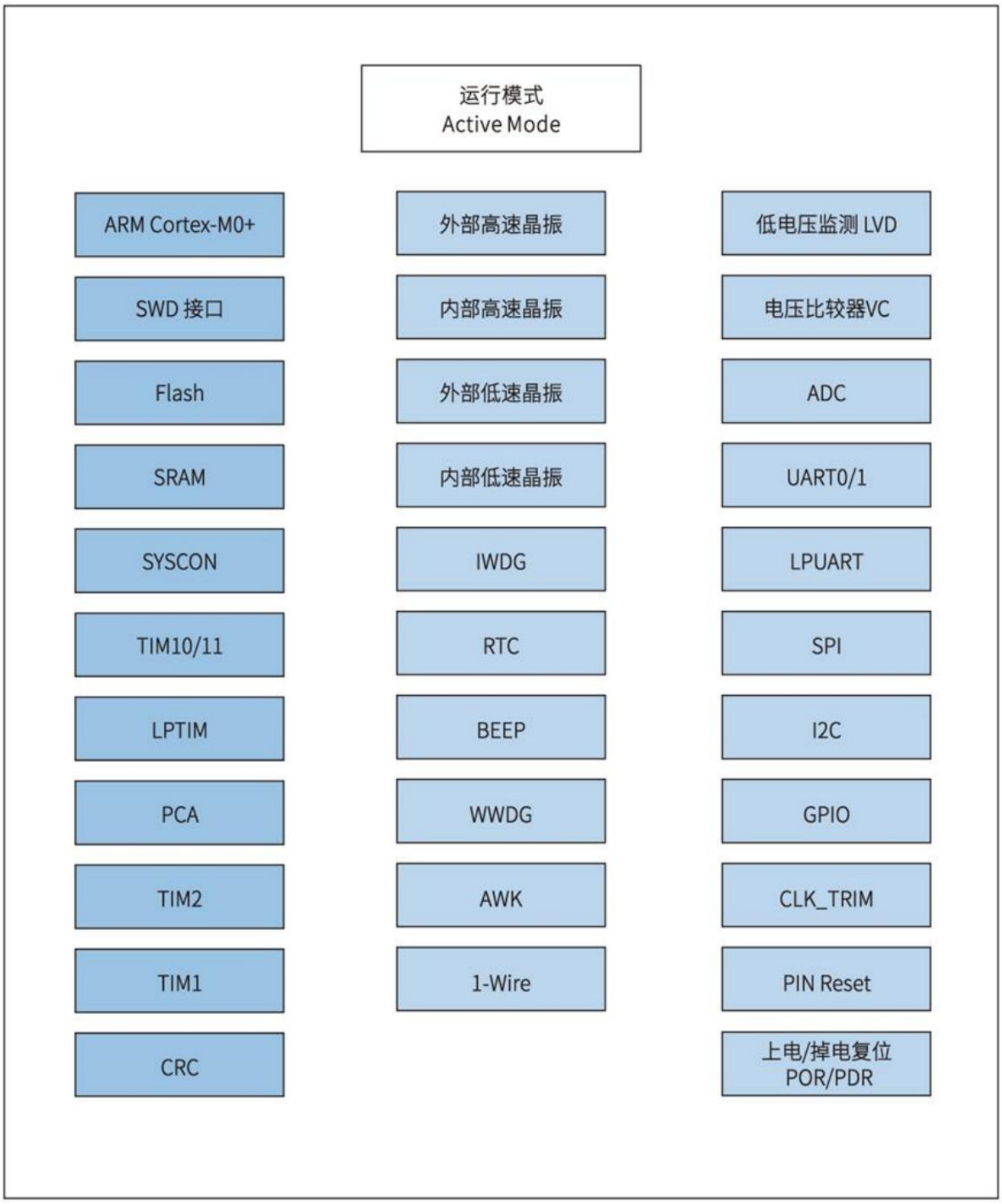


Figure 5-2 List of executable modules in running mode

Several ways to reduce chip power consumption in run mode:

1. In run mode, by programming the prescaler register (RCC_HCLKDIV, RCC_PCLKDIV), any

   One system clock (HCLK, PCLK) speed. Before entering sleep mode, the prescaler can also be used to reduce the timing of peripherals.

   bell.

2. In Run mode, turn off the clocks (RCC_HCLKEN, RCC_PCLKEN) of peripherals that are not used to reduce power consumption.

3. Use the deep sleep mode instead of the sleep mode, because the wake-up time of this product is extremely short (~3us), which can also meet the real-time requirements of the system.

   Response to needs.

## 5.2 Sleep Mode

Use the WFI instruction to enter the sleep mode. In the sleep mode, the CPU stops running, but the system clock, NVIC interrupt processing and peripheral function modules not driven by HCLK can still work.

When the system enters the dormancy state, the port state will not be changed. Before entering dormancy, please change the IO state to the state in dormancy mode as needed.

How to enter sleep mode:

Enter sleep state by executing WFI instruction. Depending on the value of the SLEEPONEXIT bit in the Cortex® -M0+ system control register, there are two options for selecting the sleep mode entry mechanism:

1. SLEEP-NOW: If SLEEPONEXIT=0, when WFI or WFE is executed, the microcontroller enters sleep mode immediately

   Mode.

2. SLEEP-ON-EXIT: If SLEEPONEXIT=1, when the system exits from the lowest priority interrupt handler, the MCU

   The controller immediately enters sleep mode.

How to exit hibernation:

If the WFI instruction is executed to enter the sleep mode, any peripheral interrupt responded by the high-priority nested vectored interrupt controller can The system wakes up from sleep mode.

Note:

ÿ SLEEP-ON-EXIT=1, automatically enter Sleep after executing the interrupt, the program does not need to write

__wfi(); ÿ SLEEP-ON-EXIT=0, main() enters Sleep after executing __wfi(), interrupt triggers and executes return after completion of interrupt routine

After main(), execute the WFI command and enter Sleep. Wait for a subsequent interrupt to fire.

ÿ The SLEEP-ON-EXIT bit does not affect the execution of the __wfi() instruction. SLEEP-ON-EXIT=0: main() executes __wfi() and advances

Enter Sleep, the interrupt is triggered and the execution of the interrupt program returns to main(), and then

continue to execute;

Low priority; interrupts with a priority lower than or equal to the current interrupt cannot wake up.

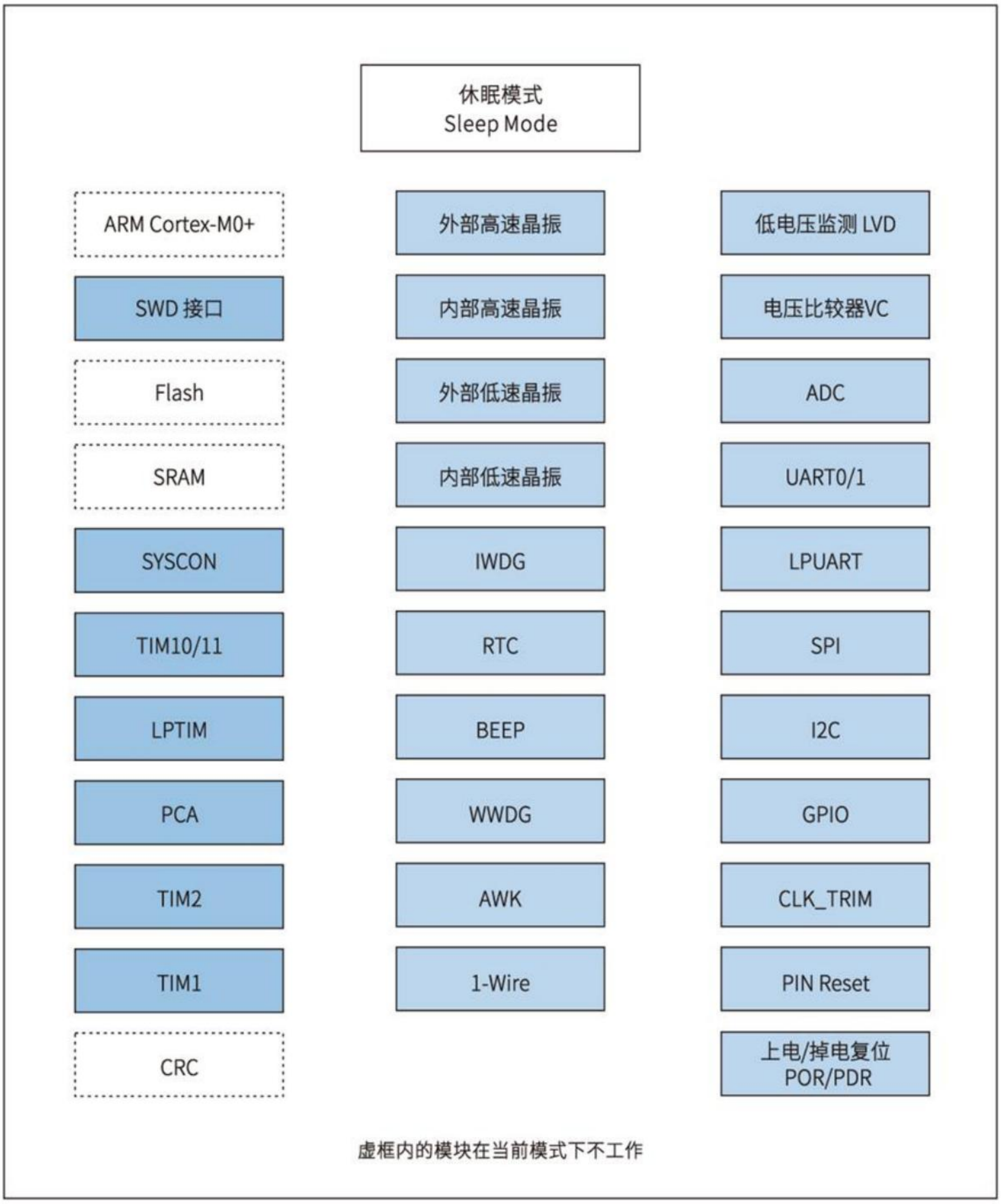


Figure 5-3 List of executable modules in sleep mode

## 5.3 Deep Sleep Mode

Use SLEEPDEEP with the WFI command to enter deep sleep mode. In deep sleep mode, the CPU stops running, the high-speed clock is turned off, and the low-speed clock can be configured to run or not. Some low-power peripheral modules can be configured to run or not. NVIC interrupt processing still work.

ÿThe system enters the deep sleep mode from the high-speed clock, the high-speed clock is automatically turned off, and the low-speed clock remains in the state before entering deep sleep state.

ÿThe system enters deep sleep mode from the low-speed clock, and the low-speed clock keeps running. Except the low-power module can run, other modules The block is automatically closed.

ÿWhen the system clock is switched, all clocks will not be automatically turned off. It needs to be turned off and turned on by the software according to power consumption and system requirements. clock.

ÿWhen the system enters the deep sleep state, the port state will not be changed. Before entering the deep sleep state, the IO state can be changed to State in Deep Sleep mode.

How to enter deep sleep mode:

First set the SLEEPDEEP bit in the Cortex® -M0+ system control register, and enter the deep sleep state by executing the WFI instruction. Depending on the value of the SLEEPONEXIT bit in the Cortex® -M0+ System Control Register, there are two options for selecting the Deep Sleep mode entry mechanism:

SLEEP-NOW: If SLEEPONEXIT=0, the microcontroller enters sleep mode immediately when WFI or WFE is executed.

SLEEP-ON-EXIT: If SLEEPONEXIT=1, the microcontroller enters sleep mode immediately when the system exits from the lowest priority interrupt handler.

How to exit deep sleep mode:

If the WFI instruction is executed to enter the sleep mode, any peripheral interrupt (peripheral module interrupt that can be run in Deep Sleep mode) that is responded by the Nested Vectored Interrupt Controller (NVIC) can wake up the system from the sleep mode. For wake-up settings, refer to 8.5 Interrupt Wake-Up Controller WIC.

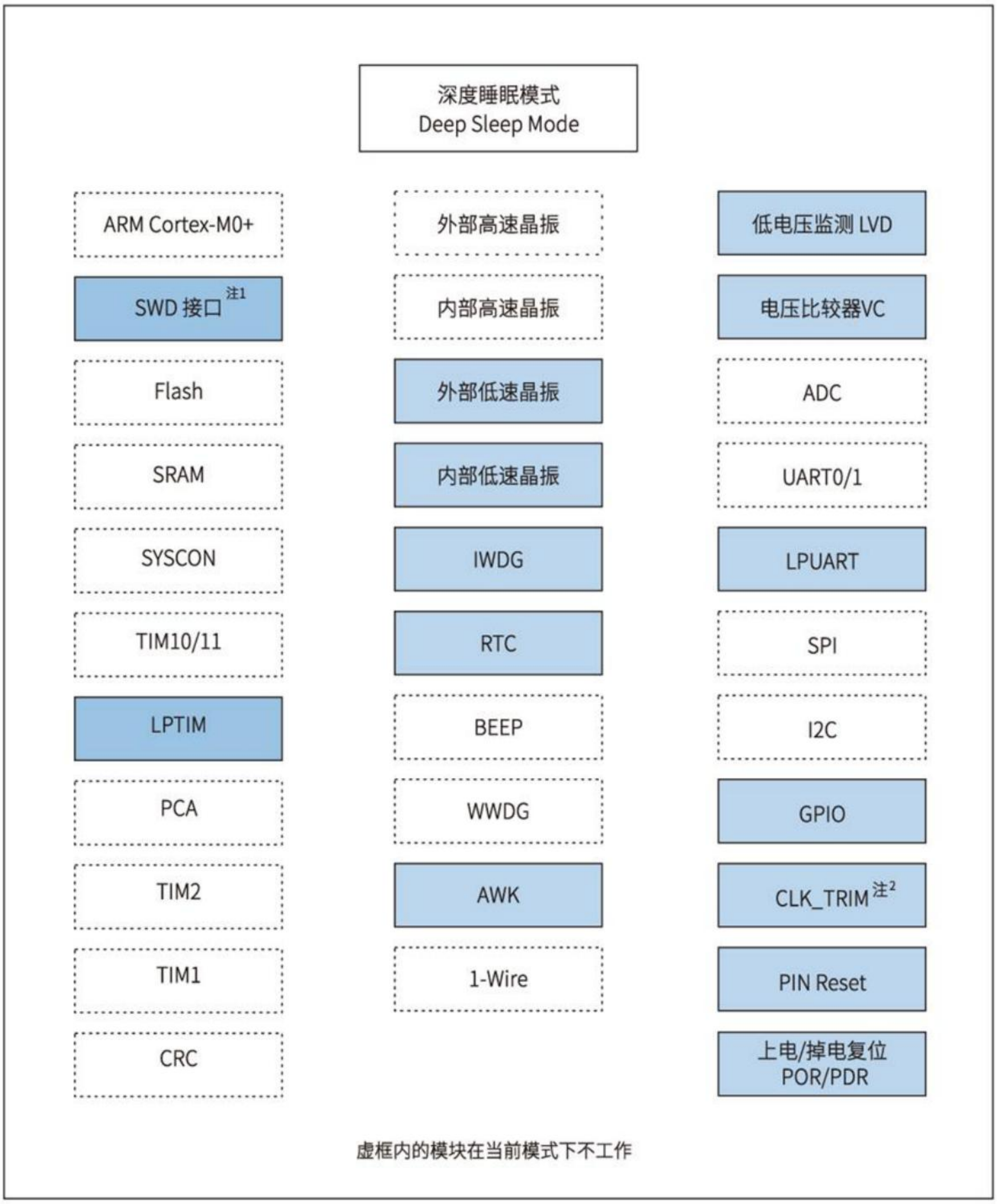


Figure 5-4 List of executable modules in deep sleep mode

Note:

1. In Deep Sleep mode, after the chip is reset, it can be woken up through the SWD interface

2. It can only wake up when the internal low-speed monitoring external low-speed clock function is selected

## 5.4 Cortex® -M0+ Core System Control Register (SCR)

Address: 0xE000ED10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | SEVON PEND | reserve | SLEEP DEEP | SLEEPO NEXIT | reserve |
| | | | | | | | | | | | R/W | | R/W | R/W | |

| bit flag | | Function | Reset value | read and write |
|---|---|---|---|---|
| 31:5 | RESERVED | reserve | 0x0 | - |
| 4 | SEVONPEND | When set to 1, an event is generated each time a new interrupt is pending, if used WFE sleep, which can be used to wake up the processor | 0 | R/W |
| 3 | RESERVED | reserve | 0 | - |
| 2 | SLEEP DEEP | When set to 1, implement WFI to enter deep sleep, and this product enters DeepSleep mode<br>When set to 0, execute WFI to enter sleep mode, and the product enters Sleep/Idle mode | 0 | R/W |
| 1 | SLEEPONE EXIT | When set to 1, when exiting exception handling and returning to the program thread, the processor automatically enters Sleep mode (WFI)<br>When set to 0, the feature is automatically disabled | 0 | R/W |
| 0 | RESERVED | Reserved | 0 | - |

After entering deep sleep, there are two options for the system clock after waking up. By default, the clock entering deep sleep is used, and the configuration register After RCC_SYSCLKCR.WKBYHIRC is 1, regardless of the clock before entering deep sleep, the internal high-speed clock will be used after waking up HIRC. If the system uses an external crystal oscillator before entering deep sleep, this setting can speed up the system wake-up.

# 6 System Reset and Clock (RCC)

## 6.1 6.1.1

## Reset Controller Introduction

This product has 9 reset signal sources, each reset signal can make the CPU run again, most of the registers will be reset to the reset value, the program counter PC will be reset to the reset address.

ÿ Digital area power-on power-off reset POR

ÿExternal pin reset (NRST PAD), low level reset

ÿ IWDG reset

ÿ WWDG reset

ÿ Low Voltage Reset (LVD)

ÿSoftware reset (Cortex® -M0+ SYSRESETREQ)

ÿ Cortex® -M0+ LOCKUP hardware reset

ÿRegister reset (CPURST)

ÿRegister reset (MCURST)

Each reset source has a dedicated reset flag to indicate, which is set to "1" by hardware and cleared by software. In addition to the POR of the digital area Reset flag, other reset flags can be cleared by POR in the digital area.

The following figure describes the 9 reset sources:

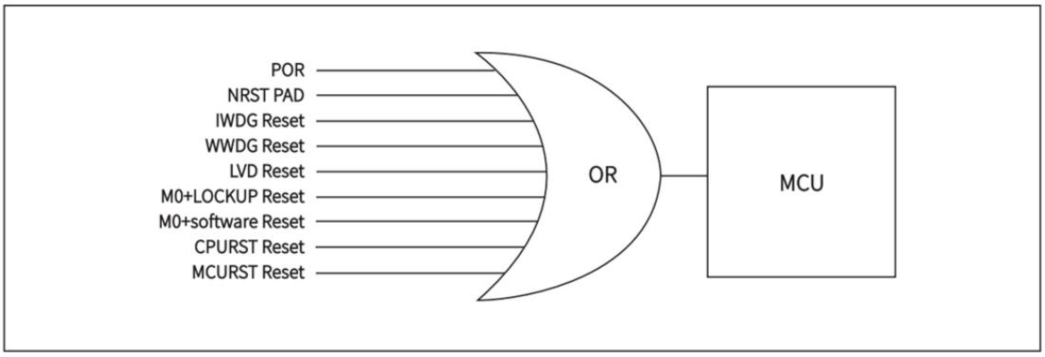


Figure 6-1 Schematic diagram of reset source

### 6.1.2 Reset sources

### 6.1.2.1 Power-on/power-off reset POR

When the chip is powered on and powered off, if the power supply is lower than a threshold voltage (2.2V), a POR signal will be generated internally. When the power supply is high When the threshold voltage is reached, the POR signal is released. The POR signal will reset the registers and control signals of the chip. This product has two voltages area, VDD area and Vcore area, so there are two PORs: the POR of the VDD area and the POR of the Vcore area.

#### 6.1.2.2 External pin reset

Pulling the external reset pin low will generate a system reset. This reset pin has an internal pull-up resistor, in addition, an internal Internally integrates a glitch filter circuit, the system will automatically filter the glitch signal less than 10us (typical value), therefore, the user is using the reset When the pin generates a reset signal, it must generate a low level greater than 10us to ensure that the chip can receive the reset signal correctly.

#### 6.1.2.3 IWDG reset

For independent watchdog reset, please refer to chapter IWDG.

#### 6.1.2.4 WWDG reset

Window watchdog reset, please refer to chapter WWDG.

#### 6.1.2.5 LVD Low Voltage Reset

For LVD reset, please refer to chapter LVD.

#### 6.1.2.6 Cortex® -M0+ software reset

A software reset is achieved by setting the SYSRESETREQ bit in the Cortex® -M0+ System Control Register. Please refer to Cortex® -M0+ Technical Reference Manual for further information.

#### 6.1.2.7 Register reset

The chip can be reset by writing MCURST and CPURST of RCC_RSTCR register.

#### 6.1.2.8 Cortex® -M0+ LOCKUP hardware reset

When the Cortex® -M0+ encounters a severe exception, it stops its PC pointer at the current address, locks itself, and resets the entire core region after a delay of one clock cycle.

# 6.2 System Clock

The clock control module mainly controls the system clock and the peripheral clock. Different clock sources can be configured as the system clock. Different clock sources can be configured. Peripheral clocks can be enabled or disabled with different system clock dividers. In addition, in order to ensure high precision, the internal clock has a calibration function.

This product supports the following four different clock sources as the system clock:

ÿ Internal high-speed RC clock HIRC (4MHz) (default main frequency) ÿ External low-speed
crystal oscillator clock LXT (32.768KHz) ÿ Internal low-speed RC clock LIRC
(38.4KHz and 32.768KHz configurable) ÿ External high-speed crystal oscillator clock HXT (4MHz~24MHz)

Note: LXT and HXT can be input from outside through terminals PB5 and PA1. When using an external oscillator input, it is necessary to enable the corresponding oscillator swing. Select the external oscillator control selection in the RCC_SYSCLKCR, RCC_LXTCR registers.

Each clock source can be turned on or off independently, and when they are not in use, they can be turned off to reduce power consumption.

Multiple dividers are available to configure the AHB and APB clock domains, and the maximum clock frequency for the AHB and APB domains is 24MHz.

The Cortex M0+ SysTick timer is driven by the AHB clock, which can be directly driven by AHB/4 or AHB clock frequency (configured through SYST_CSR.CLKSOURCE). RCC can use AHB clock (HCLK) divided by 4 as the external clock of the SysTick timer. By setting the SysTick control and status register (SYST_CSR.CLKSOURCE), the above clock or the Cortex clock (HCLK) can be selected as the SysTick clock.

Special attention should be paid to: the correct process must be followed in the process of switching the system clock source. For the specific process, refer to "6.2.7 System Clock Switching" section.

### 6.2.1 System clock tree

The following figure is the clock tree diagram of this product:

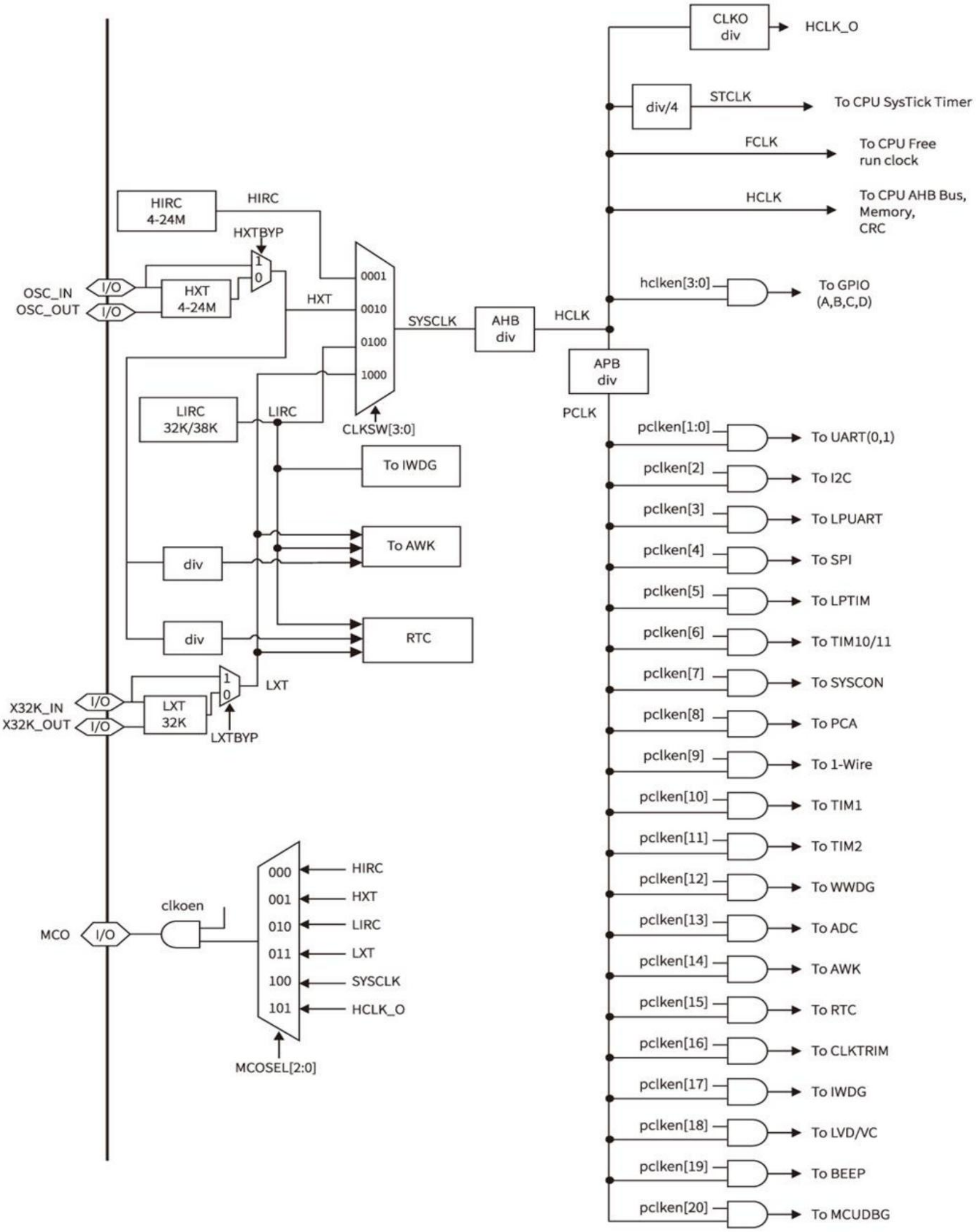


Figure 6-2 CX32L003 clock tree structure diagram

### 6.2.2 Internal High Speed RC Clock HIRC

The default system clock is the internal high-speed RC clock, which starts to work after the chip is powered on or reset, through the register

RCC_HIRCCR[11:0] to configure the frequency of the internal high-speed clock, giving accurate 4MHz, 8MHz, 16MHz, 22.12MHz,

24MHz frequency value. Because the internal high-speed clock starts quickly, about 3us, in order to make the system respond to external interrupts more quickly, the system

The system can choose to use this clock source as the system clock when it wakes up from Deep-sleep mode.

### 6.2.3 Internal Low Speed RC Clock LIRC

The internal low-speed RC clock frequency can be configured as 38.4KHz and 32.768KHz. In low-speed and low-precision application scenarios, it can be

Select this clock source as the system clock.

### 6.2.4 External high-speed crystal oscillator clock HXT

The external high-speed crystal oscillator clock needs to be connected with a 4MHz~24MHz high-speed crystal oscillator according to the user's system requirements.

External crystal oscillator clock can choose two input methods:

ÿ HXT external crystal/ceramic resonator

ÿ HXT user external clock

To reduce distortion of the clock output and shorten start-up settling time, the crystal/ceramic resonator and load capacitor must be placed as close as possible to the oscillator pin. The load capacitor value must be adjusted according to the selected oscillator.

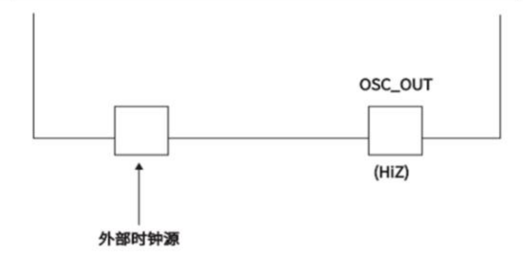
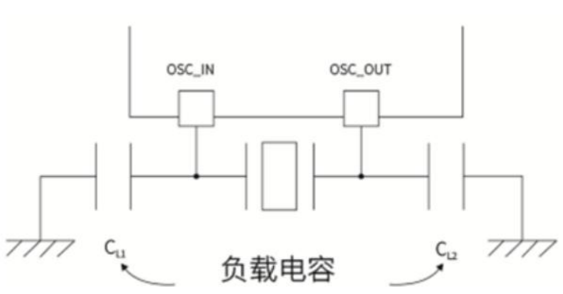
| | Hardware Configuration |
|---|---|
| external clock |  |
| Crystal/Ceramic Resonator |  |

Figure 6-3 HXT/LXT clock source

### 6.2.5 External low-speed crystal oscillator clock LXT

The external low-speed crystal oscillator clock requires an external 32.768KHz low-power crystal oscillator, which has ultra-high precision and low power consumption. Ultra Low Power Mode

Modules working in this mode can choose this clock source as the clock signal.

For external clock source bypass, please refer to Figure 6-3 HXT/LXT clock source

### 6.2.6 System clock startup process

The above four clock sources all have a start-up stabilization time. After the clock source is enabled, they will wait for a period of stabilization time before outputting the clock to the

System use, Figure 6-4 Internal high-speed clock startup diagram Take the internal high-speed RC clock HIRC as an example to illustrate the clock startup stability process

Procedure.

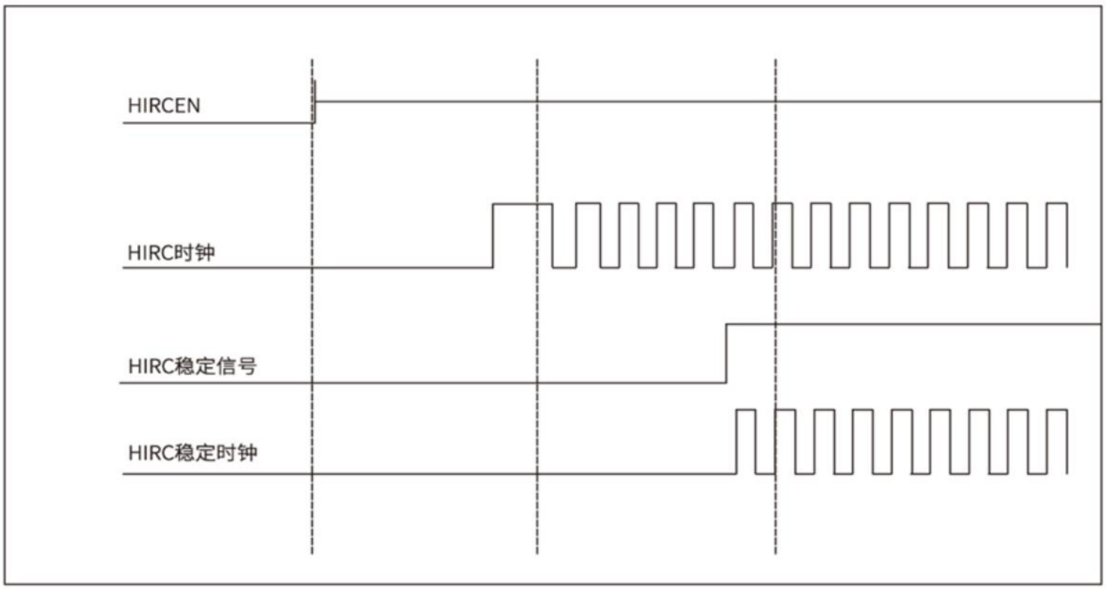


Figure 6-4 Schematic diagram of internal high-speed clock startup

After the chip is powered on, the system uses the 4MHz internal high-speed clock as the start-up clock. After the start-up is completed, the user can

To modify the frequency of the high-speed clock and switch the clock source.

### 6.2.7 System Clock Switching

The switching of the clock source is controlled by the register RCC_SYSCLKSEL[3:0]. In dual clock mode, when the system clock is from the current

When the clock is switched to the target clock, it must be implemented according to a certain process, otherwise an exception will occur.

### 6.2.7.1 Internal high speed switching to external low speed

Taking switching from HIRC (internal high-speed RC clock) to LXT (external low-speed crystal oscillator clock) as an example, the specific process is as follows:

1. Use the RCC_LXTCR.LXTPORT bit to configure the pin used by the clock LXT to be switched as an analog pin, or through the GPIOx_AFR=0x0F on the pin to configure as analog function
2. Write RCC_LXTCR.LXTEN to enable LXT clock
3. Wait for the register RCC_LXTCR.LXTRDY bit to be set to "1" by hardware
4. Write the register RCC_SYSCLKSEL.CLKSW[3:0] to switch the clock
5. Turn off the HIRC clock as needed

### 6.2.7.2 Internal high speed switching to external high speed

Switching from HIRC (internal high-speed RC clock) to HXT (external high-speed crystal oscillator clock) as an example, the specific process is as follows:

1. Use the RCC_SYSCLKCR.HXTPORT bit to configure the pin used by the clock HXT to be switched as an analog pin, Or configure it as an analog function by GPIOx_AFR=0x0F for the pin
2. Write register RCC_SYSCLKCR.HXTEN bit to enable HXT clock
3. Wait for the register RCC_HXTCR.HXTRDY bit to be set to "1" by hardware
4. Write the register RCC_SYSCLKSEL.CLKSW[3:0] to switch the clock
5. Turn off the HIRC clock as needed

Note: When using an external high-speed 24M crystal oscillator, the RCC_HXTCR.HXTSTARTUP stabilization time control bit is set to 0x3, using the default The default configuration 0x2 may not be stable enough.

The following figure shows the switching timing from HIRC to HXT:

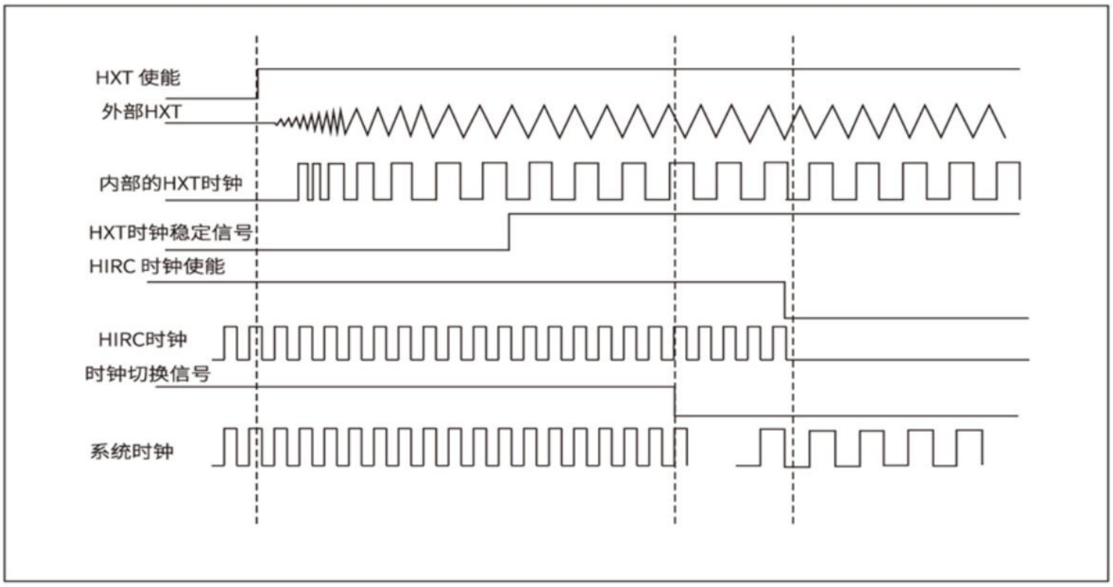


Figure 6-5 Schematic diagram of clock switching

#### 6.2.7.3 Internal low speed switching to external high speed

Switching from LIRC (internal low-speed RC clock) to HXT (external high-speed crystal oscillator clock) as an example, the specific process is as follows:

1. Use the RCC_SYSCLKCR.HXTPORT bit to configure the pin used by the clock HXT to be switched as an analog pin, or pass

   Configure as analog function by GPIOx_AFR=0x0F for the pin

2. Write register RCC_SYSCLKCR.HXTEN bit to enable HXT clock

3. Wait for the register RCC_HXTCR.HXTRDY bit to be set to "1" by hardware

4. Write the register RCC_SYSCLKSEL.CLKSW[3:0] to switch the clock

5. Turn off the LIRC clock as needed

### 6.2.8 System clock output

The microcontroller allows an output clock signal to an external MCO pin.

There are the following 6 signals that can be selected as the MCO clock output:

ÿ HIRC

ÿ HXT

ÿ LIRC

ÿ LXT

ÿ SYSCLK

ÿ FCLK and frequency division output

The selection of the MCO clock is determined by the MCOSEL[2:0] bits of the clock output control register (RCC_MCOCR).

### 6.2.9 System clock security control

When CLKFAILEN is set to be effective and the clock monitoring function of CLKTRIM is enabled, when the HXT clock or LXT clock stops

After stopping, the system clock will switch to the internal high-speed clock.

For details, please refer to the monitoring function of the CLKTRIM module.

### 6.2.10 IWDG Clock

If the independent watchdog has been enabled by hardware option or software, the LIRC oscillator will be forced on and cannot be turned off. exist

After the LIRC oscillator is stable, the clock is supplied to the IWDG.

### 6.2.11 RTC clock

RTCCLK clock source can be provided by HXT frequency division, LXT or LIRC clock.

### 6.2.12 AWK Clock

AWKCLK clock source can be provided by HXT divider, LXT or LIRC clock.

### 6.2.13 Low Power Mode

APB peripheral clocks and some AHB peripheral clocks can be disabled by software. Sleep mode stops the CPU clock, in CPU sleep

Memory interface clocks (Flash and RAM interface) are stopped.

When the SYSCON_CFGR0.DBGDLSP_DIS is configured, the CPU can also have the adjustment function in the corresponding deep sleep mode.

test function.

## 6.3 Register list

This section describes the register functions of the RCC control block in detail.

RCC base address: 0x4002 0000

Table 6-1 RCC register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | RCC_HCLKDIV | AHB Clock Divider Register | 0x0000 0000 |
| 0x04 | RCC_PCLKDIV | APB Clock Divider Register | 0x0000 0000 |
| 0x08 | RCC_HCLKEN | AHB Peripheral Module Clock Enable Register | 0x0000 0100 |
| 0x0C | RCC_PCLKEN | APB Peripheral Module Clock Enable | 0x0000 0000 |
| 0x10 | RCC_MCOCR | Register Clock Output Control | 0x0000 0000 |
| 0x18 | RCC_RSTCR | Register System Reset Control Register | 0x0000 0000 |
| 0x1C | RCC_RSTSR | Reset status register | 0x0000 00A0 |
| 0x20 | RCC_SYSCLKCR | clock source setting register | 0x0000 0001 |
| 0x24 | RCC_SYSCLKSEL | System Clock Source Selection Register Internal | 0x0000 0001 |
| 0x28 | RCC_HIRCCR | High Speed RC Oscillator Control Register | 0x0000 1312 |
| 0x2C | RCC_HXTCR | External High Speed Crystal Oscillator Control Register | 0x0000 0027 |
| 0x30 | RCC_LIRCCR | Internal Low Speed RC Oscillator Control Register | 0x0000 007F |
| 0x34 | RCC_LXTCR | External Low Speed Crystal Oscillator Control Register | 0x0000 042F |
| 0x38 | RCC_IRQLATENCY | Cortex M0+ IRQ delay control | 0x0000 0000 |
| 0x3C | RCC_STICKCR | SysTick Timer Period Calibration Register | 0x0100 9C3F |
| 0x40 | RCC_SWDIOCR | terminal special function selection register | 0x0000 0001 |
| 0x44 | RCC_PERIRST | peripheral module reset control register | 0x0000 0000 |
| 0x48 | RCC_RTCRST | RTC Control Register | 0x0000 0000 |
| 0x60 | RCC_UNLOCK | Register Write Protection | 0x0000 0000 |

## 6.4 Register description AHB

### 6.4.1 clock frequency division register (RCC_HCLKDIV)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | AHBCKDIV[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | AHBCKDIV[7:0] | System HCLK clock frequency division<br>0: HCLK=SYSCLK<br>1~255: Divide by 2×DIV (HCLK = SYSCLK/(2×AHBCKDIV)) | 0x0 | R/W |

### 6.4.2 APB Clock Divider Register (RCC_PCLKDIV)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | APBCKDIV[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | APBCKDIV[7:0] | System PCLK clock frequency division (maximum 1/16)<br>0: PCLK=HCLK<br>1~255: Divide by 2×DIV (PCLK = HCLK /(2×APBCKDIV)) | 0x0 | R/W |

### 6.4.3 AHB Peripheral Module Clock Enable Register (RCC_HCLKEN)

Address offset: 0x08

Reset value: 0x0000 0100

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | Flash<br>CKE | reserve | | | CRC<br>CKE | GPIOs<br>DCK<br>E. | GPIOs<br>CCK<br>E. | GPIOs<br>BCK<br>E. | GPIOs<br>ACK<br>E. |
| | | | | | | | R/W | | | | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:9 | – | reserve | 0x0 | – |
| 8 | FlashCKE | Flash controller block clock enable. Flash configuration registers cannot be written after shutdown, Flash Programs in it can still run.<br>0: off<br>1: enable | 1 | R/W |
| 7:5 | – | reserve | 0 | R/W |
| 4 | CRCCKE | CRC module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 3 | GPIODCKE | GPIOD module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 2 | GPIOCCKE | GPIOC module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 1 | GPIOBCKE | GPIOB module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 0 | GPIOACKE | GPIOA module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |

### 6.4.4 APB Peripheral Module Clock Enable Register (RCC_PCLKEN)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | DBG<br>CKE<br>N | BEEP<br>CKE<br>N | LVDV<br>CCK<br>EN | IWD<br>G<br>CKE<br>N | CLKT<br>RIMC<br>KEN |
| | | | | | | | | | | | R/W | R/W | R/W | R/W | R/W |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC<br>CKE<br>N | AWK<br>CKE<br>N | ADC<br>CKE<br>N | WWD<br>GCK<br>EN | TIM2<br>CKE<br>N | TIM1<br>CKE<br>N | OWI<br>REC<br>KEN | PCA<br>CKE<br>N | SYS<br>CON<br>CKE<br>N | BASE<br>TIMC<br>KEN | LPTI<br>MCK<br>EN | SPI<br>CKE<br>N | LPUA<br>RTC<br>KEN | I2C<br>CKE<br>N | UAR<br>T0CK<br>EN | UAR<br>T0CK<br>EN |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:21 | – | reserve | 0x0 | – |
| 20 | DBGCKEN | Debug PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 19 | BEEPCKEN | BEEP PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 18 | LVDVCCKEN | LVD/VC PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 17 | IWDGCKEN | IWDG PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 16 | CLKTRIMCKEN | CLKTRIM PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 15 | RTCCKEN | RTC PCLK clock enable<br>1: clock enable<br>0: clock off | 0 | R/W |
| 14 | AWKCKEN | AWK PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 13 | ADCCKEN | ADC PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 12 | WWDGCKEN | WWDG PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 11 | TIM2CKEN | TIM2 PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 10 | TIM1CKEN | TIM1 PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 9 | OWIRECKEN | 1-WIRE PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 8 | PCACKEN | PCA PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 7 | SYSCONCKEN | SYSCON PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 6 | BASETIMCKEN | TIM10/11 PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 5 | LPTIMCKEN | Low Power Timer PCLK clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 4 | SPICKEN | SPI PCLK module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 3 | LPUARTCKEN | Low Power UART PCLK register configuration clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 2 | I2CCKEN | I2C PCLK module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |
| 1 | UART1CKEN | UART1 PCLK module clock enable<br>1: clock enable<br>0: clock off | 0 | R/W |
| 0 | UART0CKEN | UART0 PCLK module clock enable<br>0: clock off<br>1: clock enable | 0 | R/W |

### 6.4.5 Clock Output Control Register (RCC_MCOCR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | MCO EN | reserve | MCOSEL[2:0] | | | MCDIV[7:0] | | | | | | | |
| | | | R/W | | R/W | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:13 | – | reserve | 0x0 | – |
| 12 | MCOEN | MCO output enable<br>Write:<br>0: MCO output disabled<br>1: MCO output enable<br>read:<br>0: MCO has not started to output<br>1: MCO starts to output<br>Note that when reading this bit, the output enable signal is synchronized by the output clock. | 0 | R/W |
| 11 | – | reserve | 0 | – |
| 10:8 | MCOSEL[2:0] | Clock output source selection<br>000: HIRC<br>001: HXT<br>010: LIRC<br>011: LXT<br>100: SYSCLK<br>101: HCLK and frequency division output<br>110, 111: Reserved, setting prohibited | 0x0 | R/W |
| 7:0 | MCDIV[7:0] | HCLK clock frequency division factor<br>0: HCLK<br>1~255: Divide by 2×DIV(HCLK_O = HCLK / (2×DIV)) | 0x0 | R/W |

### 6.4.6 System Reset Control Register (RCC_RSTCR)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSTKEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSTKEY | | | | | | | | | | | | | | CPU RST | MCU RST |
| WO | | | | | | | | | | | | | | R/W | R/W |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:2 RSTKEY | | Must be written when RCC_RSTCR[1:0]<br>Writing 0x156A99A6 (0x55AA6699>>2) is valid. Writing other values has no effect. | 0x0 | WO |
| 1 | CPURST | Register CPU reset, when this reset occurs, the ISP in the option byte area will not be reloaded set up<br>0: Normal<br>1: Reset CPU | 0 | R/W |
| 0 | MCURST | Register MCU reset, when this reset occurs, the ISP in the option byte area will be reloaded set up<br>0: Normal<br>1: Reset MCU | 0 | R/W |

Note:

ÿ MCU reset by writing 0x55AA6699 to RCC_RSTCR

ÿ CPU reset by writing 0x55AA669A to RCC_RSTCR

ÿ Only after the protection of the RCC_UNLOCK register is released, this register can be written.

### 6.4.7 System Reset Status Register (RCC_RSTSR)

Address offset: 0x1C

Reset value: 0x0000 00A0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | SFT<br>RST | PAD<br>RST | LOC<br>KUP<br>RST | POR<br>RST | LVD<br>RST | IWD<br>G<br>RST | WWD<br>GRS<br>T | CPU<br>RST | MCU<br>RST |
| | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:9 – | | reserve | 0x0 | – |
| 8 | SFTRST | Cotrex-M0+ CPU software reset flag, needs to be cleared by software,<br>0: No Cortrex-M0+ CPU software reset occurs<br>1: Cortrex-M0+ CPU software reset occurred | 0 | R/W |
| 7 | PADRST | RESET Port reset flag, can be reset by POR<br>0: No port reset occurs<br>1: A port reset occurred | 1 | R/W |
| 6 | LOCKUP RST | Cotrex-M0+ CPU Lockup reset flag,<br>0: No Cortrex-M0+ CPU Lockup reset occurs<br>1: A Cortrex-M0+ CPU Lockup reset occurred | 0 | R/W |
| 5 | PORRST | Vcore domain POR reset flag<br>0: Vcore domain POR no reset occurs<br>1: Vcore domain POR is reset | 1 | R/W |
| 4 | LVDRST | LVD reset flag<br>0: LVD no reset occurs<br>1: LVD is reset | 0 | R/W |
| 3 | IWDGRST | IWDG reset flag<br>0: IWDG no reset occurs<br>1: IWDG reset occurs | 0 | R/W |
| 2 | WWDGRST | WWDG reset flag<br>0: No reset from WWDG<br>1: WWDG reset occurs | 0 | R/W |
| 1 | CPURST | Register CPU reset flag, when this reset occurs, the option byte will not be reloaded ISP and IAP settings in the zone<br>0: Register CPURST no reset occurs<br>1: Register CPURST reset occurs | 0 | R/W |
| 0 | MCURST | Register MCU reset flag, when the reset occurs, the option byte area will be reloaded ISP and IAP settings in<br>0: No reset occurs in register MCURST<br>1: Register MCURST reset occurs | 0 | R/W |

Note: Only controlled by POR, can only be set to "1" by hardware, and cleared to "0" by software.

### 6.4.8 System clock source configuration register (RCC_SYSCLKCR)

Address offset: 0x20

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WKBYHIRC | reserve | | | | | | CLKFAILEN | reserved | HXTPORT | HXTBYP | reserve | | LIRCEN | HXTEN | HIRCEN |
| R/W | | | | | | | R/W | | R/W | R/W | | | R/W | R/W | R/W |

| bit flag | | Function Description Reset value read and write is valid only when the high bit is written to 0x5A69 to configure this register, and it is invalid when | | |
|---|---|---|---|---|
| 31:16 KEY | | other values are written. 0x0 WO | | |
| 15 | WKBYHIRC | 0: Wake up from Deep Sleep, the source of the system clock enters and exits.<br>1: Wake up from DeepSleep is to use HIRC to wake up, and the hardware is automatically enabled HIRC, and the system clock is automatically switched to HIRC, and the original clock continues to be on. | 0 | R/W |
| 14:9 | – | | 0 | – |
| 8 | CLKFAILEN | Retains clock fail detect enable control<br>0: Clock failure detection disabled<br>1: Clock failure detection is enabled, when the clock failure is detected and the system is automatically switched Bell to HIRC | 0 | R/W |
| 7 | – | Retain | 0 | – |
| 6 | HXTPORT | OSCIN/OSCOUT terminal configuration<br>0: GPIO alternate function mode (AFR determines its function).<br>1: HXT terminal mode (analog function). | 0 | R/W |
| 5 | HXTBYP | External high-speed clock input selection<br>0: HXT internal oscillator module is not bypassed, connected to OSC_IN/OSC_OUT<br>1: HXT internal oscillator module is bypassed, HXT is directly input from terminal OSCIN | 0 | R/W |
| 4:3 | – | Internal | 0x0 | – |
| 2 | LIRCEN | low-speed clock LIRC enable signal is reserved.<br>0: off<br>1: enable<br>When the system clock selects this clock, it cannot be turned off | 0 | R/W |
| 1 | HXTEN | External 4M~24M crystal oscillator HXTOSC enable signal.<br>0: off<br>1: enable<br>Notice:<br>**1.** When the system enters Deep Sleep, the high-speed clock will be automatically turned off.<br>Remark: When using, the two external ports connected with this crystal must be set as analog ports (configure RCC_SYSCLKCR.HXTPORT register).<br>2. When HXT stops detecting, this bit will be cleared to 0 by hardware when the system clock selects this When the clock is on, it cannot be turned off. | 0 | R/W |
| 0 | HIRCEN | Internal high-speed clock HIRC enable signal.<br>0: off<br>1: enable<br>Notice:<br>1. When the system enters deep sleep, the high-speed clock will be automatically turned off.<br>2. When HXT stop detection, if the system clock is selected as HXT, and CLKFAIL_EN is enabled, HIRC_EN will be automatically set to 1 by hardware.<br>3. When the system clock selects this clock, it cannot be turned off | 1 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.9 System clock source selection register (RCC_SYSCLKSEL)

Address offset: 0x24

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | CLKSW[3:0] | | | |
| | | | | | | | | | | | | R/W | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 | KEY | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:4 | – | reserve | 0x0 | – |
| 3:0 | CLKSW[3:0] | System Clock Source Select.<br>0001: HIRC selection<br>0010: HXT selection<br>0100: LIRC selection<br>1000: LXT selection<br>Note: When HXT stop detection, if the system clock is selected as HXT, and CLKFAIL_EN Enable, HIRC_EN will be automatically set to 1 by hardware. The system clock automatically selects HIRC. | 0x1 | R/W |

Note: This bit write is protected by RCC_UNLOCK

### 6.4.10 Internal High Speed RC Oscillator Control Register (RCC_HIRCCR)

Address offset: 0x28

Reset value: 0x0000 1312

This register is not controlled by Cortex-M0+ software reset

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | HIRC RDY | HIRCTRIM[11:0] | | | | | | | | | | | |
| | | | RO | R/W | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 KEY | | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:13 – | | reserve | 0x0 | – |
| 12 | HIRCRDY | HIRC clock stable flag.<br>0: It means HIRC is not stable and cannot be used by internal circuits.<br>1: It means HIRC is stable and can be used by internal circuits. | 1 | RO |
| 11:0 | HIRCTRIM<br>[11:0] | Internal high-speed clock frequency adjustment<br>Note: When leaving the factory, the frequency calibration value is saved in Flash,<br>Users need to write the Flash value into RCC_HIRCCR.HIRCTRIM[11:0] to configure<br>Precise main frequency clock.<br>For packages (TSSOP-20 and QFN-20): ÿ 24M<br>calibration value address: 0x1800_00A0<br>ÿ 22.12M calibration value address: 0x1800_00A2<br>ÿ 16M calibration value address: 0x1800_00A4<br>ÿ 8M calibration value address: 0x1800_00A6<br>ÿ 4M calibration value address: 0x1800_00A8<br>For bare die (KGD): ÿ<br>24M calibration value address: 0x1800_00C0<br>ÿ 22.12M calibration value address: 0x1800_00C2<br>ÿ 16M calibration value address: 0x1800_00C4<br>ÿ 8M calibration value address: 0x1800_00C6<br>ÿ 4M calibration value address: 0x1800_00C8 | 0x312 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.11 External High Speed Crystal Oscillator Control Register (RCC_HXTCR)

Address offset: 0x2C

Reset value: 0x0000 0027

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | HXT RDY | HXT STARTUP[1:0] | | reserve | HXTDRV[2:0] | | |
| | | | | | | | | | RO | R/W | | | R/W | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:16 KEY | | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:7 | – | reserve | 0x0 | – |
| 6 | HXTRDY | External 4M~24M crystal oscillator stable flag<br>0: It means that the external high-speed crystal oscillator clock is not stable and cannot be used by the internal circuit.<br>1: It means that the external high-speed crystal oscillator clock has been stabilized and can be used by the internal circuit. | 0 | RO |
| 5:4 | HXT STARTUP [1:0] | External 4M~24M crystal oscillator stabilization time selection<br>00: 1024 cycles<br>01: 2048 cycles<br>10: 4096 cycles<br>11: 16384 cycles<br>When using a high-speed crystal oscillator clock, the stabilization time needs to be set to 11, otherwise the stabilization time is not enough,<br>May cause system clock switching or when waking up from deep sleep with<br>cause system instability. | 0x2 | R/W |
| 3 | – | reserve | 0 | – |
| 2:0 | HXTDRV[2:0] | External 4M~24M crystal oscillator drive selection<br>000: minimum drive<br>111: Maximum drive (recommended value) | 0x7 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.12 Internal Low Speed RC Oscillator Control Register (RCC_LIRCCR)

Address offset: 0x30

Reset value: 0x0000 007F

Controlled by resets other than LVD and Cortex-M0+ software resets.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | LIRC RDY | LIRC STARTUP[1:0] | | reserve | LIRCTRIM[8:0] | | | | | | | | |
| | | | RO | R/W | | | R/W | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | KEY | Only when the high bit is written to 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:13 | – | reserve | 0x0 | – |
| 12 | LIRCRDY | Internal low-speed clock stable flag<br>0: It means that the internal low speed is not stable and cannot be used by the internal circuit.<br>1: It means that the internal low speed has been stabilized and can be used by the internal circuit. | 0 | RO |
| 11:10 | LIRC STARTUP[1:0] | Internal low-speed clock stabilization time selection<br>00: 4 cycles<br>01: 16 cycles<br>10: 64 cycles<br>11: 256 cycles | 0x0 | R/W |
| 9 | – | reserve | 0 | – |
| 8:0 | LIRCTRIM[8:0] | Internal low-speed clock frequency adjustment<br>Note: When leaving the factory, the frequency adjustment value is saved in Flash,<br>Users need to write the Flash value into RCC_LIRCCR.LIRCTRIM[8:0] to configure<br>Calibrated 38.4KHz/32.768KHz internal low-speed clock<br>For packages (TSSOP-20 and QFN-20): ÿ<br>32.768KHz calibration value address: 0x1800_00B0<br>ÿ 38.4KHz calibration value address: 0x1800_00B4<br>For bare die (KGD): ÿ<br>32.768KHz calibration value address: 0x1800_00D0<br>ÿ 38.4KHz calibration value address: 0x1800_00D4 | 0x07F | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.13 External Low Speed Crystal Oscillator Control Register (RCC_LXTCR)

Address offset: 0x34

Reset value: 0x0000 042F

This register is in the RTC domain, only POR can reset this register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | LXT<br>POR<br>T | LXT<br>AON | LXT<br>BYP | LXT<br>EN | reserve | LXT<br>RDY | LXT<br>STARTUP[1:0] | | LXTDRV[3:0] | | | |
| | | | | R/W | R/W | R/W | R/W | | RO | R/W | | R/W | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:16 KEY | | Only when the high bit is written to 0x5A69, it is valid to configure this register, and it is invalid when other values are written. effect. | 0x0 WO | |
| 15:12 | – | | 0x0 | – |
| 11 | LXTPORT | Reserve X32K_IN/X32K_OUT function selection<br>0: GPIO multiplexing function (AFR determines its function)<br>1: X32K terminal mode (analog function)<br>Note: This bit write is protected by RCC_UNLOCK | 0 | R/W |
| 10 | LXTAON | LXT can only enable but not disable control<br>0: LXT_EN allows prohibition control<br>1: LXT_EN can only enable but not disable control | 1 | R/W |
| 9 | LXTBYP | Set and cleared by software, this bit is only when the external 32KHz oscillator is turned off Write down the value.<br>0: LSE oscillator is not bypassed<br>1: LSE oscillator is bypassed<br>Note: When using an external low-speed oscillator, the enable bit of the low-speed crystal oscillator needs to be enabled LXT_EN | 0 | R/W |
| 8 | LXTEN | External 32K crystal oscillator LXT enable signal<br>0: off<br>1: enable<br>When the system clock selects this clock, it cannot be turned off | 0 | R/W |
| 7 | – | Reserve | 0 | – |
| 6 | LXTRDY | the external 32K crystal oscillator stable flag bit<br>0: It means that the external 32K crystal oscillator clock is not stable and cannot be used by the internal circuit.<br>1: It means that the external 32K crystal oscillator clock has been stabilized and can be used by the internal circuit. | 0 | RO |
| 5:4 | LXTSTARTUP[1:0] | External 32.768KHz crystal oscillator stabilization time selection<br>00: 1024 cycles<br>01: 2048 cycles<br>10: 4096 cycles<br>11: 16384 cycles<br>When using a low-speed crystal oscillator clock, the stabilization time needs to be set to 11, otherwise the stabilization time Not enough, it may cause deep sleep when the system clock is switched or when using a low-speed crystal oscillator clock Causes system instability when waking up. | 0x2 | R/W |
| 3:0 | LXTDRV[3:0] | External low-speed clock drive selection<br>0000: minimum drive<br>1111: maximum drive | 0xF | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.14 Cortex M0+ IRQ Latency Control Register (RCC_IRQLATENCY)

Address offset: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | IRQLATENCY[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | IRQLATENCY[7:0] | IRQLATENCY[7:0], refer to the ARM CortexM0+ documentation for specific usage | 0x0 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.15 SysTick Timer Control Register (RCC_STICKCR)

Address offset: 0x3C

Reset value: 0x0100 9C3F

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | NOREF | SKEW | STCALIB [23:16] | | | | | | | |
| | | | | | | R/W | R/W | R/W | | | | | | | |

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

| Bitmark 31:26 – | | Function | Reset value read and | |
|---|---|---|---|---|
| | | | write 0x0 | – |
| 25 | NOREF | Description Reserved Whether the SysTick timer uses an external reference clock<br>0: HCLK/4<br>1: Use the core clock (HCLK)<br>Notice:<br>1. This register is set with any one of the system register SYST_CSR.CLKSOURCE Set to 1 to use the core clock (HCLK)<br>2. When the divided clock HCLK/4 is used as the SysTick clock, the reference clock frequency is not allowed to be higher than the system clock HCLK | 0 | R/W |
| twenty four | SKEW | 10ms Whether the STCALIB value is accurate<br>0: accurate<br>1: Inaccurate | 1 | R/W |
| 23:0 | STCALIB[23:0] SysTick | 10ms calibration value, this value uses external reference clock HCLK/4(4MHz)<br>10ms calibration value. | 0x009C<br>3F | R/W |

### 6.4.16 SWDIO Port Control Register (RCC_SWDIOCR)

Address offset: 0x40

Reset value: 0x0000 0001

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">KEY</td></tr>
<tr><td colspan="16">WO</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15" rowspan="2">reserve</td><td>SWD POR T</td></tr>
<tr><td>R/W</td></tr>
</table>

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | – | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:1 | – | reserve | 0x0 | |
| 0 | SWDPORT | Configure the terminal function mode of PC7 and PD1<br>0: Peripheral module function mode<br>1: SWD terminal function | 1 | R/W |

Note: This bit write is protected by RCC_UNLOCK

### 6.4.17 Peripheral module reset control register (RCC_PERIRST)

Address offset: 0x44

Reset value: 0x0000 00000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | CRC<br>RST | GPIOs<br>DRS<br>T | GPIOs<br>CRS<br>T | GPIOs<br>BRST | GPIOs<br>ARST | reserve | | | DBG<br>RST | BEEP<br>RST | LVDV<br>CRS<br>T | reserved | CLKT<br>RIMR<br>ST |
| | | | R/W | R/W | R/W | R/W | R/W | | | | R/W | R/W | R/W | | R/W |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | AWK<br>RST | ADC<br>RST | WWD<br>GRS<br>T | TIM2<br>RST | TIM1<br>RST | OWR<br>IE<br>RST | PCA<br>RST | SYS<br>CON<br>RST | BASE<br>TIMR<br>ST | LPTI<br>MRS<br>T | SPIR<br>ST | LPUA<br>RTR<br>ST | I2CR<br>ST | UAR<br>T1RS<br>T | UAR<br>T0RS<br>T |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:29 – | | reserve | 0x0 | – |
| 28 | CRCRST | CRC module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 27 | GPIODRST | GPIOD module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 26 | GPIOCRST | GPIOC module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 25 | GPIOBRST | GPIOB module reset<br>0: normal work<br>1: reset | 0 | R/W |
| twenty four | GPIOARST | GPIOA module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 23:21 – | | reserve | 0x0 | – |
| 20 | DBGRST | MCU DEBUG module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 19 | BEEPRST | BEEP module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 18 | LVDVCRST | LVD module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 17 | – | reserve | 0 | – |
| 16 | CLKTRIMRST | Clock TRIM module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 15 | – | reserve | 0 | – |
| 14 | AWKRST | AWK module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 13 | ADCRST | ADC module reset<br>0: normal work | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 1: reset | | |
| 12 | WWDGRST | WWDG module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 11 | TIM2RST | TIM2 module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 10 | TIM1RST | TIM1 module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 9 | OWIREST | 1-Wire module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 8 | PCARST | PCA module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 7 | SYSCONRST | SYSCON module reset<br>1: reset<br>0: normal work | 0 | R/W |
| 6 | BASETIMRST | Base Timer10/11 reset<br>0: normal work<br>1: reset | 0 | R/W |
| 5 | LPTIMRST | Low Power Timer0/1 Reset<br>1: reset<br>0: normal work | 0 | R/W |
| 4 | SPIRST | SPI module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 3 | LPUARTRST | LPUART module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 2 | I2CRST | I2C module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 1 | UART1RST | UART1 module reset<br>0: normal work<br>1: reset | 0 | R/W |
| 0 | UART0RST | UART0 module reset<br>0: normal work<br>1: reset | 0 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.18 RTC Reset Control Register (RCC_RTCRST)

Address offset: 0x48

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | RTC RST |
| | | | | | | | | | | | | | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 KEY | | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:1 | – | Reserved, must remain at reset value | 0x0 | – |
| 0 | RTCRST | RTC module reset<br>Set or cleared by software<br>0: Reset not active<br>1: reset the entire RTC | 0 | R/W |

Note: Only after the protection of RCC_UNLOCK register is released, this register can be written.

### 6.4.19 Register Write Protection Control Register (RCC_UNLOCK)

Address offset: 0x60

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:1] | | | | | | | | | | | | | | | UNL OCK |
| WO | | | | | | | | | | | | | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | key | Only when the high bit is written to 0x2AD5334C, it is valid to configure this register, and it is invalid when writing other values. effect. | 0x0 | WO |
| 0 | UNLOCK | 0: Register write protection is enabled, and the protected register cannot be written<br>1: Register write protection is prohibited, and the protected register can be written | 0 | R/W |

Write 0x55AA6699 to remove protection

# 7 System Control (SYSCON)

This product has a set of system configuration registers. The main purposes of the system configuration controller are as follows:

ÿ Configure the interrupt generation mode of the GPIO terminal

ÿRemap the input trigger source of TIM10/11, PCA, TIM1, TIM2

ÿ CS input remapping setting for SPI slave mode

ÿSystem -level Deep Sleep debugging and Lockup reset control settings

## 7.1 Register list

SYSCON: 0x4000 1C00

Table 7-1 SYSCON register list and reset value

| Offset Address Name Description System Configuration Register 0 0x00 | | | Reset |
|---|---|---|---|
| | SYSCON_CFGR0 | | value 0x0000 0000 |
| 0x04 | Interrupt mode setting of SYSCON_PORTINTCR terminal | | 0x0000 0000 |
| 0x08 | SYSCON_PORTCR Terminal Control Register | | 0x0000 0000 |
| 0x0C | SYSCON_PCACR | PCA capture channel source selection | 0x0000 0000 |
| 0x10 | SYSCON_TIM1CR | TIM1 channel input source selection | 0x0000 0000 |
| 0x14 | SYSCON_TIM2CR | TIM2 channel input source selection | 0x0000 0000 |
| 0x50 | SYSCON_UNLOCK | SYSCON register write protection | 0x0000 0000 |

## 7.2 Register Description

### 7.2.1 System Configuration Register 0 (SYSCON_CFGR0)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve<br>3.20 | | | | | | | | | | | | | | DBG<br>DLSP<br>_DIS | LOC<br>KUP |
| | | | | | | | | | | | | | | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 Key | | Only when the high bit is written to 0x5A69, it is valid to configure this register, and it is invalid when other values are written. effect. | 0x0 | WO |
| 15:2 | – | reserve | 0x0 | – |
| 1 | DBGDLSP_DIS | Debug mode, enter Deep Sleep mode to disable the control bit<br>0: Allow to enter Deep Sleep in Debug mode<br>1: Not allowed to enter Deep Sleep in Debug mode | 0 | R/W |
| 0 | LOCKUPEN | Cortex-M0+ LookUp function enabled<br>0: off<br>1: enable<br>Note: When the Cortex-M0+ reads the wrong command, the MCU will reset to increase Strong system reliability. | 0 | R/W |

Note that this register can only be written after the SYSCON_UNLOCK register is unprotected.

Machine Translated by Google

### 7.2.2 Terminal Deep Sleep Interrupt Mode Control Register (SYSCON_PORTINTCR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | PAD DLSP CON | PAD INTS EL |
| | | | | | | | | | | | | | | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 Key | | Only when the high bit is written 0x5A69, it is valid to configure this register, and it is invalid when other values are written. | 0x0 | WO |
| 15:2 | – | reserve | 0x0 | – |
| 1 | PADDLSPCON | 0: After entering Deep Sleep, the interrupt generation mode of PAD is automatically switched to Deep Sleep interrupt generation mode (without Debounce function)<br>1: After entering DeepSleep, the interrupt generation mode of PAD will not switch automatically, The interrupt generation mode is determined by the SYSCON_PORTINTCR.PADINTSEL bit. | 0 | R/W |
| 0 | PADINTSEL | Port Interrupt Mode Selection<br>0: ACTIVE/Sleep interrupt generation mode<br>1: Deep Sleep interrupt generation mode | 0 | R/W |

Notice:

1. When the Deep Sleep interrupt generation mode is selected, the Debounce function of the GPIO terminal needs to be turned off, and the anti-interference ability is very good. Poor, it is generally recommended that users choose this mode when the chip needs to use terminal interrupts to wake up the Deep Sleep mode

2. Note that the SYSCON_UNLOCK register can only be written after the protection of the SYSCON_UNLOCK register is released.

### 7.2.3 Terminal Control Register (SYSCON_PORTCR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | LPTIM_GATE_SEL[1:0] | | TIM11_GATE_SEL[1:0] | | TIM10_GATE_SEL[1:0] | | SPINCS_SEL[3:0] | | | |
| | | | | | | R/W | | R/W | | R/W | | R/W | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:10 – | | reserve | 0x0 | – |
| 9:8 | LPTIM_GATE_SEL[1:0] | Low Power Timer gate input signal source selection<br>00: LPTIM_GATE<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 7:6 | TIM11_GATE_SEL[1:0] | Timer11 gate input signal source selection<br>00: TIM11_GATE<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 5:4 | TIM10_GATE_SEL[1:0] | Timer10 gate input signal source selection<br>00: TIM10_GATE<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 3:0 | SPINCS_SEL[3:0] | SPI Slave mode NCS signal source selection<br>0000 Fixed high level<br>0001 PA1<br>0010 PA2<br>0011 PA3<br>0100 PB4<br>0101 PB5<br>0110 PC3<br>0111 PC4<br>1000 PC5<br>1001 PC6<br>1010 PC7<br>1011 PD1<br>1100 PD2<br>1101 PD3<br>1110 PD4<br>1111 PD6 | 0x0 | R/W |

### 7.2.4 PCA capture channel control register (SYSCON_PCACR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | PCA_CAP4_SEL[1:0] | | PCA_CAP3_SEL[1:0] | | PCA_CAP2_SEL[1:0] | | PCA_CAP1_SEL[1:0] | | PCA_CAP0_SEL[1:0] | |
| | | | | | | R/W | | R/W | | R/W | | R/W | | R/W | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:10 – | | reserve | 0x0 | – |
| 9:8 | PCA_CAP4_SEL[1:0] | PCA Capture Channel 4 Source Selection<br>00: PCA_CH4<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 7:6 | PCA_CAP3_SEL[1:0] | PCA capture channel 3 signal source selection<br>00: PCA_CH3<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 5:4 | PCA_CAP2_SEL[1:0] | PCA capture channel 2 signal source selection<br>00: PCA_CH2<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 3:2 | PCA_CAP1_SEL[1:0] | PCA capture channel 1 signal source selection<br>00: PCA_CH1<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |
| 1:0 | PCA_CAP0_SEL[1:0] | PCA capture channel 0 signal source selection<br>00: PCA_CH0<br>01: UART0_RXD<br>10: UART1_RXD<br>11: LPUART_RXD | 0x0 | R/W |

### 7.2.5 TIM1 channel input source selection (SYSCON_TIM1CR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | CLKFAILBRKEN | DSLPBRKEN | TIM1BRKoutCFG | TIM1ETR_SEL[3:0] | | | |
| | | | | | | | | | R/W | R/W | R/W | R/W | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | TIM1CH4IN_SEL[2:0] | | | reserve | TIM1CH3IN_SEL[2:0] | | | reserve | TIM1CH2IN_SEL[2:0] | | | reserve | TIM1CH1IN_SEL[2:0] | | |
| | R/W | | | | R/W | | | | R/W | | | | R/W | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:23 | – | | 0x0 | – |
| twenty two | CLKFAILBRKEN | Enable TIM1 Break when system clock stop is detected<br>0: invalid<br>1: enable | 0 | |
| twenty one | DSLPBRKEN | TIM1 Break enable in Deep Sleep mode<br>0: invalid<br>1: enable | 0 | R/W |
| 20 | TIM1BRKOUTCFG | 0: The output of ocxp/ocxnp in break mode is controlled by TIM1<br>1: ocxp/ocxnp output 0 at the same time in break mode | 0 | R/W |
| 19:16 | TIM1ETR_SEL[3:0] | TIM1 ETR signal source selection<br>0000 Fixed low level<br>0001 PA1<br>0010 PA2<br>0011 PA3<br>0100 PB4<br>0101 PB5<br>0110 PC3<br>0111 PC4<br>1000 PC5<br>1001 PC6<br>1010 PC7<br>1011 PD1<br>1100 PD2<br>1101 PD3<br>1110 PD4<br>1111 PD6 | 0x0 | R/W |
| 15 | – | reserve | 0 | – |
| 14:12 | TIM1CH4IN_SEL[2:0] | TIM1 CH4 input channel signal source selection<br>000: TIM1_CH4<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |
| 11 | – | reserve | 0 | – |

| | | | | |
|---|---|---|---|---|
| 10:8 | TIM1CH3IN_SEL[2:0] | TIM1 CH3 input channel signal source selection<br>000: TIM1_CH3<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |
| 7 | – | reserve | 0 | – |
| 6:4 | TIM1CH2IN_SEL[2:0] | TIM1 CH2 input channel signal source selection<br>000: TIM1_CH2<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |
| 3 | – | reserve | 0 | – |
| 2:0 | TIM1CH1IN_SEL[2:0] | TIM1 CH1 input channel signal source selection<br>000: TIM_CH1<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |

### 7.2.6 TIM2 channel input source selection (SYSCON_TIM2CR)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | TIM2<br>ETR_SEL[3:0] | | | |
| | | | | | | | | | | | | R/W | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | TIM2CH4IN_SEL[2:0] | | | reserve | TIM2CH3IN_SEL[2:0] | | | reserve | TIM2CH2IN_SEL[2:0] | | | reserve | TIM2CH1IN_SEL[2:0] | | |
| | R/W | | | | R/W | | | | R/W | | | | R/W | | |

| Bitmark 31:20 – | | Functional description | Reset value read and | |
|---|---|---|---|---|
| | | reserved | write 0x0 | – |
| 19:16 | TIM2ETR_SEL[3:0] | TIM2 ETR signal source selection<br>0000 Fixed low level<br>0001 PA1<br>0010 PA2<br>0011 PA3<br>0100 PB4<br>0101 PB5<br>0110 PC3<br>0111 PC4<br>1000 PC5<br>1001 PC6<br>1010 PC7<br>1011 PD1<br>1100 PD2<br>1101 PD3<br>1110 PD4<br>1111 PD6 | 0x0 | R/W |
| 15 | – | reserve | 0 | – |
| 14:12 | TIM2CH4IN_SEL[2:0] | TIM2 CH4 input channel signal source selection<br>000: TIM2_CH4<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |
| 11 | – | reserve | 0 | – |
| 10:8 | TIM2CH3IN_SEL[2:0] | TIM2 CH3 input channel signal source selection<br>000: TIM2_CH3<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |
| 7 | – | reserve | 0 | – |
| 6:4 | TIM2CH2IN_SEL[2:0] | TIM2 CH2 input channel signal source selection<br>000: TIM2_CH2<br>001: UART0_RXD | 0x0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | | |
| 3 | – | reserve | 0 | – |
| 2:0 | TIM2CH1IN_SEL[2:0] | TIM2 CH1 input channel signal source selection<br>000: TIM_CH1<br>001: UART0_RXD<br>010: UART1_RXD<br>011: LPUART_RXD<br>100: LIRC<br>Other: reserved | 0x0 | R/W |

## 7.2.7 Syscon register write protection (SYSCON_UNLOCK)

Address offset: 0x50

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:1] | | | | | | | | | | | | | | | UNL OCK |
| WO | | | | | | | | | | | | | | | R/W |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:1 KEY | | Only when the high bit is written to 0x2AD5334C, it is valid to configure this register, and it is invalid when writing other values. effect. | 0x0 | – |
| 0 | UNLOCK | 0: protection is valid<br>1: Unprotect | 0 | R/W |

Write 0x55AA6699 to remove protection.

# 8 Interrupt Controller (NVIC)

## 8.1 Overview

NVIC Cortex® -M0+ provides an interrupt controller for overall management of interrupts and exceptions, called the "Nested Vectored Interrupt Controller (NVIC)". Closely connected with the processor core, it can realize low-latency interrupt processing and efficient late-arriving interrupt processing.

NVIC supports up to 32 interrupt request (IRQ) inputs, and 1 non-maskable interrupt (NMI) input (not used in this product system use). In addition, the processor supports several internal exceptions.

This chapter only introduces the 32 external interrupt requests (interrupt 0 to interrupt 31) of the processor in detail, and the specific situation of the internal exception of the processor Other relevant documents may be referred to. For details, please refer to "ARM® Cortex® -M0+ Technical Reference Manual" and "ARM® v6-MA Architecture Reference Manual".

## 8.2 Features

ÿ 32 maskable interrupt channels (excluding 16 Cortex® -M0+ interrupt lines) ÿ 4 programmable priorities (using 2-bit interrupt priority)

ÿLow -latency exception and interrupt handling

ÿ Power Management Control

ÿ Implementation of system control registers

ÿ Support nested and vectored interrupts

ÿ Dynamically change priority

## 8.3 Interrupt Priority

Software can set 4 levels of priority for external interrupts. The highest priority is "0", the lowest priority is "3", all user configurable The default value of the priority is "0".

Preemption occurs when the processor is already running an interrupt handler and the new interrupt has a higher priority than the current one. is Running interrupt handlers are suspended and new interrupts are executed, a process often referred to as interrupt nesting. The new interrupt is executed After that, the previous interrupt processing will continue and return to the program thread after it ends.

If the processor is running an interrupt handler of the same or higher priority, the new interrupt will wait and enter the pending state. hang The interrupt started will wait until the current interrupt level changes. For example, after the currently running interrupt processing completes and returns, the current priority will drop to Low enough to be less than a pending interrupt.

If two interrupts occur at the same time and they have the same priority, the interrupt with the lower interrupt number will be executed first. For example, if Interrupt 0 and Interrupt 1 are enabled and have the same priority. When they are triggered at the same time, Interrupt 0 will be executed first.

## 8.4 Interrupt vector table

When the Cortex® -M0+ responds to an interrupt, the processor automatically fetches the start address of the interrupt service routine (ISR) from the interrupt vector table in memory.

The vector table includes the initial value of the stack (MSP) after reset and the entry addresses of all exception handling. The interrupt number indicates the order in which exceptions are handled

order. Among them, the storage order of the interrupt vector is the same as the interrupt number. Since each is 1 word (4 bytes), the address of the interrupt vector

Multiplying the interrupt number by 4, each interrupt vector is the starting address for processing.

Table 8-1 Interrupt vector table

| interrupt number | External Interrupt Number (IRQ#) | priority class | priority class type | Introduction to Interrupt Sources | | address |
|---|---|---|---|---|---|---|
| 0 | - | - | - | - | MSP Initial Value | 0x0000 0000 |
| 1 | | -3 | fixed-2 | Reset | Reset Vector (RESET) | 0x0000 0004 |
| 2 | | fixed fixed | | NMI | Non-Maskable Interrupt | 0x0000 0008 |
| 3 | | -1 | | HardFault all types of errors | (fault) | 0x0000 000C |
| 4-10 | - | - | - | - | reserve | 0x0000 000C<br>0x0000 002B |
| 11 | | | Configurable | SVCall | Invoked by the generic SWI instruction system service | 0x0000 002C |
| 12-13 | - | - | - | - | reserve | 0x0000 0030-<br>0x0000 0037 |
| 14 | | | Configurable | PendSV | suspendable system service 0x0000 0038 | |
| 15 | | | Configurable | SysTick | system tick timer 0x0000 003C | |
| 16 | 0 | | Configurable | GPIO_PA | GPIOA Interrupt 0x0000 0040 | |
| 17 | 1 | | Configurable | GPIO_PB | GPIOB interrupt 0x0000 0044 | |
| 18 | 2 | | Configurable | GPIO_PC | GPIOC Interrupt 0x0000 0048 | |
| 19 | 3 | | Configurable | GPIO_PD | GPIOD interrupt 0x0000 004C | |
| 20 | 4 | | Configurable | Flash | Flash Interrupt 0x0000 0050 | |
| 21 | 5 | | Configurable | reserved 0x0000 0054 | - | |
| 22 | 6 | | Configurable | UART0 0x0000 0058 | UART0 interrupt | |
| 23 | 7 | | Configurable | UART1 | UART1 interrupt | 0x0000 005C |
| 24 | 8 | | Configurable | LPUART | LPUART interrupt | 0x0000 0060 |
| 25 | 9 | | Configurable | Reserve | - | 0x0000 0064 |
| 26 | 10 | | Configurable | SPI | SPI interrupt | 0x0000 0068 |
| 27 | 11 | | Configurable | Reserve | - | 0x0000 006C |
| 28 | 12 | | Configurable | I2C | I2C interrupt | 0x0000 0070 |
| 29 | 13 | | Configurable | Reserve | - | 0x0000 006C |
| 30 | 14 | | Configurable | TIM10 | TIM10 interrupt | 0x0000 0078 |
| 31 | 15 | | Configurable | TIM11 | TIM11 interrupt | 0x0000 007C |
| 32 | 16 | | Configurable | LPTIM | LPTIM interrupt | 0x0000 0080 |
| 33 | 17 | | Configurable | Reserve | - | 0x0000 007C |
| 34 | 18 | | Configurable | TIM1 | TIM1 interrupt | 0x0000 0088 |
| 35 | 19 | | Configurable | TIM2 | TIM2 interrupt | 0x0000 008C |
| 36 | 20 | | Configurable | Reserve | - | 0x0000 0088 |
| 37 | twenty one | | Configurable | PCA | PCA interrupt | 0x0000 0094 |
| 38 | twenty two | | Configurable | WWDG | WWDG Interruption | 0x0000 0098 |
| 39 | 23 | | Configurable | IWDG | IWDG Interruption | 0x0000 009C |
| 40 | 24 | | Configurable | ADC | ADC interrupt | 0x0000 00A0 |
| 41 | 25 | | Configurable | LVD | LVD interrupt | 0x0000 00A4 |
| 42 | 26 | | Configurable | VC | VC interrupt | 0x0000 00A8 |
| 43 | 27 | | Configurable | Reserve | - | 0x0000 00A4 |
| 44 | 28 | | Configurable | AWK | AWK interrupt | 0x0000 00B0 |
| 45 | 29 | | configurable | ONEWIRE | 1-WIRE interrupt configurable | 0x0000 00B4 |
| 46 | 30 | | RTC | | RTC interrupt | 0x0000 00B8 |
| 47 | 31 | | configurable | CLKTRIM | CLKTRIM interrupt | 0x0000 00BC |

## 8.5 Interrupt wake-up control WIC

When the processor uses the SLEEP-ON-EXIT feature or executes the WFI instruction to enter sleep, it will stop instruction execution.

line, and when an interrupt request (higher priority) occurs and needs to be serviced, the processor is woken up.

| WFI behavior | wake | ISR execution |
|---|---|---|
| PRIMASK Clear | | |
| IRQ Priority > Current Class | Y | Y |
| IRQ priorityÿcurrent level | N | N |
| PRIMASK set (interrupt disabled) | | |
| IRQ Priority > Current Class | Y | N |
| IRQ priorityÿcurrent level | N | N |

### 8.5.1 NVIC wake-up from deep-sleep mode into interrupt ISR setup

1. Enable the interrupt NVIC that needs to wake up from deep sleep

2. Enable module interrupt enable

3. Set SCR.DEEPSLEEP to 1

4. Use the WFI command to enter deep sleep mode

5. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the interrupt service subroutine after waking up

Routine:

```
SCR |= 0x00000004u;

__asm("nop");

__asm("nop");

__asm("nop");

while(1){

__asm("WFI");

}
```

### 8.5.2 NVIC Wakeup from Deep Sleep Mode Setup Does Not Execute Interrupt ISR Setup

1. Enable the interrupt NVIC that needs to wake up from deep sleep

2. Use the PRIMASK register to mask the interrupt

3. Enable module interrupt enable

4. Set SCR.DEEPSLEEP to 1

5. Use the WFI command to enter deep sleep mode

6. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the next instruction after waking up

7. Clear interrupt flag, clear interrupt pending status

### 8.5.3 Using the Exit Hibernation Feature

The exit dormancy (SLEEP-ON-EXIT) feature is very suitable for interrupt-driven applications. When this feature is enabled, as long as the completion exception process and returns to thread mode, the processor enters sleep mode. With the exit-from-sleep feature, the processor can in sleep mode.

Cortex® -M0+ uses the SLEEP-ON-EXIT feature to enter sleep, which is the same as executing immediately after abnormal exit WFI works about the same. However, in order to avoid the need to push the stack when entering an exception next time, the processor will not perform the stack popping operation. process.

1. Enable the interrupt NVIC that needs to wake up from deep sleep
2. Enable module interrupt enable
3. Set SCR.DEEPSLEEP to 1
4. Set SCR.SLEEPONEXIT to 1

5. Use the WFI command to enter deep sleep mode 6. The
system enters deep sleep mode and waits for the interrupt to wake up, after waking up, execute the interrupt service
subroutine 7. Automatically enter the sleep mode when exiting the interrupt service

Routine:

```
SCR |= 0x00000004u;
SCR |= 0x00000002u;
__asm("nop");

__asm("nop");

__asm("nop");
while(1){
__asm("WFI");

}
```

## 8.6 Basic software operation

### 8.6.1 External Interrupt Enable

Each peripheral module has its own interrupt enable register. When an interrupt operation is required, the peripheral must be enabled first.

own interrupt enable. The operation of this enable bit is not discussed in this chapter, please refer to the respective chapter descriptions of the peripheral modules.

### 8.6.2 NVIC interrupt enable and clear enable

The Cortex-M0+ processor supports up to 32 interrupt sources, and each interrupt source corresponds to an interrupt enable bit and clear bit. so that

With 32-bit interrupt enable set register NIVC_ISER and 32-bit interrupt enable clear register NVIC_ICER. If you want to make

If an interrupt is enabled, the corresponding bit of the NIVC_ISER register is set to 1. If you want to clear an interrupt enable, then

The corresponding bit of the NVIC_ICER register is set to 1.

Note that the interrupt enable mentioned here is only for the processor NVIC. Whether each peripheral interrupt is generated or not is determined by the peripheral

It is determined by the interrupt control register and has nothing to do with NVIC_ISER and NVIC_ICER.

### 8.6.3 NVIC interrupt pending and clear pending

If an interrupt occurs but cannot be handled immediately, the interrupt request will be pending. Suspend state is kept in a register if the

If the processor's current priority has not been lowered enough to handle the pending request, and the pending state has not been manually cleared, the state will

Always remain in effect.

When the processor starts to enter the interrupt processing, the hardware will automatically clear the corresponding interrupt pending status.

It can be accessed or modified by operating the two registers of interrupt setting pending NVIC_ISPR and interrupt clearing pending NVIC_ICPR

Interrupt pending state. The Interrupt Pending Status Register allows software to trigger interrupts.

### 8.6.4 NVIC interrupt priority

Setting the NVIC_IPR0-NVIC_IPR7 registers determines the priority of IRQ0-IRQ32. The interrupt priority register should be programmed in the

This is usually done at the beginning of the program, before the interrupt is enabled. Changing the interrupt priority after the interrupt is enabled should be avoided.

The results of this condition are unpredictable and are not supported by Cortex-M0+ processors.

### 8.6.5 NVIC interrupt mask

Some time-sensitive applications need to disable all interrupts for a short period of time, you can use the interrupt mask register PRIMASK implementation. Only 1 bit of PRIMASK is valid and defaults to 0 after reset. When this register is 0, all interrupts and All exceptions are allowed; and when set to 1, only NMI (not supported by this system) and hardware error exceptions are enabled. In fact, When PRIMASK is set to 1, the current priority of the processor is reduced to 0.

The PRIMASK register can be programmed in several ways, using assembly language, it can be set using the CPSIE i and CPSID i instructions and clear the PRIMASK register. If using C language and CMSIS device driver library, users can use the following functions to set and clear PRIMASK.

```
void__enable_irq(void); //clear PRIMASK

void__disable_irq(void); //Set PRIMASK
```

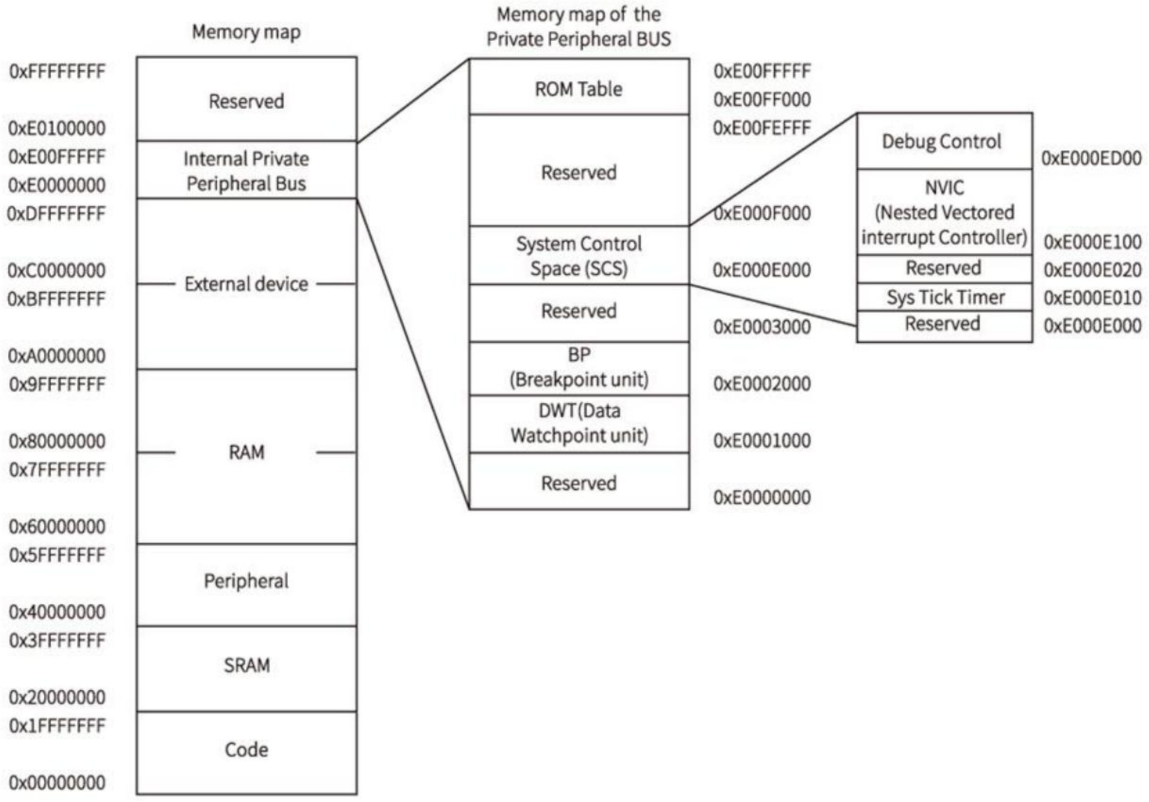
## 8.7 Register list

SCS base address: 0xE000 E000

Table 8-2 NVIC register list and reset value

| offset address name | | describe | reset value |
|---|---|---|---|
| 0x100 | NVIC_ISER | IRQ0~IRQ31 enable setting control register | 0x0000 0000 |
| 0x180 | NVIC_ICER | IRQ0~IRQ31 enable clear control register | 0x0000 0000 |
| 0x200 | NVIC_ISPR | IRQ0~IRQ31 Pending Set Control Register | 0x0000 0000 |
| 0x280 | NVIC_ICPR | IRQ0~IRQ31 pending clear control register | 0x0000 0000 |
| 0x400 | NVIC_IPR0 | IRQ0~IRQ3 Priority Control Register 0 | 0x0000 0000 |
| 0x404 | NVIC_IPR1 | IRQ4~IRQ7 Priority Control Register 1 | 0x0000 0000 |
| 0x408 | NVIC_IPR2 | IRQ8~IRQ11 Priority Control Register 2 | 0x0000 0000 |
| 0x40C | NVIC_IPR3 | IRQ12~IRQ15 priority control register 3 | 0x0000 0000 |
| 0x410 | NVIC_IPR4 | IRQ16~IRQ19 priority control register 4 | 0x0000 0000 |
| 0x414 | NVIC_IPR5 | IRQ20~IRQ23 priority control register 5 | 0x0000 0000 |
| 0x418 | NVIC_IPR6 | IRQ24~IRQ27 priority control register 6 | 0x0000 0000 |
| 0x41C | NVIC_IPR7 | IRQ28~IRQ31 priority control register 7 | 0x0000 0000 |

Note: The NVIC register is part of the SCS register, and the SCS base address is 0xE000E000, as shown in the following figure:



## 8.8 Register Description

### 8.8.1 Interrupt Enable Setting Register (NVIC_ISER)

Address offset: 0x100

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SETENA [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | SETENA | Set to enable external interrupt 0 to interrupt 31; write "1" is set, write "0" is invalid<br>[0]: IRQ0<br>[1]: IRQ1<br>[2]: IRQ2<br>…<br>[31]: IRQ31 | 0x0 | R/W |

### 8.8.2 Interrupt Enable Clear Register (NVIC_ICER)

Address offset: 0x180

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLRENA [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | CLRENA | Clear enable external interrupt 0 to interrupt 31; write "1" to clear, write "0" is invalid<br>[0]: IRQ0<br>[1]: IRQ1<br>[2]: IRQ2<br>…<br>[31]: IRQ31 | 0x0 | R/W |

### 8.8.3 Interrupt Pending Setup Register (NVIC_ISPR)

Address offset: 0x200

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SETPEND [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit flag | | Function | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | SETPEND | Description Set the pending status of external interrupt 0 to interrupt 31; write "1" to set, write "0" to disable effect<br>[0]: IRQ0<br>[1]: IRQ1<br>[2]: IRQ2<br>...<br>[31]: IRQ31 | 0x0 | R/W |

### 8.8.4 Interrupt Pending Clear Register (NVIC_ICPR)

Address offset: 0x280

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLRPEND [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit flag | | Function | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | CLRPEND | Description Clear the pending status of external interrupt 0 to interrupt 31; write "1" to clear, write "0" to disable effect<br>[0]: IRQ0<br>[1]: IRQ1<br>[2]: IRQ2<br>...<br>[31]: IRQ31 | 0x0 | R/W |

## 8.8.5 Interrupt Priority Control Register 0 (NVIC_IPR0)

Address offset: 0x400

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR0[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR0[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR0 | Priority of external interrupt 0 to interrupt 3:<br>[31:30]: Priority of interrupt 3<br>[23:22]: Priority of interrupt 2<br>[15:14]: Priority of Interrupt 1<br>[7:6]: Priority of interrupt 0<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

## 8.8.6 Interrupt Priority Control Register 1 (NVIC_IPR1)

Address offset: 0x404

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR1 [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR1[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR1 | Priority of interrupt 4 to interrupt 7:<br>[31:30]: Priority of interrupt 7<br>[23:22]: Priority of interrupt 6<br>[15:14]: Priority of interrupt 5<br>[7:6]: Priority of interrupt 4<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

## 8.8.7 Interrupt Priority Control Register 2 (NVIC_IPR2)

Address offset: 0x408

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR2 [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR2[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR2 | Priority of interrupt 8 to interrupt 11:<br>[31:30]: Priority of interrupt 11<br>[23:22]: Priority of interrupt 10<br>[15:14]: Priority of interrupt 9<br>[7:6]: Priority of interrupt 8<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

## 8.8.8 Interrupt Priority Control Register 3 (NVIC_IPR3)

Address offset: 0x40C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR3 [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR3[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR3 | Priority of interrupt 12 to interrupt 15:<br>[31:30]: Priority of interrupt 15<br>[23:22]: Priority of interrupt 14<br>[15:14]: Priority of interrupt 13<br>[7:6]: Priority of interrupt 12<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

### 8.8.9 Interrupt Priority Control Register 4 (NVIC_IPR4)

Address offset: 0x410

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR4 [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR4[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit Flag Functional Description | | | Reset value read and write | |
|---|---|---|---|---|
| 31:0 | IPR4 | Priority of interrupt 16 to interrupt 19:<br>[31:30]: Priority of interrupt 19<br>[23:22]: Priority of interrupt 18<br>[15:14]: Priority of interrupt 17<br>[7:6]: Priority of interrupt 16<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

### 8.8.10 Interrupt Priority Control Register 5 (NVIC_IPR5)

Address offset: 0x414

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR5 [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPR5[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit Flag Functional Description | | | Reset value read and write | |
|---|---|---|---|---|
| 31:0 | IPR5 | Priority of interrupt 20 to interrupt 23:<br>[31:30]: Priority of interrupt 23<br>[23:22]: Priority of interrupt 22<br>[15:14]: Priority of interrupt 21<br>[7:6]: Priority of interrupt 20<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

## 8.8.11 Interrupt Priority Control Register 6 (NVIC_IPR6)

Address offset: 0x418

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">IPR6 [31:16]</td></tr>
<tr><td colspan="16">R/W</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">IPR6[15:0]</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR6 | Priority of interrupt 24 to interrupt 27:<br>[31:30]: Priority of interrupt 27<br>[23:22]: Priority of interrupt 26<br>[15:14]: Priority of interrupt 25<br>[7:6]: Priority of interrupt 24<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

## 8.8.12 Interrupt Priority Control Register 7 (NVIC_IPR7)

Address offset: 0x41C

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">IPR7 [31:16]</td></tr>
<tr><td colspan="16">R/W</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">IPR7[15:0]</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | IPR7 | Priority of interrupt 28 to interrupt 31:<br>[31:30]: Priority of interrupt 31<br>[23:22]: Priority of interrupt 30<br>[15:14]: Priority of interrupt 29<br>[7:6]: Priority of interrupt 28<br>Among them, 00 has the highest priority and 11 has the lowest priority | 0x0 | R/W |

# 9 General purpose input and output (GPIO)

## 9.1 Introduction to GPIOs

The general-purpose input/output port is used for data transmission between the chip and the outside, and there are 4 groups of GPIO: GPIOA, GPIOB, GPIOC and GPIOD. The functions of the 4 groups of GPIO are basically the same, and the GPIO can be mapped to the corresponding chip pin through configuration, and each pin can be controlled by Independently configured as a digital input or output port, it can also be configured as an analog input. In addition, it can also be configured as an external interrupt, on-chip Multiplexing functions such as peripheral input/output. At the same time, a pin can only be mapped to one alternate function, registered through the port alternate function register (GPIOx_AFR) configuration.

Each general-purpose I/O port has 5 configuration registers (GPIOx_DIRCR, GPIOx_OTYPER, GPIOx_PUPDR, GPIOx_SLEWCR and GPIOx_DRVCR), 2 data registers (GPIOx_IDR and GPIOx_ODR), 1 output set register (GPIOx_ODSET), 1 output reset register (GPIOx_ODCLR) and 1 alternate function register (GPIOx_AFR).

Each port can be configured as an internal pull-up/pull-down input/output, high-impedance input (floating input), Push-pull output (push-pull output), open drain output (open drain output), enhanced drive capability output. Chip reset backend The port reset is a high-impedance input, the purpose is to prevent abnormal actions of external devices when the chip is reset abnormally. In order to avoid high impedance input The user should configure the port accordingly after the chip is started (configured to internally pull up the input or output). end After the port is configured as an analog port, the digital function is isolated, and the digital "1" and "0" cannot be output, and the result of the CPU reading the port is "0".

All ports can provide external interrupts, and each interrupt can be configured as high-level trigger, low-level trigger, rising edge trigger, Falling edge trigger or arbitrary edge trigger, support input debounce in edge mode. Support working mode/sleep mode/deep sleep mode generates an interrupt.

## 9.2 GPIO main features

ÿOutput state: push-pull output or open-drain output with pull-up or pull-down

ÿ Output data from data register (GPIOx_ODR) or peripherals (alternate function output)

ÿConfigurable speed of each I/O port

ÿInput status: floating, pull-up/pull-down, analog input

ÿ Input data from data register (GPIOx_IDR) or peripheral (alternate function output)

ÿThe output set/reset registers (GPIOx_ODSET, GPIOx_ODCLR) provide bit rewriting for the GPIOx_ODR register ability

ÿ Analog function pins/debug pins/digital general purpose pins/digital function pins are multiplexed

ÿ Allows flexible multiplexing of GPIO ports and peripheral pins

## 9.3 GPIO Functional Description

Each bit of a GPIO port can be individually configured by software to

Various modes:

ÿ Floating input

ÿ Pull-up input

ÿDropdown input

ÿAnalog input

ÿOpen -drain output with pull-up or pull-down capability

ÿPush -pull output with pull-up or pull-down capability

ÿPush -pull output with multiplexing function and pull-up or pull-down capability

ÿOpen -drain output with multiplexing function and pull-up or pull-down capability

Each I/O port bit is freely programmable and the I/O port registers are accessed as 32-bit words. GPIOx_ODSET, GPIOx_ODCLR register

Register allows bit read/write operations on GPIOx_ODR. In this case, an interruption between read and change accesses is avoided

And the exception occurred.

Figure 9-1 shows the basic structure of a standard IO port.

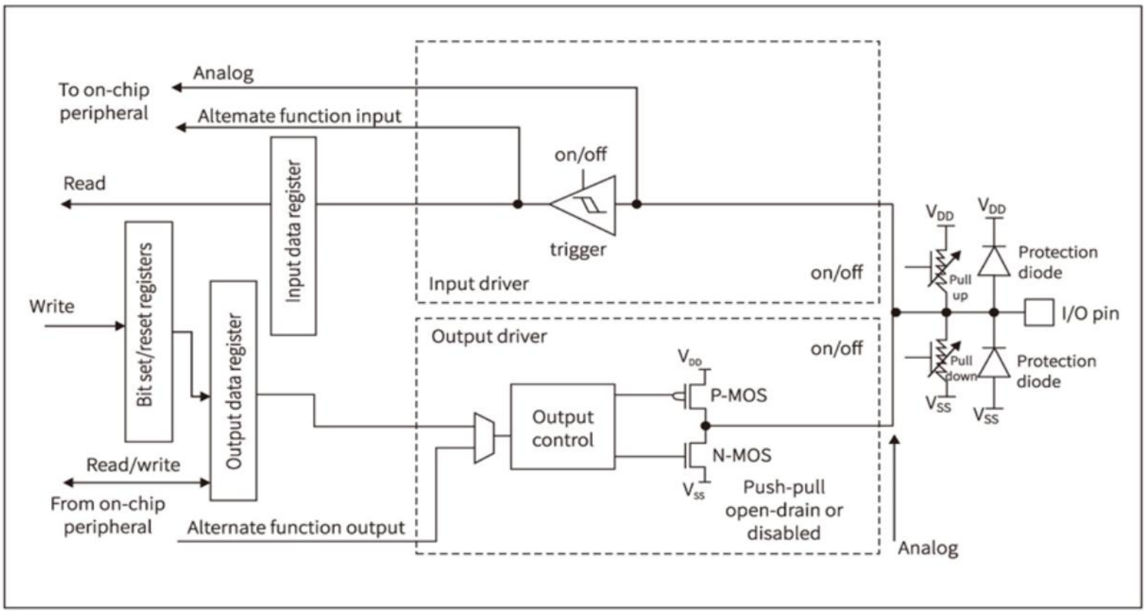


Figure 9-1 Basic bit structure of a standard I/O port

Table 9-1 Port bit configuration table shows the possible configuration of port bits, and Table 9-2 shows the multiplexed function mapping of GPIO ports and peripheral pins

shoot.

Table 9-1 Port bit configuration table

| AFRi[3:0] | DIRi | OTYPi | DRVi | PUPDi | | IO Configuration | |
|---|---|---|---|---|---|---|---|
| 0000 | 1 | 0 | DRVi | 0 | 0 | GPIO output | PP |
| | | 0 | | 0 | 1 | GPIO output | PP+PU |
| | | 0 | | 1 | 0 | GPIO output | PP+PD |
| | | 0 | | 1 | 1 | Reserved | |
| | | 1 | | 0 | 0 | GPIO output | OD |
| | | 1 | | 0 | 1 | GPIO output | OD+PU |
| | | 1 | | 1 | 0 | GPIO output | OD+PD |
| | | 1 | | 1 | 1 | Reserved(output OD) | |
| | 0 | x | x | 0 | 0 | input | Floating |
| | | x | x | 0 | 1 | input | PU |
| | | x | x | 1 | 0 | input | PD |
| | | x | x | 1 | 1 | Reserved (input floating) | |
| 0001~1110 | x | 0 | DRVi | 0 | 0 | AF | PP |
| | x | 0 | | 0 | 1 | AF | PP+PU |
| | x | 0 | | 1 | 0 | AF | PP+PD |
| | x | 0 | | 1 | 1 | Reserved | |
| | x | 1 | | 0 | 0 | AF | OD |
| | x | 1 | | 0 | 1 | AF | OD+PU |
| | x | | | 1 | 0 | AF | OD+PD |
| | x | 1 1 | | 1 | 1 | Reserved | |
| 1111 | x | x | x | 0 | 0 | input/output | Analog |
| | x | x | x | 0 | 1 | Forbidden | |
| | x | x | x | 1 | | | |
| | x | x | x | 1 | 0 1 | | |

i = 0...7

GP=general purpose, PP=push-pull output, PU=pull-up, PD=pull-down, OD=open drain, AF=alternate function.

### 9.3.1 General I/O (GPIO)

During and after reset, the alternate function is not enabled, and all I/O ports except debug pins are configured as floating input mode.

Mode.

After reset, the debug pins are put into pull-up/pull-down mode with alternate functions:

ÿ PD1: SWDCLK put in pull-down mode

ÿ PC7: Put SWDIO in pull-up mode

When configured as an output, the value written to the output data register (GPIOx_ODR) is output on the corresponding pin. push-pull mode Or open-drain mode (only low level is driven, high level appears as high impedance) output.

The Input Data Register (GPIOx_IDR) captures the data on the I/O pin every AHB clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down resistor, they are activated or disconnected depending on the GPIOx_PUPD register value.

### 9.3.2 I/O Port Control Register

Each general purpose I/O port has 5 control registers (GPIOx_DIRCR, GPIOx_OTYPER, GPIOx_PUPDR, GPIOx_SLEWCR and GPIOx_DRVCR) are used to configure the I/O port. The GPIOx_DIRCR register is used to select the direction (input/output). GPIOx_OTYPER, GPIOx_SLEWCR and GPIOx_DRVCR registers to select output type (push-pull or open-drain), voltage translation speed and drive strength. The GPIOx_PUPDR register is used to select the pull-up/pull-down mode.

### 9.3.3 I/O Port Data Register

Each GPIO port has two data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR is used for Store output data, which is read/write accessible. The input data from the I/O line is stored in the (GPIOx_IDR) register, which is Read-only register.

### 9.3.4 I/O Data Bit Handling

The Output Set Register (GPIOx_ODSET) and Output Clear Register (GPIOx_ODCLR) registers allow the application to

Each bit of the register (GPIOx_ODR) is set and cleared.

When a 1 is written to bit GPIOx_ODSET[i], the corresponding ODR[i] bit is set. When writing 1 to GPIOx_ODCLR[i], the phase is cleared

The corresponding ODR[i] bit.

Writing 0 to any bit in GPIOx_ODSET, GPIOx_ODCLR will not affect the value of GPIOx_ODR register. not only can

Use GPIOx_ODSET, GPIOx_ODCLR register to change the corresponding bit of GPIOx_ODR, also can change the GPIOx_ODR register

Access directly. GPIOx_ODSET, GPIOx_ODCLR registers provide atomic bit operation processing for GPIOx_ODR registers.

GPIOx_ODR The access mechanism of GPIOx_ODSET, GPIOx_ODCLR set or cleared does not require software to disable interrupt access

Ask GPIOx_ODR: It is possible to change 1 or more bits of data in one AHB write access cycle.

### 9.3.5 Input configuration

When I/O programming is configured as an input:

ÿ The port's output buffer is disabled

ÿThe pull-up and pull-down resistors are activated by the value of the GPIOx_PUPD register

ÿData on the I/O pin is sampled into the input data register every AHB clock cycle.

A read access to the input data register is used to obtain the synchronized input data Figure 9-2 shows the input configuration of the I/O port.

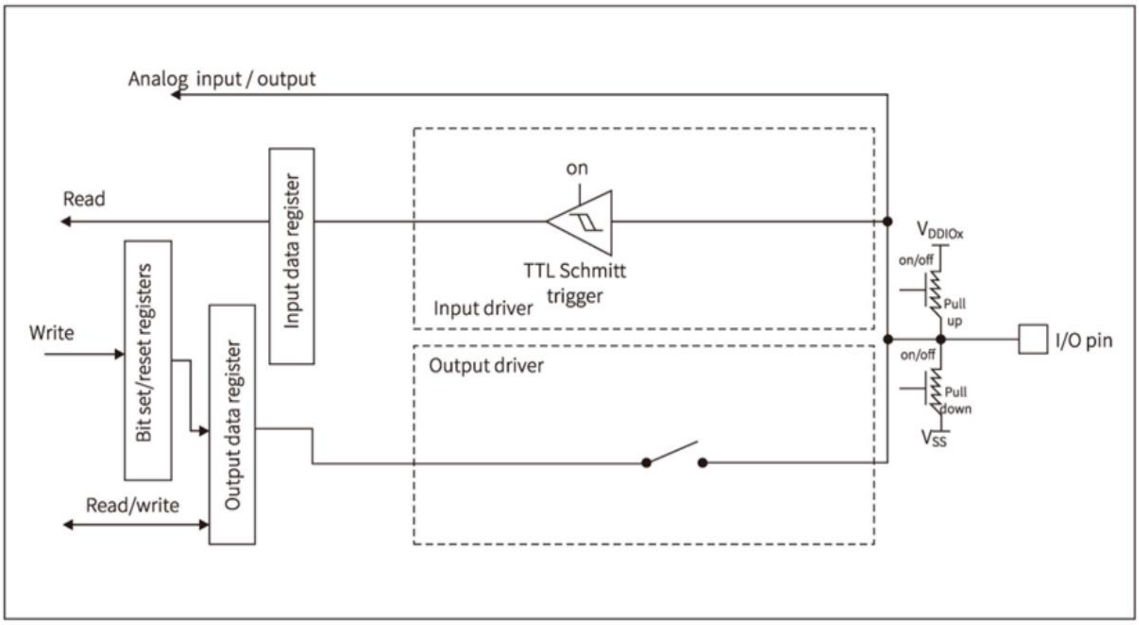


Figure 9-2 Floating input/pull-up/pull-down configuration

### 9.3.6 Output configuration

When the I/O port is configured as an output:

ÿ Output buffering enabled:

- Open-drain mode: a '0' on the output register activates the N-MOS, while a '1' on the output register puts the port in high-impedance state (P-MOS is never activated).

- Push-Pull Mode: A '0' on the output register activates the N-MOS, while a '1' on the output register will activate the P-MOS.

ÿ Schmitt trigger input is activated.

ÿ Whether the weak pull-up and weak pull-down resistors are activated depends on the value of the GPIOx_PUPDR register.

ÿData on the I/O pin is sampled into the input data register every AHB clock cycle.

ÿ Use a read access to the input data register to get the synchronized input data.

A read access to an output register is used to obtain the last value written to the register. Figure 9-3 shows the output configuration of the I/O port bits.

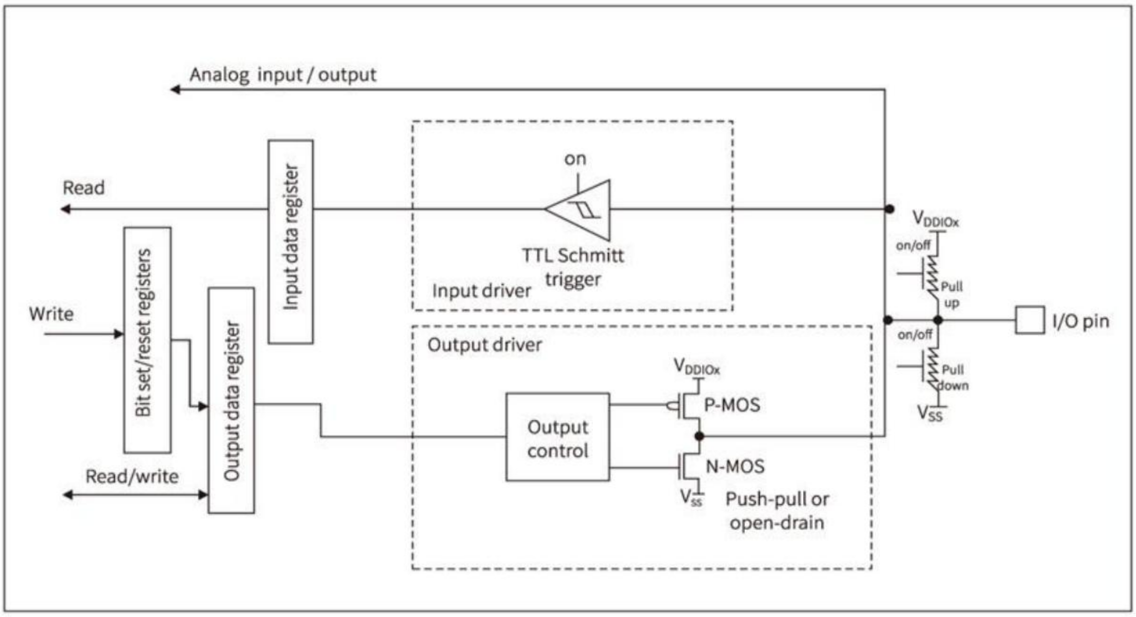


Figure 9-3 Output configuration

### 9.3.7 External interrupt/wake-up line

All ports have interrupt generation and wake-up capabilities. After entering Sleep or Deep Sleep mode, if an interrupt is detected, the WIC module block will automatically wake up the chip.

Each digital general-purpose port can be interrupted by an external signal source, which can be high level/low level/rising edge/falling Along the four types of signals, when an interrupt is triggered, it can be judged which port triggered the interrupt by querying the interrupt status register.

The corresponding interrupt status flag can be cleared by setting the interrupt clear register.

### 9.3.8 Alternate functions and remapping of I/O pins

Device I/Os are connected to embedded peripheral modules through multiplexers. Microscopically, only one pin of the multiplexing function of the peripheral is allowed to be connected to one I/O interface at the same time. Therefore, there cannot be conflicting peripheral pin assignments on the same wire.

Each I/O pin has an alternate function multiplexer, which can be implemented by configuring the GPIOx_AFR register (from pin 0 to pin 7).

After reset, all I/O ports are connected to GPIO functions. Each peripheral also has multiplexed functions mapped to different I/O pins, an approach used to optimize the number of available peripherals on a small package.

For the specific alternate function of each pin, please refer to: Table 9-2 Alternate function mapping of GPIO ports and peripheral pins

In order to use a given I/O port configuration, you must do the following:

ÿDebug function: Immediately after each device reset, these pins are configured as alternate functions to support calls. ÿ GPIO:

Configure the required I/O as output/input in GPIOx_DIRCR register. ÿ Multiplexing functions of peripherals:

-Connect the I/O to the required AFRx, AFRx is defined in the GPIOx_AFR register; -Configure the

corresponding pin's on-boarding by GPIOx_PUPDR and GPIOx_SLEWCR, GPIOx_DRVCR register

Pull/pull, output speed and output capability;

- Configure the output type through the GPIOx_OTYPER register: 0 means push-pull, 1 means open drain;

ÿAdditional features:

- For ADC/VC, configure GPIOx_AFR as 0x0F, configure the required I/O line as analog mode and set it in ADC or

Configure the desired function in the VC register.

- For additional function oscillators, configure the corresponding required functions in the associated RCC registers.

Table 9-2 Alternate function mapping of GPIO ports and peripheral pins

| encapsulation | | | GPIOx_AFR[i+3:i] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TSSOP20 | QFN20 | Config | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | f |
| 1 | 18 | | PD4 | TIM1_CH1 | PCA_CH0 | RTC_1HZ | TIM10_TOG | UART0_TXD | TIM10_EXT | BEEP | TIM2_CH1 | VCIN2 |
| 2 | 19 | | PD5 | TIM1_CH1N | PCA_CH4 | SPI_MISO | I2C_SCL | UART1_TXD | TIM10_GATE | UART0_TXD | TIM2_CH4 | AIN5 |
| 3 | 20 | | PD6 | TIM1_CH2 | PCA_CH3 | SPI_MOSI | I2C_SDA | UART1_RXD | LPTIM_EXT | UART0_RXD | TIM2_CH2 | AIN6 |
| 4 | 1 | NRST | | | | | | | | | | |
| 5 | 2 | OSC_IN | PA1 | TIM1_CH2N | | SPI_CLK | I2C_SDA | UART0_RXD | TIM10_TOG | UART1_RXD | | |
| 6 | 3 | OSC_ out | PA2 | TIM1_CH3 | | SPI_NSS | I2C_SCL | UART0_TXD | TIM10_TOGN | UART1_TXD | TIM2_CH2 | |
| 7 | 4 | VSS | | | | | | | | | | |
| 8 | 5 | VCAP | | | | | | | | | | AIN7 |
| 9 | 6 | VDD | | | | | | | | | | |
| 10 | 7 | | PA3 | TIM1_CH3N | PCA_CH2 | SPI_NSS | RTC_1HZ | LPUART_RXD | PCA_ECI | VC0_OUT | TIM2_CH3 | |
| 11 | 8 | X32K_IN | PB5 | TIM1_BKIN | PCA_CH4 | SPI_CLK | I2C_SDA | UART0_RXD | TIM11_TOG | LVD_OUT | TIM2_CH1 | |
| 12 | 9 | X32K_OUT | PB4 | LPTIM_GATE | PCA_ECI | SPI_NSS | I2C_SCL | UART0_TXD | TIM11_TOGN | | | |
| 13 | 10 | | PC3 | TIM1_CH3 | TIM1_CH1N | | I2C_SDA | UART1_TXD | PCA_CH1 | 1-WIRE | TIM2_CH3 | AIN1 |
| 14 | 11 | | PC4 | TIM1_CH4 | TIM1_CH2N | | I2C_SCL | UART1_RXD | PCA_CH0 | CLK_MCO | TIM2_CH4 | AIN2 |
| 15 | 12 | | PC5 | TIM1_BKIN | PCA_CH0 | SPI_CLK | | LPUART_TXD | TIM11_GATE | LVD_OUT | TIM2_CH1 | VCIN1 |
| 16 | 13 | | PC6 | TIM1_CH1 | PCA_CH3 | SPI_MOSI | | LPUART_RXD | TIM11_EXT | CLK_MCO | TIM2_CH4 | AIN0 |
| 17 | 14 | SWDIO | PC7 | TIM1_CH2 | PCA_CH4 | SPI_MISO | | UART1_RXD | LIRC_OUT | LXT_OUT | | |
| 18 | 15 | SWDCLK | PD1 | | PCA_ECI | | | UART1_TXD | HIRC_OUT | VC0_OUT | | |
| 19 | 16 | | PD2 | TIM1_CH2 | PCA_CH2 | SPI_MISO | RTC_1HZ | LPUART_TXD | LPTIM_TOG | 1-WIRE | | AIN3/<br>VCIN0 |
| 20 | 17 | | PD3 | TIM1_CH3N | PCA_CH1 | SPI_MOSI | HXT_OUT | UART0_RXD | LPTIM_TOGN | | TIM2_CH2 | AIN4 |

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

ÿOutput buffer can be configured in open-drain or push-pull mode

ÿ Signals from peripherals drive output buffers

ÿSchmitt trigger input is activated

ÿ Whether the weak pull-up and weak pull-down resistors are activated depends on the value of the GPIOx_PUPDR register

ÿData on the I/O pin is sampled into the input data register every AHB clock cycle

ÿ Use a read access to the input data register to get the synchronized input data

Figure 9-4 shows the alternate function configuration of the I/O port bits.

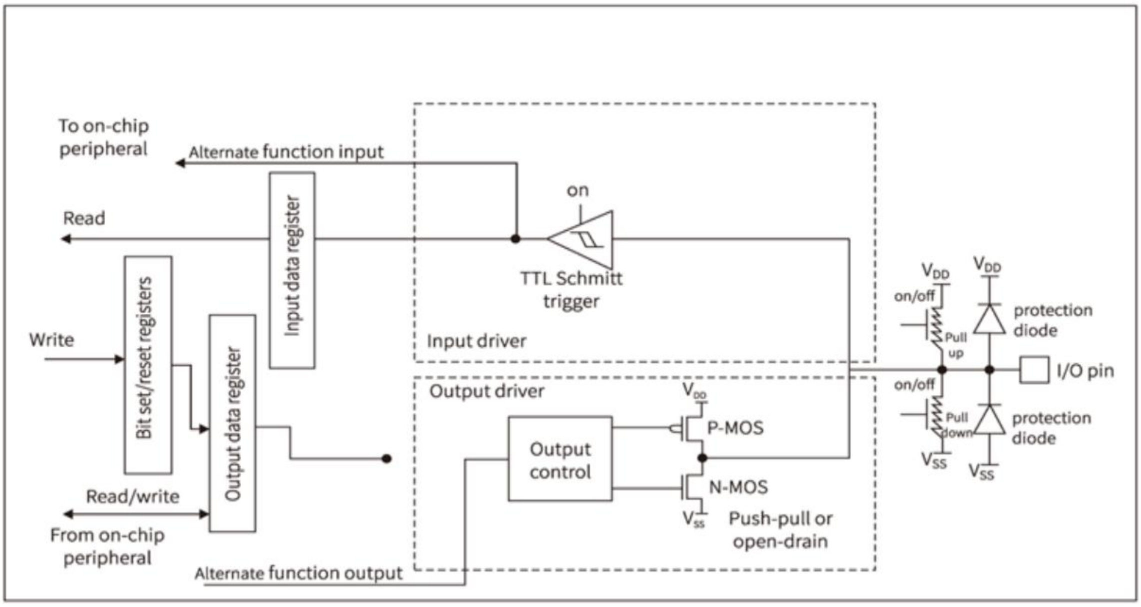


Figure 9-4 Multiplexing function configuration

### 9.3.9 Simulation configuration

When the I/O port is programmed as analog configuration (GPIOx_AFR.AFR=0x0F):

ÿ output buffer off

ÿ Schmitt trigger input disabled, enabling zero consumption on each analog I/O pin. Schmitt trigger output value is forced to '0'

ÿ Weak pull-up and pull-down resistors are disabled

ÿ When reading the input data register, the value is 0

Figure 9-5 shows the high-impedance analog configuration of the I/O port bits.

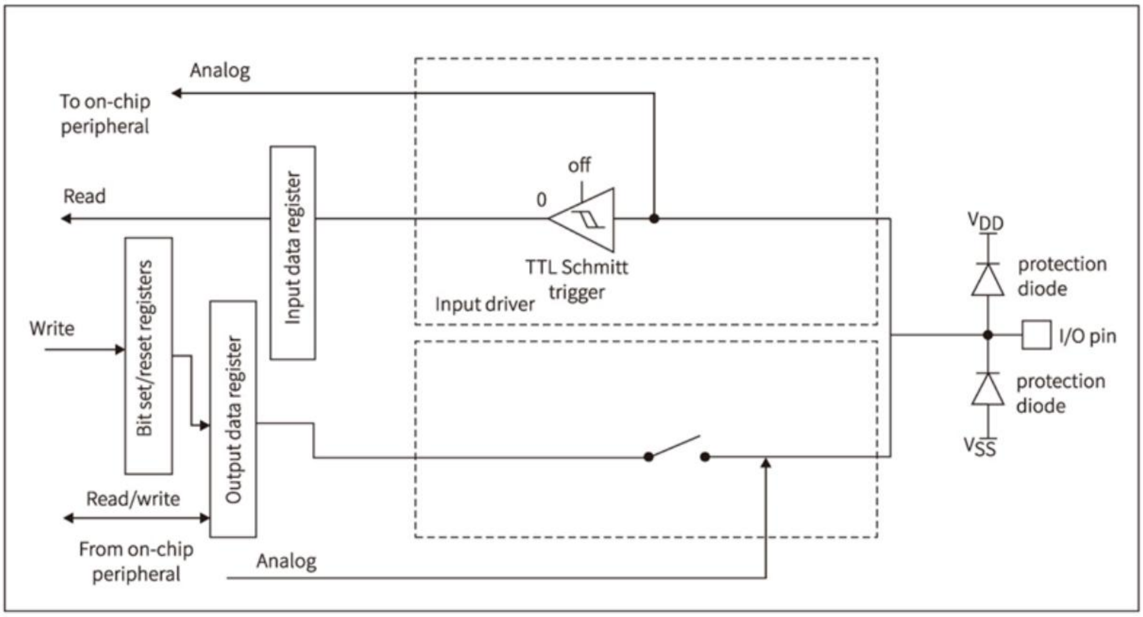


Figure 9-5 High impedance analog configuration

### 9.3.10 HXT or LXT pin used as GPIO

When the HXT or LXT oscillator is turned off (the default state after reset), the associated oscillator pin can be used as a normal GPIO port.

When the HXT or LXT oscillator is turned on (set the RCC_SYSCLKCR.HXTEN or RCC_LXTCR.LXTEN bit to turn it on), it must be configured

The corresponding pin is an analog function, the oscillator controls its related pin and the GPIO configuration of the related pin is invalid.

The method of configuring the oscillator pin as an analog function:

ÿConfigure by RCC_SYSCLKCR.HXTPORT=1 or RCC_LXTCR.LXTPORT=1

or

ÿ Configure by setting GPIOx_AFR=0xF of the corresponding pin

When the oscillator is configured as user external clock input mode (RCC_SYSCLKCR.HXTBYP=1 or

RCC_LXTCR.LXTBYP=1), there is no need to configure the corresponding pins as analog functions, only use OSC_IN or X32K_IN pins as

Clock input processing, OSC_OUT or X32K_OUT pins can still be configured as normal GPIO pins.

## 9.4 GPIO register list

x=A,B,C,D

GPIOx base address 0x4002 1000

| GPIOx | offset address | describe |
|---|---|---|
| GPIOA | 0x000 | GPIOA offset address |
| GPIOB | 0x400 | GPIOB offset address |
| GPIOC | 0x800 | GPIOC offset address |
| GPIOD | 0xC00 | GPIOD offset address |

Table 9-3 GPIOx register list and reset value

| Offset | name | describe | reset value |
|---|---|---|---|
| Address I/O Mode Register 0x00 GPIOx_DIRCR 0x04 GPIOx_OTYPER<br>Output Type Register Output | | | 0x0000 0000 |
| Data Register 0x08 GPIOx_ODR Input Data Register 0x0C | | | 0x0000 0000 |
| Enable Register 0x10 | | GPIOx_IDR Interrupt | 0x0000 0000 |
| Interrupt Raw Status GPIOx_INTEN 0x14 GPIOx_RAWINTSR | | | 0x0000 00XX |
| | Register, read-only | | 0x0000 0000 |
| | | The interrupt status can be read regardless of whether the interrupt is enabled or not. | 0x0000 0000 |
| 0x18 GPIOx_MSKINTCR interrupt status register, read only 0x1C | | | 0x0000 0000 |
| GPIOx_INTCLR interrupt clear register 0x20 GPIOx_INTTYPCR | | | 0x0000 0000 |
| interrupt type register 0x24 GPIOx_INTPOLCR interrupt type | | | 0x0000 0000 |
| value register 0x28 GPIOx_INTANY any edge trigger interrupt | | | 0x0000 0000 |
| register 0x2C GPIOx_ODSET output set register 0x30 GPIOx_ODCLR output | | | 0x0000 0000 |
| clear Register 0x34 GPIOx_INDBEN Input Debounce and Sync | | | 0x0000 0000 |
| Enable Enable register 0x38 GPIOx_DBCLKCR input debounce | | | 0x0000 0000 |
| clock configuration register 0x3C GPIOx_PUPDR pull-up/pull-down register | | | 0x0000 0000 |
| | | | 0x0000 0000 |
| | | | 0x0000 0000 |
| 0x40 GPIOx_SLEWCR Voltage slew rate control | | | GPIOA: 0x0000000E<br>GPIOB: 0x0000 0030<br>GPIOC: 0x000000F8<br>GPIOD: 0x0000007E |
| 0x44 GPIOx_DRVCR drive strength configuration | | | 0x0000 0000 |
| 0x48 GPIOx_AFR | | Alternate function register | 0x0000 0000 |

0: Indicates a logical value of 0

1: Indicates logical value 1

X: Indicates uncertain

## 9.5 GPIO register description

### 9.5.1 GPIO Port Direction Register (GPIOx_DIRCR) (x = A..D)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | PxDIR7 | PxDIR6 | PxDIR5 | PxDIR4 | PxDIR3 | PxDIR2 | PxDIR1 | PxDIR0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxDIR7 | Terminal Px7 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 6 | PxDIR6 | Terminal Px6 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 5 | PxDIR5 | Terminal Px5 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 4 | PxDIR4 | Terminal Px4 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 3 | PxDIR3 | Terminal Px3 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 2 | PxDIR2 | Terminal Px2 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 1 | PxDIR1 | Terminal Px1 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |
| 0 | PxDIR0 | Terminal Px0 input/output direction selection bit<br>0: input mode<br>1: output mode | 0 | R/W |

### 9.5.2 GPIO port output type register (GPIOx_OTYPER) (x = A..D)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya<br>YP7 | wxya<br>YP6 | wxya<br>YP5 | wxya<br>YP4 | wxya<br>YP3 | wxya<br>YP2 | wxya<br>YP1 | wxya<br>YP0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:8 | – | | 0x0 | – |
| 7 | PxOTYP7 | Reserved terminal Px7 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 6 | PxOTYP6 | Terminal Px6 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 5 | PxOTYP5 | Terminal Px5 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 4 | PxOTYP4 | Terminal Px4 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 3 | PxOTYP3 | Terminal Px3 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 2 | PxOTYP2 | Terminal Px2 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 1 | PxOTYP1 | Terminal Px1 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |
| 0 | PxOTYP0 | Terminal Px0 output type control bit<br>0: push-pull output (reset state)<br>1: open drain output | 0 | R/W |

### 9.5.3 GPIO port output data register (GPIOx_ODR) (x = A..D)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | wxya<br>7 | wxya<br>6 | wxya<br>5 | wxya<br>4 | wxya<br>3 | wxya<br>2 | wxya<br>1 | wxya<br>0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For the setting/clearing of atomic bits, it can be realized by operating GPIOx_ODSET and GPIOx_ODCLR(x=A..D) registers independently.

| Bit | Flag | Functional Description | reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | Reserved | 0x0 | – |
| 7 | PxOD7 | terminal Px7 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 6 | PxOD6 | Terminal Px6 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 5 | PxOD5 | Terminal Px5 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 4 | PxOD4 | Terminal Px4 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 3 | PxOD3 | Terminal Px3 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 2 | PxOD2 | Terminal Px2 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 1 | PxOD1 | Terminal Px1 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |
| 0 | PxOD0 | Terminal Px0 output value configuration bit<br>0: output low level<br>1: output high level, if it is an open-drain output, it needs to be configured or connected with an external pull-up resistor | 0 | R/W |

### 9.5.4 GPIO Port Input Data Register (GPIOx_IDR) (x = A..D)

Offset address: 0x0C

Reset value: 0x0000 00xx

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | Reserve | 0x0 | – |
| 7 | PxID7 | the input value of terminal Px7<br>0: input low level<br>1: input high level | x | RO |
| 6 | PxID6 | Terminal Px6 input value<br>0: input low level<br>1: input high level | x | RO |
| 5 | PxID5 | Terminal Px5 input value<br>0: input low level<br>1: input high level | x | RO |
| 4 | PxID4 | Terminal Px4 input value<br>0: input low level<br>1: input high level | x | RO |
| 3 | PxID3 | Terminal Px3 input value<br>0: input low level<br>1: input high level | x | RO |
| 2 | PxID2 | Terminal Px2 input value<br>0: input low level<br>1: input high level | x | RO |
| 1 | PxID1 | Terminal Px1 input value<br>0: input low level<br>1: input high level | x | RO |
| 0 | PxID0 | Terminal Px0 input value<br>0: input low level<br>1: input high level | x | RO |

Note: x represents an indeterminate value

### 9.5.5 GPIO Port Interrupt Enable Register (GPIOx_INTEN) (x = A..D)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya<br>N7 | wxya<br>N6 | wxya<br>N5 | wxya<br>N4 | wxya<br>N3 | wxya<br>N2 | wxya<br>N1 | wxya<br>N0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | | 0x0 | – |
| 7 | PxIEN7 | Reserved terminal Px7 Interrupt mask release bit<br>0: Px7 terminal interrupt shielding<br>1: Px7 terminal interrupt is valid | 0 | R/W |
| 6 | PxIEN6 | Terminal Px6 interrupt mask release bit<br>0: Px6 terminal interrupt shielding<br>1: Px6 terminal interrupt is valid | 0 | R/W |
| 5 | PxIEN5 | Terminal Px5 interrupt mask release bit<br>0: Px5 terminal interrupt shielding<br>1: Px5 terminal interrupt is valid | 0 | R/W |
| 4 | PxIEN4 | Terminal Px4 interrupt mask release bit<br>0: Px4 terminal interrupt shielding<br>1: Px4 terminal interrupt is valid | 0 | R/W |
| 3 | PxIEN3 | Terminal Px3 interrupt mask release bit<br>0: Px3 terminal interrupt shielding<br>1: Px3 terminal interrupt is valid | 0 | R/W |
| 2 | PxIEN2 | Terminal Px2 interrupt mask release bit<br>0: Px2 terminal interrupt shielding<br>1: Px2 terminal interrupt is valid | 0 | R/W |
| 1 | PxIEN1 | Terminal Px1 interrupt mask release bit<br>0: Px1 terminal interrupt shielding<br>1: Px1 terminal interrupt is valid | 0 | R/W |
| 0 | PxIEN0 | Terminal Px0 interrupt mask release bit<br>0: Px0 terminal interrupt mask<br>1: Px0 terminal interrupt is valid | 0 | R/W |

### 9.5.6 GPIO Port Interrupt Raw Status Register (GPIOx_RAWINTSR) (x = A..D)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | QUR S7 | QUR S6 | QUR S5 | QUR S4 | QUR S3 | QUR S2 | QUR S1 | QUR S0 |
| | | | | | | | | RO | RO | RO | RO | RO | RO | RO | RO |

The interrupt status can be read regardless of whether the interrupt is enabled or not.

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxRIS7 | Terminal Px7 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 6 | PxRIS6 | Terminal Px6 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 5 | PxRIS5 | Terminal Px5 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 4 | PxRIS4 | Terminal Px4 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 3 | PxRIS3 | Terminal Px3 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 2 | PxRIS2 | Terminal Px2 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 1 | PxRIS1 | Terminal Px1 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 0 | PxRIS0 | Terminal Px0 interrupt raw status bit<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |

### 9.5.7 GPIO Port Interrupt Status Register (GPIOx_MSKINTSR) (x = A..D)

Offset address: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya<br>S7 | wxya<br>S6 | wxya<br>S5 | wxya<br>S4 | wxya<br>S3 | wxya<br>S2 | wxya<br>S1 | wxya<br>S0 |
| | | | | | | | | RO | RO | RO | RO | RO | RO | RO | RO |

These bits are read-only and are set by hardware. The interrupt status can only be read if the interrupt is enabled.

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxMIS7 | Interrupt status bit after masking of terminal Px7<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 6 | PxMIS6 | Interrupt status bit after masking of terminal Px6<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 5 | PxMIS5 | Interrupt status bit after terminal Px5 is shielded<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 4 | PxMIS4 | Interrupt status bit after terminal Px4 is masked<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 3 | PxMIS3 | Interrupt status bit after masking of terminal Px3<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 2 | PxMIS2 | Interrupt status bit after masking of terminal Px2<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 1 | PxMIS1 | Interrupt status bit after masking of terminal Px1<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |
| 0 | PxMIS0 | Interrupt status bit after terminal Px0 is masked<br>0: No interrupt occurs<br>1: Interrupt occurs | 0 | RO |

### 9.5.8 GPIO Port Interrupt Clear Register (GPIOx_INTCLR) (x = A..D)

Offset address: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | PxC LR7 | PxC LR6 | PxC LR5 | PxC LR4 | PxC LR3 | PxC LR2 | PxC LR1 | PxC LR0 |
| | | | | | | | | WO WO | WO WO | WO WO | WO WO | | | | |

These bits are write only and are used by software to clear interrupts.

| bit | mark | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxICLR7 | Write 1 to clear the interrupt status of terminal Px7<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 6 | PxICLR6 | Write 1 to clear the interrupt status of terminal Px6<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 5 | PxICLR5 | Write 1 to clear the interrupt status of terminal Px5<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 4 | PxICLR4 | Write 1 to clear the interrupt status of terminal Px4<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 3 | PxICLR3 | Write 1 to clear the interrupt status of terminal Px3<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 2 | PxICLR2 | Write 1 to clear the interrupt status of terminal Px2<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 1 | PxICLR1 | Write 1 to clear the interrupt status of terminal Px1<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |
| 0 | PxICLR0 | Write 1 to clear the interrupt status of terminal Px0<br>0: Reserve the interrupt flag bit<br>1: Clear the corresponding interrupt flag bit | 0 | WO |

### 9.5.9 GPIO Port Interrupt Type Register (GPIOx_INTTYPCR) (x = A..D)

Offset address: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | PxIT<br>YPE7 | PxIT<br>YPE6 | PxIT<br>YPE5 | PxIT<br>YPE4 | PxIT<br>YPE3 | PxIT<br>YPE2 | PxIT<br>YPE1 | PxIT<br>YPE0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxITYPE7 | Interrupt type configuration bit of terminal Px7<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 6 | PxITYPE6 | Interrupt type configuration bit of terminal Px6<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 5 | PxITYPE5 | Interrupt type configuration bit of terminal Px5<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 4 | PxITYPE4 | Interrupt type configuration bit of terminal Px4<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 3 | PxITYPE3 | Interrupt type configuration bit of terminal Px3<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 2 | PxITYPE2 | Interrupt type configuration bit of terminal Px2<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 1 | PxITYPE1 | Interrupt type configuration bit of terminal Px1<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |
| 0 | PxITYPE0 | Interrupt type configuration bit of terminal Px0<br>0: Edge trigger interrupt type<br>1: Level trigger interrupt type | 0 | R/W |

### 9.5.10 GPIO Port Interrupt Type Register (GPIOx_INTPOLCR) (x = A..D)

Offset address: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | wxya<br>AL7 | wxya<br>AL6 | wxya<br>AL5 | wxya<br>AL4 | wxya<br>AL3 | wxya<br>AL2 | wxya<br>AL1 | wxya<br>AL0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value reac | and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxIVAL7 | Interrupt polarity configuration bit for terminal Px7<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 6 | PxIVAL6 | Interrupt polarity configuration bit for terminal Px6<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 5 | PxIVAL5 | Interrupt polarity configuration bit for terminal Px5<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 4 | PxIVAL4 | Interrupt polarity configuration bit for terminal Px4<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 3 | PxIVAL3 | Interrupt polarity configuration bit for terminal Px3<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 2 | PxIVAL2 | Interrupt polarity configuration bit for terminal Px2<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 1 | PxIVAL1 | Interrupt polarity configuration bit for terminal Px1<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |
| 0 | PxIVAL0 | Interrupt polarity configuration bit for terminal Px0<br>0: low level or falling edge trigger interrupt<br>1: High level or rising edge trigger interrupt | 0 | R/W |

### 9.5.11 GPIO Port Any Edge Triggered Interrupt Register (GPIOx_INTANY) (x = A..D)

Offset address: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya<br>NY7 | wxya<br>NY6 | wxya<br>NY5 | wxya<br>NY4 | wxya<br>NY3 | wxya<br>NY2 | wxya<br>NY1 | wxya<br>NY0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxIANY7 | Any edge trigger interrupt configuration bit of terminal Px7<br>0: Interrupt trigger edge is determined by PxIVAL7<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 6 | PxIANY6 | Any edge trigger interrupt configuration bit of terminal Px6<br>0: Interrupt trigger edge is determined by PxIVAL6<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 5 | PxIANY5 | Any edge trigger interrupt configuration bit of terminal Px5<br>0: Interrupt trigger edge is determined by PxIVAL5<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 4 | PxIANY4 | Any edge trigger interrupt configuration bit of terminal Px4<br>0: Interrupt trigger edge is determined by PxIVAL4<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 3 | PxIANY3 | Any edge trigger interrupt configuration bit of terminal Px3<br>0: Interrupt trigger edge is determined by PxIVAL3<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 2 | PxIANY2 | Any edge trigger interrupt configuration bit of terminal Px2<br>0: Interrupt trigger edge is determined by PxIVAL2<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 1 | PxIANY1 | Any edge trigger interrupt configuration bit of terminal Px1<br>0: Interrupt trigger edge is determined by PxIVAL1<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |
| 0 | PxIANY0 | Terminal Px0 any edge trigger interrupt configuration bit<br>0: Interrupt trigger edge is determined by PxIVAL0<br>1: Both rising and falling edges trigger interrupt | 0 | R/W |

## 9.5.12 GPIO port output set register (GPIOx_ODSET) (x = A..D)

Offset address: 0x2C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya SET7 | wxya SET6 | wxya SET5 | wxya SET4 | wxya SET3 | wxya SET2 | wxya SET1 | wxya SET0 |
| | | | | | | | | WO WO | WO WO | WO WO | WO WO WO | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxODSET7 | Terminal Px7 output set 1 control bit, when read<br>0: output hold<br>1: output is set high | 0 | WO |
| 6 | PxODSET6 | Terminal Px6 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 5 | PxODSET5 | Terminal Px5 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 4 | PxODSET4 | Terminal Px4 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 3 | PxODSET3 | Terminal Px3 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 2 | PxODSET2 | Terminal Px2 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 1 | PxODSET1 | Terminal Px1 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |
| 0 | PxODSET0 | Terminal Px0 output set 1 control bit<br>0: output hold<br>1: output is set high | 0 | WO |

### 9.5.13 GPIO port output clear register (GPIOx_ODCLR) (x = A..D)

Offset address: 0x30

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya CLR7 | wxya CLR6 | wxya CLR5 | wxya CLR4 | wxya CLR3 | wxya CLR2 | wxya CLR1 | wxya CLR0 |
| | | | | | | | | WO WO | WO WO | WO WO | WO WO | | | | |

If ODSETx and ODCLRx are set at the same time, ODSETx has priority.

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxODCLR7 | Terminal Px7 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 6 | PxODCLR6 | Terminal Px6 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 5 | PxODCLR5 | Terminal Px5 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 4 | PxODCLR4 | Terminal Px4 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 3 | PxODCLR3 | Terminal Px3 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 2 | PxODCLR2 | Terminal Px2 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 1 | PxODCLR1 | Terminal Px1 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |
| 0 | PxODCLR0 | Terminal Px0 output clear 0 control bit<br>0: output hold<br>1: Reset the corresponding ODRx bit | 0 | WO |

Machine Translated by Google

### 9.5.14 GPIO Port Input Debounce Register (GPIOx_INDBEN) (x = A..D)

Offset address: 0x34

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | SYNC_EN | wxya | | | | | | | |
| | | | | | | | R/W | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:9 | – | reserve | 0x0 | – |
| 8 | SYNC_EN | When the debounce is not enabled, set this register to configure whether the input uses 2-level synchronization to eliminate Metastable (only valid for interrupts).<br>0: Do not use two-level synchronization<br>1: Use two-level synchronization | 0 | R/W |
| 7:0 | wxya<br>n=(0~7) | Terminal Pxn (n=0~7) debounce enable configuration bit, if the input signal cannot be If the debounce sampling period is sampled, the input signal is regarded as a signal jitter and does not trigger an interrupt. Only for edge-triggered "edge-trigger" interrupts, not for level-triggered ("level trigger") interrupt. In level mode, it is a two-level synchronous input<br>0: Disable port debounce function<br>1: Enable port debounce function | 0 | R/W |

### 9.5.15 GPIO Port Input Debounce Clock Configuration Register (GPIOx_DBCLKCR) (x = A..D)

Offset address: 0x38

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserve</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="11" rowspan="2">reserve</td><td>DBCL<br>KEN</td><td colspan="4">DBCLK_DIV[3:0]</td></tr>
<tr><td>R/W</td><td colspan="4">R/W</td></tr>
</table>

<table>
<tr><th>bit flag</th><th></th><th>Functional description</th><th>Reset value</th><th>read and write</th></tr>
<tr><td>31:5 –</td><td></td><td>reserve</td><td>0x0</td><td>–</td></tr>
<tr><td>4</td><td>DBCLKEN</td><td>Whether to enable the debounced clock<br>0: Disable debounced clock<br>1: Enable debounced clock</td><td>0</td><td>R/W</td></tr>
<tr><td>3:0</td><td>DBCLK_DIV</td><td>Debounce Sampling Period Selection<br>The debounce clock is the (2^dbclk_div[3:0]) frequency division of hclk .<br>Debounce sampling period selection:
<table>
<tr><th>DBCLK_DIV Description</th><th></th></tr>
<tr><td>0x0</td><td>Debounce sampling 1 HCLK cycle 1 time</td></tr>
<tr><td>0x1</td><td>Debounce sampling 1 time in 2 HCLK cycles</td></tr>
<tr><td>0x2</td><td>Debounce sampling once in 4 HCLK cycles</td></tr>
<tr><td>0x3</td><td>Debounce sampling once in 8 HCLK cycles</td></tr>
<tr><td>0x4</td><td>Debounce sampling once in 16 HCLK cycles</td></tr>
<tr><td>0x5</td><td>Debounce sampling once in 32 HCLK cycles</td></tr>
<tr><td>0x6</td><td>Debounce sampling once in 64 HCLK cycles</td></tr>
<tr><td>0x7</td><td>Debounce sampling once every 128 HCLK cycles</td></tr>
<tr><td>0x8</td><td>Debounce sampling once every 256 HCLK cycles</td></tr>
<tr><td>0x9</td><td>Debounce sampling once every 512 HCLK cycles</td></tr>
<tr><td>0xA</td><td>Debounce sampling once in 1024 HCLK cycles</td></tr>
<tr><td>0xB</td><td>Debounce sampling 2*1024 HCLK cycles 1 time</td></tr>
<tr><td>0xC</td><td>Debounce sampling 4*1024 HCLK cycle 1 time</td></tr>
<tr><td>0xD</td><td>Debounce sampling 8*1024 HCLK cycle 1 time</td></tr>
<tr><td>0xE</td><td>Debounce sampling 16*1024 HCLK cycle 1 time</td></tr>
<tr><td>0xF</td><td>Debounce sampling 32*1024 HCLK cycle 1 time</td></tr>
</table></td><td>0</td><td>R/W</td></tr>
</table>

### 9.5.16 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..D)

Offset address: 0x3C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PxPUPD7[1:0] | | PxPUPD6[1:0] | | PxPUPD5[1:0] | | PxPUPD4[1:0] | | PxPUPD3[1:0] | | PxPUPD2[1:0] | | PxPUPD1[1:0] | | PxPUPD0[1:0] | |
| R/W | | R/W | | R/W | | R/W | | R/W | | R/W | | R/W | | R/W | |

| bit flag | | Function | Reset value read | and write |
|---|---|---|---|---|
| 31:16 – | | | 0x0 | – |
| 15:14 PxPUPD7 | | description Reserved terminal Px7 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 13:12 PxPUPD6 | | Terminal Px6 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 11:10 PxPUPD5 | | Terminal Px5 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enable<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 9:8 | PxPUPD4 | Terminal Px4 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enable<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 7:6 | PxPUPD3 | Terminal Px3 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 5:4 | PxPUPD2 | Terminal Px2 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 3:2 | PxPUPD1 | Terminal Px1 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |
| 1:0 | PxPUPD0 | Terminal Px0 pull-up/pull-down configuration control bit<br>00: Pull up, pull down disabled<br>01: pull-up enabled<br>10: pull-down enable<br>11: reserved | 0x0 | R/W |

### 9.5.17 GPIO port voltage slew rate configuration (GPIOx_SLEWCR) (x = A..D)

Offset address: 0x40

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | wxya 7 | wxya 6 | wxya 5 | wxya 4 | wxya 3 | wxya 2 | wxya 1 | wxya 0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:8 | – | | 0x0 | – |
| 7 | PxSR7 | Reserved terminal Px7 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 6 | PxSR6 | Terminal Px6 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 5 | PxSR5 | Terminal Px5 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 4 | PxSR4 | Terminal Px4 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 3 | PxSR3 | Terminal Px3 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 2 | PxSR2 | Terminal Px2 voltage slew rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 1 | PxSR1 | Terminal Px1 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |
| 0 | PxSR0 | Terminal Px0 voltage conversion rate configuration control bit<br>0: High voltage slew rate<br>1: Low voltage slew rate | Note 1 | R/W |

Note 1: For the reset value, please refer to Table 9-3 GPIOx register list and reset value

### 9.5.18 GPIO Port Drive Strength Configuration Register (GPIOx_DRVCR) (x = A..D)

Offset address: 0x44

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | PxD<br>V7 | PxD<br>V6 | PxD<br>V5 | PxD<br>V4 | PxD<br>V3 | PxD<br>V2 | PxD<br>V1 | PxD<br>V0 |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value reac | and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | PxDRV7 | Terminal Px7 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 6 | PxDRV6 | Terminal Px6 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 5 | PxDRV5 | Terminal Px5 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 4 | PxDRV4 | Terminal Px4 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 3 | PxDRV3 | Terminal Px3 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 2 | PxDRV2 | Terminal Px2 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 1 | PxDRV1 | Terminal Px1 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |
| 0 | PxDRV0 | Terminal Px0 drive strength selection control bit<br>0: high drive strength<br>1: Low drive strength | 0 | R/W |

### 9.5.19 GPIO Port Alternate Function Register (GPIOx_AFR)(x = A..D)

Offset address: 0x48

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PxAFR7[3:0] | | | | PxAFR6[3:0] | | | | PxAFR5[3:0] | | | | PxAFR4[3:0] | | | |
| R/W | | | | R/W | | | | R/W | | | | R/W | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PxAFR3[3:0] | | | | PxAFR2[3:0] | | | | PxAFR1[3:0] | | | | PxAFR0[3:0] | | | |
| R/W | | | | R/W | | | | R/W | | | | R/W | | | |

#### 9.5.19.1 GPIOA multiplexing configuration

| bit flag | | Functional description | Reset value reac | and write |
|---|---|---|---|---|
| 31:16 | – | | 0x0 | – |
| 15:12 | PAAFR3 | Reserved port PA3 function selection<br>0000: PA3<br>0001: TIM1_CH3N<br>0010: PCA_CH2<br>0011: SPI_NSS<br>0100: RTC_1HZ<br>0101: LPUART_RXD<br>0110: PCA_ECI<br>0111: VC0_OUT<br>1000: TIM2_CH3<br>1001~1111: reserved | 0x0 | R/W |
| 11:8 | PAAFR2 | Port PA2 function selection<br>0000: PA2<br>0001: TIM1_CH3<br>0010: reserved<br>0011: SPI_NSS<br>0100: I2C_SCL<br>0101: UART0_TXD<br>0110: TIM10_TOGN<br>0111: UART1_TXD<br>1000: TIM2_CH2<br>1001~1110: reserved<br>1111: OSC_OUT | 0x0 | R/W |
| 7:4 | PAAFR1 | Port PA1 function selection<br>0000: PA1<br>0001: TIM1_CH2N<br>0010: reserved<br>0011: SPI_CLK<br>0100: I2C_SDA<br>0101: UART0_RXD<br>0110: TIM10_TOG<br>0111: UART1_RXD<br>1000: reserved<br>1001~1110: reserved<br>1111: OSC_IN | 0x0 | R/W |
| 3:0 | – | reserve | 0x0 | – |

### 9.5.19.2 GPIOB multiplexing configuration

| bit | mark | Functional description | Reset value rea | d and write |
|---|---|---|---|---|
| 31:24 | – | reserve | 0x0 | – |
| 23:20 | PBAFR5 | Port PB5 function selection<br>0000: PB5<br>0001: TIM1_BKIN<br>0010: PCA_CH4<br>0011: SPI_CLK<br>0100: I2C_SDA<br>0101: UART0_RXD<br>0110: TIM11_TOG<br>0111: LVD_OUT<br>1000: TIM2_CH1<br>1001~1110: reserved<br>1111: X32K_IN | 0x0 | R/W |
| 19:16 | PBAFR4 | Port PB4 function selection<br>0000: PB4<br>0001: LPTIM_GATE<br>0010: PCA_ECI<br>0011: SPI_NSS<br>0100: I2C_SCL<br>0101: UART0_TXD<br>0110: TIM11_TOGN<br>0111: reserved<br>1000: reserved<br>1001~1110: reserved<br>1111: X32K_OUT | 0x0 | R/W |
| 15:0 | – | reserve | 0x0 | – |

### 9.5.19.3 GPIOC multiplexing configuration

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:28 | PCAFR7 | Port PC7 function selection<br>0000: PC7<br>0001: TIM1_CH2<br>0010: PCA_CH4<br>0011: SPI_MISO<br>0100: reserved<br>0101: UART1_RXD<br>0110: LIRC_OUT<br>0111: LXT_OUT<br>1000: reserved<br>1001~1111: reserved | 0x0 | R/W |
| 27:24 | PCAFR6 | port PC6 function selection<br>0000: PC6<br>0001: TIM1_CH1<br>0010: PCA_CH3<br>0011: SPI_MOSI<br>0100: reserved<br>0101: LPUART_RXD<br>0110: TIM11_EXT<br>0111: CLK_MCO<br>1000: TIM2_CH4<br>1001~1110: reserved<br>1111: AIN0 | 0x0 | R/W |
| 23:20 | PCAFR5 | port PC5 function selection<br>0000: PC5<br>0001: TIM1_BKIN<br>0010: PCA_CH0<br>0011: SPI_CLK<br>0100: reserved<br>0101: LPUART_TXD<br>0110: TIM11_GATE<br>0111: LVD_OUT<br>1000: TIM2_CH1<br>1001~1110: reserved<br>1111: VCIN1 | 0x0 | R/W |
| 19:16 | PCAFR4 | port PC4 function selection<br>0000: PC4<br>0001: TIM1_CH4<br>0010: TIM1_CH2N<br>0011: reserved<br>0100: I2C_SCL<br>0101: UART1_RXD<br>0110: PCA_CH0<br>0111: CLK_MCO<br>1000: TIM2_CH4<br>1001~1110: reserved<br>1111: AIN2 | 0x0 | R/W |

| | | | | |
|---|---|---|---|---|
| 15:12 | PCAFR3 | Port PC3 function selection<br>0000: PC3<br>0001: TIM1_CH3<br>0010: TIM1_CH1N<br>0011: reserved<br>0100: I2C_SDA<br>0101: UART1_TXD<br>0110: PCA_CH1<br>0111: 1-WIRE<br>1000: TIM2_CH3<br>1001~1110: reserved<br>1111: AIN1 | 0x0 | R/W |
| 11:0 | – | reserve | 0x0 | – |

9.5.19.4 GPIOD multiplexing configuration

| bit | mark | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:28 | – | reserve | 0x0 | – |
| 27:24 | PDAFR6 | Port PD6 function selection<br>0000: PD6<br>0001: TIM1_CH2<br>0010: PCA_CH3<br>0011: SPI_MOSI<br>0100: I2C_SDA<br>0101: UART1_RXD<br>0110: LPTIM_EXT<br>0111: UART0_RXD<br>1000: TIM2_CH2<br>1001~1110: reserved<br>1111: AIN6 | 0x0 | R/W |
| 23:20 | PDAFR5 | Port PD5 function selection<br>0000: PD5<br>0001: TIM1_CH1N<br>0010: PCA_CH4<br>0011: SPI_MISO<br>0100: I2C_SCL<br>0101: UART1_TXD<br>0110: TIM10_GATE<br>0111: UART0_TXD<br>1000: TIM2_CH4<br>1001~1110: reserved<br>1111: AIN5 | 0x0 | R/W |
| 19:16 | PDAFR4 | port PD4 function selection<br>0000: PD4<br>0001: TIM1_CH1<br>0010: PCA_CH0<br>0011: RTC_1HZ<br>0100: TIM10_TOG<br>0101: UART0_TXD<br>0110: TIM10_EXT<br>0111: BEEP<br>1000: TIM2_CH1<br>1001~1110: reserved<br>1111: VCIN2 | 0x0 | R/W |
| 15:12 | PDAFR3 | Port PD3 function selection<br>0000: PD3<br>0001: TIM1_CH3N<br>0010: PCA_CH1<br>0011: SPI_MOSI<br>0100: HXT_OUT<br>0101: UART0_RXD<br>0110: LPTIM_TOGN<br>0111: Reserved<br>1000: TIM2_CH2<br>1001~1110: reserved<br>1111: AIN4 | 0x0 | R/W |

| | | | | |
|---|---|---|---|---|
| 11:8 | PDAFR2 | Port PD2 function selection<br>0000: PD2<br>0001: TIM1_CH2<br>0010: PCA_CH2<br>0011: SPI_MISO<br>0100: RTC_1HZ<br>0101: LPUART_TXD<br>0110: LPTIM_TOG<br>0111: 1-WIRE<br>1000~1110: reserved<br>1111: VCIN0/AIN3 port | 0x0 | R/W |
| 7:4 | PDAFR1 | PD1 function selection<br>0000: PD1<br>0001: Reserved<br>0010: PCA_ECI<br>0011: Reserved<br>0100: Reserved<br>0101: UART1_TXD<br>0110: HIRC_OUT<br>0111: VC0_OUT<br>1000: reserved<br>1001~1110: reserved<br>1111: reserved | 0x0 | R/W |
| 3:0 | – | reserve | 0x0 | – |

# 10 Flash controller (Flash)

## 10.1 Flash Controller Overview

This chip contains a 64K / 32K Byte embedded Flash memory, including a 128 / 64 sector MainArray area,

An 8sector NVR area. The capacity of each sector is 512Byte. The MainArray area of Flash is for users,

Programs and data developed by users can be stored. In the NVR area, one sector is used to store system configuration, and one sector is used to store options

bytes, and the remaining 6 sectors are used to store the ISP program of the system. This module supports erasing, programming and reading of Flash memory

operate. In addition, this module supports the protection of flash memory erasing and writing, and the writing protection of control registers.

## 10.2 Flash Architecture Block Diagram

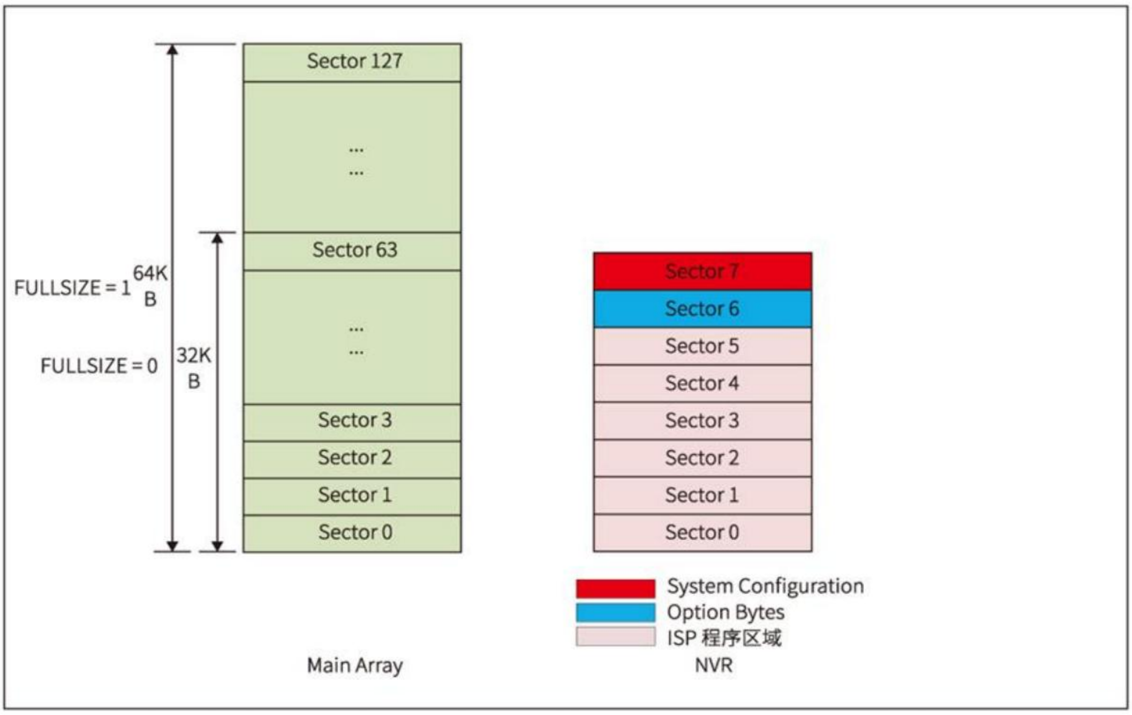


Figure 10-1 Flash structure block diagram

The MainArray area is used to store user codes.

Sector 0~5 in the NVR area These 6 sectors are used to store the system ISP (in-system programmable) code provided by the manufacturer, the user can pass

The application program developed by downloading the ISP code, the user program cannot read or erase the ISP code. of the NVR area

Sector 6 is the option byte area, which is used for the user to realize the configuration of some system functions. Sector 7 in the NVR area is the system configuration area, use

Used to store system configuration values.

## 10.3 Functional description

This controller supports read and write operations on three bit widths of Flash: byte (8bits), half-word (16bits), and word (32bits). Notice, The address of byte operation must be byte-aligned, and the target address of half-word operation must be half-word aligned (the lowest bit of the address is 0), the target address of the word operation must be word-aligned (the lowest two bits of the address are 0). If the address of the read and write operation is not according to the bit width If the alignment is specified, this operation is invalid, and the system will enter a hard fault error interrupt.

### 10.3.1 Erase operation

#### 10.3.1.1 Sector Erase

Sector erase operation steps are shown in the figure below:

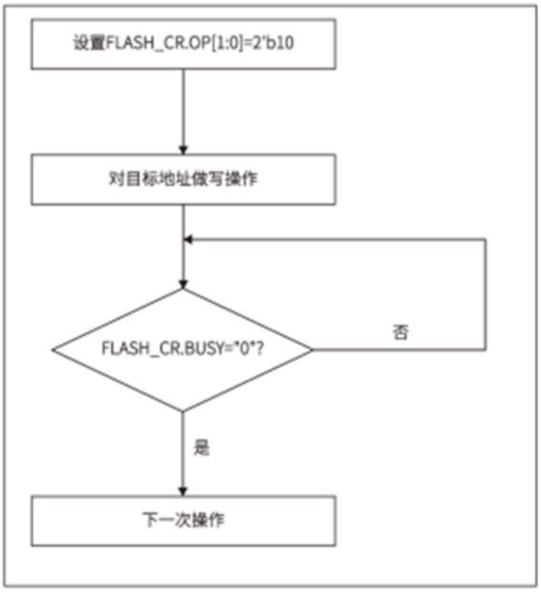


Figure 10-2 Sector erase operation steps

Notice:

1. The controller ignores the lower 9 bits of the target address as long as the target address falls within the address range of the page.
2. The write operation is used to trigger the sector erase operation, and the written data can be any data.
3. If the current sector erase command is executed in Flash, the CPU will suspend the action of fetching and executing instructions, and wait for Continue to fetch and execute instructions after the BUSY state of Flash is over.
4. If the current sector erase command is executed in RAM, the CPU will not stop fetching and executing instructions. Before operation, the software must judge whether the BUSY state of Flash is over.

#### 10.3.1.2 Chip erase

Chip erase operation steps are shown in the figure below:

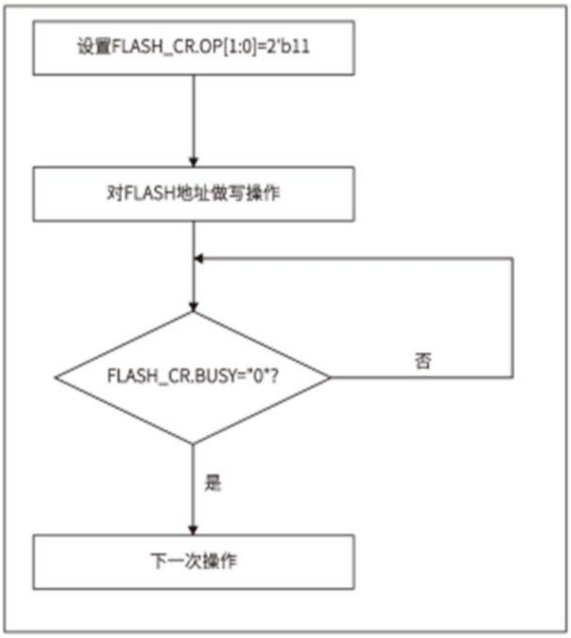


Figure 10-3 chip erase operation steps

Notice:

1. Chip erasing can only be performed on the main array of Flash, and only sector erasing can be performed on the NVR area;
2. The controller ignores the lower 15 bits of the target address, as long as the target address falls within the Flash address range;
3. The write operation is used to trigger the chip erase operation, and the written data can be any data;
4. If the current chip erase command is executed in the Flash, the CPU will suspend the action of fetching and executing the command, and wait automatically.

   Continue to fetch and execute instructions after the BUSY state of Flash is over.
5. If the current chip erase command is executed in RAM, the CPU will not stop fetching and executing instructions, and it will not stop when performing any operation on Flash.

   Before operation, the software must judge whether the BUSY state of Flash is over.

### 10.3.2 Write operation

The write operation steps are shown in the figure below:

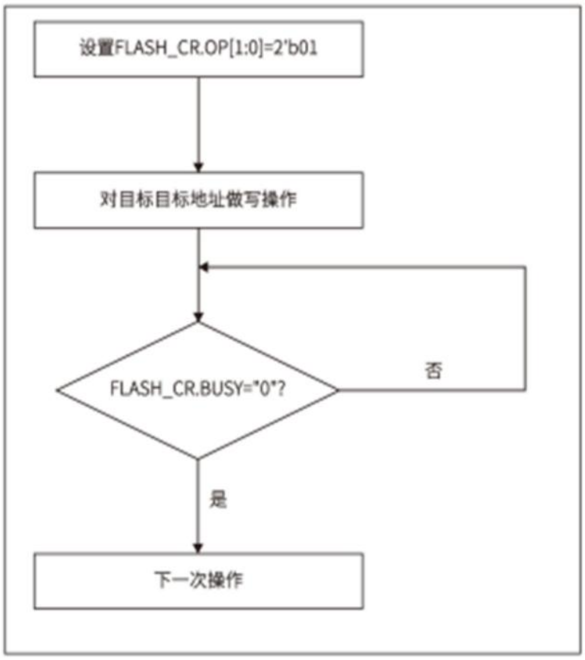


Figure 10-4 Write operation steps

Notice:

1. If the current write operation instruction is executed in Flash, the CPU will suspend the action of fetching and executing instructions, and automatically wait for the flash

Continue to fetch and execute instructions after the BUSY state ends.

2. If the current write operation instruction is executed in RAM, the CPU will not stop fetching and executing instructions, and perform any operation on Flash

Before, the software must judge whether the BUSY status of Flash is over.

### 10.3.3 Read operation

The operation of reading Flash is the same as that of CPU reading SRAM memory, and there is no special requirement. The maximum reading speed of Flash in this chip

It is 35ns, and the accuracy after on-chip HIRC trimming is ±1% (typ.), converted into the fastest reading time of CPU single cycle is 41.3ns, so

This chip meets the single-cycle read operation of Flash when the CPU uses the on-chip HIRC as the clock source.

### 10.3.4 Erase and write time

Flash memory has strict time requirements for the control signals of erasing and programming operations, and the effective time of the control signals exceeds the design requirements

Erase and program operations will fail. This chip sets the Flash sector erase time register (Flash_TSERASE), Flash

Chip erasing time register (Flash_TMERASE) and Flash programming time register (Flash_TPROG) can be divided into two registers

Adjust the sector, chip erase and program (write) time.

Note: In the CP stage, the chip will test the time parameters of Flash erasing and programming and write them into the system parameter configuration area.

After power-on, the circuit will automatically load these parameters into the sector, chip erase time register and programming time register, so generally

It is not recommended to change these set erasing time parameters under certain circumstances.

### 10.3.5 Erase and write protection

#### 10.3.5.1 Erase and write protection bit

The entire 64K Byte Flash memory is divided into 128 sectors, in order to prevent accidental erase and write operations in the application from changing

A total of 64 erase and write protection bits are set, and each erase and write protection bit is responsible for protecting 2 sector areas. Protection Bit Register

The default value of Flash_SLOCK.SLOCK0/1[31:0] is "0000_0000", which means erasing and writing are not allowed. Only modify the corresponding protection bits to

"1", the sector can be erased. When any sector in the Flash memory is protected by erasing and writing, the chip of the Flash

Erasing and writing will also be automatically shielded by the controller, and an alarm flag and interrupt signal will occur. If you need to erase and write the chip, you must modify the storage

The value of the guard register Flash_SLOCK.SLOCK0/1[31:0] is "0xFFFFFFFF".

#### 10.3.5.2 PC address erase and write protection

When the CPU runs a program in Flash, it will encounter a situation: the currently running PC pointer just falls on the sector where the software erases and writes the Flash.

within the address range, then the erasing and writing operation will also be automatically shielded by the controller, and an alarm flag and an interrupt signal will be generated.

#### 10.3.5.3 Register write protection

In order to prevent accidental Flash erase and write operations from changing the contents of the Flash during application, the write and erase operations of the Flash controller registers

During operation, the write operation to Flash must be modified by writing sequence. The specific operation steps are shown in the figure below:

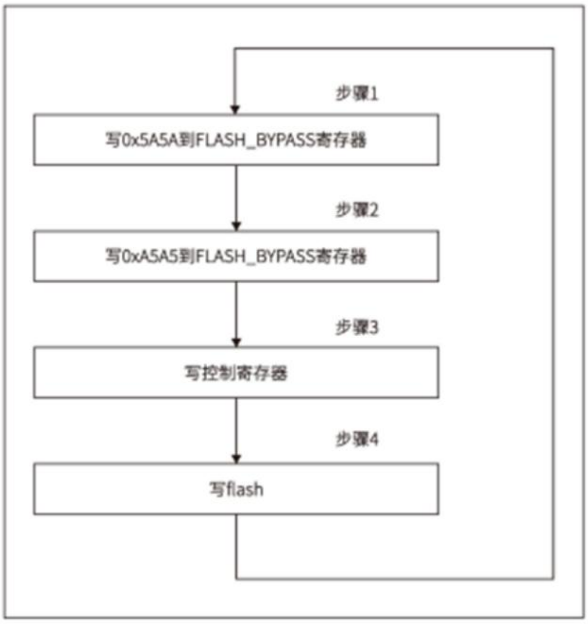


Figure 10-5 Write register BYPASS sequence

Notice:

Do not insert any other write operations between steps 1~4, otherwise the BYPASS sequence operation will be invalid and you need to rewrite 0x5A5A,

0xA5A5 sequence.

## 10.3.6 System BOOT address mapping

### 10.3.6.1 APP program boot

The figure below is the address mapping of APP program area boot.

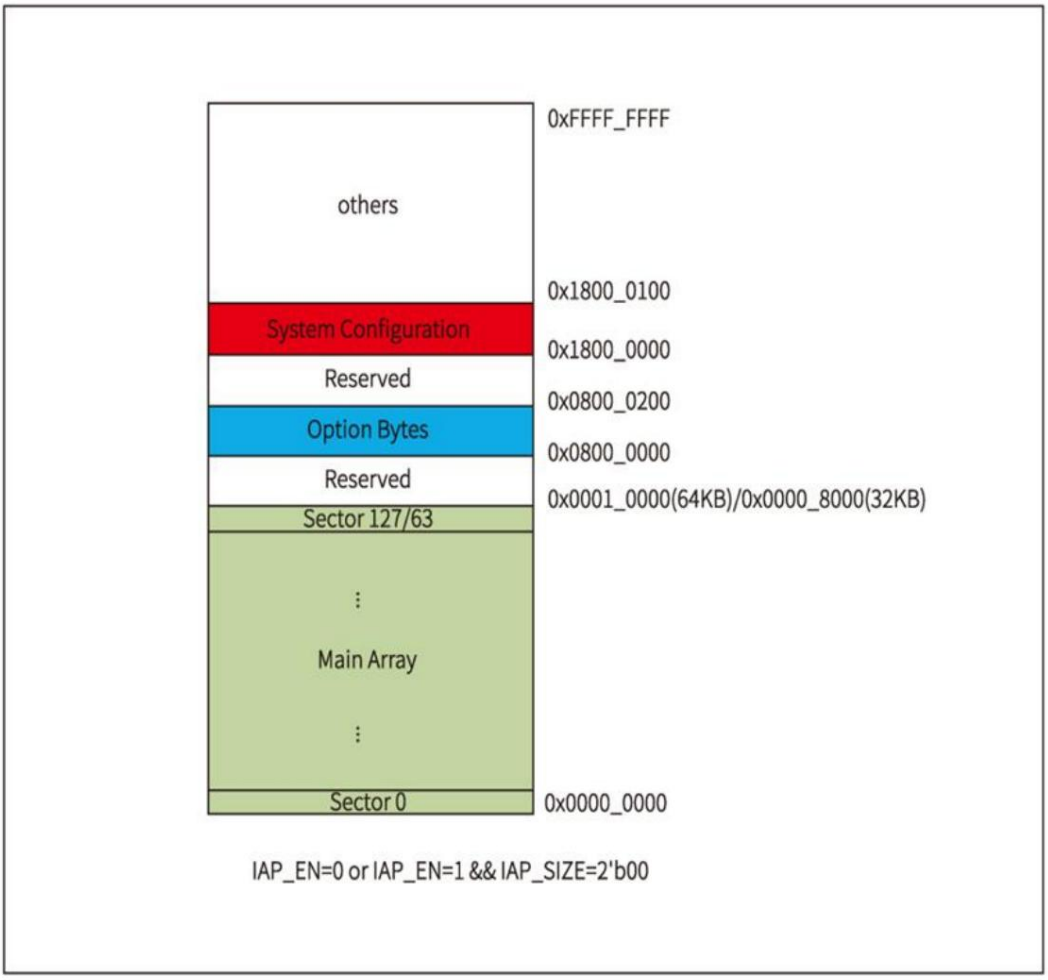


Figure 10-6 APP program area boot address mapping

The boot state of the APP program area is also the initial state provided by the chip to customers. At this point the Main Array is mapped to a logical address 64K/32K Byte area starting from 0x0000_0000, the option byte area is mapped to 512Byte starting from logical address 0x0800_0000 area, the lower 256 Byte of the system configuration area is mapped to the 256 Byte starting at 0x1800_0000, and the program can only read the 256 Byte. Cannot be erased.

## 10.3.7 Option Bytes

For details, refer to CHAPTER 31 OPTION BYTE AREA.

### 10.3.8 In-System Programming (ISP)

This chip has the function of in-system programming (ISP). The ISP program is provided by the chip manufacturer and burned in the NVR area of Flash. user

By executing the ISP program and programming the application program into the application program area of Flash, there are two software methods to start the execution of the ISP program.

The first method is to write the ISP_CON of the option byte area USERCFG0 to 0. After the chip is powered on or reset other than CPURST, the hardware will automatically load the ISP_CON

of the option byte area USERCFG0 to the ISP_CON of the FMC register Flash_ISPCON.

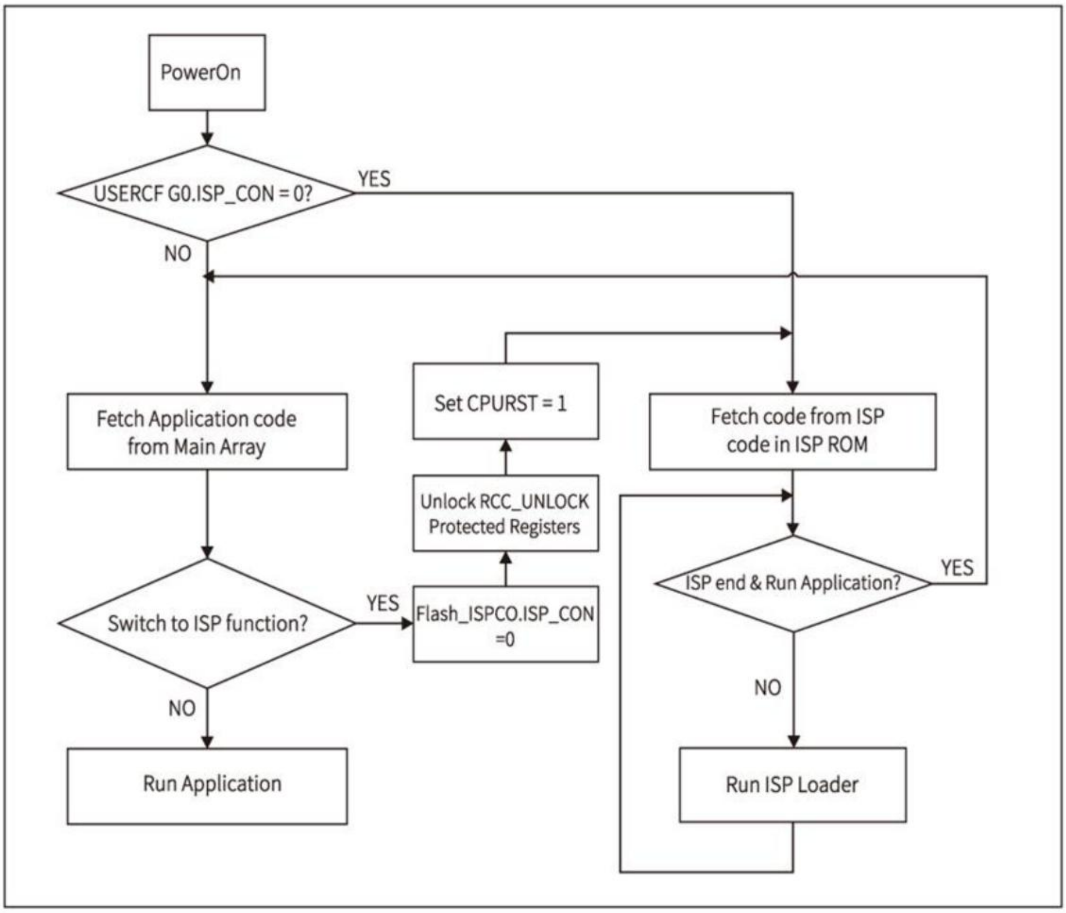
After the CPU starts, the Bootloader program recognizes that the ISP_CON of the FMC register Flash_ISPCON is 0 and calls the ISP program.

Cooperate with the ISP host computer software provided by the manufacturer to download the application program to the application program area of Flash. The second method customers do not need to repair

Changing the ISP_CON of the option byte area USERCFG0 does not need to be powered on again, just directly modify the ISP_CON bit of the FMC register Flash_ISPCON to 0, and then

generate CPURST to make the setting take effect. The flow of programming in the system is shown in the figure below

shown.


PowerOn
USERCF G0.ISP_CON = 0?
YES
NO
Fetch Application code from Main Array
Set CPURST = 1
Unlock RCC_UNLOCK Protected Registers
Fetch code from ISP code in ISP ROM
Switch to ISP function?
YES
Flash_ISPCO.ISP_CON =0
ISP end & Run Application?
YES
NO
NO
Run Application
Run ISP Loader

In addition to the software method to start the ISP, the chip also has a hardware way to start the ISP, through the specific ISP tool provided by the external manufacturer to register

And cooperate with the upper computer software provided by the factory to start the ISP. It should be noted that if you use hardware to start the ISP, you need to use a chip

The three pins of NRST, PD1 and PC7, it is recommended to lead out these three pins to the interface with the specific ISP tool when making the PCB board

on the appropriate connector.

Note: For specific ISP tools, please contact the factory for application or purchase.

## 10.4 Register List

Base address: 0x4002 0400

| offset address | name | | Defaults |
|---|---|---|---|
| 0x00 | Flash_CR | Describe Control Register | 0x0000 0000 |
| 0x04 | Flash_IFR | Interrupt Flag Register | 0x0000 0000 |
| 0x08 | Flash_ICLR | Interrupt Flag Clear Register | 0x0000 0000 |
| 0x0C | Flash_BYPASS | 0x5A5A-0xA5A5 Sequence Register Sector | 0x0000 0000 |
| 0x10 | Flash_SLOCK0 | Erase and Write Protection Register 0 | 0x0000 0000 |
| 0x14 | Flash_SLOCK1 | Sector Erase and Write Protection Register 1 | 0x0000 0000 |
| 0x18 | Flash_ISPCON | Flash ISP Control Register | 0x0000 0001 |

Note: All Flash registers can only be read and written in word mode.

## 10.5 Register description

### 10.5.1 Flash_CR (Flash Control Register)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | IE[1:0] | | BUSY | OP[1:0] | |
| | | | | | | | | | | | R/W | | RO | R/W | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:5 | - | Reserved, always reads as 0. | 0 | - |
| 4:3 | IE[1:0] | IE[1]: Flash erase and write protected address interrupt enable; 0: Disable; 1: Enable.<br>IE[0]: Flash erase PC value interrupt enable; 0: Disable; 1: Enable. | 2'b00 | R/W |
| 2 | BUSY | Free/busy flag; 0: idle state; 1: busy state. | 1'b0 | RO |
| 1:0 | OP[1:0] | Flash operation mode register<br>2'b00: Read<br>2'b01: Program<br>2'b10: Sector erase<br>2'b11: Chip erase | 2'b00 | R/W |

### 10.5.2 Flash_IFR (Flash Interrupt Flag Register)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | | IF1 | IF0 |
| | | | | | | | | | | | | | | RO | RO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:2 | - | Reserved, always reads as 0. | 0 | - |
| 1 | IF1 | Erase and write protection alarm interrupt flag bit | 1'b0 | RO |
| 0 | IF0 | Erase and write PC address alarm interrupt flag bit | 1'b0 | RO |

### 10.5.3 Flash_ICLR (Flash Interrupt Flag Clear Register)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | | | | | | | ICLR1 | ICLR0 |
| | | | | | | | | | | | | | | WO | WO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:2 | - | Reserved, always reads as 0. | 0 | - |
| 1 | ICLR1 | clear protection alarm interrupt flag: write 0 to clear; write 1 to invalid | - | WO |
| 0 | ICLR0 | Clear PC address alarm interrupt flag: write 0 to clear; write 1 to invalid | - | WO |

### 10.5.4 Flash_BYPASS (BYPASS sequence register)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| BYPASSSEQ[15:0] | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | BYPASSSEQ[15:0] | Before modifying the registers of this module, you must write to the BYPASSSEQ[15:0] register Enter 0x5A5A, 0xA5A5 sequence. Only one write is allowed after a correct write sequence Secondary register, if you need to modify the register again, you must enter the correct BYPASS sequence | 16'h0000 WO | |

### 10.5.5 Flash_SLOCK0 (sector erase protection register 0)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLOCK0[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLOCK0[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | Corresponding | Reset value read and write |
|---|---|---|
| 31 SLOCK0[31] | function description: sector 62-63<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 30 SLOCK0[30] | Correspondence: sector 60-61<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 29 SLOCK0[29] | Correspondence: sector 58-59<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 28 SLOCK0[28] | Correspondence: sector 56-57<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 27 SLOCK0[27] | Correspondence: sector 54-55<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 26 SLOCK0[26] | Correspondence: sector 52-53<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 25 SLOCK0[25] | Correspondence: sector 50-51<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 24 SLOCK0[24] | Correspondence: sector 48-49<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 23 SLOCK0[23] | Correspondence: sector 46-47<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 22 SLOCK0[22] | Correspondence: sector 44-45<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 21 SLOCK0[21] | Correspondence: sector 42-43<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 20 SLOCK0[20] | Correspondence: sector 40-41<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 19 SLOCK0[19] | Correspondence: sector 38-39<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 R/W |
| 18 SLOCK0[18] | Correspondence: sector 36-37<br>0: Erase and write are not allowed; | 1'b0 R/W |

| bit | flag | Functional description | Reset value | read and write |
|---|---|---|---|---|
| | | 1: Erase and write are allowed. | | |
| 17 | SLOCK0[17] | Corresponding to: sector 34-35 0:<br>Erase and write not allowed;<br>1: Erase and write | 1'b0 | R/W |
| 16 | SLOCK0[16] | allowed. Corresponding to: sector<br>32-33 0: Erase and write not<br>allowed; 1: Erase and | 1'b0 | R/W |
| 15 | SLOCK0[15] | write allowed. Corresponding: sector<br>30-31 0: Erase and write not<br>allowed; 1: Erase and | 1'b0 | R/W |
| 14 | SLOCK0[14] | write allowed. Corresponding to:<br>sector 28-29 0: Erase and<br>write not allowed; 1: | 1'b0 | R/W |
| 13 | SLOCK0[13] | Erase and write allowed.<br>Corresponding to: sector<br>26-27 0: Erase and write | 1'b0 | R/W |
| 12 | SLOCK0[12] | not allowed; 1: Erase and write<br>allowed. Corresponding to:<br>sector 24-25 0: Erase | 1'b0 | R/W |
| 11 | SLOCK0[11] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 22-23 0: Erase | 1'b0 | R/W |
| 10 | SLOCK0[10] | and write not allowed; 1: Erase and<br>write allowed. Corresponding:<br>sector 20-21 0: Erase | 1'b0 | R/W |
| 9 | SLOCK0[9] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 18-19 0: Erase | 1'b0 | R/W |
| 8 | SLOCK0[8] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 16-17 0: Erase | 1'b0 | R/W |
| 7 | SLOCK0[7] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 14-15 0: Erase | 1'b0 | R/W |
| 6 | SLOCK0[6] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 12-13 0: Erase | 1'b0 | R/W |
| 5 | SLOCK0[5] | and write not allowed; 1: Erase and<br>write allowed. Corresponding:<br>sector 10-11 0: Erase | 1'b0 | R/W |
| 4 | SLOCK0[4] | and write not allowed; 1: Erase<br>and write allowed.<br>Corresponding: sector | 1'b0 | R/W |
| 3 | SLOCK0[3] | 8-9 0: Erase and write not<br>allowed; 1: Erase and write<br>allowed. Corresponding | 1'b0 | R/W |
| 2 | SLOCK0[2] | to: sector 6-7 0: Erase and write<br>not allowed; 1: Erase and<br>write allowed. | 1'b0 | R/W |
| 1 | SLOCK0[1] | Corresponding: sector 4-5 0:<br>Erase and write not allowed;<br>1: Erase and write | 1'b0 | R/W |
| 0 | SLOCK0[0] | allowed. Corresponding: sector<br>2-3 0: Erase and write not<br>allowed; 1: Erase and write allowed. Corresponding: sector 0-1 0: Erase and write not allowed; 1: Erase and write allowed. | 1'b0 | R/W |

### 10.5.6 Flash_SLOCK1 (sector erase protection register 1)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLOCK1[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLOCK1[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit | flag | Corresponding | Reset value | read and write |
|---|---|---|---|---|
| 31 | SLOCK1[31] | function description: sector 126-127<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 30 | SLOCK1[30] | Correspondence: sector 124-125<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 29 | SLOCK1[29] | Correspondence: sector 122-123<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 28 | SLOCK1[28] | Correspondence: sector 120-121<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 27 | SLOCK1[27] | Correspondence: sector 118-119<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 26 | SLOCK1[26] | Correspondence: sector 116-117<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 25 | SLOCK1[25] | Correspondence: sector 114-115<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 24 | SLOCK1[24] | Correspondence: sector 112-113<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 23 | SLOCK1[23] | Correspondence: sector 110-111<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 22 | SLOCK1[22] | Correspondence: sector 108-109<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 21 | SLOCK1[21] | Correspondence: sector 106-107<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 20 | SLOCK1[20] | Correspondence: sector 104-105<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 19 | SLOCK1[19] | Correspondence: sector 102-103<br>0: Erase and write are not allowed;<br>1: Erase and write are allowed. | 1'b0 | R/W |
| 18 | SLOCK1[18] | corresponds to: sector 100-101<br>0: Erase and write are not allowed; | 1'b0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 1: Erase and write are allowed. | | |
| 17 | SLOCK1[17] | Corresponding to: sector 98-99 0:<br>Erase and write not allowed;<br>1: Erase and write | 1'b0 | R/W |
| 16 | SLOCK1[16] | allowed. Corresponding to: sector<br>96-97 0: Erase and write not<br>allowed; 1: Erase and | 1'b0 | R/W |
| 15 | SLOCK1[15] | write allowed. Corresponding to:<br>sector 94-95 0: Erase and<br>write not allowed; 1: | 1'b0 | R/W |
| 14 | SLOCK1[14] | Erase and write allowed.<br>Corresponding to: sector<br>92-93 0: Erase and write | 1'b0 | R/W |
| 13 | SLOCK1[13] | not allowed; 1: Erase and write<br>allowed. Corresponding to:<br>sector 90-91 0: Erase | 1'b0 | R/W |
| 12 | SLOCK1[12] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 88-89 0: Erase | 1'b0 | RW |
| 11 | SLOCK1[11] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 86-87 0: Erase | 1'b0 | R/W |
| 10 | SLOCK1[10] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 84-85 0: Erase | 1'b0 | R/W |
| 9 | SLOCK1[9] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 82-83 0: Erase | 1'b0 | R/W |
| 8 | SLOCK1[8] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 80-81 0: Erase | 1'b0 | R/W |
| 7 | SLOCK1[7] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 78-79 0: Erase | 1'b0 | R/W |
| 6 | SLOCK1[6] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 76-77 0: Erase | 1'b0 | R/W |
| 5 | SLOCK1[5] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 74-75 0: Erase | 1'b0 | R/W |
| 4 | SLOCK1[4] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 72-73 0: Erase | 1'b0 | R/W |
| 3 | SLOCK1[3] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 70-71 0: Erase | 1'b0 | R/W |
| 2 | SLOCK1[2] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 68-69 0: Erase | 1'b0 | R/W |
| 1 | SLOCK1[1] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 66-67 0: Erase | 1'b0 | R/W |
| 0 | SLOCK1[0] | and write not allowed; 1: Erase and<br>write allowed. Corresponding<br>to: sector 64-65 0: Erase and write not allowed; 1: Erase and write allowed. | 1'b0 | R/W |

Note: When Flash_FULLSIZE is configured as 0, that is, when the Flash capacity is configured as 32K Byte, the setting values of the Flash_SLOCK1 register are all invalid.

### 10.5.7 Flash_ISPCON (Flash ISPCON configuration register)

Address offset: 0x18

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | ISP_<br>CON |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 - | | Reserved, always reads as 0. | 0 | - |
| 0 | ISP_CON | ISP function configuration bits:<br>0: Enable ISP function during BOOTLOAD process.<br>1: Skip the ISP function in the BOOTLOAD process and switch directly to the application program;<br>ISP_CON can only be reset by MCURST, CPURST cannot reset ISP_CON.<br>After the user modifies ISP_CON, the settings will take effect through CPURST. | 1'b1 R/W | |

Note: When writing ISP_CON of the Flash_ISPCON register, the upper 16-bit data must be 16'h5A69.

# 11 Cyclic redundancy check calculation unit (CRC)

## 11.1 Overview

The cyclic redundancy check (CRC) calculation unit obtains the CRC calculation result of any byte data according to a fixed generator polynomial. In the application, CRC technology is mainly used to verify the correctness and integrity of data transmission or data storage. The following shows the CRC algorithm in data transmission One of the most typical applications of:

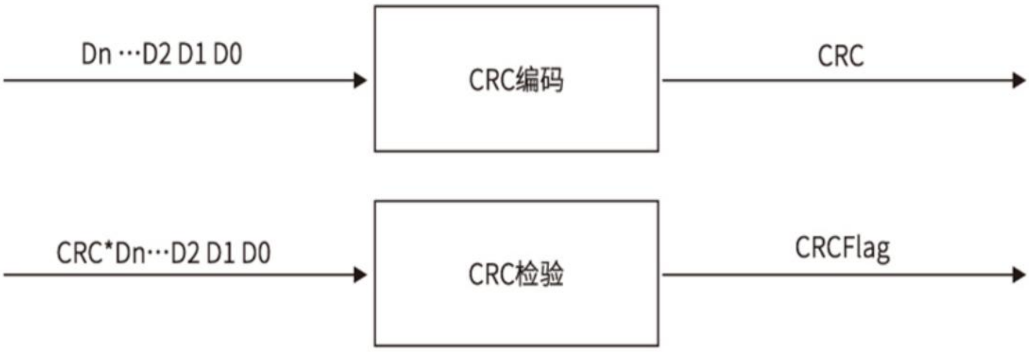


Figure 11-1 Schematic diagram of CRC application

## 11.2 Functional description

The algorithm of this module complies with the definition of ISO/IEC13239, adopts 16-bit CRC, and the calculation polynomial is:

$$16 + 12 + 5 +$$

The calculated initial value is 0xFFFF.

The functions of this module include:

ÿ CRC code and CRC check

ÿ 3 access modes of bit width: 8-bit, 16-bit, 32-bit

ÿ Examples of input data in 8-bit width are 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77

ÿ Examples of input data in 16-bit width are 0x1100, 0x3322, 0x5544, 0x7766

ÿ The input data example under 32-bit width is 0x33221100, 0x77665544

### 11.2.1 CRC encoding mode

Encoding mode can encode the original data to calculate its CRC value, the operation process is as follows:

Step1: Write 0xFFFF to CRC_RESULT.RESULT to initialize CRC calculation.

Step2: Write the original data to be encoded into the CRC_DATA register sequentially according to the organization mode of 8 bits/16 bits/32 bits.

Step3: Read CRC_RESULT.RESULT, which is the CRC value.

### 11.2.2 CRC check mode

The verification mode can verify whether the encoded data has been tampered with, and the operation process is as follows

Step1: Write 0xFFFF to CRC_RESULT.RESULT to initialize CRC calculation.

Step2: Write the coded data into the CRC_DATA register sequentially according to the organization mode of 8 bits/16 bits/32 bits.

Note: When writing the CRC value to the CRC_DATA register according to the 8-bit organization method, the lower 8 bits should be written first, and then the upper 8 bits should be written.

Step3: Read CRC_RESULT.FLAG to determine whether the CRC check is successful.

## 11.3 Register List

Base address: 0x 4002 0800

Table 11-1 CRC register list and reset value

| offset address | name | | reset value |
|---|---|---|---|
| 0x04 | CRC_RESULT | Describes the CRC result register, which is read after the calculation is complete<br>That is, the result is obtained. | 0x0000 0000 |
| 0x80-0xFF | CRC_DATA | CRC data register, used to input the data to be calculated. 0x0000 0000 | |

## 11.4 Register description

### 11.4.1 CRC Result Register (CRC_RESULT)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | FLAG |
| | | | | | | | | | | | | | | | RO |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:17 | - | reserve | 0x0 | - |
| 16 | FLAG | Verification result flag; 0: current verification error, 1: current verification is correct.<br>NOTE: Register [16] is a read-only bit and writing to it has no effect. conduct When CRC checking, it should be after all data and 16-bit CRC code input data register Read this bit, if it is 1, it means the verification is successful. | 0x0 | RO |
| 15:0 | RESULT | This register is used to update and save each CRC calculation result. After operation, read this register The device will get 16-bit CRC encoding result.<br>Note: According to the standard, after the operation is completed, the 16-bit CRC code value is the operation register The result after inversion, so the reading of this register [15:0] will get the current [15:0] of this register Negates the value. | 0x0000 | R/W |

### 11.4.2 CRC data register (CRC_DATA)

Offset address: 0x80-0xFF

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRC_DATA[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:0 | CRC_DATA | This register is used to input the data to be calculated.<br>Note: The address of this register is a range (0x80-0xFF), any address in this range The operation on the address will be regarded as the operation on this register. The purpose of this definition is to facilitate the The software can use the STM instruction to write continuous 32-bit data to this register to add Fast calculation speed. This register supports 8/16/32-bit input mode. | 0x0 | R/W |

# 12 Advanced Control Timer (TIM1)

## 12.1 Introduction to TIM1

The advanced control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler. it fits It can be used for a variety of purposes, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output compare, PWM, embedded Complementary PWM with dead time, etc.).

Using timer prescaler and RCC clock control prescaler, pulse width and waveform period can be realized from several microseconds to several milliseconds adjustment.

## 12.2 TIM1 main features

The functions of the TIM1 timer include:

ÿ 16-bit up, down, up/down autoload counter

ÿ 16-bit programmable (can be modified in real time) prescaler, the frequency division coefficient of the counter clock frequency is any number between 1 and 65535 value

ÿUp to 4 independent channels:

- input capture
- output compare
- PWM generation (edge or center aligned mode)
-Single pulse mode output

ÿ Complementary outputs with programmable dead time

ÿUsing external signals to control timers and synchronous circuits interconnected with timers

ÿ A repeat counter that allows updating the timer register after a specified number of counter cycles

ÿBrake input signal can put timer output signal in reset state or a known state

ÿ An interrupt is generated when the following events occur:

-Update : counter overflow/underflow, counter initialization (by software or internal/external trigger)
- Trigger event (counter start, stop, initialization or counting by internal/external trigger)
- input capture
- output compare
- Brake signal input

ÿSupports incremental (quadrature) encoder and Hall sensor circuits for positioning

ÿTrigger input as external clock or cycle-by-cycle current management

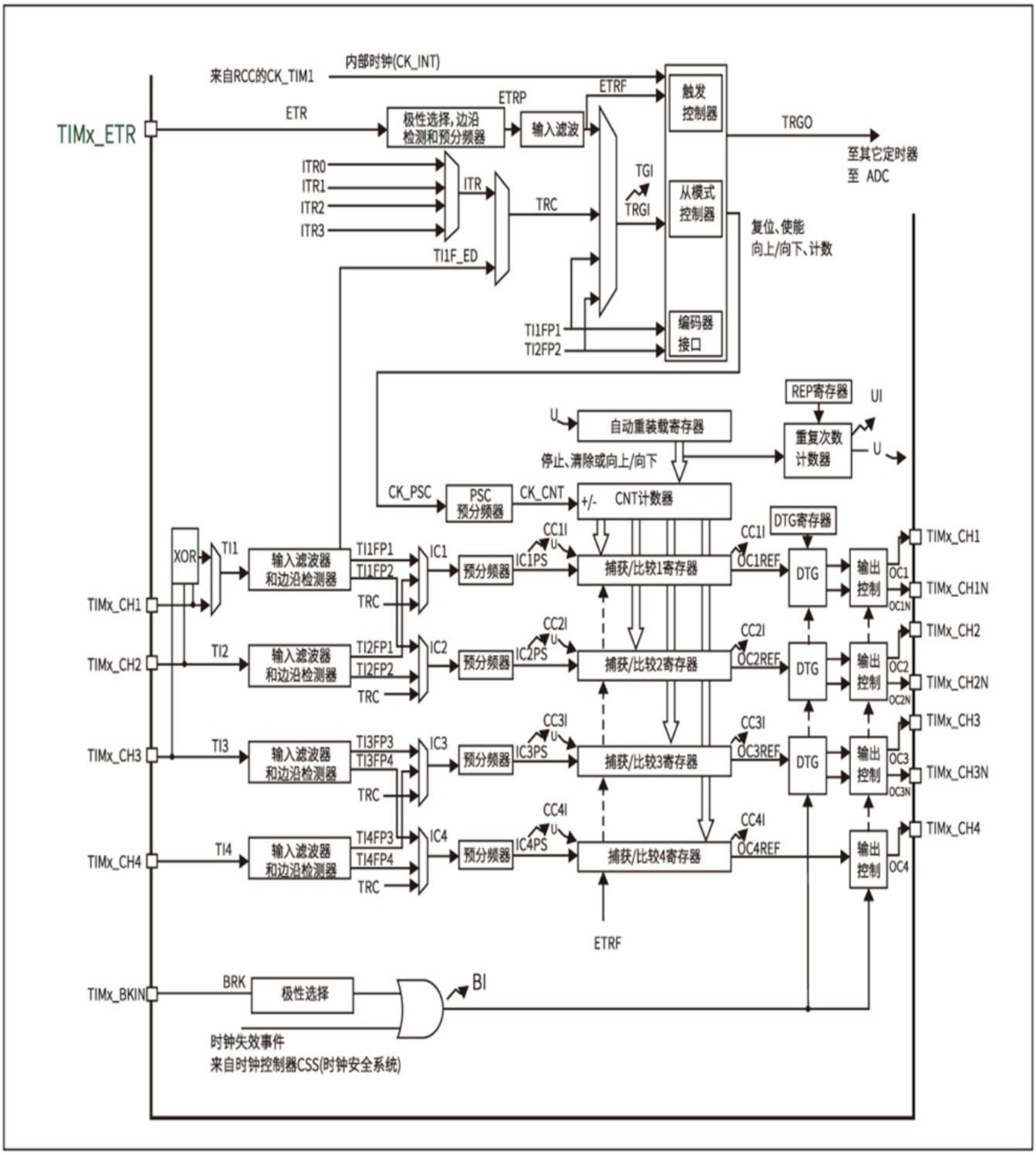


Figure 12-1 Advanced Control Timer Block Diagram

Note:

Reg According to the setting of the control bit, the content of the preload register is transferred to the working register on the U (update) event

event

to interrupt

## 12.3 TIM1 Functional Description

### 12.3.1 Time base unit

The main part of the Programmable Advanced Control Timer is a 16-bit counter and its associated autoload register. This counter can Count up, down, or up/down. This counter clock is divided by a prescaler.

The counter, autoload register and prescaler register can be read and written by software, even if the counter is still running, reading and writing are still valid.

The time base unit contains:

ÿ Counter register (TIM1_CNT)

ÿ Prescaler register (TIM1_PSC)

ÿAuto Load Register (TIM1_ARR)

ÿ Repeat count register (TIM1_RCR)

The auto-reload register is preloaded, and writing or reading the auto-reload register will access the preload register. According to register in TIM1_CR1 The setting of the automatic reload preload enable bit (ARPE) in the register, the content of the preload register is immediately or at each update event UEV is transferred to the shadow register. When the counter reaches an overflow condition (underflow condition when counting down) and when the TIM1_CR1 register When the UDIS bit is equal to 0, an update event is generated. Update events can also be generated by software. Updates under each configuration will be described in detail later event generation.

The counter is driven by the clock output CK_CNT of the prescaler, only if the counter enable bit in the counter TIM1_CR1 register is set (CEN), CK_CNT is valid. (See the controller's slave mode description for more details on enabling the counter).

Note that the counter starts counting one clock cycle after the CEN bit of the TIM1_CR1 register is set.

Prescaler Description

The prescaler can divide the clock frequency of the counter by any value between 1 and 65536. It is based on a (in the TIM1_PSC register

16-bit counter controlled by a 16-bit register. Because this control register is buffered, it can be changed at runtime.

The new prescaler parameters are adopted at the next update event.

Figure 12-2 and Figure 12-3 show examples of changing the counter parameters while the prescaler is running.

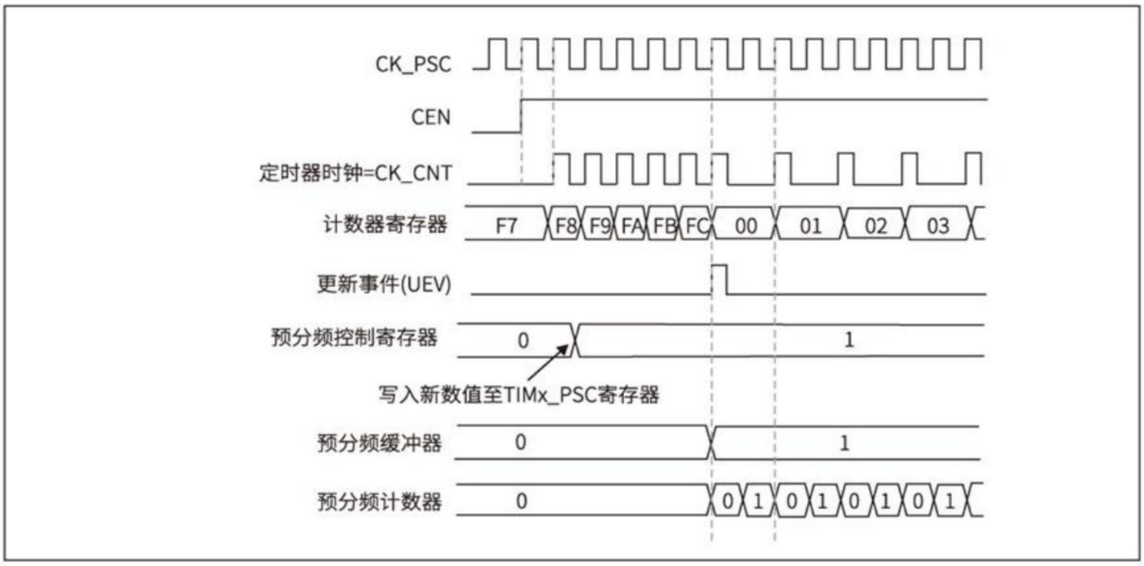


Figure 12-2 When the parameter of the prescaler changes from 1 to 2, the timing diagram of the counter

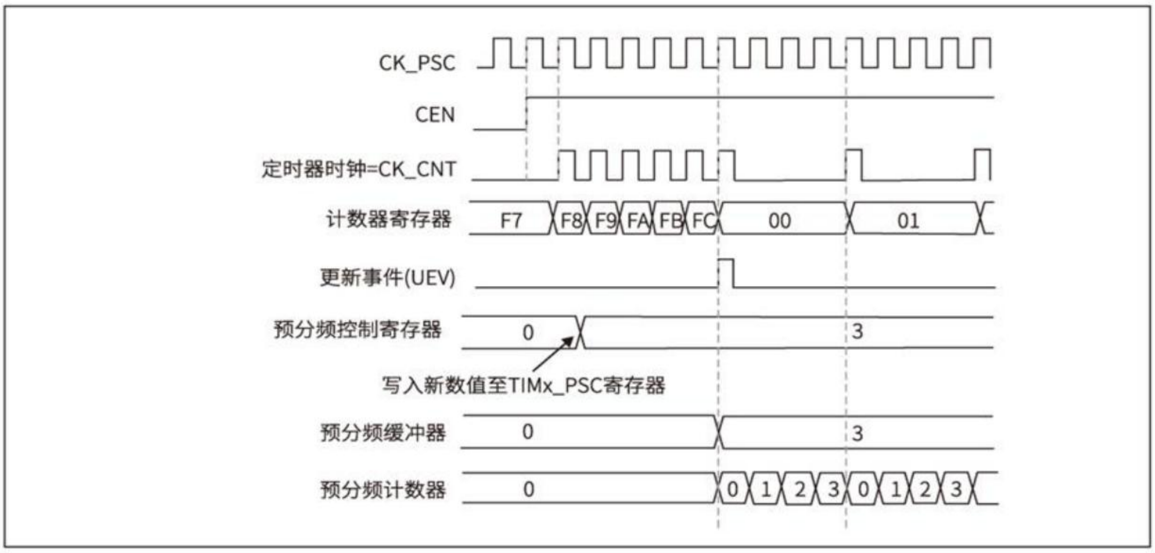


Figure 12-3 When the parameter of the prescaler changes from 1 to 4, the timing diagram of the counter

## 12.3.2 Counter mode

### 12.3.2.1 Count Up Mode

In count-up mode, the counter counts from 0 to the autoload value (contents of the TIM1_ARR counter) and then restarts counting from 0

And generate a counter overflow event.

If the repeat counter function is used, an update event will be generated when the count-up reaches the set repeat count times (TIM1_RCR)

(UEV); otherwise an update event is generated each time the counter overflows.

Setting the UG bit in the TIM1_EGR register (by software or using a slave mode controller) can also generate an update event.

pieces.

Update events can be disabled by setting the UDIS bit in the TIM1_CR1 register; this avoids writing new

Update shadow registers when value is set. No update event will be generated until the UDIS bit is cleared to '0'. But when an update event should be generated,

The counter will still be cleared to '0', and the count of the prescaler will also be reset to 0 (but the value of the prescaler remains unchanged). Additionally, if the

The URS bit in the TIM1_CR1 register (select update request), setting the UG bit will generate an update event UEV, but the hardware does not set

UIF flag (i.e. no interrupt generated). This is to avoid simultaneous update and capture interrupts when the counter is cleared in capture mode.

When an update event occurs, all registers are updated and the hardware sets the update flag (UIF bit in the TIM1_SR register) at the same time (according to the URS bit).

ÿ The repetition counter is reloaded with the contents of the TIM1_RCR register.

ÿ The autoload shadow register is reset to the value of the preload register (TIM1_ARR).

ÿ The prescaler buffer is loaded with the value of the preload register (content of the TIM1_PSC register).

The figure below gives some examples, when TIM1_ARR=0x36, the actions of the counter at different clock frequencies.

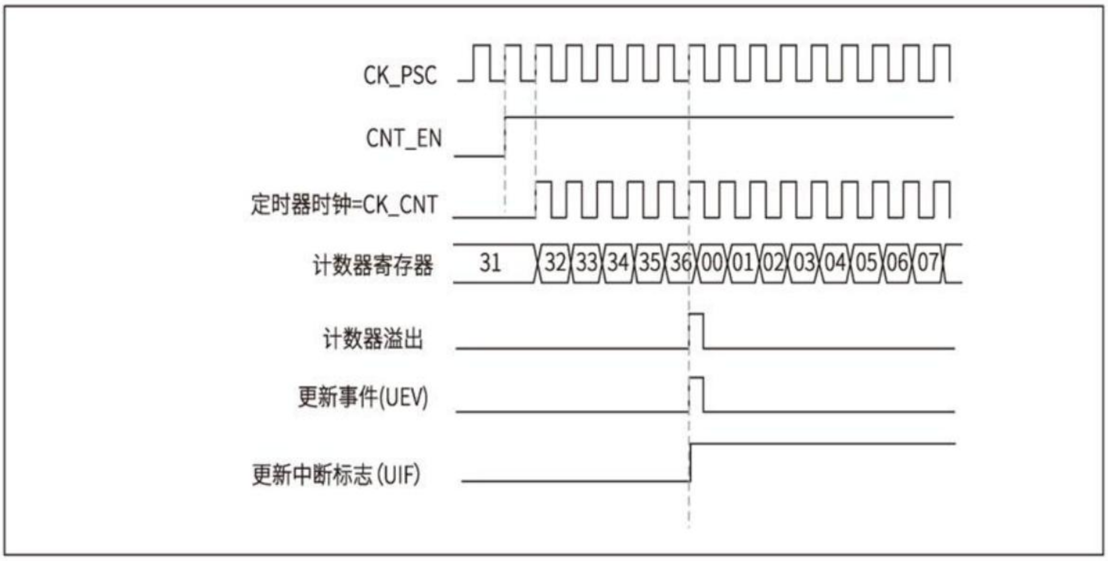


Figure 12-4 Counter timing diagram: internal clock frequency division factor is 1

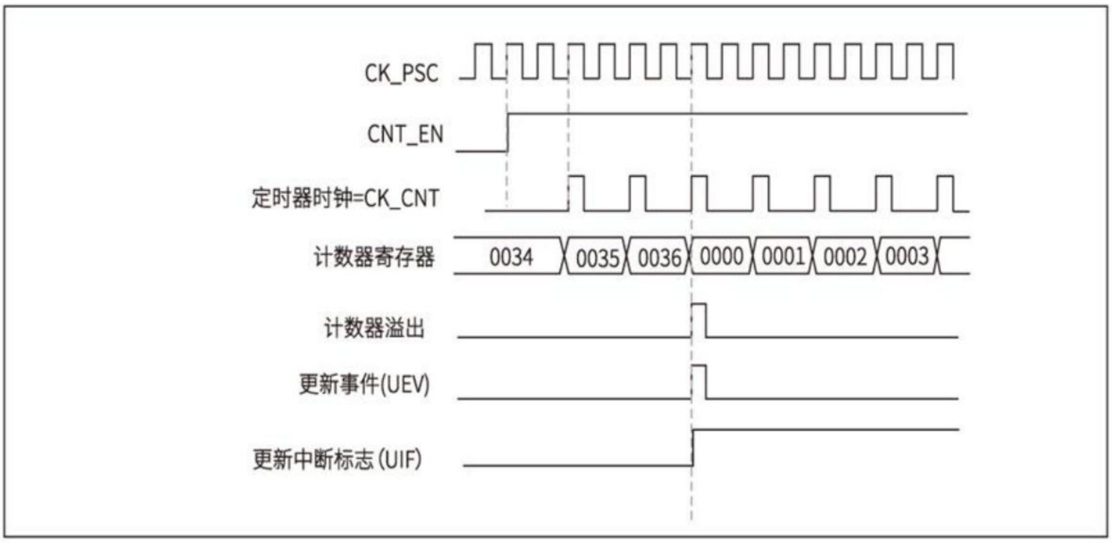


Figure 12-5 Counter timing diagram: internal clock frequency division factor is 2

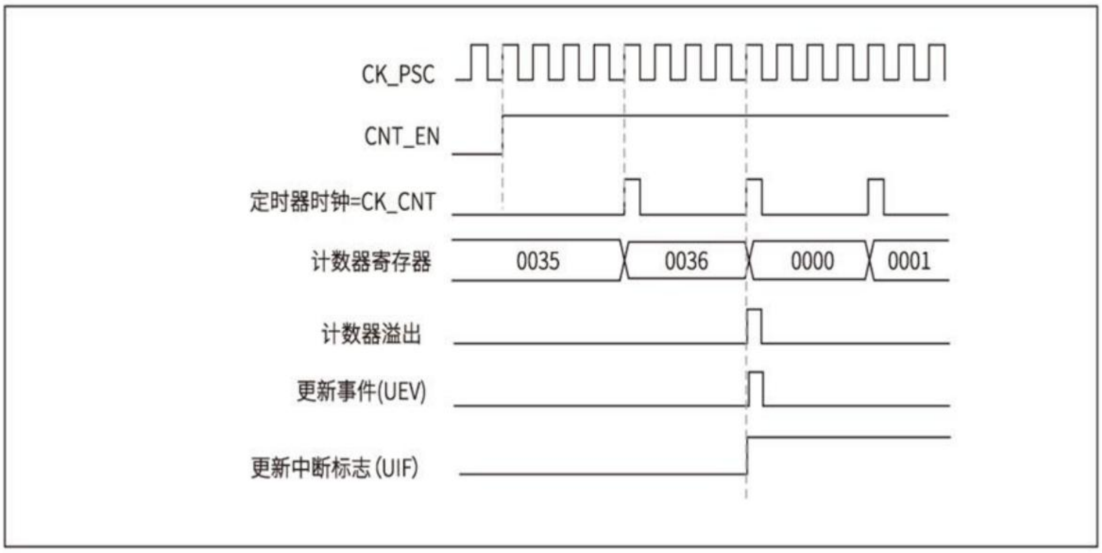


Figure 12-6 Counter timing diagram: internal clock division factor is 4

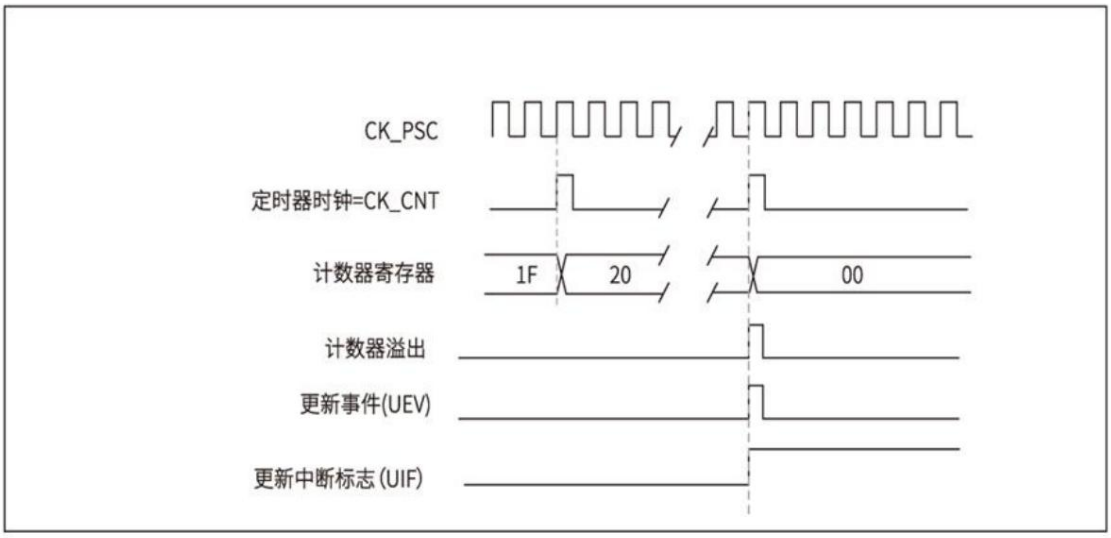


Figure 12-7 Timing diagram of the counter: internal clock division factor is N

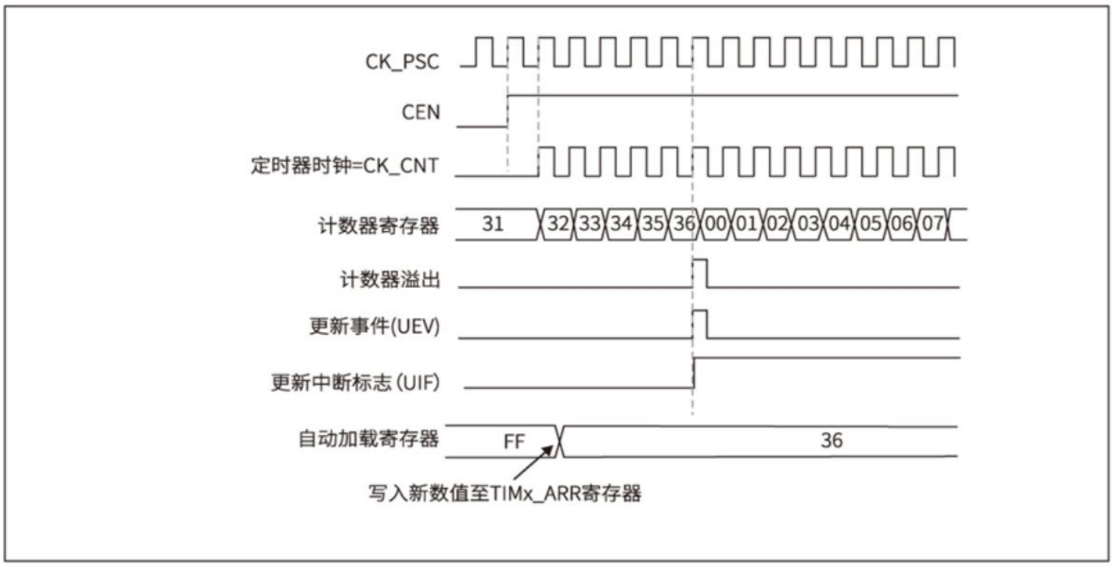


Figure 12-8 Counter timing diagram: update event when ARPE=0 (TIM1_ARR is not preloaded)

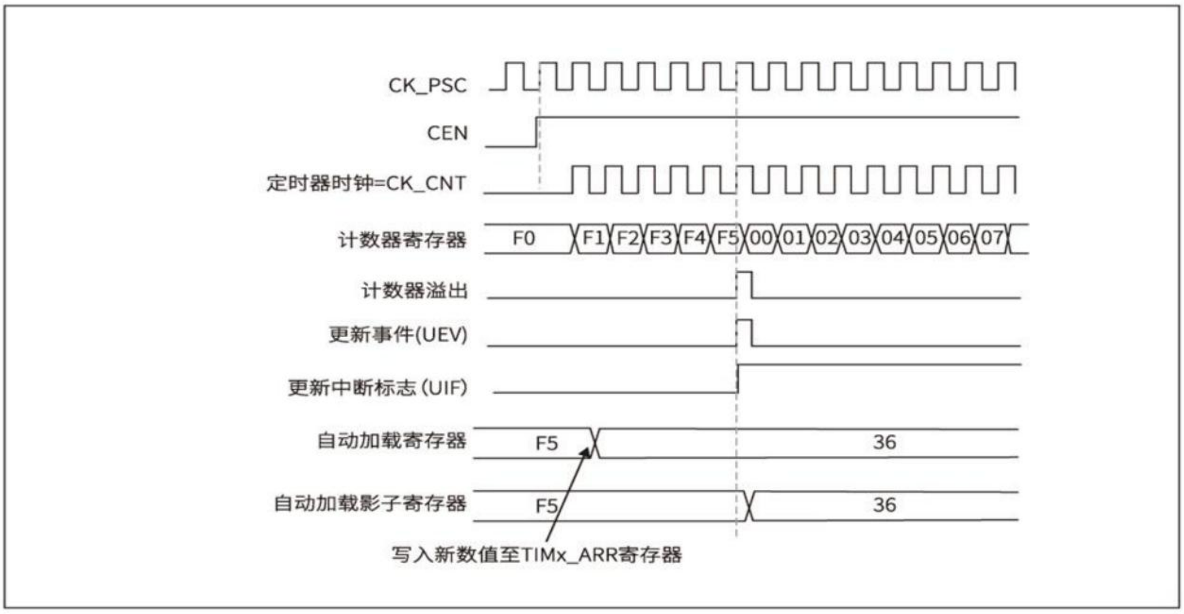


Figure 12-9 Counter timing diagram: Update event when ARPE=1 (TIM1_ARR is preloaded)

12.3.2.2 Down Counting Mode

In down mode, the counter counts down from the autoloaded value (TIM1_ARR counter value) to 0, then restarts from the autoloaded value and generates a counter underflow event.

If a repeat counter is used, an update event (UEV) will be generated after counting down for the number of times set in the repeat count register (TIM1_RCR), otherwise an update event will be generated each time the counter underflows.

An update event can also be generated by setting the UG bit in the TIM1_EGR register (by software or using a slave mode controller).

The UEV event can be disabled by setting the UDIS bit of the TIM1_CR1 register. This avoids updating the shadow registers when new values are written to the preload registers. Therefore no update event will be generated until the UDIS bit is cleared to 0. However, the counter will still restart counting from the current autoload value, and the counter of the prescaler will restart from 0 (but the prescaler factor will not change).

Also, if the URS bit in the TIM1_CR1 register is set (selecting an update request), setting the UG bit will generate an update event UEV but not set the UIF flag (and thus not generate an interrupt), this is to avoid that when a capture event occurs and clears the counter , generating update and capture interrupts at the same time.

When an update event occurs, all registers are updated and (according to the setting of the URS bit) the update flag bit (UIF bit in the TIM1_SR register) is also set.

ÿThe repeat counter is reset to the content of the TIM1_RCR register ÿThe prescaler buffer is loaded with the preloaded value (TIM1_PSC register value) ÿThe current autoload register is updated with the preloaded value (TIM1_ARR register in the content)

Note: The autoload is updated before the counter is reloaded, so the next cycle will be the expected value.

The following are some examples of counter operation at different clock frequencies when TIM1_ARR=0x36.

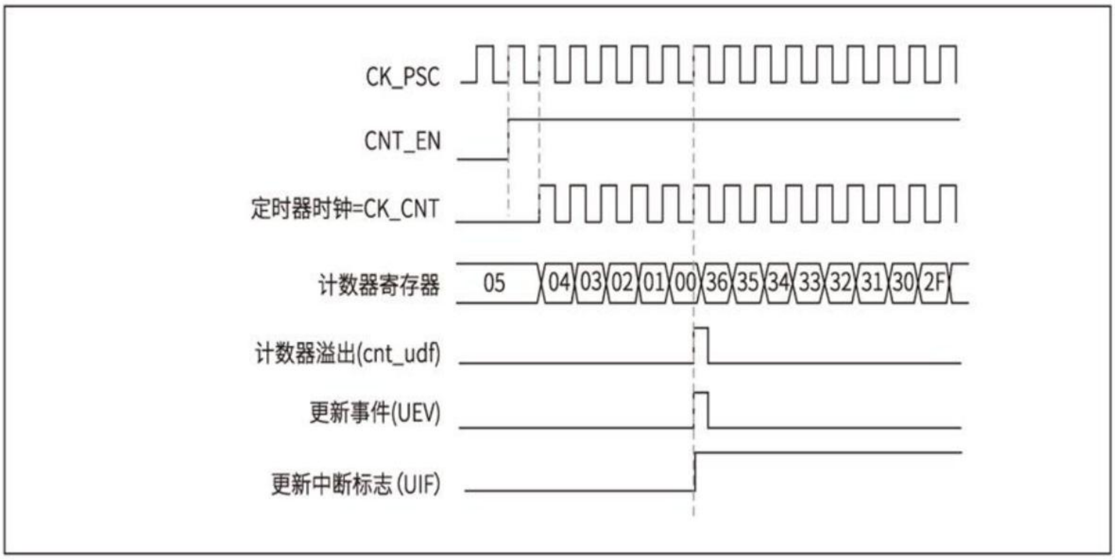


Figure 12-10 Counter timing diagram: internal clock frequency division factor is 1

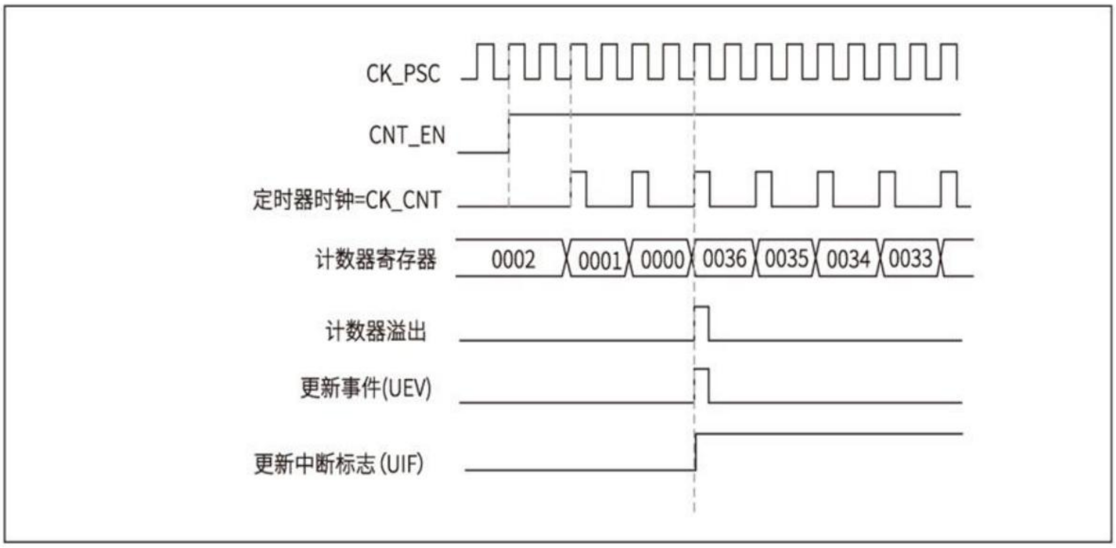


Figure 12-11 Counter timing diagram: internal clock frequency division factor is 2

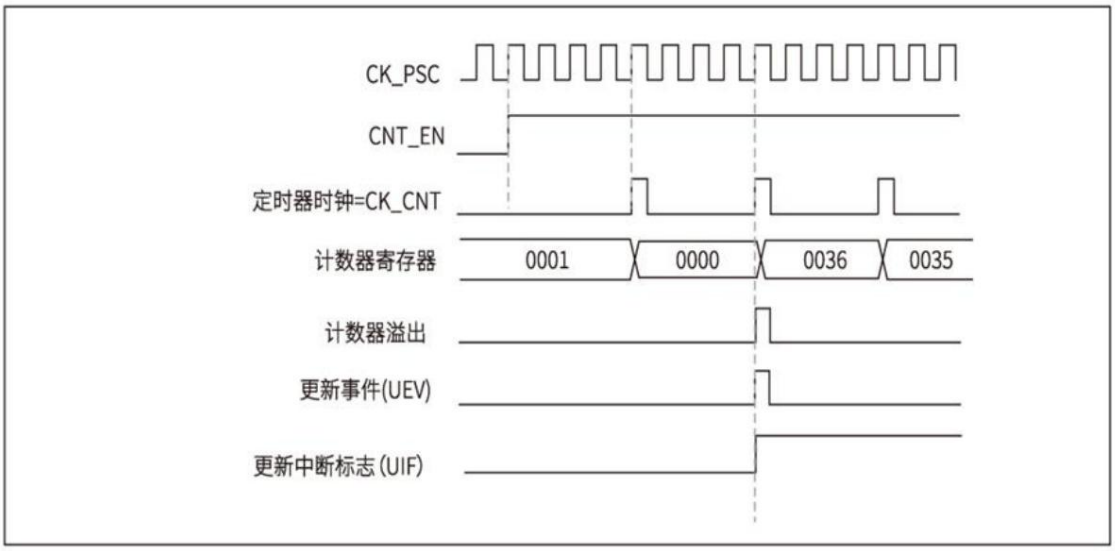


Figure 12-12 Counter Timing Diagram: The internal clock frequency division factor is 4

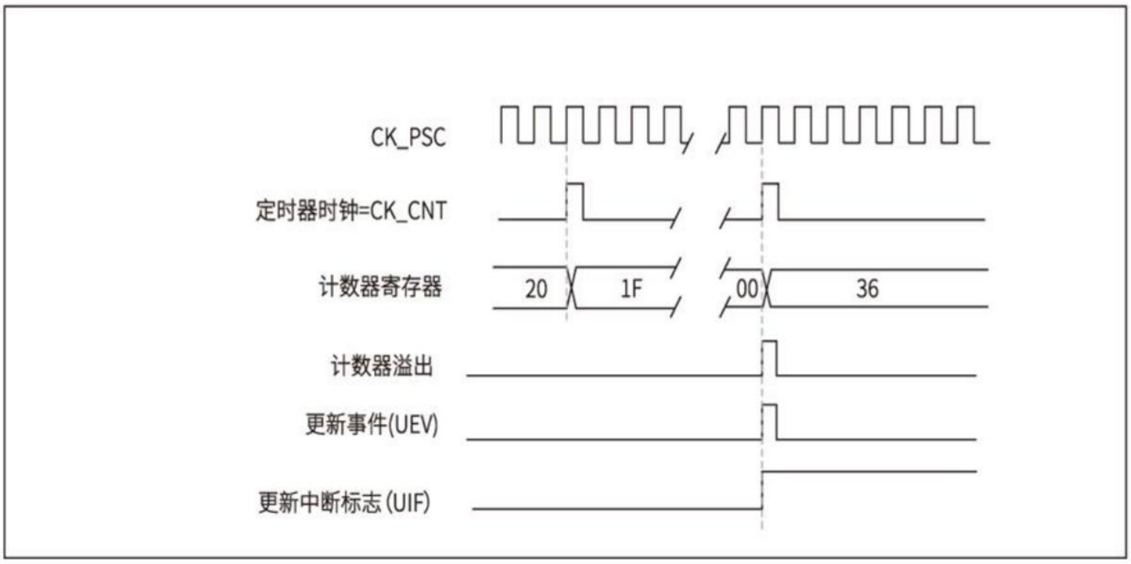


Figure 12-13 Counter timing diagram: internal clock frequency division factor is N

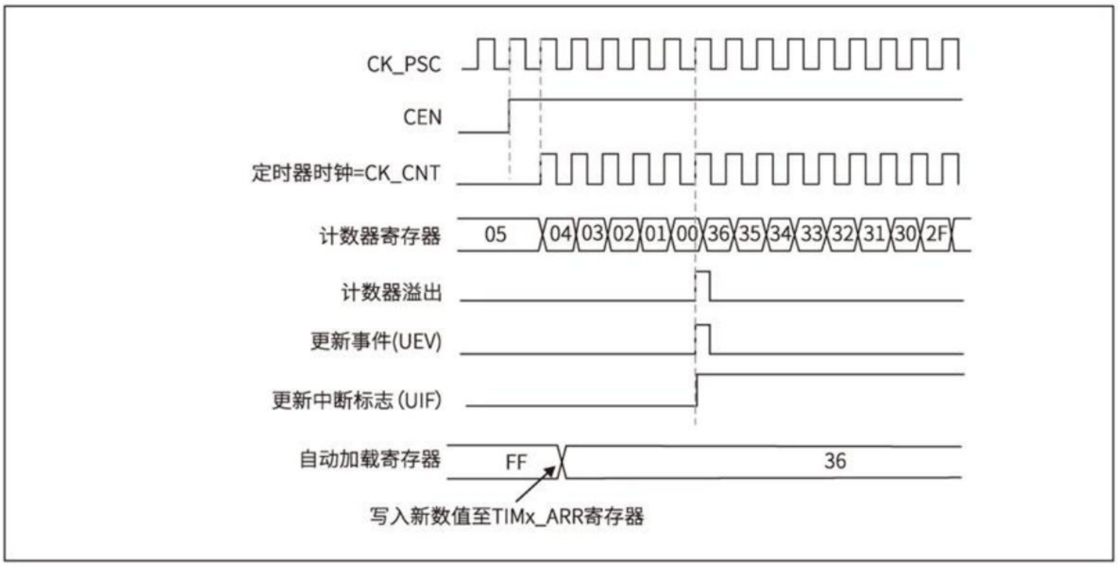


Figure 12-14 Counter Timing Diagram: Update Events When Repeat Counters Are Not Used

### 12.3.2.3 Center Aligned Mode (Count Up/Down)

In center-aligned mode, the counter counts from 0 to the autoload value (TIM1_ARR register)ÿ1, generating a counter overflow event.

event, then counts down to 1 and generates a counter underflow event; then restarts counting from 0.

In this mode, the DIR direction bit in TIM1_CR1 cannot be written. It is updated by hardware and indicates the current counting direction. available every

Generates update events on every count overflow and every count underflow; can also be passed (software or using a slave mode controller)

An update event is generated by setting the UG bit in the TIM1_EGR register. Then, the counter restarts counting from 0, and the prescaler restarts from

0 to start counting.

The UEV event can be disabled by setting the UDIS bit in the TIM1_CR1 register. This avoids writing new

Update shadow registers when value is set. Therefore no update event will be generated until the UDIS bit is cleared to 0. However, the counter will still automatically

Reloaded value to continue counting up or down.

In addition, setting the UG bit will generate an update event if the URS bit in the TIM1_CR1 register is set (select update request)

UEV but does not set the UIF flag (so no interrupt is generated), this is to avoid simultaneous generation of

Update and capture interrupts.

When an update event occurs, all registers are updated and (according to the setting of the URS bit) the update flag bit (UIF bit in the TIM1_SR register) is also set.

ÿ The repeat counter is reset to the content in the TIM1_RCR register ÿ The prescaler

buffer is loaded with the preloaded (TIM1_PSC register) value ÿ The current autoload register is updated

with the preloaded value (TIM1_ARR register) content)

Note: If an update occurs due to a counter overflow, the auto-reload will be updated before the counter is reloaded, so the next cycle will

is the expected value (the counter is loaded with the new value).

The following are some examples of counter operation at different clock frequencies:

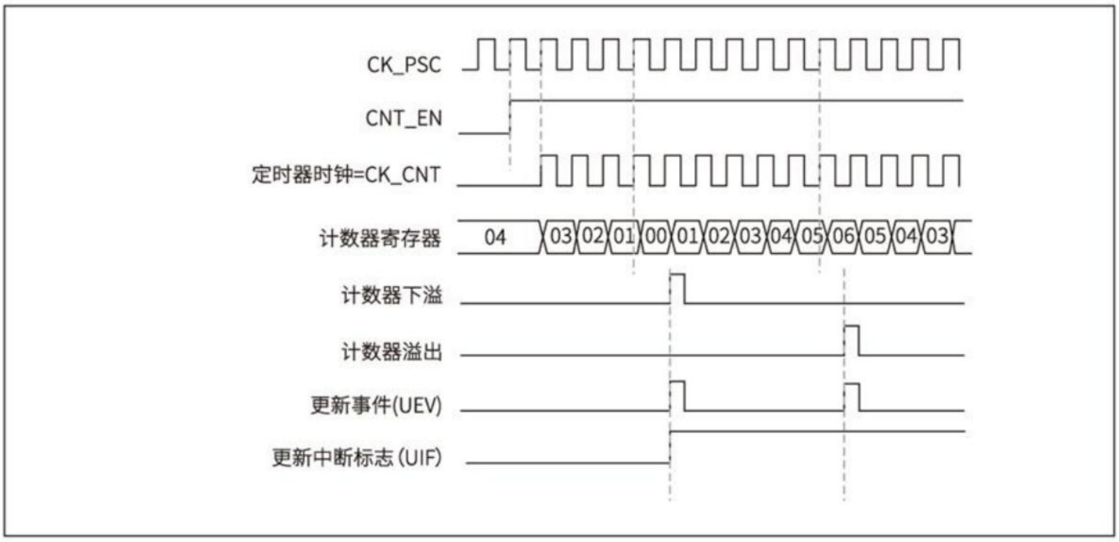


Figure 12-15 Counter timing diagram: internal clock frequency division factor is 1, TIM1_ARR=0x6

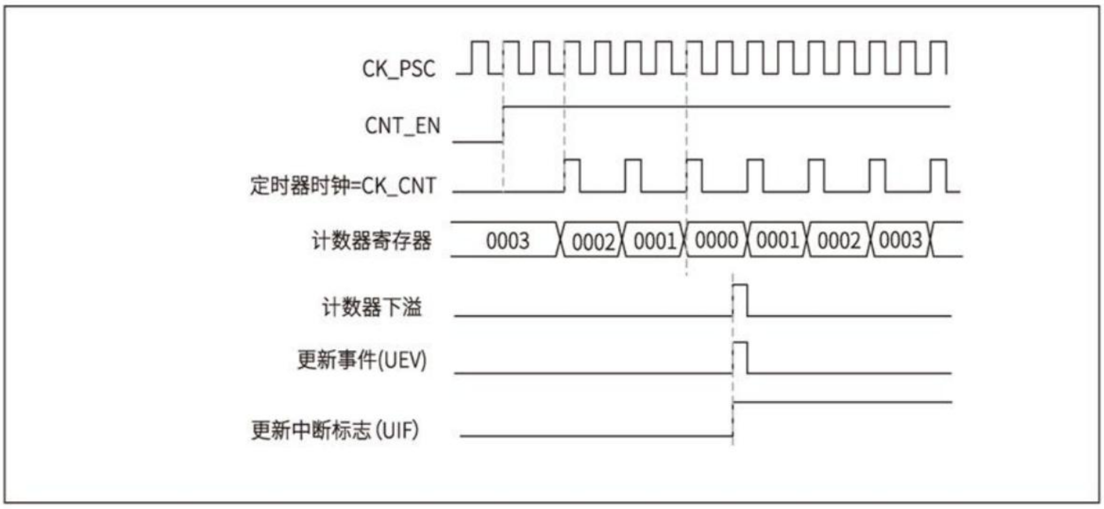


Figure 12-16 Counter timing diagram: internal clock frequency division factor is 2

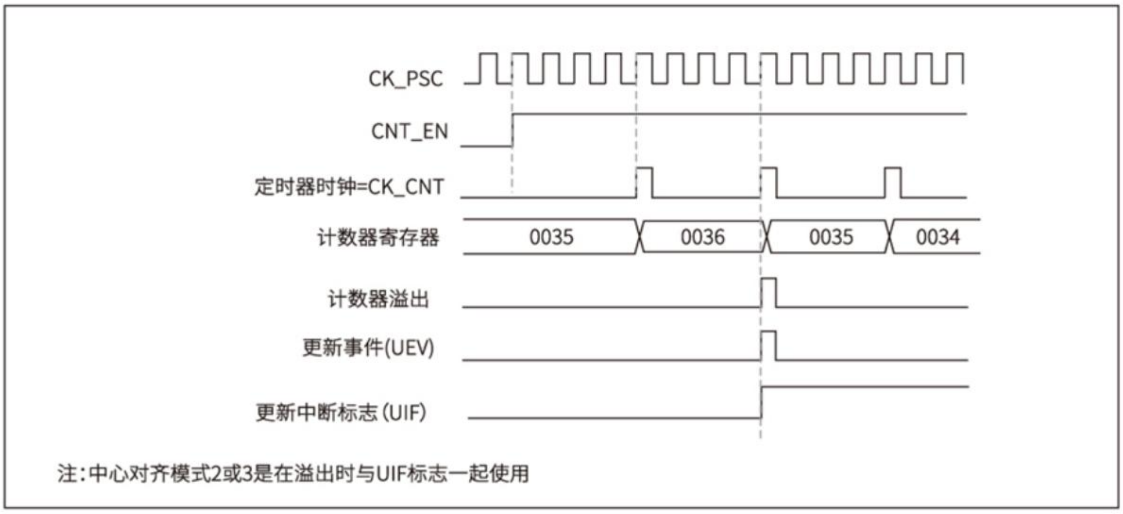


Figure 12-17 Counter timing diagram: internal clock frequency division factor is 4, TIM1_ARR=0x36

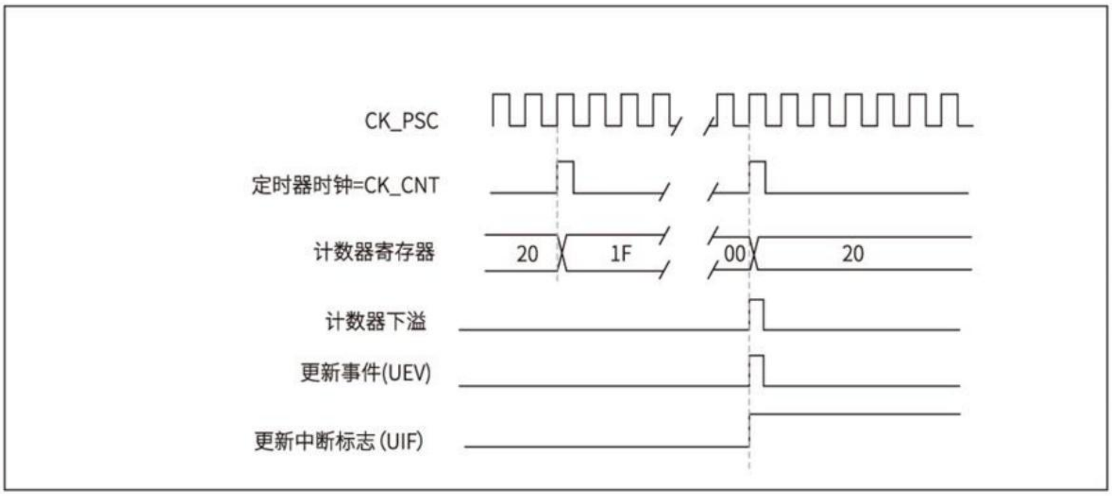


Figure 12-18 Counter timing diagram: internal clock frequency division factor is N

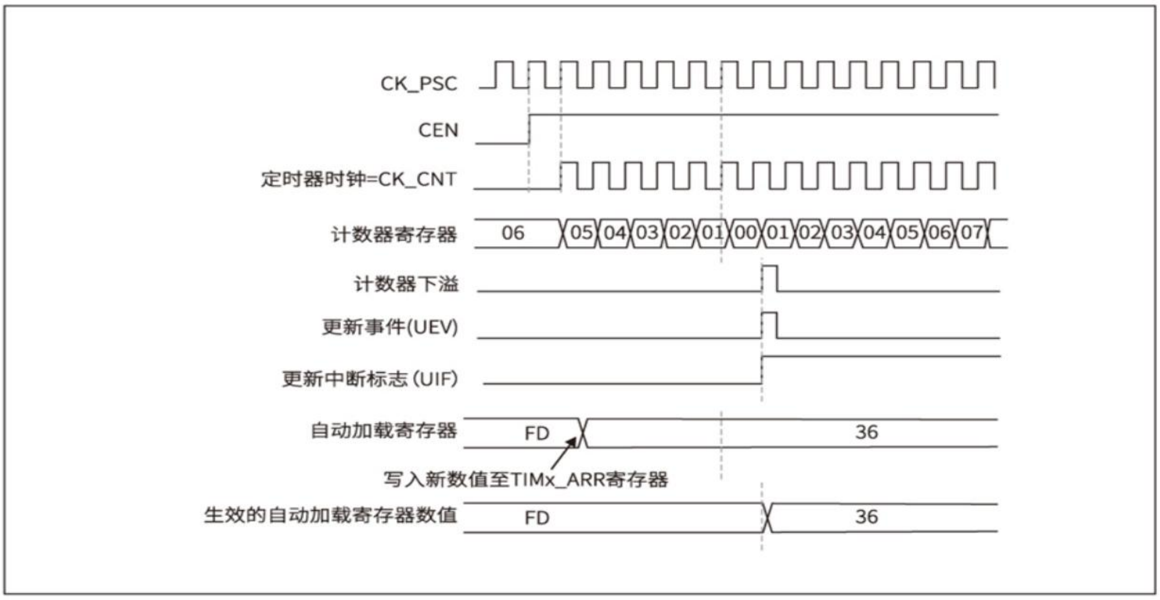


Figure 12-19 Counter timing diagram: update event when ARPE=1 (counter underflow)

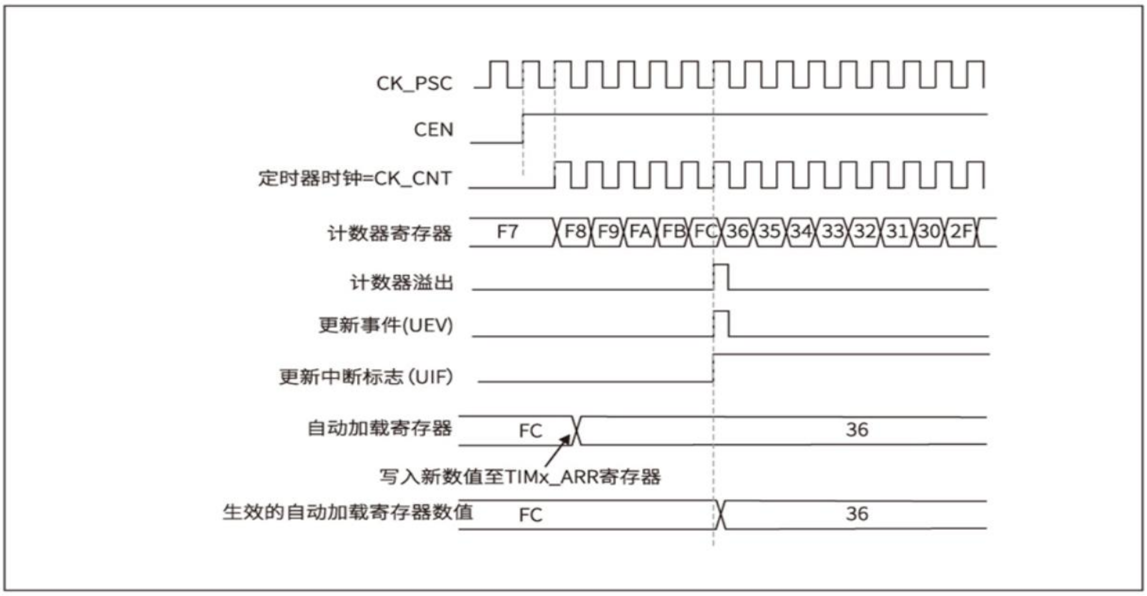


Figure 12-20 Counter timing diagram: update event when ARPE=1 (counter overflow)

## 12.3.3 Repeat counter

12.3.1 Time base unit explains how the update event (UEV) is generated when the counter overflows/underflows, however in fact it can only

Fired when the count reaches 0. This feature is very useful for generating PWM signals.

This means that at every N count overflows or underflows, data is transferred from the preload register to the shadow register (TIM1_ARR auto-reload register, TIM1_PSC preload register, and capture/compare register in compare mode TIM1_CCRx), N is the value in the TIM1_RCR repeat count register.

The repeat counter is decremented when any of the following conditions are true:

ÿ Every time the counter overflows in the up-counting mode.

ÿ Every time the counter underflows in down counting mode.

ÿOn every overflow and every underflow in center-aligned mode. Although this limits the maximum PWM cycle period to 128, it can

Enough to update the duty cycle 2 times per PWM cycle. In center-aligned mode, since the waveform is symmetrical, if each

The compare register is refreshed only once in a PWM cycle, and the maximum resolution is 2xTck.

The repetition counter is automatically loaded and the repetition rate is defined by the value of the TIM1_RCR register (see Figure 12-21). When the update event is generated by software (by setting the UG bit in TIM1_EGR) or by the slave mode controller of the hardware, no matter what the value of the repeat counter is, the update event occurs immediately, and the contents of the TIM1_RCR register are reloaded into Repeat counter.

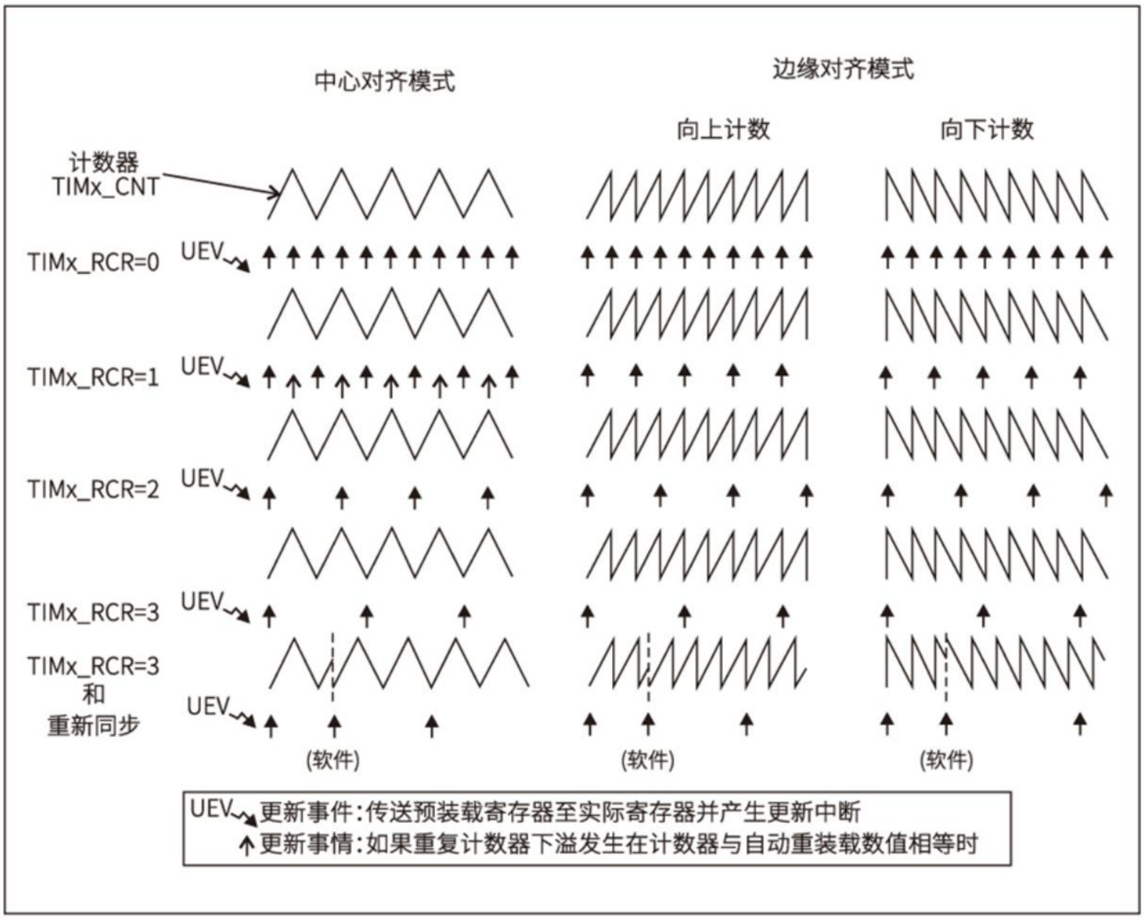


Figure 12-21 Examples of update rates in different modes, and register settings for TIM1_RCR

## 12.3.4 Clock Selection

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

ÿInternal clock (CK_INT)

ÿExternal clock mode 1: External input pin

ÿExternal clock mode 2: External trigger input ETR

ÿInternal trigger input (ITRx): Use one timer as a prescaler for another timer. If you can configure a timer TIM1 acts as a prescaler for another timer TIM2, see Section 13.3.15.1 for details.

ÿ Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), the CEN, DIR (TIM1_CR1 register) and UG bits (TIM1_EGR register) are De facto control bit, and can only be modified by software (the UG bit is still automatically cleared). As long as the CEN bit is written to '1', the prescaler The clock is provided by the internal clock CK_INT.

The figure below shows the operation of the control circuit and up-counter in normal mode without prescaler.

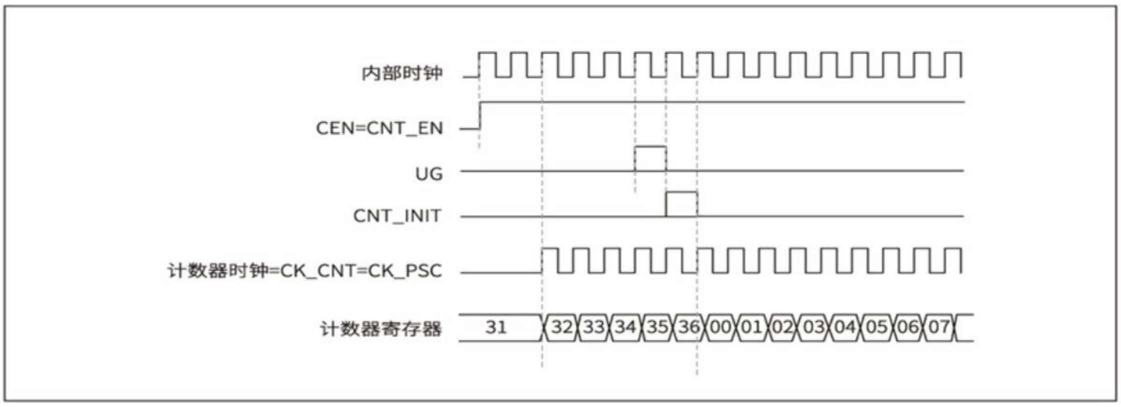


Figure 12-22 Control circuit in normal mode, internal clock frequency division factor is 1

12.3.4.1 External Clock Source Mode 1

This mode is selected when SMS=111 in the TIM1_SMCR register. The counter can be activated on every rising or falling edge of the selected input

Falling edge count.

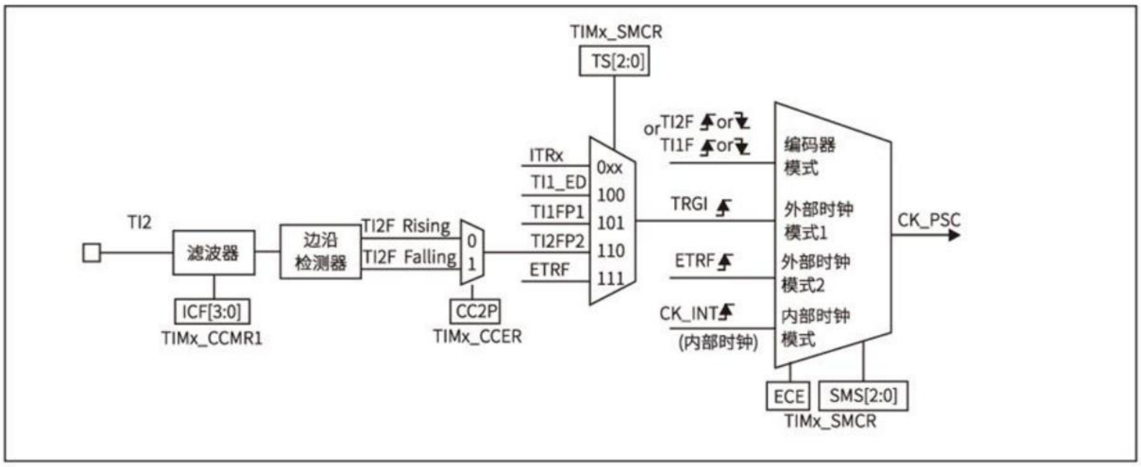


Figure 12-23 TI2 external clock connection example

For example, to configure an up counter to count on rising edges at the T12 input, use the following steps:

1. Configure TIM1_CCMR1 register CC2S=01, configure channel 2 to detect the rising edge of TI2 input
2. Configure IC2F[3:0] of the TIM1_CCMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000)
3. Configure CC2P=0 of the TIM1_CCER register, select the rising edge polarity
4. Configure SMS=111 in the TIM1_SMCR register to select the timer external clock mode
5. Configure TS=110 in the TIM1_SMCR register, and select TI2 as the trigger input source
6. Set CEN=1 in the TIM1_CR1 register to start the counter

Note: The capture prescaler is not used as a trigger, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge of TI2 and the actual clocking of the counter depends on the resynchronization circuit at the TI2 input.

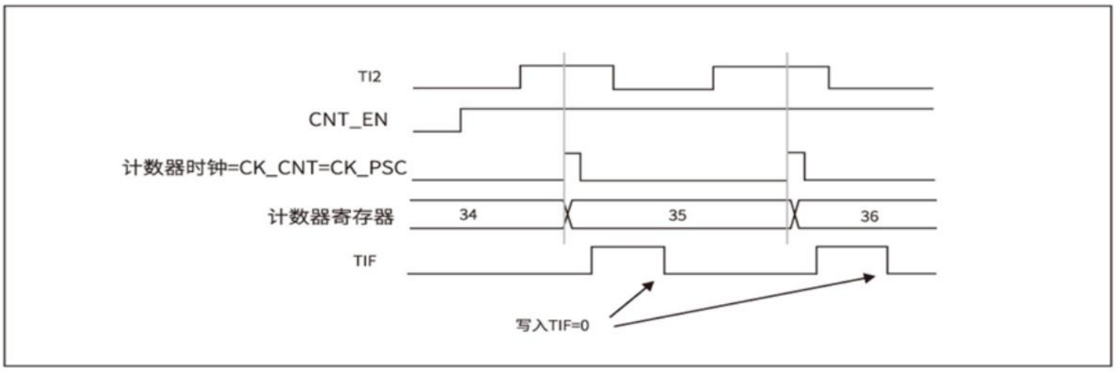


Figure 12-24 Control circuit in external clock mode 1

12.3.4.2 External Clock Source Mode 2

The method to select this mode is: make ECE=1 in the TIM1_SMCR register

The counter can count on every rising or falling edge of the external trigger ETR.

The figure below is a block diagram of the external trigger input

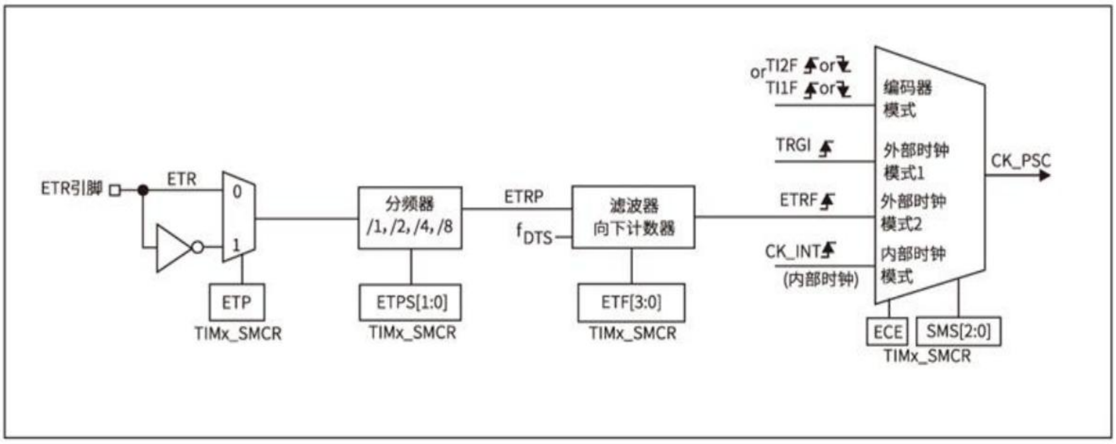


Figure 12-25 External trigger input block diagram

For example, to configure an up counter that counts every 2 rising edges at ETR, use the following steps:

1. No filter is needed in this example, set ETF[3:0]=0000 in the TIM1_SMCR register;
2. Set the prescaler, set ETPS[1:0]=01 in the TIM1_SMCR register;
3. Select the rising edge detection of ETR, set ETP=0 in the TIM1_SMCR register;
4. Turn on the external clock mode 2, write ECE=1 in the TIM1_SMCR register;
5. Start the counter, write CEN=1 in the TIM1_CR1 register;

The counter counts every 2 rising edges of ETR.

The delay between the rising edge of ETR and the actual clocking of the counter depends on the resynchronization circuitry on the ETRP signal.

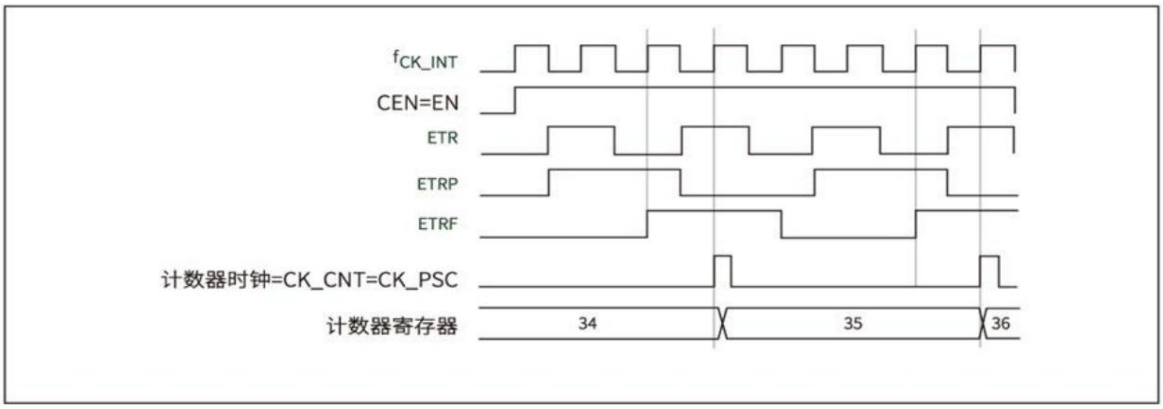


Figure 12-26 Control circuit in external clock mode 2

### 12.3.5 Capture/Compare Channels

Each capture/compare channel is built around a capture/compare register (including shadow registers), including the input part of the capture (digital word filter, multiplexer, and prescaler), and the output section (comparator and output control).

Figure 12-27 through Figure 12-30 are an overview of the capture/compare channel.

The input section samples the corresponding TIx input signal and produces a filtered signal TIxF. Then, an edge with polarity selection

The monitor generates a signal (TIxFPx) which can be used as an input trigger for the slave mode controller or as a capture control. The signal passes

The prescaler goes into the capture register (ICxPS).

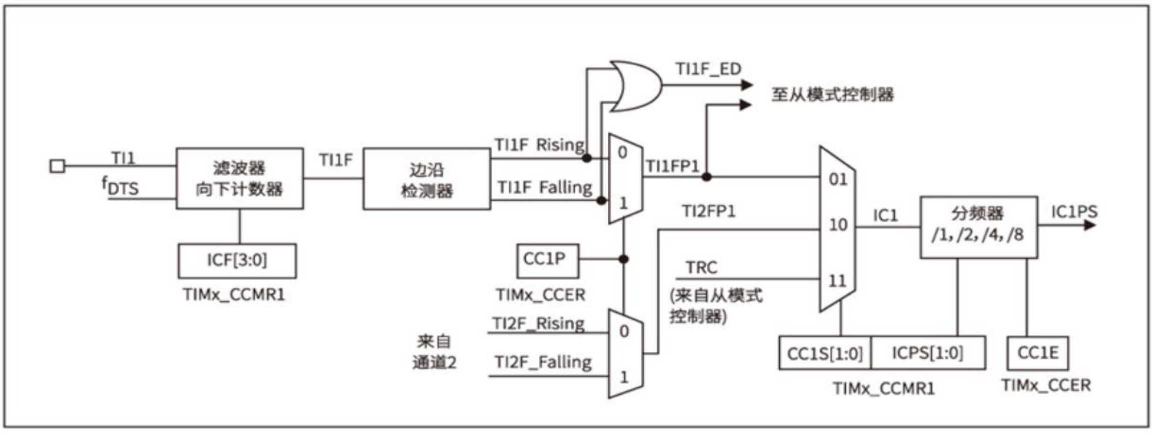


Figure 12-27 Capture/compare channel (eg: channel 1 input part)

The output section generates an intermediate waveform OCxRef (active high) as a reference, and the end of the chain determines the polarity of the final output signal.

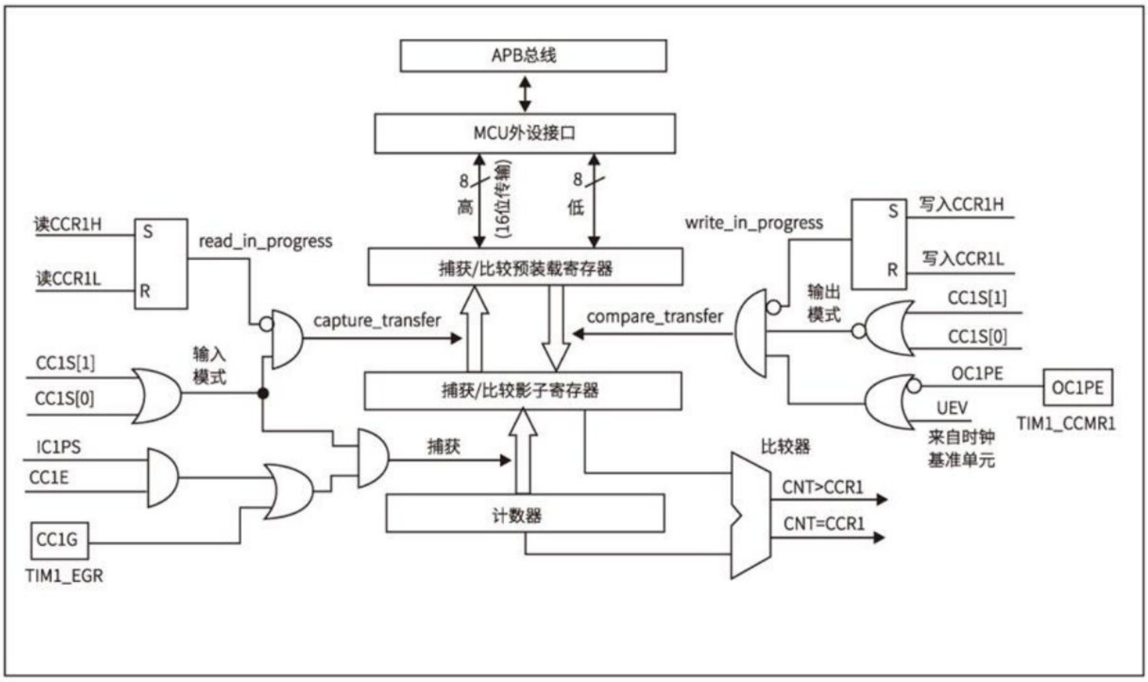


Figure 12-28 The main circuit of capture/compare channel 1

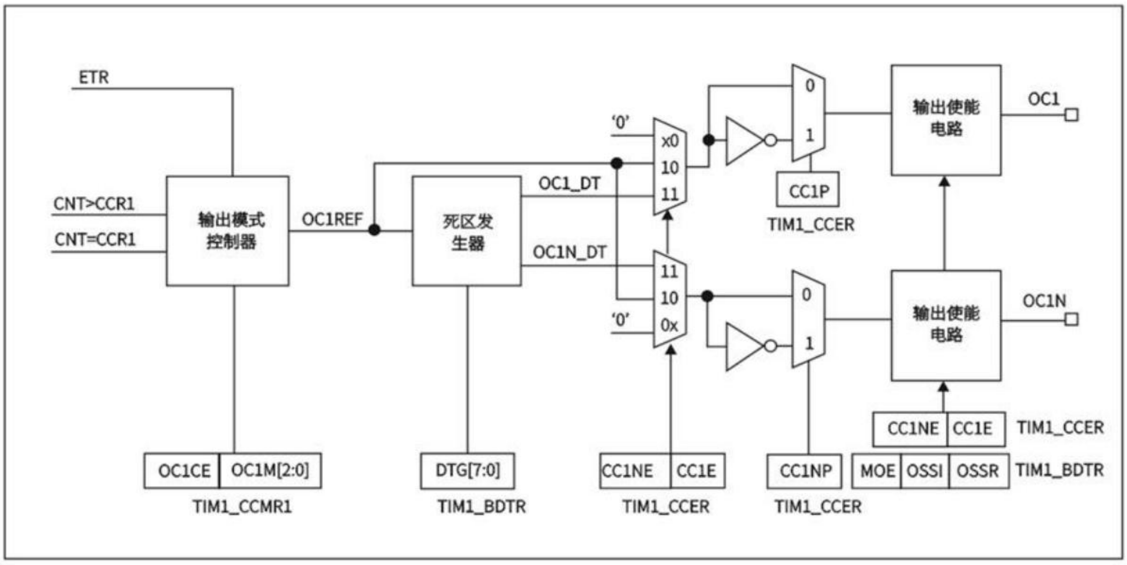


Figure 12-29 Capture/Compare Channel Output Section (Channels 1 to 3)

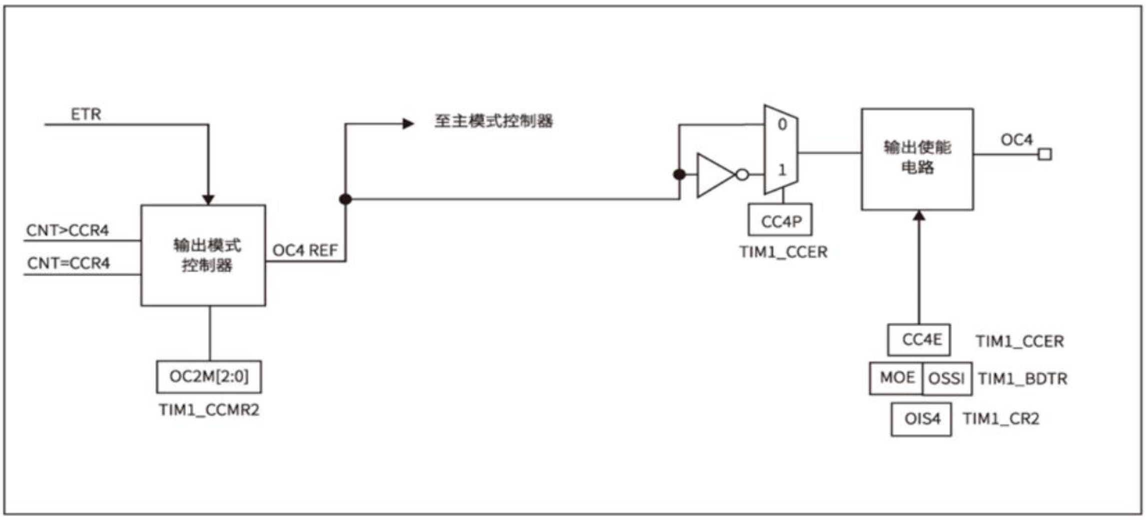


Figure 12-30 Capture/Compare Channel Output Section (Channel 4)

The capture/compare module consists of a preload register and a shadow register. The read and write process only operates the preload register. in capture In mode, the capture occurs on the shadow register, which is then copied into the preload register. In compare mode, the preload register's The content is copied into the shadow register, and then the content of the shadow register is compared with the counter.

### 12.3.6 Input capture mode

In input capture mode, the current value of the counter is latched into the capture/compare register (TIM1_CCRx) when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIM1_SR register) is set and an interrupt will be generated if enabled. If the CCxIF flag is already high when a capture event occurs, the repeated capture flag CCxOF (TIM1_SR register) is set. CCxIF is cleared by writing CCxIF=0, or by reading the capture data stored in the TIM1_CCRx register.

Write CCxOF=0 to clear CCxOF.

The following example shows how to capture the value of the counter to the TIM1_CCR1 register on the rising edge of the TI1 input. The steps are as follows:

ÿSelect valid input: TIM1_CCR1 must be connected to TI1 input, so write to TIM1_CCMR1 register

CC1S=01, as long as CC1S is not '00', the channel is configured as an input, and the TIM1_CCR1 register becomes read-only.

ÿ According to the characteristics of the input signal, configure the input filter to the required bandwidth (that is, when the input is TIx, the input filter control bit is the ICxF bit in the TIM1_CCMRx register). Assuming the input signal jitters for at most 5 internal clock cycles, we have to configure the filter bandwidth to be longer than 5 clock cycles; thus we can take 8 consecutive samples (at fDTS frequency) to confirm a true edge on TI1 Conversion, that is, write IC1F=0011 in the TIM1_CCMR1 register. ÿ To select the valid conversion edge of the TI1 channel, write CC1P=0 (rising edge) in the

TIM1_CCER register. ÿ Configure the input prescaler. In this example, we want the capture to occur at every valid level transition, so the prescaler

The register is disabled (write IC1PS=00 in TIM1_CCMR1 register).

ÿ Set CC1E=1 in the TIM1_CCER register to allow the value of the capture counter to be captured in the capture register. ÿ If required,

enable the associated interrupt request by setting the CC1IE bit in the TIM1_DIER register.

When an input capture occurs:

ÿWhen a valid level transition occurs, the counter value is transferred to the TIM1_CCR1 register. ÿ CC1IF flag is set

(interrupt flag). When at least 2 consecutive captures occur, CC1IF is not cleared, CC1OF is also cleared

set to 1.

ÿIf the CC1IE bit is set, an interrupt will be generated.

In order to deal with the capture overflow, it is recommended to read the data before reading the capture overflow flag, this is to avoid losing the capture overflow information that may be generated after reading the capture overflow flag and before reading the data.

Note: An input capture interrupt can be generated by software by setting the corresponding CCxG bit in the TIM1_EGR register.

## 12.3.7 PWM Input Mode

This mode is a special case of input capture mode and operates the same as input capture mode except for the following differences:

ÿ Two ICx signals are mapped to the same TIx input. ÿThe 2 ICx

signals are edge active, but opposite in polarity. ÿ One of the TIxFP signals

is used as a trigger input signal, and the slave mode controller is configured in reset mode.

For example, you need to measure the length (TIM1_CCR1 register) and duty cycle (TIM1_CCR2 register) of the PWM signal input to TI1, the specific

steps are as follows (depending on the frequency of CK_INT and the value of the prescaler)

ÿSelect valid input of TIM1_CCR1: set CC1S=01 in TIM1_CCMR1 register (select TI1). ÿSelect the effective polarity of TI1FP1

(used to capture data into TIM1_CCR1 and clear the counter): set CC1P=0 (rising edge has

effect).

ÿSelect valid input of TIM1_CCR2: Set CC2S=10 in TIM1_CCMR1 register (select TI1). ÿSelect the active polarity of TI1FP2

(capture data to TIM1_CCR2): set CC2P=1 (falling edge active). ÿSelect a valid trigger input signal: set TS=101 in the

TIM1_SMCR register (select TI1FP1). ÿ Configure the slave mode controller as reset mode: set SMS=100 in TIM1_SMCR.

ÿEnable capture: set CC1E=1 and CC2E=1 in TIM1_CCER register.

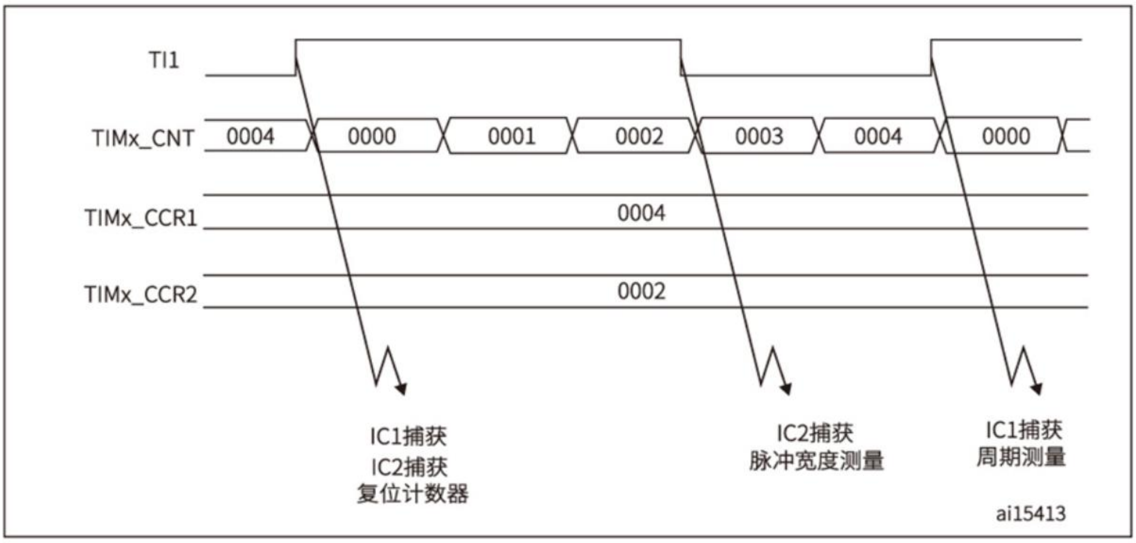


Figure 12-31 PWM input mode timing

Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can only be used

TIM1_CH1/TIM1_CH2 signal.

## 12.3.8 Forced output mode

In output mode (CCxS=00 in TIM1_CCMRx register), the output compare signal (OCxREF and corresponding OCx/OCxN) can be directly

It can be forced to active or inactive by software, independent of the comparison result between the output compare register and the counter.

Set the corresponding OCxM=101 in the TIM1_CCMRx register to force the output compare signal (OCxREF/OCx) to be valid. In this way, OCxREF is forced high (OCxREF is always active high), and OCx gets the opposite signal of CCxP polarity.

For example: CCxP=0 (OCx active high), then OCx is forced to be high.

Set OCxM=100 in TIM1_CCMRx register to force OCxREF signal low.

In this mode, the comparison between the TIM1_CCRx shadow register and the counter is still in progress and the corresponding flags will be modified. therefore The corresponding interrupt will still be generated. This is described in the Output Compare Mode section below.

## 12.3.9 Output Compare Mode

This function is used to control an output waveform, or to indicate that a given time has elapsed. When the counter is registered with the capture/compare When the contents of the registers are the same, the output compare function does the following operations:

ÿ Output the value defined by the output compare mode (OCxM bit in the TIM1_CCMRx register) and output polarity (CCxP bit in the TIM1_CCER register) to the corresponding pin. On a compare match, the output pin can maintain its level (OCxM=000), set to an active level (OCxM=001), set to an inactive level (OCxM=010), or toggle (OCxM=011).

ÿ Set the flag bit in the interrupt status register (CCxIF bit in the TIM1_SR register). ÿ An interrupt is generated if the corresponding interrupt mask is set (CCxIE bit in the TIM1_DIER register).

The OCxPE bit in TIM1_CCMRx selects whether the TIM1_CCRx register needs to use the preload register. In output compare mode, the Update event UEV has no effect on OCxREF and OCx outputs. The accuracy of the synchronization can reach one count cycle of the counter. output The compare mode (in single pulse mode) can also be used to output a single pulse.

Configuration steps for output compare mode:

1. Select the counter clock (internal, external, prescaler). 2. Write the corresponding data into the TIM1_ARR and TIM1_CCRx registers.

3. To generate an interrupt request, set the CCxIE bit.

4. Select the output mode, for example:

   - To toggle the output pin of OCx when the counter matches CCRx, set OCxM=011

   - Set OCxPE = 0 to disable preload register

   - Set CCxP = 0 to select polarity as active high

   - Set CCxE = 1 to enable the output

5. Set the CEN bit of the TIM1_CR1 register to start the counter

The TIM1_CCRx register can be updated by software at any time to control the output waveform, provided that the preload register is not used (OCxPE='0', otherwise the shadow register of TIM1_CCRx can only be updated at the next update event). The figure below gives the one example.

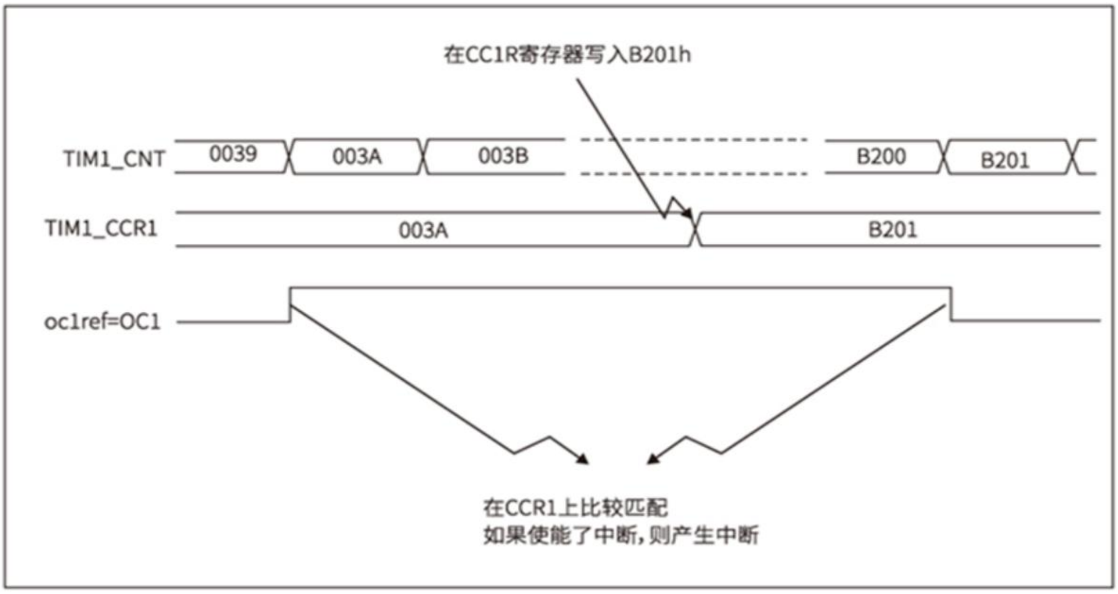


Figure 12-32 Output compare mode, flip OC1

## 12.3.10 PWM Mode

The Pulse Width Modulation mode can generate a frequency determined by the TIM1_ARR register and a duty cycle determined by the TIM1_CCRx register. Signal.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bits in the TIM1_CCMRx register can independently Set each OCx output channel to generate one PWM. The corresponding Preload the registers, and finally set the ARPE bit of the TIM1_CR1 register (in up-counting or centrosymmetric mode) to enable the automatic Preload registers for dynamic loads.

Only when an update event occurs, the preload register can be transferred to the shadow register, so before the counter starts counting, it must be All registers must be initialized by setting the UG bit in the TIM1_EGR register.

The polarity of OCx can be set by software with the CCxP bit in the TIM1_CCER register, it can be set to active high or low flat and effective. The output enable of OCx is via (in TIM1_CCER and TIM1_BDTR registers) CCxE, CCxNE, MOE, OSSI and Combination control of OSSR bits. See the description of the TIM1_CCER register for details.

In PWM mode (mode 1 or mode 2), TIM1_CNT and TIM1_CCRx are always comparing, (according to the counting method of the counter Direction) to determine whether TIM1_CCRxÿTIM1_CNT or TIM1_CNTÿTIM1_CCRx.

Depending on the state of the CMS bit in the TIM1_CR1 register, the timer can generate edge-aligned PWM signals or center-aligned PWM Signal.

### 12.3.10.1 PWM Edge-Aligned Mode

Count Up Configuration

Counting up is performed when the DIR bit in the TIM1_CR1 register is low.

Below is an example of PWM mode 1. When TIM1_CNT<TIM1_CCRx, the PWM reference signal OCxREF is high, otherwise it is Low. If the compare value in TIM1_CCRx is greater than the auto-reload value (TIM1_ARR), OCxREF remains '1'. if compare value is 0, OCxREF remains '0'. The figure below is an example of an edge-aligned PWM waveform when TIM1_ARR=8.

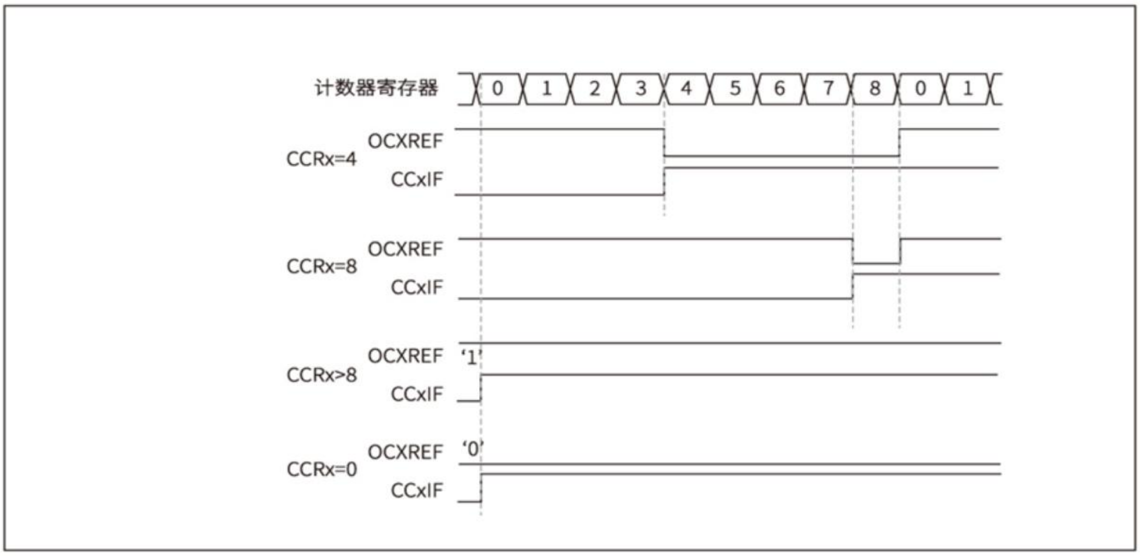


Figure 12-33 Edge-aligned PWM waveform (ARR=8)

Count Down Configuration

Down counting is performed when the DIR bit of the TIM1_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when TIM1_CNT>TIM1_CCRx, otherwise it is high. If the TIM1_CCRx in If the compare value is greater than the auto-reload value in TIM1_ARR, OCxREF remains '1'. 0% PWM cannot be generated in this mode waveform.

### 12.3.10.2 PWM Center-Aligned Mode

Center-aligned mode when the CMS bit in the TIM1_CR1 register is not '00' (all other configurations are valid for OCxREF/OCx signal numbers have the same effect). Depending on the CMS bit setting, the compare flag can be set to 1 when the counter counts up, Set to 1 when counting down, or when the counter is counting up and down. The counting direction bit (DIR) in the TIM1_CR1 register is controlled by the hard software update, do not modify it with software.

The figure below gives some examples of center-aligned PWM waveforms

ÿ TIM1_ARR=8

ÿ PWM mode 1

ÿ CMS=01 of TIM1_CR1 register, in center-aligned mode 1, set compare flag when counter counts down.

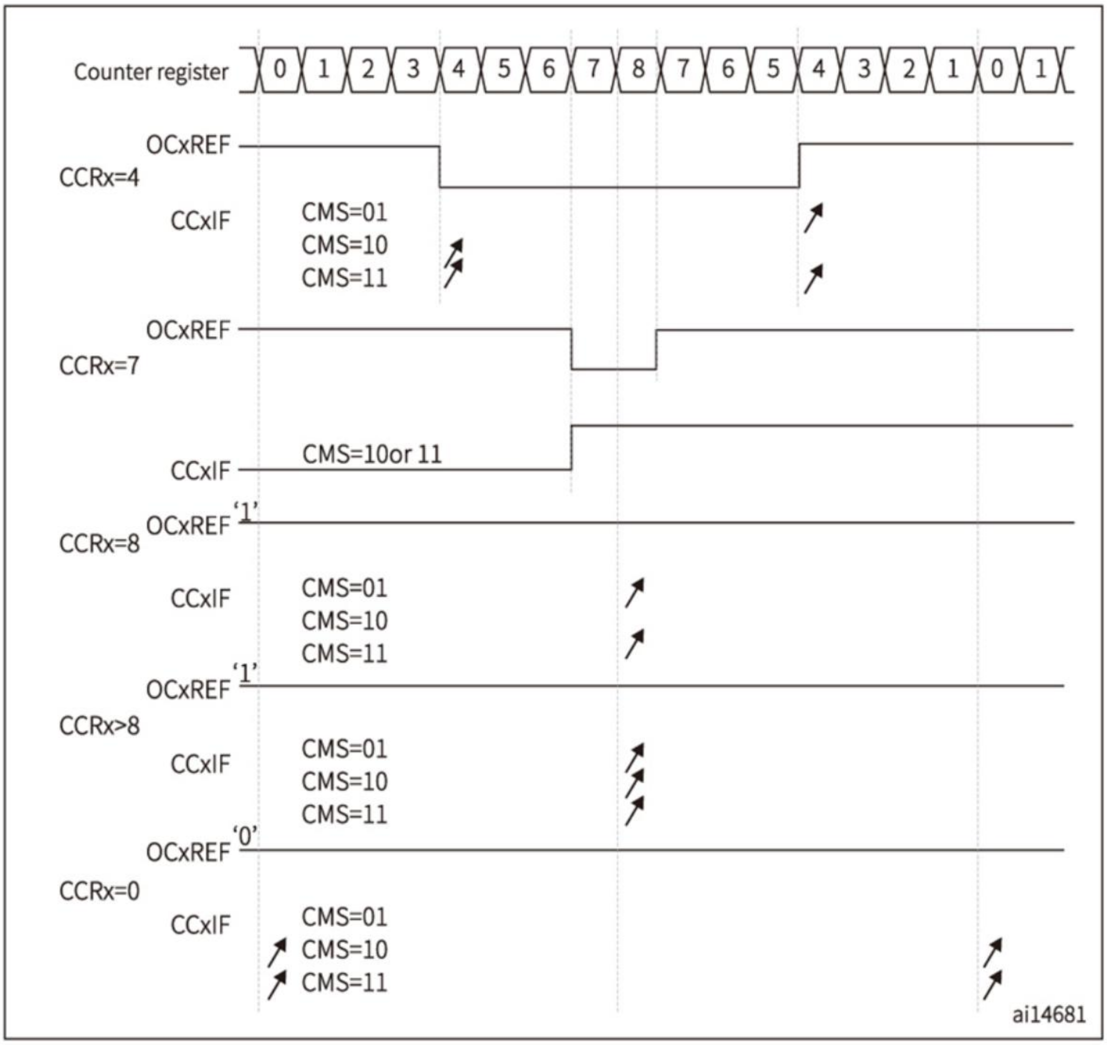


Figure 12-34 Center aligned PWM waveform (APR=8)

Tips for using center-aligned modes:

ÿ Use current up/down count configuration when entering center-aligned mode. This means that whether the counter counts up or down takes

Depends on the current value of the DIR bit in the TIM1_CR1 register. Also, software cannot modify the DIR and CMS bits at the same time.

ÿ It is not recommended to overwrite counters when running in center-aligned mode, as this can produce unpredictable results. In particular:

- If the value written to the counter is greater than the auto-reload value (TIM1_CNT>TIM1_ARR), the direction will not be updated. example

For example, if the counter is counting up, it continues to count up.

- If 0 or the value of TIM1_ARR is written to the counter, the direction is updated, but no update event UEV is generated.

ÿThe safest way to use center-aligned mode is to generate a software update (set TIM1_EGR

UG bit in the bit), and do not modify the value of the counter while counting is in progress.

### 12.3.11 Complementary output and dead time insertion

The advanced control timer (TIM1) is capable of outputting two complementary signals and can manage the momentary turn-off and turn-on of the output. Usually during this time Known as the dead zone, the user should adjust it according to the connected output devices and their characteristics (delay of level shifting, delay of power switching, etc.) total dead time.

Polarity can be selected independently for each output by configuring the CCxP and CCxNP bits in the TIM1_CCER register (main output OCx or complementary output OCxN).

The complementary signals OCx and OCxN are controlled by a combination of the following control bits: CCxE and CCxNE bits of the TIM1_CCER register, The MOE, OISx, OISxN, OSSI and OSSR bits in the TIM1_BDTR and TIM1_CR2 registers are detailed in Table 12-4 with brake function Control bits for the complementary output channels OCx and OCxN. In particular, the deadband is activated when transitioning to the IDLE state (MOE drops to 0) live.

Setting the CCxE and CCxNE bits at the same time will insert the deadband, and if the brake circuit is present, also set the MOE bit. each channel There is a 10-bit dead-time generator. The reference signal OCxREF can generate 2 outputs OCx and OCxN. If OCx and OCxN Effective for high:

ÿ The OCx output signal is the same as the reference signal, but its rising edge has a delay relative to the rising edge of the reference signal.

ÿ The OCxN output signal is opposite to the reference signal, but its rising edge has a delay relative to the falling edge of the reference signal.

If the delay is greater than the currently active output width (OCx or OCxN), no corresponding pulse will be generated.

The following figures show the relationship between the output signal of the dead-band generator and the current reference signal OCxREF. (assuming CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1)

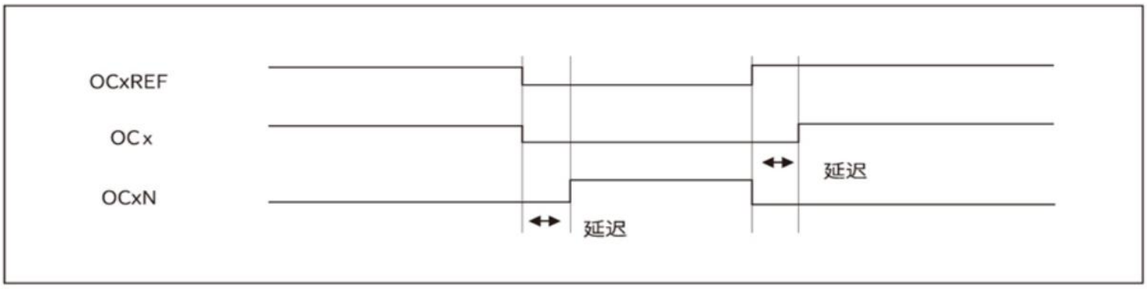


Figure 12-35 Complementary output with dead zone insertion

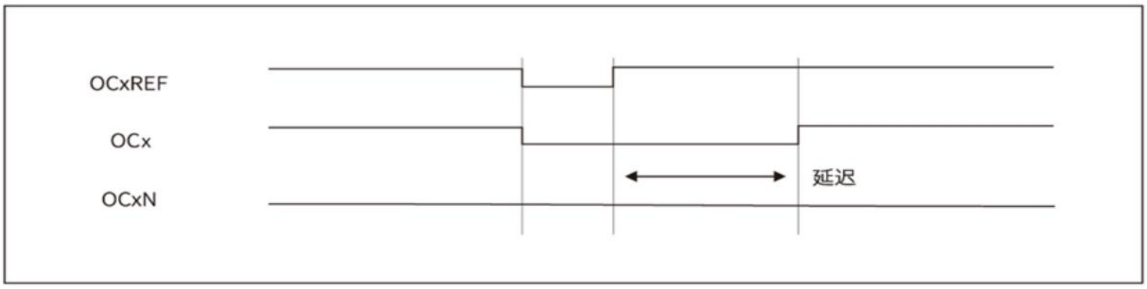


Figure 12-36 Dead zone waveform delay is greater than negative pulse

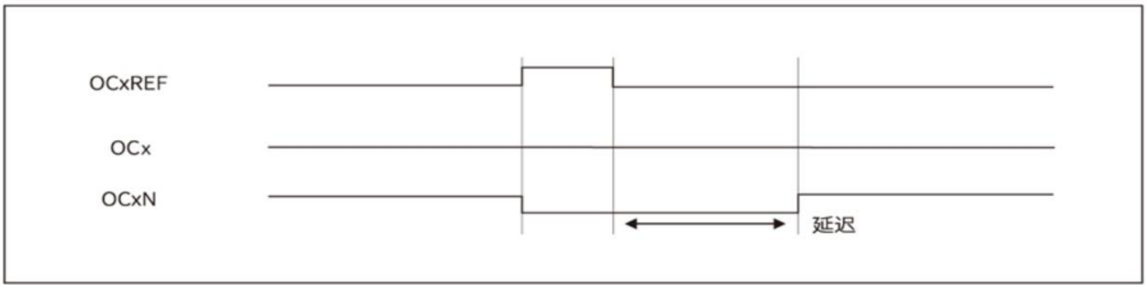


Figure 12-37 Dead zone waveform delay is greater than positive pulse

The dead-band delay is the same for each channel and is programmed by the DTG bit in the TIM1_BDTR register. See TIM1 Brake and Delay calculation in dead-band register (TIM1_BDTR).

### 12.3.11.1 Redirect OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), by configuring the CCxE and CCxNE bits of the TIM1_CCER register, OCxREF can be redirected to OCx or OCxN output.

This function can be used to send a special waveform (such as PWM or static active level). Another effect is to have both outputs at an inactive level at the same time, or at an active level and a complementary output with deadband.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it will not invert and go high immediately when OCxREF is valid. For example, if CCxNP=0, then OCxN=OCxREF. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1), when OCxREF OCx is active when OCx is high; and OCxN, on the contrary, becomes active when OCxREF is low.

### 12.3.12 Using the brake function

When using the brake function, according to the corresponding control bits (MOE, OSSI and OSSR bits in the TIM1_BDTR register, OISx and OISxN bits in the TIM1_CR2 register), the output enable signal and the inactive level will be modified. But whenever OCx and OCxN output outputs cannot be active at the same time. See Table 12-4 for the control of complementary output channels OCx and OCxN with brake function. position.

The break source can be either a break input pin or a clock fail event. A clock fail event is reset by the clock in the clock controller security system generated.

After system reset, the braking circuit is disabled and the MOE bit is low. Setting the BKE bit in the TIM1_BDTR register can enable the brake function, The polarity of the brake input signal can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. when When writing BKE and BKP bits, there will be a delay of 1 APB clock cycle before actually writing, so you need to wait for an APB clock cycle before the written bits can be read back correctly.

Because the MOE falling edge can be asynchronous, between the actual signal (acted on the output) and the synchronous control bit (in the TIM1_BDTR register) A resynchronization circuit is provided between them. This resynchronization circuit introduces a delay between the asynchronous and synchronous signals. In particular, if when When it is low and write MOE=1, a delay (empty instruction) must be inserted before reading it to read the correct value. This is because writing The read is an asynchronous signal and the read is a synchronous signal.

When braking occurs (selected level present at the brake input), the following actions take place:

ÿ The MOE bit is cleared asynchronously, putting the output in an inactive state, an idle state, or a reset state (selected by the OSSI bit). this special The behavior is still valid when the MCU's oscillator is turned off.

ÿOnce MOE=0, each output channel outputs the level set by the OISx bits in the TIM1_CR2 register. If OSSI=0, then The timer releases the enable output, otherwise the enable output is always high.

ÿ When using complementary output:

- The output is first placed in a reset state, ie an inactive state (depending on polarity). This is an asynchronous operation, even when the timer does not This function is also effective when the clock is turned on.
- If the timer clock is still present, the deadband generator will be reactivated, after the deadband according to OISx and OISxN The level indicated by the bit drives the output port. Even in this case, OCx and OCxN cannot be driven to valid Level. Note that due to resynchronization of the MOE, the dead time is longer than usual (about 2 clocks of ck_tim cycle).
- If OSSI = 0, the timer releases the enable output, otherwise keeps the enable output; or once one of CCxE and CCxNE goes high , the enable output goes high.

ÿIf the BIE bit in the TIM1_DIER register is set, when the brake status flag (BIF bit in the TIM1_SR register) is '1' , an interrupt is generated.

ÿIf the AOE bit in the TIM1_BDTR register is set, the MOE bit is automatically set at the next update event UEV; This can be used for shaping, for example. Otherwise, MOE remains low until it is set to '1' again; at this time, this feature can Used for safety, you can connect the brake input to the alarm output of the power driver, thermal sensor or other safety devices superior.

Note: Brake input is level effective. So, MOE cannot be set (automatically or by software) at the same time when the brake input is active. same

, the status flag BIF cannot be cleared.

The brake is generated by the BRK input, whose active polarity is programmable and enabled by the BKE bit in the TIM1_BDTR register.

In addition to brake input and output management, write protection is also implemented in the brake circuit to keep the application safe. It allows the user to freeze several

Configuration parameters (deadband length, OCx/OCxN polarity and disabled state, OCxM configuration, brake enable and polarity).

The user can select one of the three levels of protection through the LOCK bit in the TIM1_BDTR register, see section 0 TIM1 brake and dead zone

register (TIM1_BDTR). The LOCK bit can only be modified once after MCU reset.

The figure below shows an example output in response to braking.

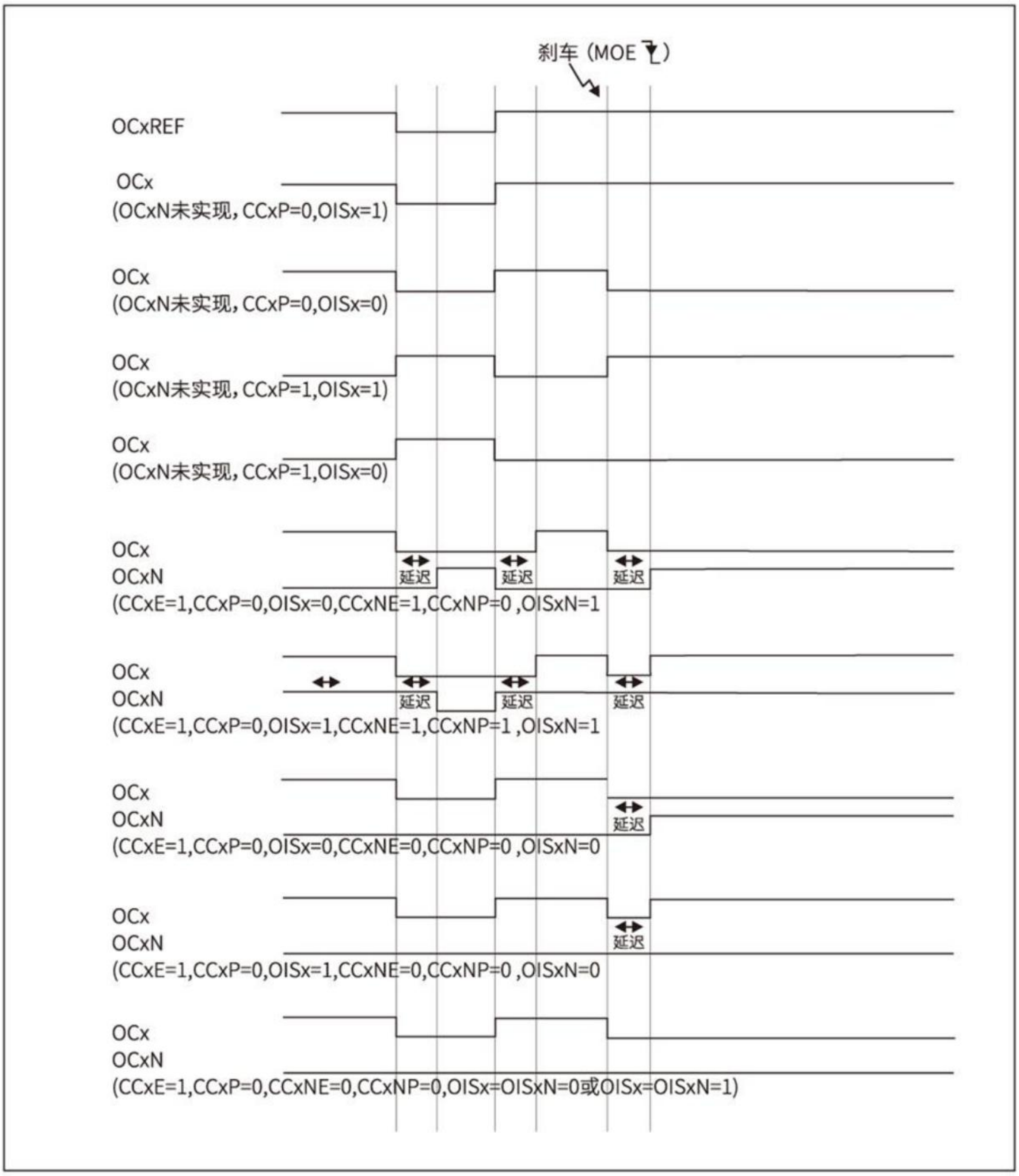

刹车（MOE ↓）
OCxREF
OCx
(OCxN未实现, CCxP=0,OISx=1)
OCx
(OCxN未实现, CCxP=0,OISx=0)
OCx
(OCxN未实现, CCxP=1,OISx=1)
OCx
(OCxN未实现, CCxP=1,OISx=0)
OCx
OCxN
延迟
(CCxE=1,CCxP=0,OISx=0,CCxNE=1,CCxNP=0 ,OISxN=1
OCx
OCxN
延迟
(CCxE=1,CCxP=0,OISx=1,CCxNE=1,CCxNP=1 ,OISxN=1
OCx
OCxN
延迟
(CCxE=1,CCxP=0,OISx=0,CCxNE=0,CCxNP=0 ,OISxN=0
OCx
OCxN
延迟
(CCxE=1,CCxP=0,OISx=1,CCxNE=0,CCxNP=0 ,OISxN=0
OCx
OCxN
(CCxE=1,CCxP=0,CCxNE=0,CCxNP=0,OISx=OISxN=0或OISx=OISxN=1)

Figure 12-38 Output in response to braking

### 12.3.13 CLEARING THE OCxREF SIGNAL ON EXTERNAL EVENT

For a given channel, set the corresponding OCxCE bit in the TIM1_CCMRx register to '1' to enable

A high level pulls the OCxREF signal low, and the OCxREF signal will remain low until the next update event UEV occurs.

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

For example, the OCxREF signal can be tied to the output of a comparator for current control. At this time, ETR must be configured as follows:

1. External trigger prescaler must be off: ETPS[1:0]=00 in TIM1_SMCR register.
2. External clock mode 2 must be disabled: ECE=0 in TIM1_SMCR register.
3. External Trigger Polarity (ETP) and External Trigger Filter (ETF) can be configured as required.

The figure below shows the behavior of the OCxREF signal for different values of OCxCE when the ETRF input goes high. In this example, set

Timer TIM1 is placed in PWM mode.

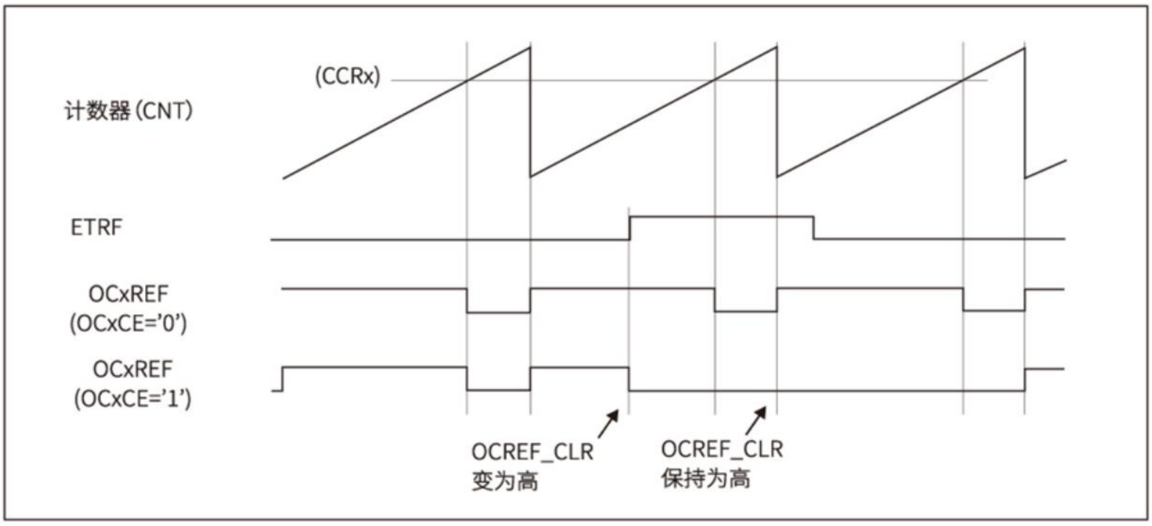


Figure 12-39 Clear OCxREF of TIM1

### 12.3.14 Generate six-step PWM output

When complementary outputs are required on one channel, the preloaded bits are OCxM, CCxE, and CCxNE. During a COM commutation event, these

The preload bits are transferred to the shadow register bits. In this way, you can pre-set the next step configuration and modify it at the same time

Change the configuration of all channels. COM can be generated by software by setting the COM bit of the TIM1_EGR register, or on the rising edge of TRGI

Generated by hardware.

When a COM event occurs, a flag bit (COMIF bit in the TIM1_SR register) will be set, and if it has been set

The COMIE bit of the TIM1_DIER register generates an interrupt.

The figure below shows the OCx and OCxN output for three different configurations when a COM event occurs.

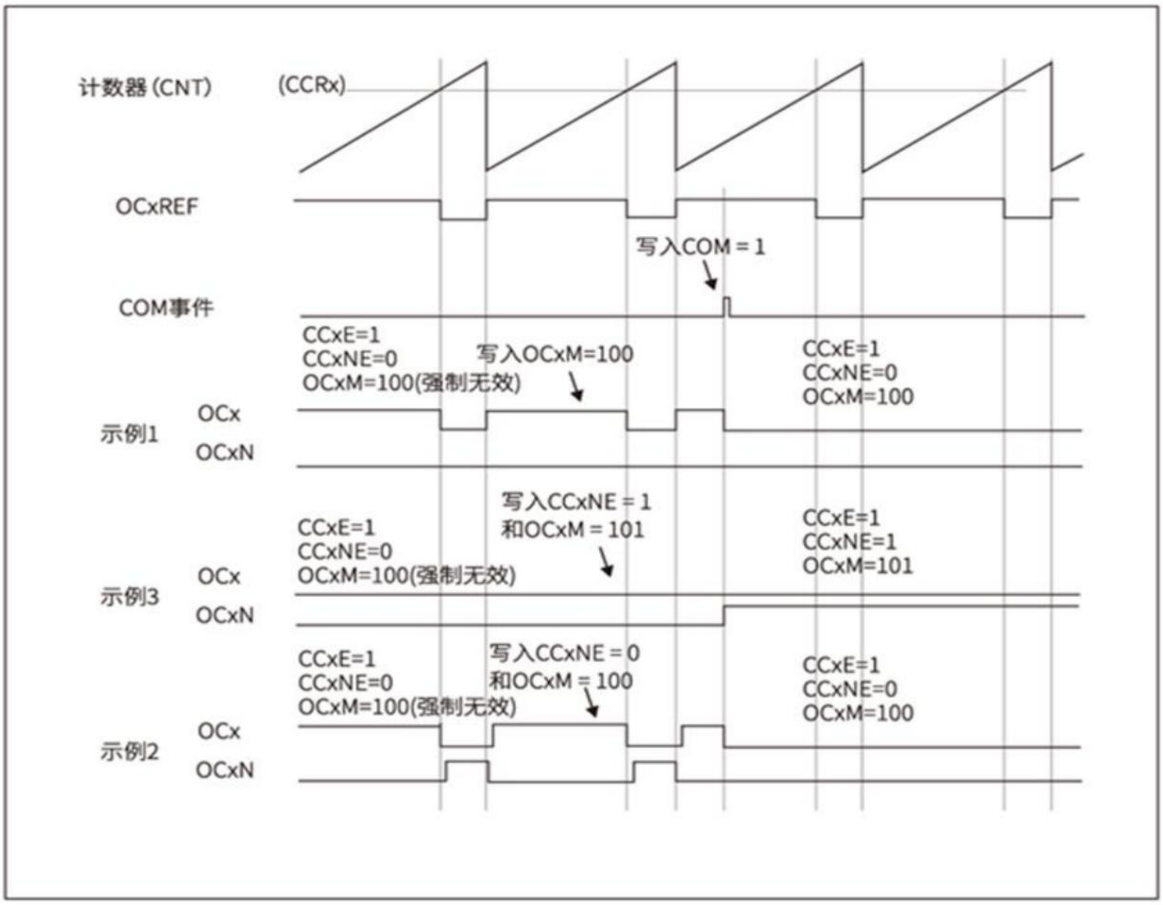


Figure 12-40 Generate six-step PWM, using COM example (OSSR=1)

### 12.3.15 Single Pulse Mode

One-Pulse Mode (OPM) is a special case of the aforementioned modes. This mode allows the counter to respond to a stimulus and generate a pulse of programmable pulse width after a programmable delay.

The counter can be started from the mode controller to generate waveforms in output compare mode or PWM mode. Setting the OPM bit in the TIM1_CR1 register selects the single pulse mode, which allows the counter to automatically stop when the next update event UEV is generated.

A pulse is generated only when the comparison value differs from the counter's initial value. Before starting (while the timer is waiting to fire), the Configure as follows:

ÿ Up counting method: counter CNT < CCRx ÿ ARR (especially, 0 < CCRx)

ÿ Down counting method: counter CNT > CCRx

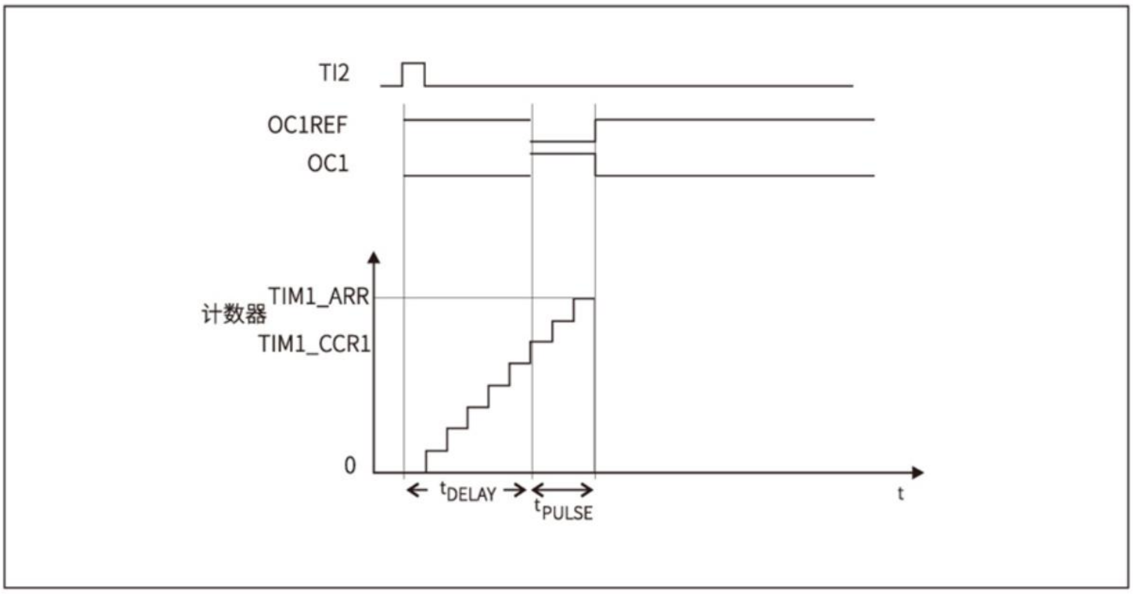


Figure 12-41 Example of single pulse mode

For example, you need to generate a length of

Positive pulse of tPULSE.

Assuming TI2FP2 as trigger 1:

ÿ Set CC2S=01 in the TIM1_CCMR1 register to map TI2FP2 to TI2. ÿ Set CC2P=0 in the TIM1_CCER

register to enable TI2FP2 to detect the rising edge. ÿSet TS=110 in the TIM1_SMCR register, TI2FP2 is used

as the trigger (TRGI) of the slave mode controller. ÿ Set SMS=110 in TIM1_SMCR register (trigger mode), TI2FP2 is used to

start the counter.

The waveform of the OPM is determined by the value written to the compare register (considering the clock frequency and counter prescaler)

ÿ tDELAY is defined by the value in the TIM1_CCR1 register. ÿ

tPULSE is defined by the difference between the autoload value and the compare value (TIM1_ARR - TIM1_CCR1).

ÿAssuming that a waveform from 0 to 1 is to be generated when a compare match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preload value

Waveform; first set OC1M=111 in the TIM1_CCMR1 register to enter PWM mode 2; selectively use

Can preload registers: set OC1PE=1 in TIM1_CCMR1 and ARPE in TIM1_CR1 register; then in

Fill in the comparison value in the TIM1_CCR1 register, fill in the autoload value in the TIM1_ARR register, and set the UG bit to generate

An update event, then waits for an external trigger event on TI2. In this example, CC1P=0.

In this example, the DIR and CMS bits in the TIM1_CR1 register should be low. Since only one pulse is required, it must be set

Set OPM=1 in the TIM1_CR1 register to stop counting at the next update event (when the counter rolls over from the autoload value to 0).

Special case: OCx fast enable

In Single Pulse mode, the edge detection logic at the TIx input pin sets the CEN bit to start the counter. Then between the counter and the compare value

The comparison operation produces an inversion of the output. But these operations require a certain clock cycle, so it limits the minimum delay that can be obtained

tDELAY.

If you want to output the waveform with the minimum delay, you can set the OCxFE bit in the TIM1_CCMRx register; at this time, OCxREF (and OCx) directly

In response to the stimulus without relying on the result of the comparison, the output waveform is the same as when the comparison matched. OCxFE is only available if the channel is configured as

It works in PWM1 and PWM2 mode.

## 12.3.16 Encoder interface mode

The method to select the encoder interface mode is: if the counter only counts on the edge of TI2, set the TIM1_SMCR register SMS=001; if only counting on the edge of TI1, set SMS=010; if the counter counts on the edge of TI1 and TI2 at the same time, set SMS=011.

TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIM1_CCER register;

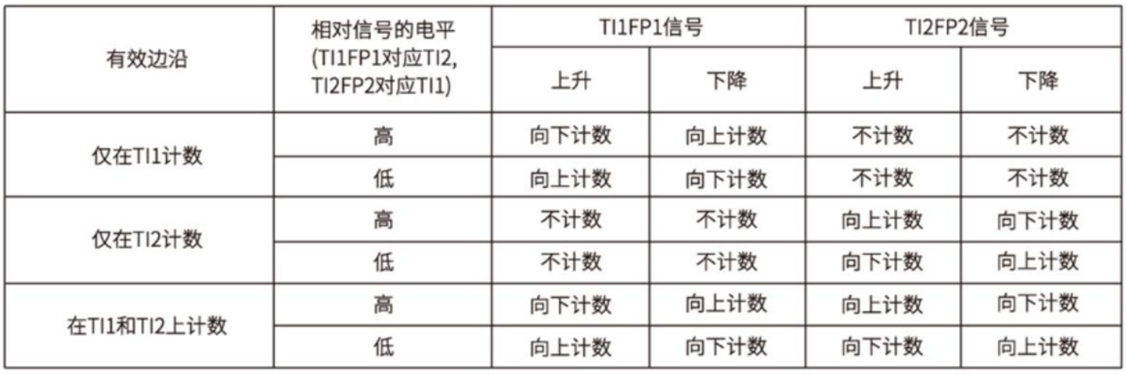
Input filter programming.

Two inputs TI1 and TI2 are used to interface the incremental encoder. Referring to Table 12-1, assuming that the counter has been enabled (CEN=1 in the TIM1_CR1 register), the counter is driven by every valid transition on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are TI1 and the signal of TI2 after passing through the input filter and polarity control; if there is no filter and phase change, then TI1FP1=TI1, TI2FP2=TI2. According to the transition sequence of the two input signals, count pulses and direction signals are generated. According to the transition sequence of the two input signals, the counter Count up or down, and the hardware sets the DIR bit of the TIM1_CR1 register accordingly. Regardless of whether the counter is counted by TI1 Counting, relying on TI2 counting or relying on TI1 and TI2 counting at the same time, the transition at any input terminal (TI1 or TI2) will be recalculated DIR bit.

Encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously between 0 and the autoload value of the TIM1_ARR register (either counting from 0 to ARR or counting from ARR to 0, depending on the direction). So TIM1_ARR must be configured before counting; likewise, capture, comparator, prescaler, repeat counter, trigger input Features, etc. still work as usual. Encoder mode and external clock mode 2 are not compatible and therefore cannot operate simultaneously.

In this mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so the content of the counter always indicates the position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The table below lists all possible combinations, assuming TI1 and TI2 are different time conversion.

Table 12-1 Relationship between counting direction and encoder signal



| 有效边沿 | 相对信号的电平<br>(TI1FP1对应TI2,<br>TI2FP2对应TI1) | TI1FP1信号 | | TI2FP2信号 | |
|---|---|---|---|---|---|
| | | 上升 | 下降 | 上升 | 下降 |
| 仅在TI1计数 | 高 | 向下计数 | 向上计数 | 不计数 | 不计数 |
| | 低 | 向上计数 | 向下计数 | 不计数 | 不计数 |
| 仅在TI2计数 | 高 | 不计数 | 不计数 | 向上计数 | 向下计数 |
| | 低 | 不计数 | 不计数 | 向下计数 | 向上计数 |
| 在TI1和TI2上计数 | 高 | 向下计数 | 向上计数 | 向上计数 | 向下计数 |
| | 低 | 向上计数 | 向下计数 | 向下计数 | 向上计数 |

An external incremental encoder can be directly interfaced with the MCU without external interface logic. However, it is common to use a comparator to convert The differential output of the coder is converted to a digital signal, which greatly increases the ability to resist noise interference. The third signal output by the encoder indicates mechanical zero, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing count signal generation and direction control. It also shows that when both edges are selected, How input jitter is suppressed; jitter can occur when the sensor is positioned close to a transition point. In this example, we Suppose the configuration is as follows:

ÿ CC1S='01' (TIM1_CCMR1 register, IC1FP1 is mapped to TI1)

ÿ CC2S='01' (TIM1_CCMR2 register, IC2FP2 is mapped to TI2)

ÿ CC1P='0' (TIM1_CCER register, IC1FP1 is not inverted, IC1FP1=TI1)

ÿ CC2P='0' (TIM1_CCER register, IC2FP2 is not inverted, IC2FP2=TI2)

ÿ SMS='011' (TIM1_SMCR register, all inputs are valid on rising and falling edges)

ÿ CEN='1' (TIM1_CR1 register, counter enable)

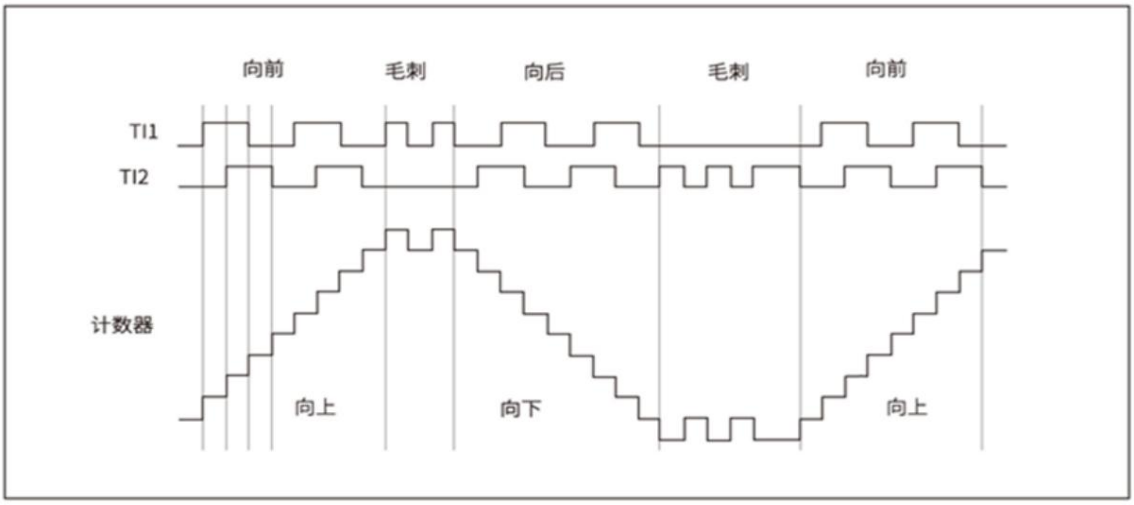


Figure 12-42 Example of counter operation in encoder mode

The figure below is an example of the operation of the counter when the polarity of IC1FP1 is reversed (CC1P='1', other configurations are the same as the above example)

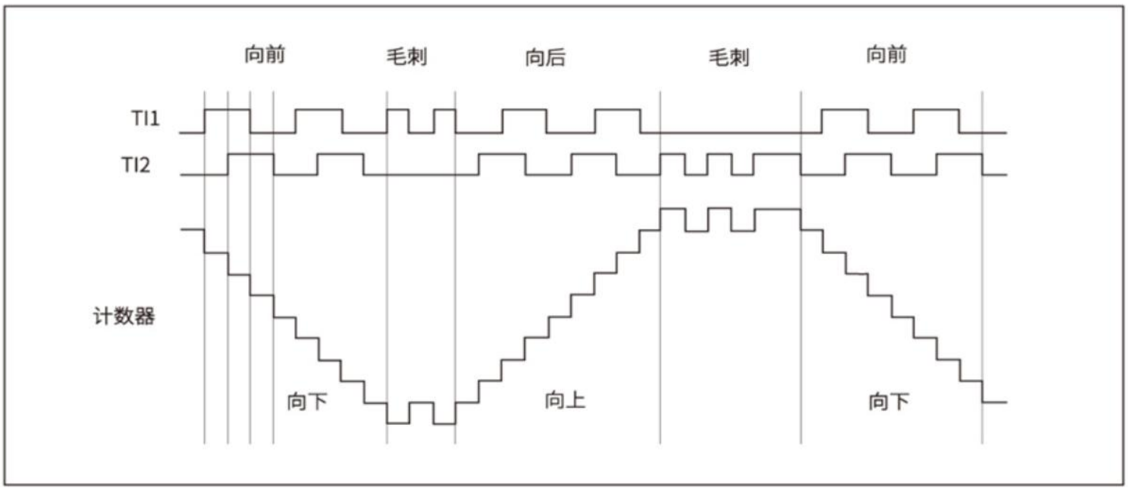


Figure 12-43 IC1FP1 Inverted Encoder Interface Mode Example

When the timer is configured in encoder interface mode, it provides information on the current position of the sensor. Timing in capture mode using the second configuration

The encoder can measure the interval between two encoder events and obtain dynamic information (velocity, acceleration, deceleration). Code indicating mechanical zero point

The encoder output can be used for this purpose. Depending on the interval between two events, the counter can be read out at a fixed time. If possible, you can

to latch the counter value into the third input capture register (the capture signal must be periodic and can be generated by another timer).

### 12.3.17 Timer input XOR function

The TI1S bit in the TIM1_CR2 register allows the input filter of channel 1 to be connected to the output of an exclusive OR gate whose three inputs are TIM1_CH1, TIM1_CH2 and TIM1_CH3.

The XOR output can be used for all timer input functions such as trigger or input capture. The next section 12.3.18 gives an example of this feature being used to connect Hall sensors.

### 12.3.18 Interface with Hall sensors

When using the advanced control timer (TIM1) to generate a PWM signal to drive the motor, another general-purpose TIM2 timer can be used as an "interface timer" to connect the Hall sensor, see Figure 12-44, 3 timer input pins (CC1, CC2, CC3) are connected to the TI1 input channel (selected by setting the TI1S bit in the TIM1_CR2 register) through an exclusive OR gate, and the "interface timer" captures this signal.

The slave mode controller is configured in reset mode and the slave input is TI1F_ED. The counter restarts counting from 0 each time one of the 3 inputs changes. This produces a time base that is triggered by any change in the Hall input.

The capture/compare channel 1 on the "interface timer" is configured as capture mode, and the capture signal is TRC (see Figure 12-27). The captured value reflects the time delay between two input changes, giving information on the motor speed.

The "interface timer" can be used to generate a pulse in output mode, which can be used (by triggering a COM event) to change the properties of each channel of the advanced timer TIM1, while the advanced control timer generates a PWM signal to drive the motor. Therefore the "interface timer" channel must be programmed to generate a positive pulse after a specified delay (output compare or PWM mode), which is sent to the advanced control timer TIM1 via the TRGO output.

Example: The Hall input is connected to the TIM1 timer, and it is required to change the PWM configuration of the advanced control timer TIM1 at a specified time each time any Hall input changes.

ÿSet the TI1S bit of TIM1_CR2 register to '1', configure three timer input logic or to TI1 input, ÿTime base programming: set TIM1_ARR to its maximum value (counter must be cleared by TI1 change). Set the prescaler to get a
The maximum counter period that is longer than the time between two changes on the sensor.

ÿSet channel 1 to capture mode (select TRC): set CC1S=01 in the TIM1_CCMR1 register, if necessary, you can also set
set digital filter.

ÿSet channel 2 to PWM2 mode with required delay: set OC2M=111 and
CC2S=00.

ÿSelect OC2REF as trigger output on TRGO: set MMS=101 in TIM1_CR2 register.

In the advanced control register TIM1, the correct ITR input must be a trigger input, the timer is programmed to generate a PWM signal, the capture/ compare control signal is preloaded (CCPC=1 in the TIM1_CR2 register), and the trigger input controls COM Event (CCUS=1 in TIM1_CR2 register). After a COM event, write the next PWM control bits (CCxE, OCxM), which can be implemented in the interrupt subroutine that handles the rising edge of OC2REF.

The figure below shows this instance.

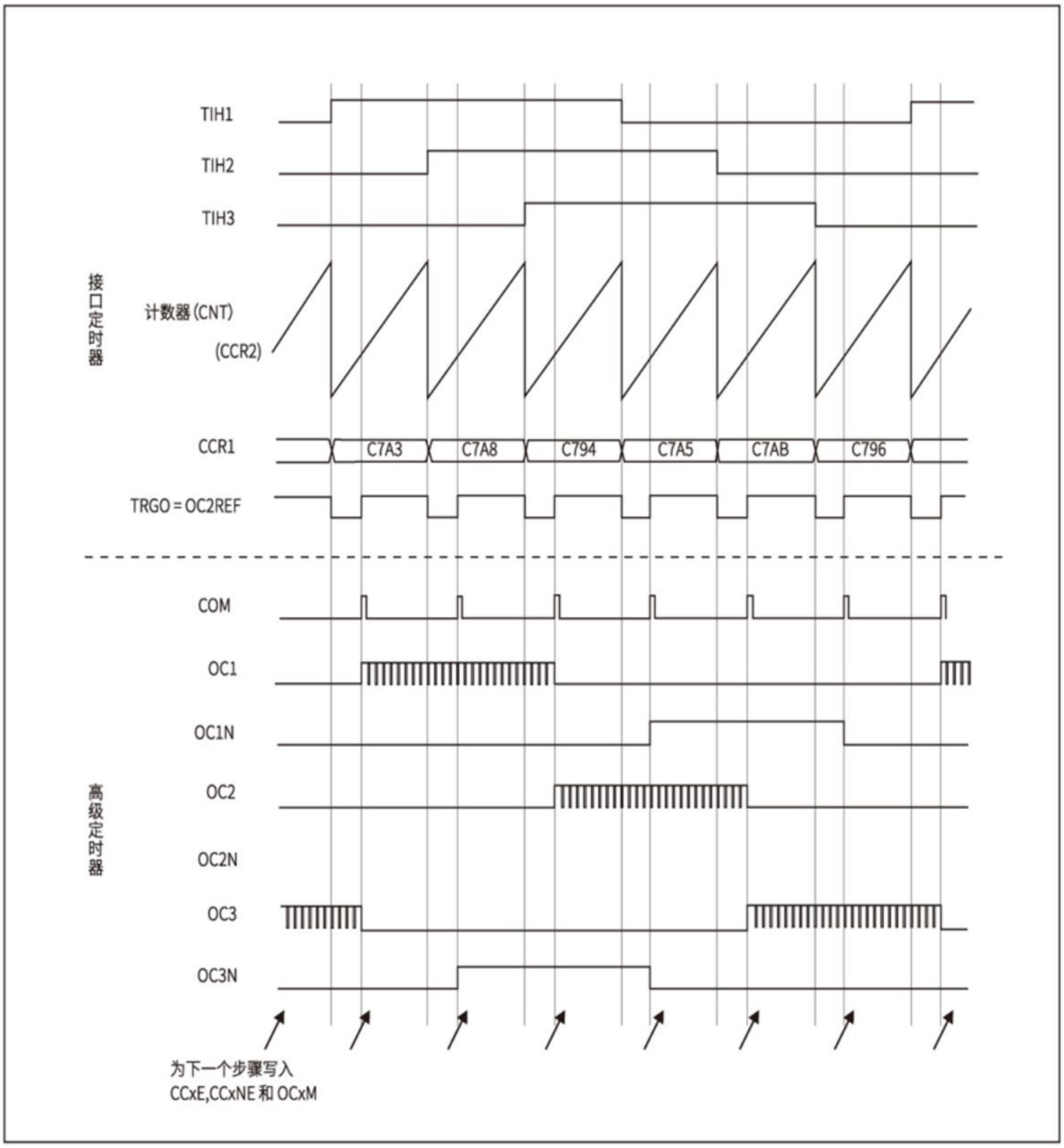


Figure 12-44 Example of Hall sensor interface

## 12.3.19 Synchronization of TIM1 timer and external trigger

The TIM1 timer can be synchronized to an external trigger in several modes: reset mode, gated mode and trigger mode.

### 12.3.19.1 Slave mode: Reset mode

When a trigger input event occurs, the counter and its prescaler can be re-initialized; at the same time, if the TIM1_CR1

The URS bit of the register is low, and an update event UEV is generated; then all preload registers (TIM1_ARR, TIM1_CCRx) have been updated.

In the following example, a rising edge at the TI1 input causes the up-counter to be cleared:

ÿ Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is needed, so Keep IC1F=0000). The capture prescaler is not used in trigger operation, so no configuration is required. The CC1S bit only selects the input capture Get the source, that is, CC1S=01 in the TIM1_CCMR1 register. Set CC1P=0 in the TIM1_CCER register to determine the polarity (only detect rising edge).

ÿSet SMS=100 in the TIM1_SMCR register, configure the timer to reset mode; set in the TIM1_SMCR register TS=101, select TI1 as the input source.

ÿ Set CEN=1 in the TIM1_CR1 register to start the counter.

The counter starts to count according to the internal clock, and then runs normally until a rising edge occurs on TI1; at this time, the counter is cleared and then starts from 0 restarts counting. At the same time, the trigger flag (TIF bit in the TIM1_SR register) is set, according to the The setting of the TIE (interrupt enable) bit generates an interrupt request.

The figure below shows the action when the auto-reload register TIM1_ARR=0x36. Between the rising edge of TI1 and the actual reset of the counter The delay depends on the resynchronization circuit at the TI1 input.

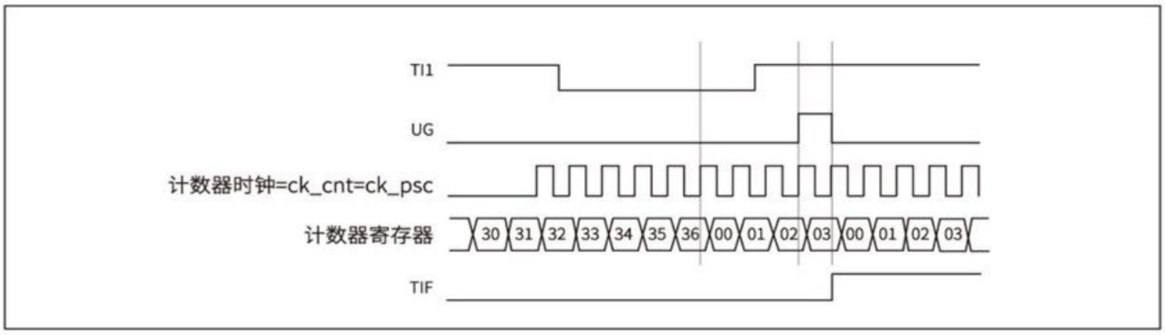


Figure 12-45 Control circuit in reset mode

12.3.19.2 Slave Mode: Gated Mode

Enables the counter according to the selected input level.

In the following example, the counter counts up only when TI1 is low:

ÿ Configure channel 1 to detect low level on TI1. Configure the input filter bandwidth (in this example, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, set CC1S=01 in the TIM1_CCMR1 register. Set CC1P=1 in the TIM1_CCER register to determine the polarity (only detect low level).

ÿSet SMS=101 in the TIM1_SMCR register, configure the timer as gated mode; set in the TIM1_SMCR register TS=101, select TI1 as the input source.

ÿ Set CEN=1 in the TIM1_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot be started active regardless of the trigger input level.

As long as TI1 is low, the counter starts counting according to the internal clock, and stops counting once TI1 becomes high. when the counter starts or stops Both set the TIF flag in TIM1_SR.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the TI1 input.

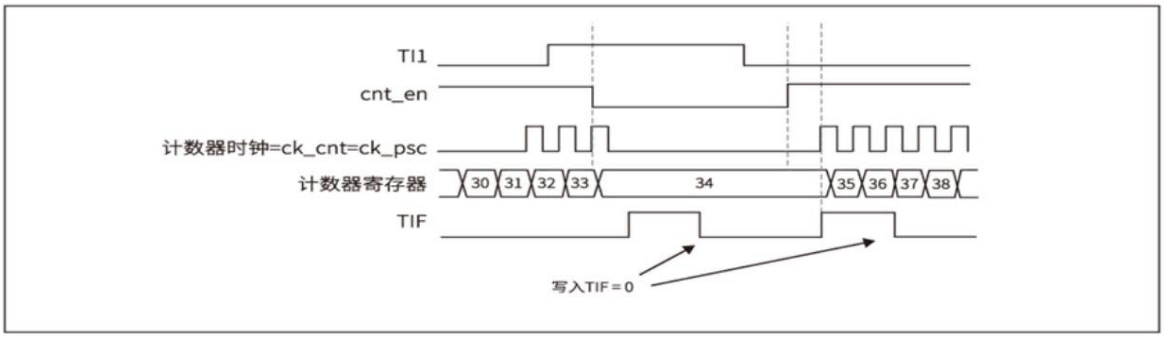


Figure 12-46 Control circuit in gated mode

### 12.3.19.3 Slave Mode: Trigger Mode

A selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

ÿ Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keep IC2F=0000). The capture prescaler is not used in trigger operation and does not require configuration. The CC2S bit is only used to select the input capture source, set CC2S=01 in the TIM1_CCMR1 register. Set CC2P=1 in the TIM1_CCER register to determine the polarity (only detect low level).

ÿSet SMS=110 in the TIM1_SMCR register, configure the timer as trigger mode; set in the TIM1_SMCR register TS=110, select TI2 as the input source.

When a rising edge occurs on TI2, the counter starts counting driven by the internal clock and sets the TIF flag at the same time.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

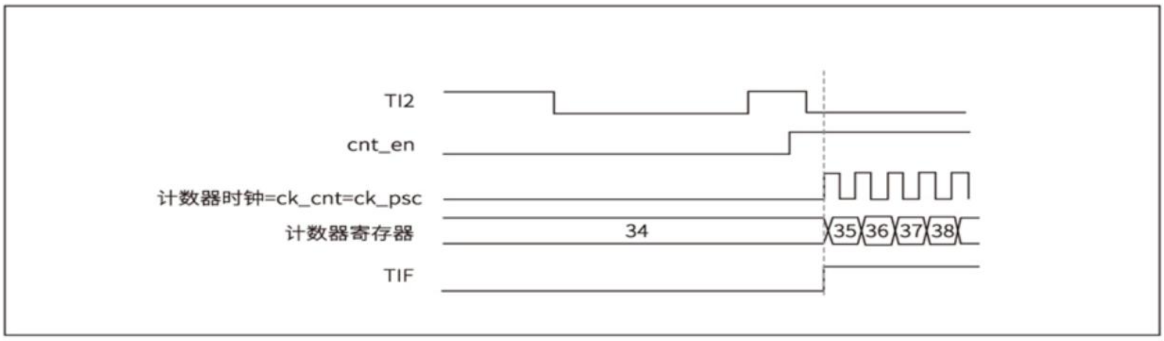


Figure 12-47 Control circuit in trigger mode

12.3.19.4 Slave Mode: External Clock Mode 2 + Trigger Mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). At this time, the ETR signal is Used as an input for an external clock, another input can be selected as a trigger input in reset mode, gate control mode or trigger mode. do not build It is recommended to use the TS bit of the TIM1_SMCR register to select ETR as TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up on every rising edge of ETR:

1. Configure the external trigger input circuit through the TIM1_SMCR register:

   - ETF=0000: no filtering

   - ETPS=00: no prescaler

   - ETP=0: detect rising edge of ETR, set ECE=1 to enable external clock mode 2.

2. Configure channel 1 as follows to detect the rising edge of TI:

   - IC1F=0000: no filtering

   - Capture prescaler is not used in trigger operation, no configuration required

   - Set CC1S=01 in the TIM1_CCMR1 register to select the input capture source

   -Set CC1P=0 in TIM1_CCER register to determine polarity (only detect rising edge)

3. Set SMS=110 in the TIM1_SMCR register to configure the timer as trigger mode. Set in TIM1_SMCR register

   TS=101, select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR. Rising edge of ETR signal The delay between the actual reset of the counter depends on the resynchronization circuit at the ETRP input.

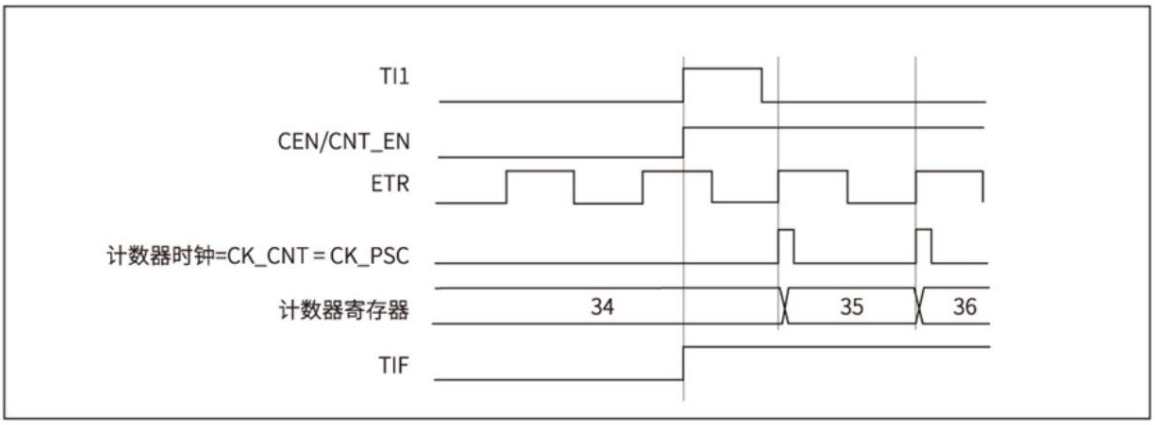


Figure 12-48 Control circuit in external clock mode 2 + trigger mode

## 12.3.20 Timer Synchronization

All TIM timers are connected internally for timer synchronization or chaining. See next chapter 13.3.15 Timer Synchronization for details

## 12.3.21 Debug mode

When the microcontroller enters debug mode (Cortex-M0+ core stops), according to the setting of DBG_TIM1_STOP in the DBG module, The TIM1 counter can either continue normal operation or stop.

## 12.4 TIM1 register list

These peripheral registers can be operated in word (32 bits), TIM1 base address 0x4000 1000.

Table 12-2 TIM1 register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | TIM1_CR1 | TIM1 Control Register 1 | 0x0000 0000 |
| 0x04 | TIM1_CR2 | TIM1 Control Register 2 | 0x0000 0000 |
| 0x08 | TIM1_SMCR | TIM1 slave mode control register | 0x0000 0000 |
| 0x0C | TIM1_DIER | TIM1 Interrupt Enable Register | 0x0000 0000 |
| 0x10 | TIM1_SR | TIM1 Status Register | 0x0000 0000 |
| 0x14 | TIM1_EGR | TIM1 event generation register | 0x0000 0000 |
| 0x18 | TIM1_CCMR1 | TIM1 capture/compare mode register 1 | 0x0000 0000 |
| 0x1C | TIM1_CCMR2 | TIM1 capture/compare mode register 2 | 0x0000 0000 |
| 0x20 | TIM1_CCER | TIM1 capture/compare enable register | 0x0000 0000 |
| 0x24 | TIM1_CNT | TIM1 counter | 0x0000 0000 |
| 0x28 | TIM1_PSC | TIM1 prescaler | 0x0000 0000 |
| 0x2C | TIM1_ARR | TIM1 auto-reload register | 0x0000 0000 |
| 0x30 | TIM1_RCR | TIM1 Repeat Count Register | 0x0000 0000 |
| 0x34 | TIM1_CCR1 | TIM1 capture/compare register 1 | 0x0000 0000 |
| 0x38 | TIM1_CCR2 | TIM1 capture/compare register 2 | 0x0000 0000 |
| 0x3C | TIM1_CCR3 | TIM1 capture/compare register 3 | 0x0000 0000 |
| 0x40 | TIM1_CCR4 | TIM1 capture/compare register 4 | 0x0000 0000 |
| 0x44 | TIM1_BDTR | TIM1 brake and dead zone register | 0x0000 0000 |

## 12.5 TIM1 register description

### 12.5.1 TIM1 Control Register 1 (TIM1_CR1)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | CKD[1:0] | | ARP<br>E. | CMS[1:0] | | DIR | OPM | URS | UDIS | CEN |
| | | | | | | R/W | | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:10 | - | Reserved, always reads as 0. | 0 | - |
| 9:8 | CKD[1:0] | Clock division factor (Clock division)<br>These 2 bits define the timer clock (CK_INT) frequency, dead time and<br>Divide ratio between sampling clocks used by filters (ETR, TIx).<br>00: tDTS = tCK_INT<br>01: tDTS = 2 x tCK_INT<br>10: tDTS = 4 x tCK_INT<br>11: Reserved, do not use this configuration | 0x0 | R/W |
| 7 | ARPE | Auto-reload preload enable bit (Auto-reload preload enable)<br>0: TIM1_ARR register is not buffered;<br>1: TIM1_ARR register is loaded into buffer. | 0 | R/W |
| 6:5 | CMS[1:0] | Select Center-aligned mode selection<br>00: Edge-aligned mode. The counter counts up or down according to the direction bit (DIR).<br>01: center alignment mode 1. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM1_CCMRx register) output comparison interrupt flag bit, only when the counter sends Set when counting down.<br>10: Center alignment mode 2. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM1_CCMRx register) output comparison interrupt flag bit, only when the counter sends Set when counting up.<br>11: Center alignment mode 3. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM1_CCMRx register) output comparison interrupt flag bit, when the counter is up Both are set when counting down.<br>Note: Transition from edge-aligned mode to center-aligned mode is not allowed while the counter is on (CEN=1). Mode. | 0x0 | R/W |
| 4 | DIR | Direction<br>0: The counter counts up;<br>1: The counter counts down.<br>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode. | 0 | R/W |
| 3 | OPM | Single pulse mode (One pulse mode)<br>0: the counter does not stop when an update event occurs;<br>1: The counter stops at the next update event (clearing the CEN bit). | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 2 | URS | Update request source<br>Software selects the source of UEV events through this bit<br>0: If set to 0, an update interrupt is generated by any of the following events:<br>– Counter overflow/underflow<br>– set the UG bit<br>– Updates generated from the mode controller<br>1: If set, an update interrupt will only be generated on counter overflow/underflow. | 0 | R/W |
| 1 | UDIS | Update disable<br>The software enables/disables the generation of UEV events through this bit<br>0: Allow UEVs. Update (UEV) events are generated by any of the following events:<br>– Counter overflow/underflow<br>– set the UG bit<br>– Updates generated from the mode controller<br>Registers with cache are loaded with their preloaded values. (Annotation: Update shadow registers)<br>1: Disable UEVs. Update events are not generated, shadow registers (ARR, PSC, CCRx) keep their value. If the UG bit is set or a hardware reset is issued from the mode controller, the count register and prescaler are reinitialized. | 0 | R/W |
| 0 | CEN | Counter enable<br>0: disable counter;<br>1: Enable counter.<br>Note: The external clock, gated mode and encoder mode will work only after the CEN bit is set by software.<br>Trigger mode can automatically set the CEN bit by hardware. | 0 | R/W |

## 12.5.2 TIM1 Control Register 2 (TIM1_CR2)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | OIS4 | OIS3N | OIS3 | OIS2N | OIS2 | OIS1N | OIS1 | TI1S | MMS[1:0] | | | reserve | CCUS | reserve | CCPC |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | | | R/W | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:15-14 | | Reserved, always reads | 0x0 | - |
| | OIS4 | as 0. Output idle state 4 (OC4 output). See OIS1 bit. | 0 | R/W |
| 13 | OIS3N | Output idle state 3 (OC3N output). See OIS1N bit. | 0 | R/W |
| 12 | OIS3 | Output idle state 3 (OC3 output). See OIS1 bit. | 0 | R/W |
| 11 | OIS2N | Output idle state 2 (OC2N output). See OIS1N bit. | 0 | R/W |
| 10 | OIS2 | Output idle state 2 (OC2 output). See OIS1 bit. Output | 0 | R/W |
| 9 | OIS1N | idle state 1 (OC1N output) (Output Idle state 1) 0:<br>When MOE=0, OC1N=0 after dead zone;<br>1: When MOE=0, OC1N=1 after dead time.<br>Note: After LOCK (TIM1_BKR register) level 1, 2 or 3 has been set, this bit cannot modified. | 0 | R/W |
| 8 | OIS1 | Output idle state 1 (OC1 output) (Output Idle state 1)<br>0: When MOE=0, if OC1N is realized, OC1=0 after dead zone;<br>1: When MOE=0, if OC1N is realized, then OC1=1 after dead time.<br>Note: After LOCK (TIM1_BKR register) level 1, 2 or 3 has been set, this bit cannot modified. | 0 | R/W |
| 7 | TI1S | TI1 selection (TI1 selection) 0:<br>The TIM1_CH1 pin is connected to the TI1 input;<br>1: TIM1_CH1, TIM1_CH2 and TIM1_CH3 pins are XORed to TI1 input. | 0 | R/W |
| 6:4 | MMS[2:0] | Master mode selection<br>These 3 bits are used to select the synchronization message (TRGO) sent to the slave timer in master mode. possible group The combination is as follows: 000: Reset – The UG bit of the TIM1_EGR register is used as a trigger output (TRGO). If the reset is generated by a trigger input (slave mode controller is in reset mode), then The signal on TRGO has a delay relative to the actual reset.<br>001: Enable – The counter enable signal CNT_EN is used as trigger output (TRGO). sometimes It is necessary to start multiple timers at the same time or control enabling slave timers within a period of time. count The device enable signal is the logical OR product of the CEN control bit and the trigger input signal in gated mode. born. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO unless Master/Slave mode selected (see description of MSM bit in TIM1_SMCR register). 010: more New – Update event selected as trigger input (TRGO). For example, a master timer clock can be Used as a prescaler for the slave timer.<br>011: Compare Pulse – When a capture occurs or a compare succeeds, when the CC1IF flag is to be set When (even if it is already high), the trigger output sends a positive pulse (TRGO).<br>100: Compare – OC1REF signal is used as trigger output (TRGO).<br>101: Compare – OC2REF signal is used as trigger output (TRGO).<br>110: Compare - OC3REF signal is used as trigger output (TRGO).<br>111: Compare - OC4REF signal is used as trigger output (TRGO). | 0x0 | R/W |

| | | | | |
|---|---|---|---|---|
| 3 | - | Reserved, always reads as 0. | 0 | - |
| 2 | CCUS | Capture/compare control update selection<br>0: If the capture/compare control bit is preloaded (CCPC=1), it can only be updated by setting the COM bit they;<br>1: If the capture/compare control bit is preloaded (CCPC=1), it can be set by setting the COM bit or A rising edge on TRGI updates them.<br>Note: This bit only works on channels with complementary outputs. | 0 | R/W |
| 1 | - | Reserved, always reads as 0. | 0 | - |
| 0 | CCPC | Capture/compare preloaded control bit (Capture/compare preloaded control) 0: CCxE, CCxNE and OCxM bits are not preloaded;<br>1: CCxE, CCxNE and OCxM bits are preloaded; when this bit is set, they is updated after setting the COM bit.<br>Note: This bit only works on channels with complementary outputs. | 0 | R/W |

### 12.5.3 TIM1 Slave Mode Control Register (TIM1_SMCR)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETP | ECE | ETPS[1:0] | | ETF[3:0] | | | | MSM | TS[2:0] | | | reserve | SMS[2:0] | | |
| R/W | R/W | R/W | | R/W | | | | R/W | R/W | | | | R/W | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15 | ETP | External trigger polarity (External trigger polarity) This bit selects whether to use ETR or ETR inversion to act as a trigger operation<br>0: ETR is not inverted, high level or rising edge is valid;<br>1: ETR is inverted, active low or falling edge. | 0 | R/W |
| 14 | ECE | External clock enable bit (External clock enable)<br>This bit enables external clock mode 2<br>0: disable external clock mode 2;<br>1: Enable External Clock Mode 2. The counter is driven by any active edge on the ETRF signal.<br>Note 1: Setting the ECE bit is the same as selecting the external clock mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111) have the same effect.<br>Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gated mode and trigger mode; however, TRGI cannot be connected to ETRF at this time (TS bit cannot be '111').<br>Note 3: When external clock mode 1 and external clock mode 2 are enabled at the same time, the output of the external clock The entry is ETRF. | 0 | R/W |
| 13:12 | ETPS[1:0] | External trigger prescaler<br>The frequency of the external trigger signal ETRP must be at most 1/4 of the frequency of TIM1CLK. When the input compares When using a fast external clock, you can use prescaler to reduce the frequency of ETRP.<br>00: close the prescaler;<br>01: ETRP frequency divided by 2;<br>10: ETRP frequency divided by 4;<br>11: ETRP frequency divided by 8. | 0 | R/W |
| 11:8 | ETF[3:0] | External trigger filter<br>These bits define the frequency at which the ETRP signal is sampled and the bandwidth of the ETRP digital filter. Reality In effect, the digital filter is an event counter that registers N events and generates a output transitions.<br>0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6<br>0001: sampling frequency fSAMPLING=fCK_INT, N=2 1001: sampling frequency fSAMPLING=fDTS/8, N=8<br>0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5<br>0011: sampling frequency fSAMPLING=fCK_INT, N=8 1011: sampling frequency fSAMPLING=fDTS/16, N=6<br>0100: sampling frequency fSAMPLING=fDTS/2, N=6 1100: sampling frequency fSAMPLING= fDTS /16, N=8<br>0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING= fDTS /32, N=5<br>0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING= fDTS /32, N=6<br>0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING= fDTS /32, N=8 | 0 | R/W |
| 7 | MSM | Master/slave mode (Master/slave mode)<br>0: no effect;<br>1: Events on the trigger input (TRGI) are delayed to allow TRGO) with perfect synchronization between its slave timers. This pair requires synchronizing several timers into one Useful when a single external event is present. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 6:4 | TS[2:0] | Trigger selection<br>These 3 bits select the trigger input for the synchronous counter.<br>000: Internal Trigger 0 (ITR0) 100: TI1 Edge Detector (TI1F_ED)<br>001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1)<br>010: Internal Trigger 2 (ITR2) 110: Filtered Timer Input 2 (TI2FP2)<br>011: Internal trigger 3 (ITR3) 111: External trigger input (ETRF)<br>See Table 12-3 for more details on ITRx.<br>Note: These bits can only be changed when they are not used (such as SMS=000) to avoid False edge detection. | 0 | R/W |
| 3 | - | Reserved, always reads as 0. | 0 | - |
| 2:0 | SMS[2:0] | Slave mode selection<br>When the external signal is selected, the active edge of the trigger signal (TRGI) and the selected external input polarity related (see description of Input Control Register and Control Register)<br>000: Disable Slave Mode – If CEN=1, the prescaler is directly driven by the internal clock.<br>001: Encoder Mode 1 – Counter clocks out on edge of TI2FP2 according to the level of TI1FP1 Up/down counting.<br>010: Encoder Mode 2 – The counter is clocked on the edge of TI1FP1 according to the level of TI2FP2. Up/down counting.<br>011: Encoder Mode 3 - Depending on the input level of another signal, the counter is on TI1FP1 and TI2FP2 edge count up/down.<br>100: Reset Mode – The rising edge of the selected trigger input (TRGI) re-initializes the counter and And generate a signal to update the register.<br>101: Gated Mode – When the trigger input (TRGI) is high, the counter's clock is on. Once touched If the send input goes low, the counter is stopped (but not reset). The start and stop of the counter are controlled of.<br>110: Trigger mode – the counter is started (but not reset) on a rising edge at the trigger input TRGI, only Startup with counters is controlled.<br>111: External Clock Mode 1 - The rising edge of the selected trigger input (TRGI) drives the counter.<br>Note: Do not use gated mode if TI1F_EN is selected as trigger input (TS=100). this The reason is that TI1F_ED outputs a pulse every time TI1F changes, but the gated mode is to check the level of the trigger input. | 0 | R/W |

Table 12-3 TIM1 internal trigger connection

| from the timer | ITR0(TS=000) | ITR1(TS=001) | ITR2(TS=010) | ITR3(TS=011) |
|---|---|---|---|---|
| TIM1 | Tim2_trgo | irq_timer10 | irq_timer11 | irq_pca |

### 12.5.4 TIM1 Interrupt Enable Register (TIM1_DIER)

Offset address: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | BIE | TIE | COMI E. | CC4I E. | CC3I E. | CC2I E. | CC1I E. | UIE |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit | Mark | function description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | Reserved, always reads as 0. | 0 | - |
| 7 | BIE | Allow brake interrupt (Break interrupt enable)<br>0: Disable brake interruption;<br>1: Enable brake interruption. | 0 | R/W |
| 6 | TIE | Trigger interrupt enable (Triggerinterrupt enable)<br>0: Disable trigger interrupt;<br>1: Enable trigger interrupt. | 0 | R/W |
| 5 | COMIE | Allow COM interrupt (COM interrupt enable)<br>0: disable COM interrupt;<br>1: COM interrupt enabled. | 0 | R/W |
| 4 | CC4IE | Enable capture/compare 4 interrupt (Capture/Compare 4 interrupt enable)<br>0: disable capture/compare 4 interrupt;<br>1: Enable capture/compare 4 interrupt. | 0 | R/W |
| 3 | CC3IE | Enable capture/compare 3 interrupt (Capture/Compare 3 interrupt enable)<br>0: disable capture/compare 3 interrupt;<br>1: Enable capture/compare 3 interrupt. | 0 | R/W |
| 2 | CC2IE | Enable capture/compare 2 interrupt (Capture/Compare 2 interrupt enable)<br>0: disable capture/compare 2 interrupt;<br>1: Capture/Compare 2 interrupt enabled. | 0 | R/W |
| 1 | CC1IE | Enable capture/compare 1 interrupt (Capture/Compare 1 interrupt enable)<br>0: disable capture/compare 1 interrupt;<br>1: Enable capture/compare 1 interrupt. | 0 | R/W |
| 0 | UIE | Update interrupt enable<br>0: disable update interruption;<br>1: Enable update interruption. | 0 | R/W |

### 12.5.5 TIM1 Status Register (TIM1_SR)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | CC4OF | CC3OF | CC2OF | CC1OF | Save<br>Keep | BIF | TIF | COMIF | CC4IF | CC3IF | CC2IF | CC1IF | UIF |
| | | | RC W0 | RC W0 | RC W0 | RC W0 | | RC<br>W0 | RC<br>W0 | RC W0 | RC W0 RC | W0 RC W0 | | RC W0 | RC<br>W0 |

| Bit flag function description is reserved | | | Reset value read and write | |
|---|---|---|---|---|
| 31:13 - | | and always reads as 0. | 0 | - |
| 12 | CC4OF | capture/compare 4 overcapture flag (Capture/Compare 4 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 11 | CC3OF | Capture/Compare 3 overcapture flag (Capture/Compare 3 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 10 | CC2OF | capture/compare 2 overcapture flag (Capture/Compare 2 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 9 | CC1OF | Capture/Compare 1 overcapture flag<br>This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear the bit.<br>0: No duplicate capture occurs;<br>1: When the counter value is captured into the TIM1_CCR1 register, the state of CC1IF has been to '1'. | 0 | RC<br>W0 |
| 8 | - | Reserved, always reads | 0 | - |
| 7 | BIF | as 0. Brake interrupt flag (Break interrupt flag)<br>Once the brake input is valid, this bit is '1' by hardware. If the brake input is inactive, this bit Can be cleared to '0' by software.<br>0: No braking event occurs;<br>1: Active level detected on brake input. | 0 | RC<br>W0 |
| 6 | TIF | Trigger interrupt flag<br>When a trigger event occurs (when the slave mode controller is in a mode other than gated mode, in When a valid edge is detected at the TRGI input, or any edge in gated mode), the bit is reset by hardware. set to '1'. It is cleared to '0' by software.<br>0: No trigger event is generated;<br>1: Trigger an interrupt and wait for a response. | 0 | RC<br>W0 |
| 5 | COMIF | COM interrupt flag (COM interrupt flag)<br>Once a COM event is generated (when capture/compare control bits: CCxE, CCxNE, OCxM have been changed new) This bit is set to '1' by hardware. It is cleared to '0' by software.<br>0: No COM event is generated;<br>1: COM interrupt pending. | 0 | RC<br>W0 |
| 4 | CC4IF | Capture/Compare 4 interrupt flag (Capture/Compare 4 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 3 | CC3IF | Capture/Compare 3 interrupt flag (Capture/Compare 3 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 2 | CC2IF | Capture/Compare 2 interrupt flag (Capture/Compare 2 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 1 | CC1IF | Capture/Compare 1 interrupt flag<br>If channel CC1 is configured in output mode:<br>This bit is set by hardware when the counter value matches the compare value, except in centrosymmetric mode (see (see the CMS bit of the TIM1_CR1 register). It is cleared to '0' by software.<br>0: no match occurs;<br>1: The value of TIM1_CNT matches the value of TIM1_CCR1. | 0 | RC<br>W0 |

| | | | | |
|---|---|---|---|---|
| | | When the content of TIM1_CCR1 is greater than the content of TIM1_APR, the counter overflows in the up or up/down counting mode, or the counter underflow condition in the down counting mode, the CC1IF bit goes high If the channel CC1 is configured as input mode: When the capture event occurs when this bit is set to '1' by hardware, it is cleared to '0' by software or by reading TIM1_CCR1 is cleared to '0'. 0: No input capture occurs; 1: Counter value has been captured (copied) to TIM1_CCR1 (an edge detected on IC1 with the same polarity | | |
| 0 | UIF | as selected). Update interrupt flag This bit is set to '1' by hardware when an update event occurs. It is cleared to '0' by software. 0: No update event generated; 1: Update interrupt waiting for response. This bit is set to '1' by hardware when the register is updated: - If UDIS=0 of the TIM1_CR1 register, when the repeat counter value overflows or underflows (an update event occurs when the repeat counter=0). - If URS=0 and UDIS=0 of the TIM1_CR1 register, an update event is generated when UG=1 of the TIM1_EGR register is set, and when the counter CNT is re-initialized by software. - If URS=0, UDIS=0 of the TIM1_CR1 register, when the counter CNT is reinitialized by the trigger event. (Refer to 12.5.3 TIM1 Slave Mode Control Register (TIM1_SMCR)). | 0 | RC W0 |

Machine Translated by Google

### 12.5.6 TIM1 Event Generation Register (TIM1_EGR)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | BG | TG | COMG | CC4G | CC3G | CC2G | CC1G | UG |
| | | | | | | | | WO WO | WO WO WO | WO WO WO | | | | | WO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | Reserved, always reads as 0. | 0x0 | - |
| 7 | BG | Generate a brake event (Break generation)<br>This bit is set to '1' by software to generate a brake event, and is automatically cleared to '0' by hardware.<br>0: no action;<br>1: Generate a brake event. At this time, MOE=0, BIF=1, if the corresponding interrupt is enabled, the corresponding interruption. | 0 | WO |
| 6 | TG | Generate a trigger event (Trigger generation)<br>This bit is set to '1' by software to generate a trigger event, and is automatically cleared to '0' by hardware.<br>0: no action;<br>1: TIF=1 in the TIM1_SR register, if the corresponding interrupt is enabled, the corresponding interrupt will be generated. | 0 | WO |
| 5 | COMG | Capture/Compare control update generation (Capture/Comparecontrol update generation)<br>This bit is set to '1' by software and automatically cleared to '0' by hardware.<br>0: no action;<br>1: When CCPC=1, allow to update CCxE, CCxNE, OCxM bits.<br>Note: This bit is only valid for channels with complementary outputs. | 0 | WO |
| 4 | CC4G | generates capture/compare 4 events (Capture/Compare 4 generation)<br>Refer to the CC1G description. | 0 | WO |
| 3 | CC3G | generates capture/compare 3 events (Capture/Compare 3 generation)<br>Refer to the CC1G description. | 0 | WO |
| 2 | CC2G | generates capture/compare 2 events (Capture/Compare 2 generation)<br>Refer to the CC1G description. | 0 | WO |
| 1 | CC1G | Generate capture/compare 1 event (Capture/Compare 1 generation)<br>This bit is set to '1' by software to generate a capture/compare event and automatically cleared to '0' by hardware.<br>0: no action;<br>1: Generate a capture/compare event on channel CC1:<br>If channel CC1 is configured as an output:<br>Set CC1IF=1, if the corresponding interrupt is enabled, the corresponding interrupt will be generated.<br>If channel CC1 is configured as an input:<br>The current counter value is captured to the TIM1_CCR1 register; set CC1IF=1, if enabled corresponding interrupt, a corresponding interrupt is generated. If CC1IF is already 1, set CC1OF=1. | 0 | WO |
| 0 | UG | Generate update events (Update generation)<br>This bit is set to '1' by software and automatically cleared to '0' by hardware.<br>0: no action;<br>1: Reinitializes the counter and generates an update event. Note that the prescaler counter is also Clear '0' (but the prescaler coefficient remains unchanged). Count if in centrosymmetric mode or DIR=0 (count up) The counter is cleared to '0'; if DIR=1 (count down), the counter takes the value of TIM1_ARR. | 0 | WO |

### 12.5.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)

Offset address: 0x18

Reset value: 0x0000 0000

A channel can be used as input (capture mode) or output (compare mode), the direction of the channel is defined by the corresponding CCxS bit. The other bits of this register are The effect is different in input and output mode. OCxx describes the function of the channel in output mode, ICxx describes the function of the channel in input mode function below. It must therefore be noted that the same bit functions differently in output mode than in input mode.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC2CE | OC2M[2:0] | | | OC2PE | OC2FE | CC2S[1:0] | | OC1CE | OC1M[2:0] | | | OC1PE | OC1FE | CC1S[1:0] | |
| IC2F[3:0] | | | | IC2PSC[1:0] | | | | IC1F[3:0] | | | | IC1PSC[1:0] | | | |
| R/W | | | | | | | | | | | | | | | |

Output compare mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0x0 | - |
| 15 | OC2CE | Output Compare 2 clear enable (Output Compare 2 clear enable) | 0 | R/W |
| 14:12 | OC2M[2:0] | Output Compare 2 mode (Output Compare 2 mode) | 0x0 | R/W |
| 11 | OC2PE | Output Compare 2 preload enable (Output Compare 2 preload enable) | 0 | R/W |
| 10 | OC2FE | Output Compare 2 fast enable (Output Compare 2 fast enable) | 0 | R/W |
| 9:8 | CC2S[1:0] | Capture/Compare 2 selections. (Capture/Compare 2 selection)<br>This bit defines the direction of the channel (input/output), and the selection of the input pin:<br>00: CC2 channel is configured as output;<br>01: CC2 channel is configured as input, IC2 is mapped on TI2;<br>10: CC2 channel is configured as input, IC2 is mapped on TI1;<br>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode only works internally When the trigger input is selected (selected by the TS bit of the TIM1_SMCR register).<br>Note: CC2S is writable only when the channel is closed (CC2E=0 in TIM1_CCER register). | 0x0 | R/W |
| 7 | OC1CE | Output Compare 1 clear '0' enable (Output Compare 1 clear enable)<br>0: OC1REF is not affected by ETRF input;<br>1: Once ETRF input high level is detected, clear OC1REF=0. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 6:4 | OC1M[2:0] | Output Compare 1 mode (Output Compare 1 mode) The 3 bits define the action of the output reference signal OC1REF, and OC1REF determines the OC1, The value of OC1N. OC1REF is active high, while the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits.<br>000: Freeze. The comparison between the output compare register TIM1_CCR1 and the counter TIM1_CNT has no effect on OC1REF; 001: set channel 1 to active level when matched. When the value of the counter TIM1_CNT is the same as the capture/compare register 1 (TIM1_CCR1), force OC1REF high. 010: Set channel 1 to invalid level when matching. When the value of the counter TIM1_CNT is the same as the capture/compare register 1 (TIM1_CCR1), force OC1REF low. 011: Flip. When TIM1_CCR1=TIM1_CNT, flip the level of OC1REF. 100: Forced to inactive level. Force OC1REF low. 101: Forced to active level. Force OC1REF high.<br>110: PWM mode 1 - when counting up, once TIM1_CNT<TIM1_CCR1 channel 1 is active level, otherwise it is inactive level; when counting down, once When TIM1_CNT>TIM1_CCR1, channel 1 is inactive level (OC1REF=0), otherwise it is active level (OC1REF=1).<br>111: PWM mode 2 - when counting up, once TIM1_CNT<TIM1_CCR1 channel 1 is inactive level, otherwise it is active level; when counting down, once When TIM1_CNT>TIM1_CCR1, channel 1 is active level, otherwise it is inactive level.<br>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIM1_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.<br>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the compare result changes or when switching from freeze mode to PWM mode in output compare mode. | 0x0 | R/W |
| 3 | OC1PE | Output Compare 1 preload enable (Output Compare 1 preload enable) 0: Disable the preload function of the TIM1_CCR1 register, the TIM1_CCR1 register can be written at any time, and the newly written value takes effect immediately. 1: Enable the preload function of the TIM1_CCR1 register, read and write operations only operate on the preload register, and the preload value of TIM1_CCR1 is loaded into the current register when the update event arrives. Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIM1_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified. Note 2: Only in single pulse mode (OPM=1 of TIM1_CR1 register), PWM mode can be used without confirming the preload | 0 | R/W |
| 2 | OC1FE | register, otherwise its action is undefined. Output Compare 1 fast enable This bit is used to speed up the CC output response to trigger input events. 0: According to the value of the counter and CCR1, CC1 operates normally even if the flip-flop is on. When the flip-flop input has an active edge, the minimum delay to activate the CC1 output is 5 clock cycles. 1: The active edge input to the flip-flop acts as a compare match. Therefore, OC is set as the comparison level regardless of the comparison result. The delay between the active edge of the sample flip-flop and the output of CC1 is shortened to 3 clock cycles.<br>OCFE works only when the channel is configured as PWM1 or PWM2 mode. | 0 | R/W |
| 1:0 | CC1S[1:0] | Capture/Compare 1 selection. (Capture/Compare 1 selection) These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIM1_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in TIM1_CCER register). | 0x0 | R/W |

Machine Translated by Google

Enter capture mode:

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:12 IC2F[3:0] Input capture 2 filter (Input capture 2 filter) | | | 0x0 | R/W |
| 11:10 IC2PSC[1:0] Input/capture 2 prescaler (Input capture 2 prescaler) | | | 0x0 | R/W |
| 9:8 | CC2S[1:0] | Capture/Compare 2 selection (Capture/Compare 2 selection) These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC2 channel is configured as output; 01: CC2 channel is configured as input, IC2 Mapping on TI2; 10: CC2 channel is configured as input, IC2 is mapped on TI1; 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIM1_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the | 0x0 | R/W |
| 7:4 | IC1F[3:0] | TIM1_CCER register). Input capture 1 filter (Input capture 1 filter) These bits define the sampling frequency and digital filter length of TI1 input. The digital filter consists of an event counter, which generates an output jump after recording N events: 0000: No filter, sampling at fDTS 1000: Sampling frequency fSAMPLING=fDTS/8, N=6 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: sampling frequency fSAMPLING=fDTS/8, N=8<br>0010: sampling frequency fSAMPLING=fCK_INT, N=4 1010: sampling frequency fSAMPLING=fDTS/16, N=5<br>0011: sampling frequency fSAMPLING=fCK_INT, N=8 1011: sampling frequency fSAMPLING=fDTS/16, N=6<br>0100: sampling frequency fSAMPLING=fDTS/2, N=6 1100: sampling frequency fSAMPLING=fDTS/16, N=8<br>0101: sampling frequency fSAMPLING= fDTS /2, N=8 1101: sampling frequency fSAMPLING=fDTS/32, N=5<br>0110: sampling frequency fSAMPLING= fDTS /4, N=6 1110: sampling frequency fSAMPLING=fDTS/ 32, N=6<br>0111: sampling frequency fSAMPLING= fDTS /4, N=8 1111: sampling frequency fSAMPLING=fDTS/32, N=8 | 0x0 | R/W |
| 3:2 | IC1PSC[1:0] | Input/capture 1 prescaler (Input capture 1 prescaler) These 2 bits define the prescaler coefficient of CC1 input (IC1). Once CC1E=0 (in TIM1_CCER register), the prescaler is reset. 00: No prescaler, each edge detected on the capture input triggers a capture; 01: Triggers a capture every 2 events; 10: Triggers a capture every 4 events; 11: Triggers a capture every 8 events capture. Capture/Compare 1 | 0x0 | R/W |
| 1:0 | CC1S[1:0] | Selection (Capture/Compare 1 Selection) These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 Mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIM1_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIM1_CCER register). | 0x0 | R/W |

### 12.5.8 TIM1 capture/compare mode register 2 (TIM1_CCMR2)

Offset address: 0x1C

Reset value: 0x0000 0000

See the description of the CCMR1 register above.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OC4CE | OC4M[2:0] | | | OC4PE | OC4FE | CC4S[1:0] | | OC3CE | OC3M[2:0] | | | OC3PE | OC3FE | CC3S[1:0] | |
| IC4F[3:0] | | | | IC4PSC[1:0] | | | | IC3F[3:0] | | | | IC3PSC[1:0] | | | |
| R/W | | | | | | | | | | | | | | | |

Output compare mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0x0 | - |
| 15 | OC4CE | Output Compare 4 clear enable (Output Compare 4 clear enable) | 0 | R/W |
| 14:12 | OC4M[2:0] | Output Compare 4 mode (Output Compare 4 mode) | 0x0 | R/W |
| 11 | OC4PE | Output Compare 4 preload enable (Output Compare 4 preload enable) | 0 | R/W |
| 10 | OC4FE | Output Compare 4 fast enable (Output Compare 4 fast enable) | 0 | R/W |
| 9:8 | CC4S[1:0] | Capture/Compare 4 selections. (Capture/Compare 4 selection)<br>The 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC4 channel is configured as output;<br>01: CC4 channel is configured as input, IC4 is mapped on TI4;<br>10: CC4 channel is configured as input, IC4 is mapped on TI3;<br>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode only works within external trigger input is selected (selected by the TS bit of the TIM1_SMCR register).<br>Note: CC4S is writable only when the channel is closed (CC4E=0 of the TIM1_CCER register) of. | 0x0 | R/W |
| 7 | OC3CE | Output Compare 3 clear '0' enable (Output Compare 3 clear enable) | 0 | R/W |
| 6:4 | OC3M[2:0] | Output Compare 3 mode (Output Compare 3 mode) | 0x0 | R/W |
| 3 | OC3PE | Output Compare 3 preload enable (Output Compare 3 preload enable) | 0 | R/W |
| 2 | OC3FE | Output Compare 3 fast enable (Output Compare 3 fast enable) | 0 | R/W |
| 1:0 | CC3S[1:0] | Capture/Compare 3 selections. (Capture/Compare 3 selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC3 channel is configured as output;<br>01: CC3 channel is configured as input, IC3 is mapped on TI3;<br>10: CC3 channel is configured as input, IC3 is mapped on TI4;<br>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode only works within external trigger input is selected (selected by the TS bit of the TIM1_SMCR register).<br>Note: CC3S is writable only when the channel is closed (CC3E=0 of the TIM1_CCER register) of. | 0x0 | R/W |

Enter capture mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0x0 | - |
| 15:12 | IC4F[3:0] | Input capture 4 filter (Input capture 4 filter) | 0x0 | R/W |
| 11:10 | IC4PSC[1:0] | Input/capture 4 prescaler (Input capture 4 prescaler) | 0x0 | R/W |
| 9:8 | CC4S[1:0] | Capture/Compare 4 selection (Capture/Compare 4 selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC4 channel is configured as output;<br>01: CC4 channel is configured as input, IC4 is mapped on TI4;<br>10: CC4 channel is configured as input, IC4 is mapped on TI3;<br>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode only works in<br>When the internal trigger input is selected (selected by the TS bit of the TIM1_SMCR register).<br>Note: CC4S is only available when the channel is closed (CC4E=0 in the TIM1_CCER register) written. | 0x0 | R/W |
| 7:4 | IC3F[3:0] | Input capture 3 filter (Input capture 3 filter) | 0x0 | R/W |
| 3:2 | IC3PSC[1:0] | Input/capture 3 prescaler (Input capture 3 prescaler) | 0x0 | R/W |
| 1:0 | CC3S[1:0] | Capture/Compare 3 Selection<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC3 channel is configured as output;<br>01: CC3 channel is configured as input, IC3 is mapped on TI3;<br>10: CC3 channel is configured as input, IC3 is mapped on TI4;<br>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode only works in<br>When the internal trigger input is selected (selected by the TS bit of the TIM1_SMCR register).<br>Note: CC3S is only available when the channel is closed (CC3E=0 in the TIM1_CCER register) written. | 0x0 | R/W |

### 12.5.9 TIM1 Capture/Compare Enable Register (TIM1_CCER)

Offset address: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | CC4P | CC4E | CC3 NP | CC3 NE | CC3P | CC3E | CC2 NP | CC2 NE | CC2P | CC2E | CC1 NP | CC1 NE | CC1P | CC1E |
| | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:14 | - | Reserved, always reads as 0. | 0 | - |
| 13 | CC4P | input/capture 4 output polarity (Capture/Compare 4 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 12 | CC4E | input/capture 4 output enable (Capture/Compare 4 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 11 | CC3NP | input/capture 3 complementary output polarity (Capture/Compare 3 complementary output polarity) refer to the description of CC1NP. | 0 | R/W |
| 10 | CC3NE | Input/Capture 3 Complementary Output Enable (Capture/Compare 3 complementary output enable)<br>Refer to the description of CC1NE. | 0 | R/W |
| 9 | CC3P | input/capture 3 output polarity (Capture/Compare 3 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 8 | CC3E | input/capture 3 output enable (Capture/Compare 3 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 7 | CC2NP | input/capture 2 complementary output polarity (Capture/Compare 2 complementary output polarity) refer to the description of CC1NP. | 0 | R/W |
| 6 | CC2NE | Input/Capture 2 Complementary Output Enable (Capture/Compare 2 complementary output enable)<br>Refer to the description of CC1NE. | 0 | R/W |
| 5 | CC2P | input/capture 2 output polarity (Capture/Compare 2 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 4 | CC2E | input/capture 2 output enable (Capture/Compare 2 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 3 | CC1NP | Input/Capture 1 Complementary Output Polarity (Capture/Compare 1 complementary output polarity)0: OC1N active high;<br>1: OC1N active low.<br>Note: Once the LOCK level (LOCK bit in TIM1_BDTR register) is set to 3 or 2 and<br>CC1S=00 (the channel is configured as output), then this bit cannot be modified. | 0 | R/W |
| 2 | CC1NE | : Input/capture 1 complementary output enable (Capture/Compare 1 complementary output enable)0: Off - OC1N disables the output, so the level of OC1N depends on MOE,<br>The value of the OSSI, OSSR, OIS1, OIS1N, and CC1E bits.<br>1: On - OC1N signal is output to the corresponding output pin, and its output level depends on MOE,<br>The value of the OSSI, OSSR, OIS1, OIS1N, and CC1E bits. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 1 | CC1P | Input/Capture 1 output polarity (Capture/Compare 1 output polarity)<br>CC1 channel configured as output:<br>0: OC1 active high;<br>1: OC1 active low.<br>CC1 channel configured as input:<br>This bit selects whether IC1 or the inverted signal of IC1 is used as trigger or capture signal.<br>0: No inversion: Capture occurs on the rising edge of IC1; when used as an external trigger, IC1 does not invert.<br>1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.<br>Note: Once the LOCK level (LOCK bit in the TIM1_BDTR register) is set to 3 or 2, the<br>Bits cannot be modified. | 0 | R/W |
| 0 | CC1E | Input/Capture 1 output enable (Capture/Compare 1 output enable)<br>CC1 channel configured as output:<br>0: Off - OC1 disables the output, so the output level of OC1 depends on MOE, OSSI,<br>The value of the OSSR, OIS1, OIS1N, and CC1NE bits.<br>1: On - OC1 signal is output to the corresponding output pin, and its output level depends on MOE,<br>The value of the OSSI, OSSR, OIS1, OIS1N, and CC1NE bits.<br>CC1 channel configured as input:<br>This bit determines whether the counter value can be captured into the TIM1_CCR1 register.<br>0: capture disabled;<br>1: Capture enabled. | 0 | R/W |

Table 12-4 Control bits for complementary output channels OCx and OCxN with break function

| control bit | | | | | output status | |
|---|---|---|---|---|---|---|
| MOE bit | OSSI bit | OSSR bit | CCxE bit | CCxNE bit | OCx output status output disabled (disconnect from | OCxN output state |
| 1 | x | 0 | 0 | 0 | timer)<br>OCx=0, OCx_EN=0 | Output disabled (disconnected from timer)<br>OCxN=0, OCxN_EN=0 |
| | | 0 | 0 | 1 | Output disabled (disconnected from timer)<br>OCx=0, OCx_EN=0 | OCxREF + Polarity,<br>OCxN= OCxREF xor<br>CCxNP, OCxN_EN=1 |
| | | 0 | 1 | 0 | OCxREF + Polarity,<br>OCx= OCxREF xor CCxP,<br>OCx_EN=1 | Output disabled (disconnected from timer)<br>OCxN=0, OCxN_EN=0 |
| | | 0 | 1 | 1 | OCxREF + Polarity + Deadband,<br>OCx_EN=1 | OCxREF inversion + polarity + dead zone,<br>OCxN_EN=1 |
| | | 1 | 0 | 0 | Output disabled (disconnected from timer)<br>OCx=CCxP, OCx_EN=0 | Output disabled (disconnected from timer)<br>OCxN=CCxNP,<br>OCxN_EN=0 |
| | | 1 | 0 | 1 | Off state ( output enabled and inactive<br>flat)<br>OCx=CCxP, OCx_EN=1 | OCxREF + Polarity,<br>OCxN= OCxREF xor<br>CCxNP, OCxN_EN=1 |
| | | 1 | 1 | 0 | OCxREF + Polarity,<br>OCx= OCxREF xor CCxP,<br>OCx_EN=1 | Off state (output enabled and inactive<br>electricity<br>Flat) OCxN=CCxNP,<br>OCxN_EN=1 |
| | | 1 | 1 | 1 | OCxREF + Polarity + Deadband,<br>OCx_EN=1 | OCxREF inversion + polarity + dead zone,<br>OCxN_EN=1 |
| 0 | 0 | x | 0 | | Output disabled (disconnected from timer) | |
| | 0 | | 0 | 0 1 | Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0; | |
| | 0 | | 1 | 0 | If clock exists: OCx=OISx, OCxN=OISxN after a dead time , false | |
| | 0 | | 1 | 1 | It is assumed that OISx and OISxN do not both correspond to active levels of OCx and OCxN . | |
| | 1 | | 0 | | Off state (output enabled and inactive level) | |
| | 1 | | 0 | 0 1 | Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1; | |
| | 1 | | 1 | 0 | If clock exists: OCx=OISx, OCxN=OISxN after a dead time , false | |
| | 1 | | 1 | 1 | It is assumed that OISx and OISxN do not both correspond to active levels of OCx and OCxN . | |

If both outputs of a channel are not used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP and CCxNP must be cleared.

Note: The state of external I/O pins whose pins are connected to complementary OCx and OCxN channels depends on the OCx and OCxN channel states and GPIO and AFIO registers.

## 12.5.10 TIM1 counter (TIM1_CNT)

Offset address: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CNT[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:0 | CNT[15:0] | Counter value (Counter value) | 0x0 | R/W |

## 12.5.11 TIM1 Prescaler (TIM1_PSC)

Offset address: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PSC[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit Flag | Functional Description | | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:0 | PSC[15:0] | Prescaler value (Prescaler value)<br>The clock frequency of the counter (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1).<br>PSC contains the value loaded into the current prescaler register each time an update event occurs; more<br>New events include the counter being cleared to '0' by the UG bit of TIM_EGR or being operated in reset mode<br>from<br>Cleared to '0' by the controller. | 0x0 | R/W |

### 12.5.12 TIM1 Auto-Reload Register (TIM1_ARR)

Offset address: 0x2C

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserve</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ARR[15:0]</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | ARR[15:0] | Auto-reload value (Prescaler value)<br>ARR contains the value to be loaded into the actual auto-reload register.<br>For details, refer to Section 12.3.1: Updates and actions related to ARR.<br>When the auto-reload value is empty, the counter does not work. | 0 | R/W |

### 12.5.13 TIM1 Repeat Count Register (TIM1_RCR)

Offset address: 0x30

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserve</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8" rowspan="2">reserve</td><td colspan="8">REP[7:0]</td></tr>
<tr><td colspan="8">R/W</td></tr>
</table>

| Bit Flag Functional Description | | | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | - | Reserved, always reads as 0. | 0 | - |
| 7:0 | REP[7:0] | Repetition counter value<br>After the preload function is turned on, these bits allow the user to set the update rate of the compare register (ie Periodically transfer from preload register to current register); if enabled generate update interrupt, it will also affect the rate at which update interrupts are generated.<br>Every time the down counter REP_CNT reaches 0, an update event is generated and the counter REP_CNT restarts counting from the REP value. Since REP_CNT is only updated periodically The REP value is only reloaded when the event U_RC occurs, so the new value written to the TIM1_RCR register Only takes effect when the next periodic update event occurs.<br>This means that in PWM mode, (REP+1) corresponds to:<br>— in edge-aligned mode, the number of PWM periods;<br>— in centrosymmetric mode, the number of PWM half-cycles; | 0 | R/W |

### 12.5.14 TIM1 capture/compare register 1 (TIM1_CCR1)

Offset address: 0x34

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR1[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR1[15:0] | Capture/Compare 1 value (Capture/Compare 1 value)<br>If the CC1 channel is configured as an output:<br>CCR1 contains the value loaded into the current capture/compare 1 register (preload value).<br>If the preload function is not selected in the TIM1_CCMR1 register (OC1PE bit), write The value of will be transferred to the current register immediately. Otherwise only when an update event occurs, this The preloaded value is transferred to the current capture/compare 1 register.<br>The current capture/compare register participates in the comparison with the counter TIM1_CNT, and in OC1 An output signal is generated on the port. like<br>CC1 channel configured as input:<br>CCR1 contains the counter value transmitted by the last Input Capture 1 event (IC1). | 0 | R/W |

### 12.5.15 TIM1 capture/compare register 2 (TIM1_CCR2)

Offset address: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR2[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR2[15:0] | Capture/Compare Channel 2 Value (Capture/Compare 2 value)<br>If the CC2 channel is configured as an output:<br>CCR2 contains the value loaded into the current capture/compare 2 register (preload value).<br>If the preload feature is not selected in the TIM1_CCMR2 register (OC2PE bit), write The value of will be transferred to the current register immediately. Otherwise only when an update event occurs, this The preloaded value is transferred to the current capture/compare 2 register.<br>The current capture/compare register participates in the comparison with the counter TIM1_CNT, and in OC2 An output signal is generated on the port.<br>If the CC2 channel is configured as an input:<br>CCR2 contains the counter value transmitted by the last Input Capture 2 event (IC2). | 0 | R/W |

### 12.5.16 TIM1 capture/compare register 3 (TIM1_CCR3)

Offset address: 0x3C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CCR3[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR3[15:0] | Capture/Compare Channel 3 Value (Capture/Compare 3 value)<br>If the CC3 channel is configured as an output:<br>CCR3 contains the value loaded into the current capture/compare 3 register (preload value). If the preload feature is not selected in the TIM1_CCMR3 register (OC3PE bit), the written The value is immediately transferred to the current register. Otherwise, only when an update event occurs, this preset The load value is transferred to the current capture/compare 3 register. The current capture/compare register participates in the comparison with the counter TIM1_CNT, and at the OC3 terminal The output signal is generated on the port.<br>If the CC3 channel is configured as an input:<br>CCR3 contains the counter value transmitted by the last Input Capture 3 event (IC3). | 0 | R/W |

### 12.5.17 TIM1 capture/compare register 4 (TIM1_CCR4)

Offset address: 0x40

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CCR4[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:0 | CCR4[15:0] | Capture/Compare 4 value (Capture/Compare 4 value)<br>If the CC4 channel is configured as an output:<br>CCR4 contains the value loaded into the current capture/compare 4 register (preload value). If the preload feature is not selected in the TIM1_CCMR4 register (OC4PE bit), the written The value is immediately transferred to the current register. Otherwise, only when an update event occurs, this preset The loaded value is transferred to the current capture/compare 4 register. The current capture/compare register participates in the comparison with the counter TIM1_CNT, and at the OC4 terminal The output signal is generated on the port.<br>If the CC4 channel is configured as an input:<br>CCR4 contains the counter value transmitted by the last Input Capture 4 event (IC4). | 0x0 | R/W |

### 12.5.18 TIM1 Brake and Deadband Register (TIM1_BDTR)

Offset address: 0x44

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| MOE | AOE | BKP | BKE | OSSR | OSSI | LOCK[1:0] | | DTG[7:0] | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | | R/W | | | | | | | |

Note: AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits can all be write-protected according to the lock setting, it is necessary to

They are configured when writing to the TIM1_BDTR register.

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15 | MOE | Main output enable<br>Once the brake input is active, this bit is asynchronously cleared to '0' by hardware. Depending on the setting value of the AOE bit, This bit can be cleared to '0' by software or set to 1 automatically. It is only available for channels configured as outputs effect.<br>0: disable OC and OCN output or force to idle state;<br>1: If the corresponding enable bit is set (CCxE, CCxNE of the TIM1_CCER register bit), the OC and OCN outputs are turned on.<br>For details on OC/OCN enable, see 11.5.9 TIM1 and TIM8 capture/compare enable registers register (TIM1_CCER). | 0 | R/W |
| 14 | AOE | Automatic output enable<br>0: MOE can only be set to '1' by software;<br>1: MOE can be set to '1' by software or automatically set to '1' at the next update event (if brake car input is invalid).<br>Note: Once the LOCK level (LOCK bit in TIM1_BDTR register) is set to '1', then This bit cannot be modified. | 0 | R/W |
| 13 | BKP | Brake input polarity (Break polarity)<br>0: Brake input is active at low level;<br>1: Brake input is active at high level.<br>Note: Once the LOCK level (LOCK bit in TIM1_BDTR register) is set to '1', then This bit cannot be modified.<br>Note: Any write operation to this bit requires a delay of one APB clock to take effect use. | 0 | R/W |
| 12 | BKE | Brake function enable (Break enable)<br>0: Disable brake input (BRK and CCS clock failure events);<br>1: Enable brake input (BRK and CCS clock failure events).<br>Note: When LOCK level 1 is set (LOCK bit in TIM1_BDTR register), the Bits cannot be modified.<br>Note: Any write operation to this bit requires a delay of one APB clock to take effect use. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 11 | OSSR | Off-state selection for Run mode This bit is used when MOE=1 and the channel is complementary output. The OSSR bit does not exist in timers without complementary outputs. Refer to the detailed description of OC/OCN enable (Section 11.5.9, TIM1 and TIM8 capture/compare enable register (TIM1_CCER)).<br>0: When the timer is not working, disable OC/OCN output (OC/OCN enable output signal=0); 1: When the timer is not working, once CCxE=1 or CCxNE=1, firstly enable OC/OCN and output invalid level, then set OC/OCN enable output signal=1. Note: Once the LOCK level (LOCK bit in TIM1_BDTR register) is set to 2, this bit cannot be modified. | 0 | R/W |
| 10 | OSSI | Off-state selection for Idle mode This bit is used when MOE=0 and the channel is set as output. Refer to the detailed description of OC/ OCN enable (Section 11.5.9, TIM1 and TIM8 Capture/Compare Enable Register (TIM1_CCER)). 0: When the timer is not working, disable OC/OCN output (OC/OCN enable output signal=0); 1: When the timer is not working, once CCxE=1 or CCxNE=1, OC/OCN will output its Idle level, then OC/ OCN enable output signal=1. Note: Once the LOCK level (LOCK bit in TIM1_BDTR register) is set to 2, this bit cannot be modified. Lock Configuration (Lock | 0 | R/W |
| 9:8 | LOOK[1:0] | Configuration) This bit provides write protection to prevent software errors. 00: lock off, registers are not write-protected; 01: lock level 1, can not write DTG, BKE, BKP, AOE bit and OISx/OISxN bit of TIM1_CR2 register; 10: Lock level 2, cannot write to each bit in lock level 1, and cannot write to CC polarity bit (once the relevant channel is set as output by CCxS bit, CC polarity bit It is the CCxP/CCNxP bit of the TIM1_CCER register) and OSSR/OSSI bit; 11: Lock level 3, you cannot write to each bit in lock level 2, and you cannot write to the CC control bit (once the relevant channel is set as an output through the CCxS bit, CC The control bit is the OCxM/OCxPE bit of the TIM1_CCMRx register); Note: After the system is reset, the LOCK bit can only be written once, once written to the TIM1_BDTR register, its content will be | 0x0 | R/W |
| 7:0 | DTG[7:0] | frozen until reset. Dead-time generator setup These bits define the duration of the dead-time inserted between complementary outputs. Suppose DT represents its duration: DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg, Tdtg = TDTS; DTG[7:5]=10x => DT=(64+DTG[5 :0])× Tdtg, Tdtg = 2 × TDTS; DTG[7:5]=110 => DT=(32+DTG[4:0])× Tdtg, Tdtg = 8 × TDTS; DTG[7:5] =111 => DT=(32+DTG[4:0])×Tdtg, Tdtg = 16×TDTS; Example: If TDTS = 125ns(8MHz), the possible dead time is: 0 to 15875ns, if the step time 125ns; 16us to 31750ns, if the step time is 250ns; 32us to 63us, if the step time is 1us; 64us to 126us, if the step time is 2us; Note: Once the LOCK level (LOCK bit in the TIM1_BDTR register) is set 1, 2, or 3, these bits cannot be modified. | 0x0 | R/W |

# 13 General purpose timer (TIM2)

## 13.1 Introduction to TIM2

The general purpose timer consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It is suitable for many occasions, including measuring the pulse length of an input signal (input capture) or generating an output waveform (output compare and PWM).

Using the timer prescaler and the RCC clock controller prescaler, the pulse length and waveform period can be from a few microseconds to a few milliseconds

time adjustment.

Each timer is completely independent and does not share any resources with each other. They can operate synchronously together.

## 13.2 TIM2 main functions

General TIM2 timer functions include:

ÿ 16-bit up, down, up/down autoload counter

ÿ 16-bit programmable (can be modified in real time) prescaler, the frequency division coefficient of the counter clock frequency is any number between 1 and 65536

value

ÿ 4 independent channels:

- input capture
- output compare
- PWM generation (edge or center aligned mode)
-Single pulse mode output

ÿ Use external signals to control timers and synchronize circuits interconnected by timers

ÿ An interrupt is generated when the following events occur:

-Update : counter overflow/underflow, counter initialization (by software or internal/external trigger)
- Trigger event (counter start, stop, initialization or counting by internal/external trigger)
- input capture
- output compare

ÿSupports incremental (quadrature) encoder and Hall sensor circuits for positioning

ÿTrigger input as external clock or cycle-by-cycle current management

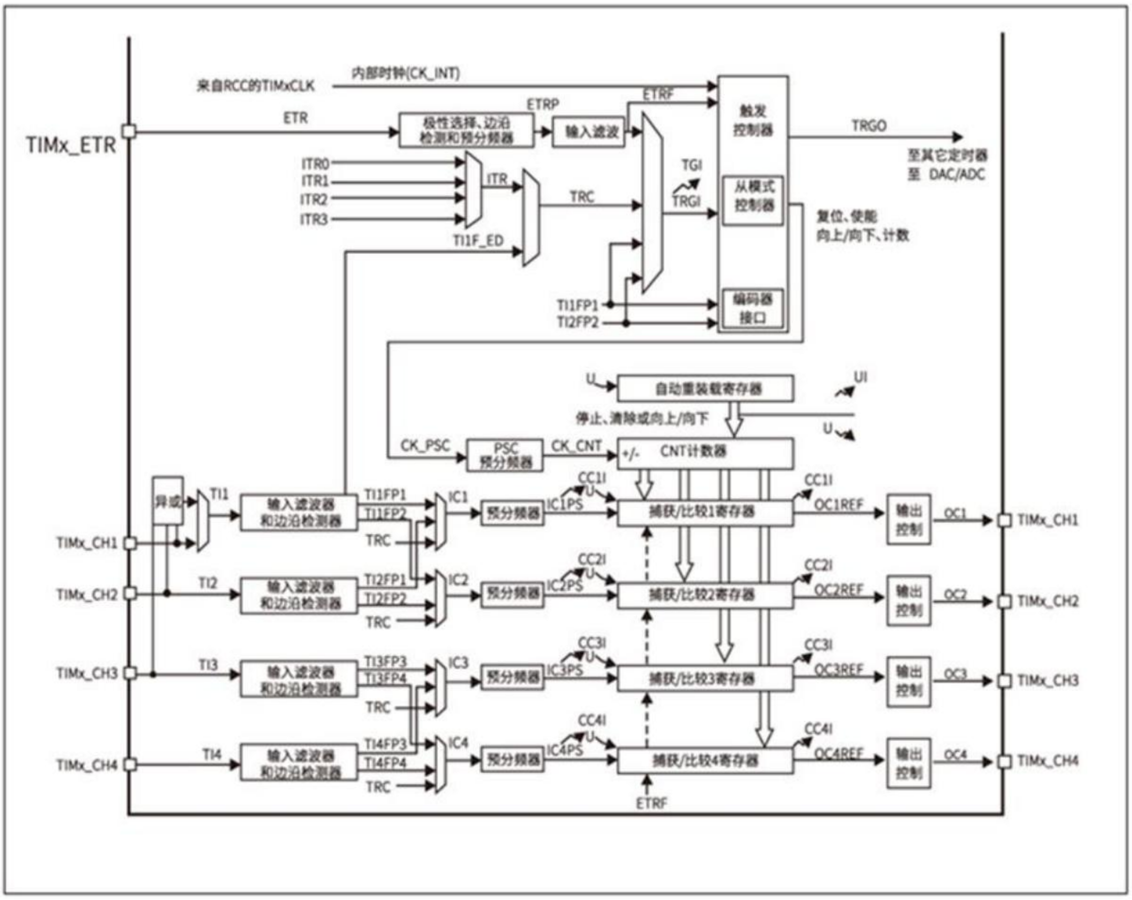


Figure 13-1 General-purpose timer block diagram

Note:

Reg

According to the setting of the control bit, the content of the preload register is transferred to the working register on the U (update) event

event

to interrupt

## 13.3 TIM2 Functional Description

### 13.3.1 Time base unit

The main part of the programmable general-purpose timer is a 16-bit counter and its associated autoload register. This counter can be

Count up, count down, or count up and down. This counter clock is divided by a prescaler.

The counter, autoload register and prescaler register can be read and written by software and can still be read and written while the counter is running. time base unit

Include:

ÿ Counter register (TIM2_CNT)

ÿ Prescaler register (TIM2_PSC)

ÿAuto Load Register (TIM2_ARR)

The auto-reload register is preloaded, and writing or reading the auto-reload register will access the preload register. According to register in TIM2_CR1

The setting of the automatic reload preload enable bit (ARPE) in the register, the content of the preload register is immediately or at each update event UEV

is transferred to the shadow register. When the counter reaches an overflow condition (underflow condition when counting down) and when the TIM2_CR1 register

An update event is generated when the UDIS bit is equal to '0'. Update events can also be generated by software. Each configuration is described in detail later

Generation of update events.

The counter is driven by the clock output CK_CNT of the prescaler, only if the counter enable bit in the counter TIM2_CR1 register is set

(CEN), CK_CNT is valid. (See the controller's slave mode description for details on counter enable).

Note: The actual counter enable signal CNT_EN is set after one clock cycle of CEN.

Prescaler Description

The prescaler can divide the clock frequency of the counter by any value between 1 and 65536. It is based on a (in the TIM2_PSC register

16-bit counter controlled by a 16-bit register. This control register is buffered and it can be changed on the fly. new

The prescaler parameters are used when the next update event arrives.

Figure 13-2 and Figure 13-3 show examples of changing the counter parameters while the prescaler is running.

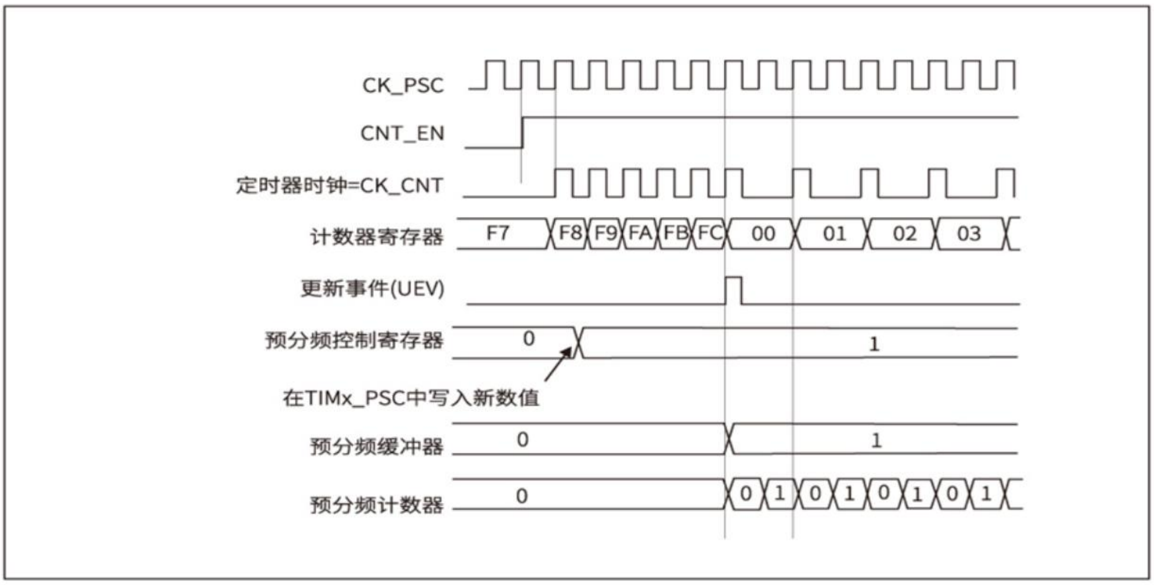


Figure 13-2 When the parameter of the prescaler changes from 1 to 2, the timing diagram of the counter

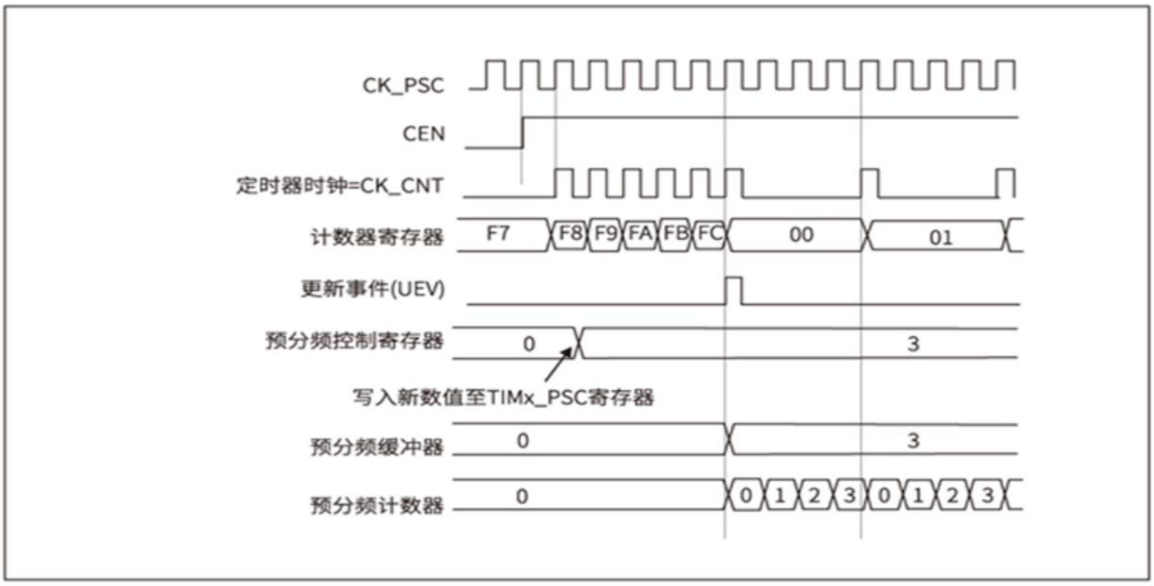


Figure 13-3 When the parameter of the prescaler changes from 1 to 4, the timing diagram of the counter

## 13.3.2 Counter Mode

### 13.3.2.1 Count Up Mode

In count-up mode, the counter counts from 0 to the autoload value (contents of the TIM2_ARR counter) and then restarts counting from 0

And generate a counter overflow event.

An update event can be generated each time the counter overflows, set in the TIM2_EGR register (by software or using a slave mode controller)

Setting the UG bit can also generate an update event.

The update event can be disabled by setting the UDIS bit in the TIM2_CR1 register; this avoids writing new

Update shadow registers when value is set. No update event will be generated until the UDIS bit is cleared to '0'. But when an update event should be generated,

The counter will still be cleared to '0', and the count of the prescaler will also be reset to 0 (but the prescaler coefficient remains unchanged). Additionally, if the

The URS bit in the TIM2_CR1 register (select update request), setting the UG bit will generate an update event UEV, but the hardware does not set

UIF flag (i.e. no interrupt generated); this is to avoid simultaneous update and capture interrupts being generated when the counter is cleared in capture mode.

When an update event occurs, all registers are updated, and the hardware simultaneously (according to the URS bit) sets the update flag (TIM2_SR

UIF bit in register).

ÿ The prescaler buffer is loaded with the value of the preload register (content of the TIM2_PSC register).

ÿ The autoload shadow register is reset to the value of the preload register (TIM2_ARR).

The figure below gives some examples, when TIM2_ARR=0x36, the actions of the counter at different clock frequencies.

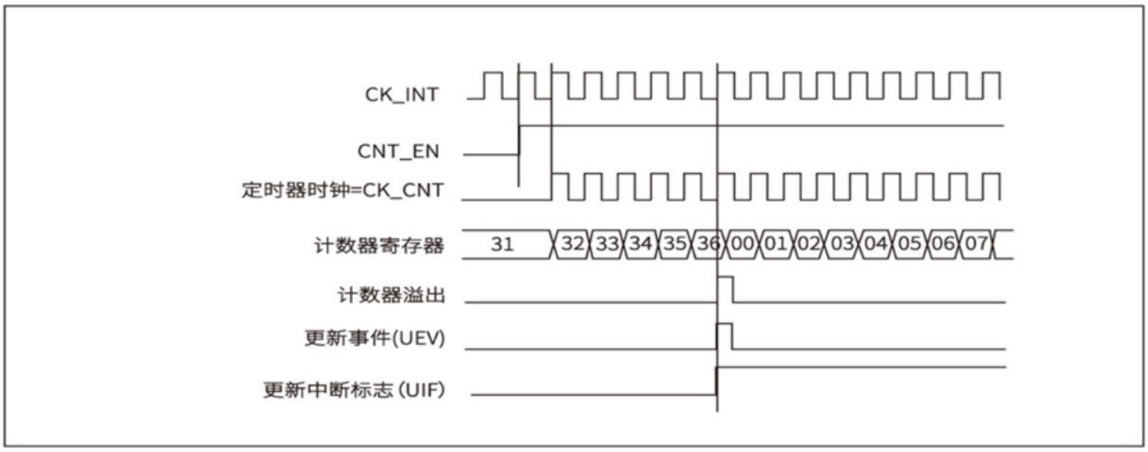


Figure 13-4 Timing diagram of the counter: internal clock frequency division factor is 1

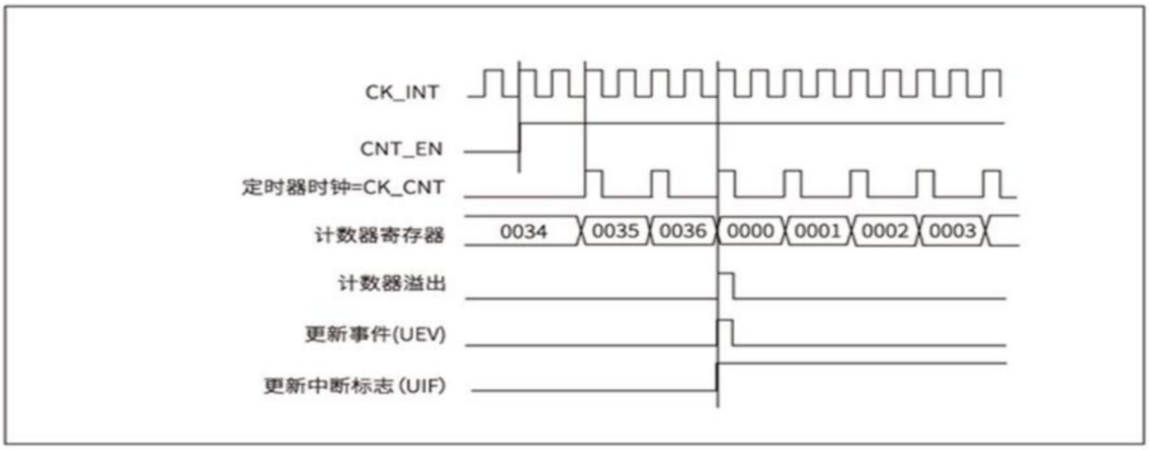


Figure 13-5 Counter timing diagram: internal clock frequency division factor is 2

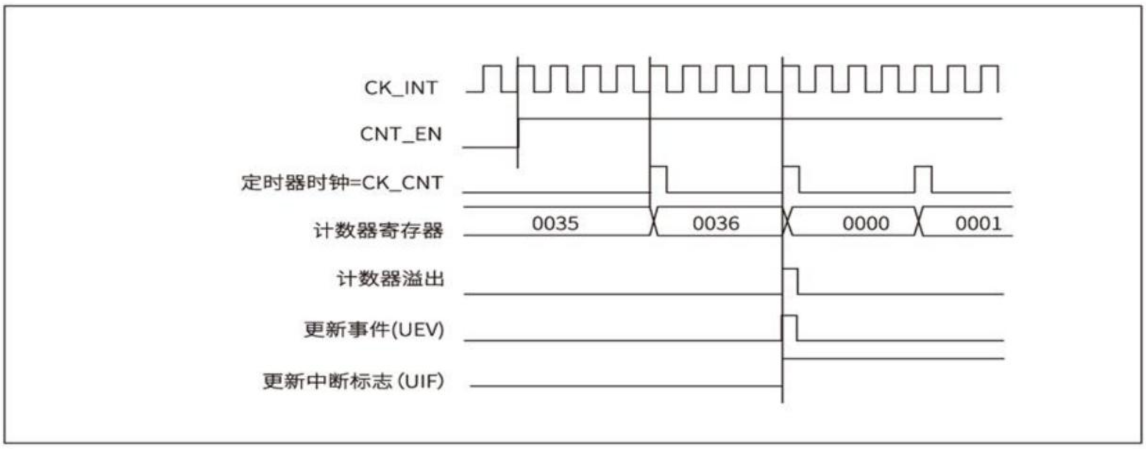


Figure 13-6 Counter timing diagram: internal clock frequency division factor is 4

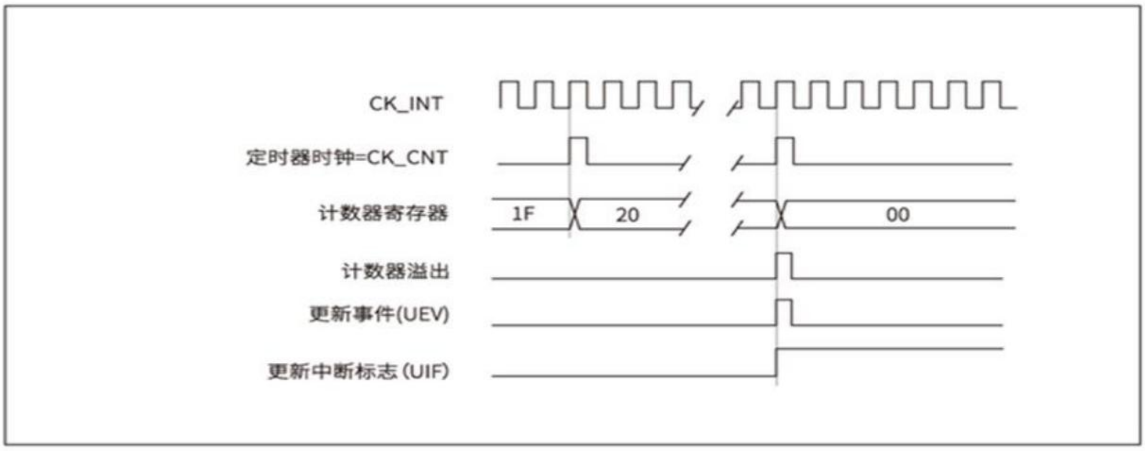


Figure 13-7 Timing diagram of the counter: internal clock division factor is N

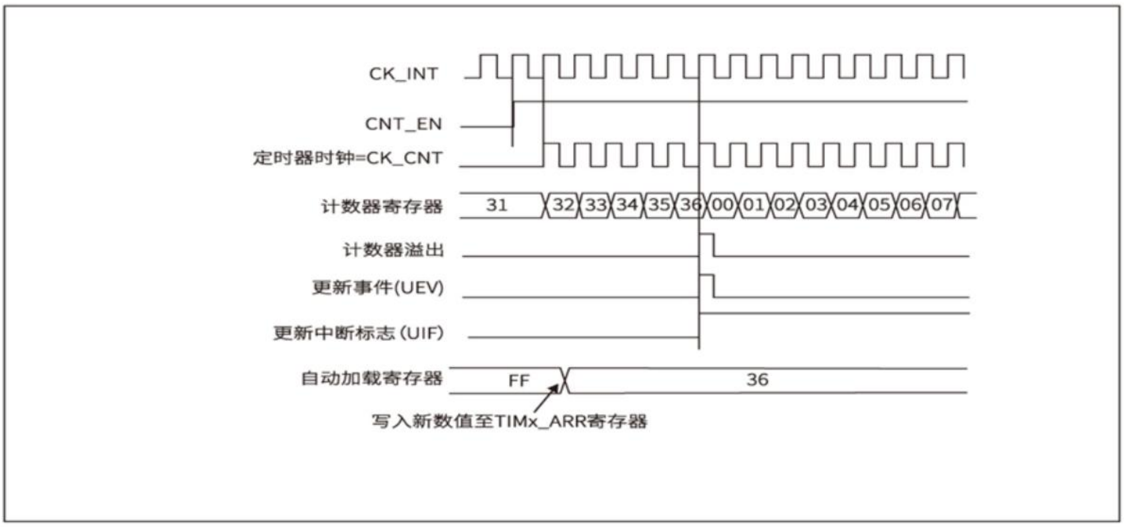


Figure 13-8 Counter timing diagram: update event when ARPE=0 (TIM2_ARR is not preloaded)

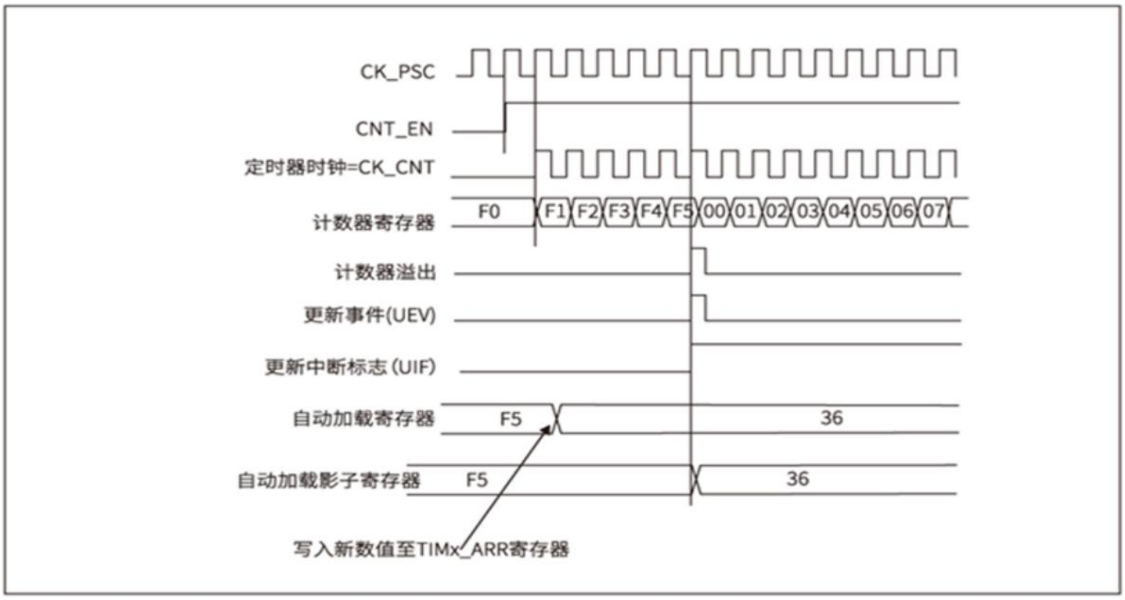


Figure 13-9 Counter timing diagram: Update event when ARPE=1 (TIM2_ARR is preloaded)

13.3.2.2 Down Counting Mode

In down mode, the counter counts down from the autoloaded value (TIM2_ARR counter value) to 0, then restarts from the autoloaded value and generates a counter underflow event.

An update event can be generated each time the counter overflows, set in the TIM2_EGR register (by software or using a slave mode controller)

Setting the UG bit also generates an update event.

The UEV event can be disabled by setting the UDIS bit of the TIM2_CR1 register. This avoids updating the shadow registers when new values are written to the preload registers. Therefore no update event will be generated until the UDIS bit is cleared to '0'. However, the counter will still restart counting from the current autoload value, and the counter of the prescaler will restart from 0 (but the prescaler factor will not change).

In addition, setting the UG bit will generate an update event if the URS bit in the TIM2_CR1 register is set (select update request)

UEV but does not set the UIF flag (so no interrupt is generated), this is to avoid simultaneous generation of

Update and capture interrupts.

When an update event occurs, all registers are updated and (according to the setting of the URS bit) the update flag bit (UIF bit in the TIM2_SR register) is also set.

ÿ The prescaler register is loaded with the value of the preload register (the value of the TIM2_PSC register).

ÿThe current autoload register is updated with the preload value (content in the TIM2_ARR register). NOTE: Automatic loading counts in

The counter is updated before reloading, so the next cycle will be the expected value.

The following are some examples of counter operation at different clock frequencies when TIM2_ARR=0x36.

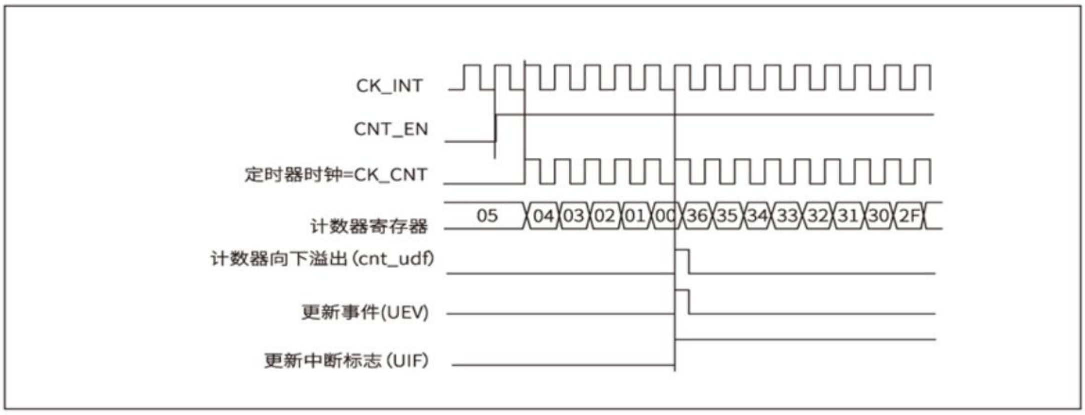


Figure 13-10 Counter timing diagram: internal clock frequency division factor is 1

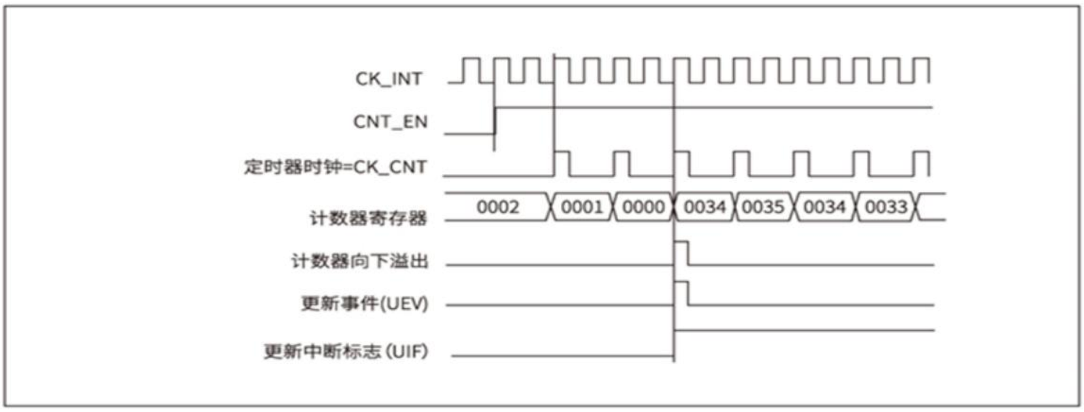


Figure 13-11 Counter timing diagram: internal clock frequency division factor is 2

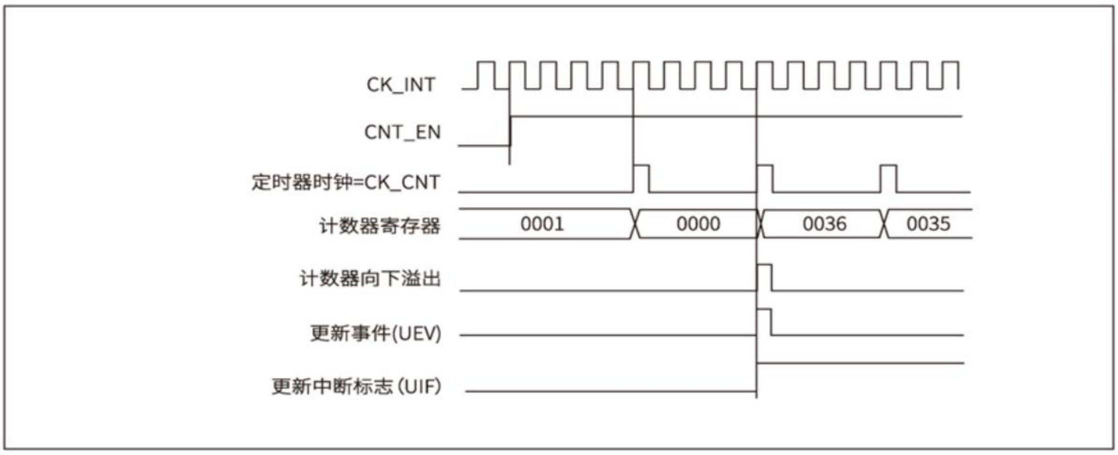


Figure 13-12 Counter Timing Diagram: Internal Clock Division Factor is 4

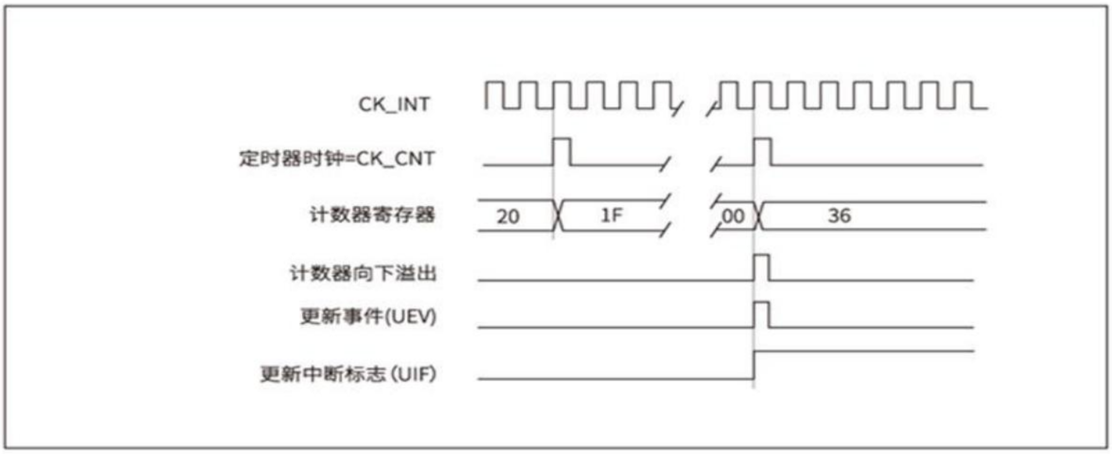


Figure 13-13 Counter Timing Diagram: Internal Clock Division Factor is N

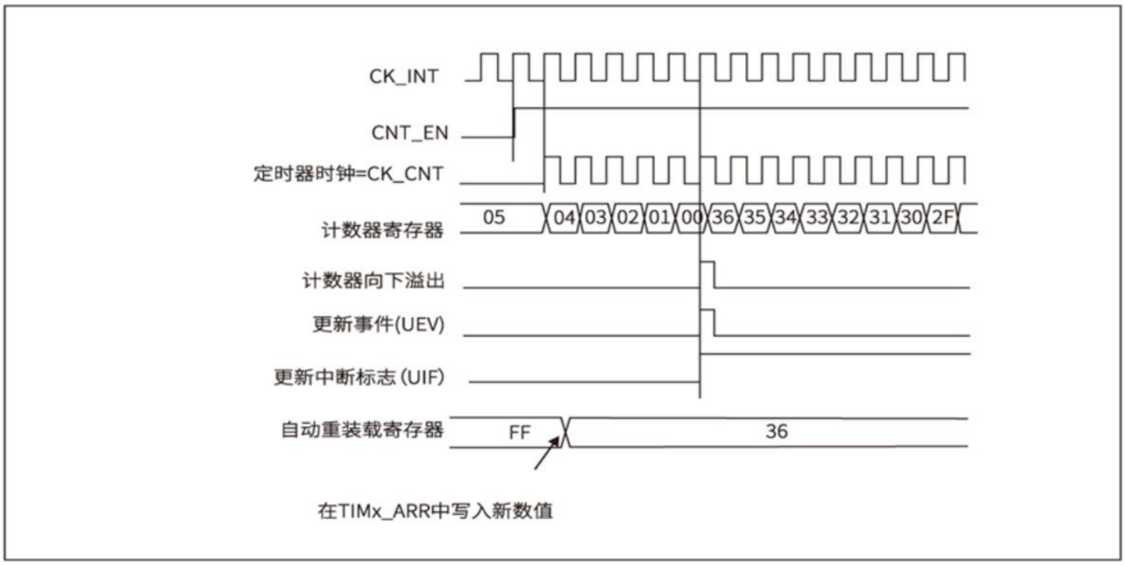


Figure 13-14 Counter Timing Diagram: Update Events When Repeat Counters Are Not Used

13.3.2.3 Center Aligned Mode (Count Up/Down)

In center-aligned mode, the counter counts from 0 to the autoload value (TIM2_ARR register)ÿ1, generating a counter overflow event, then counts down to 1 and generates a counter underflow event; then restarts counting from 0.

In this mode, the DIR direction bit in TIM2_CR1 cannot be written. It is updated by hardware and indicates the current counting direction. available every An update event is generated on each count overflow and each count underflow; can also be set by (software or using a slave mode controller) The UG bit in the TIM2_EGR register generates an update event. Then, the counter restarts counting from 0, and the prescaler restarts from 0 Start counting.

The UEV event can be disabled by setting the UDIS bit in the TIM2_CR1 register. This avoids writing new Update shadow registers when value is set. Therefore no update event will be generated until the UDIS bit is cleared to '0'. However, the counter will still The previous auto-reloaded value continues to count up or down.

Also, if the URS bit in the TIM2_CR1 register is set (selecting an update request), setting the UG bit will generate an update event UEV but not set the UIF flag (and thus not generate an interrupt), this is to avoid that when a capture event occurs and clears the counter , generating update and capture interrupts at the same time.

When an update event occurs, all registers are updated and (according to the setting of the URS bit) the update flag bit (UIF bit in the TIM2_SR register) is also set.

ÿ The prescaler register is loaded with the preload (TIM2_PSC register) value.

ÿThe current autoload register is updated with the preload value (content in the TIM2_ARR register).

Note: If an update occurs due to a counter overflow, the auto-reload will be updated before the counter is reloaded, so the next cycle will is the expected value (the counter is loaded with the new value).

The following are some examples of counter operation at different clock frequencies:

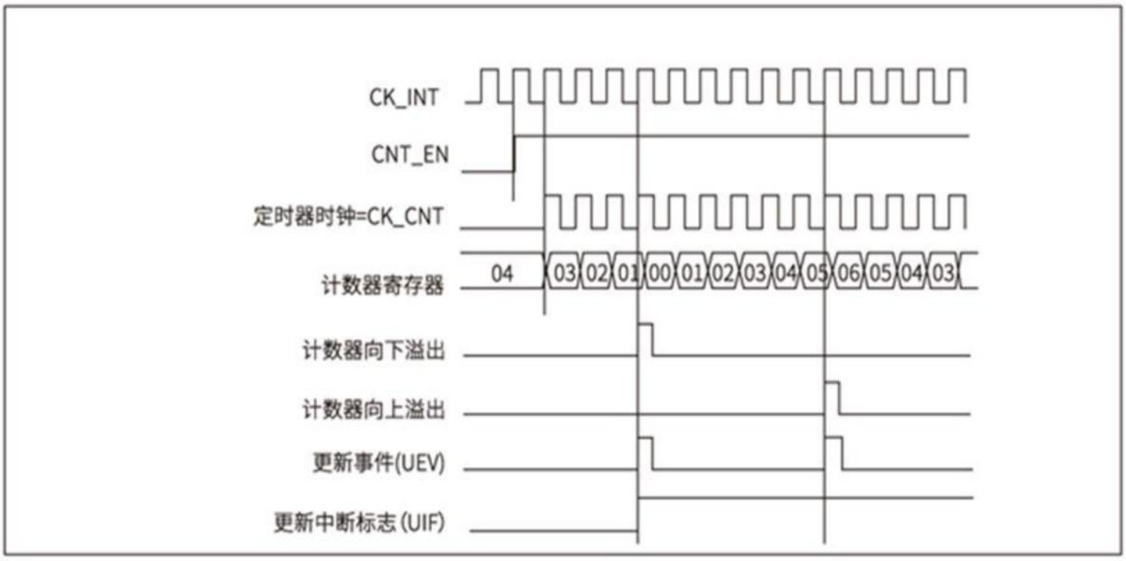


Figure 13-15 Counter timing diagram: internal clock frequency division factor is 1, TIM2_ARR=0x6

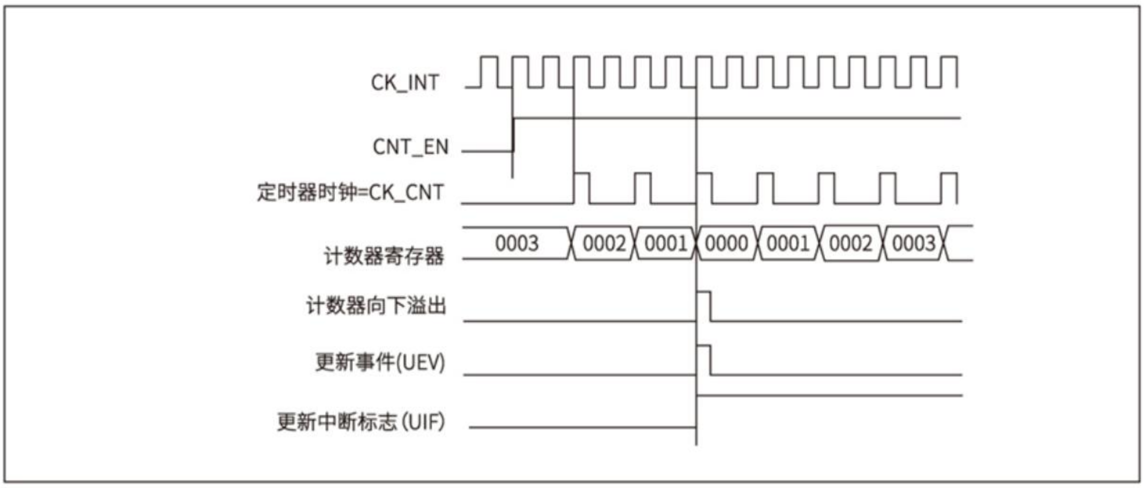
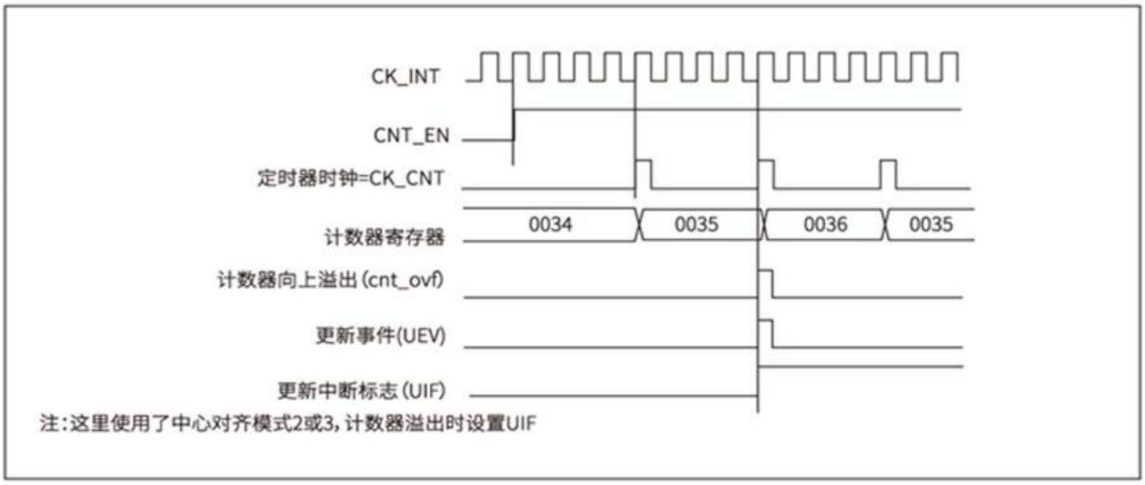


Figure 13-16 Counter timing diagram: internal clock frequency division factor is 2



注：这里使用了中心对齐模式2或3，计数器溢出时设置UIF

Figure 13-17 Counter timing diagram: internal clock frequency division factor is 4, TIM2_ARR=0x36

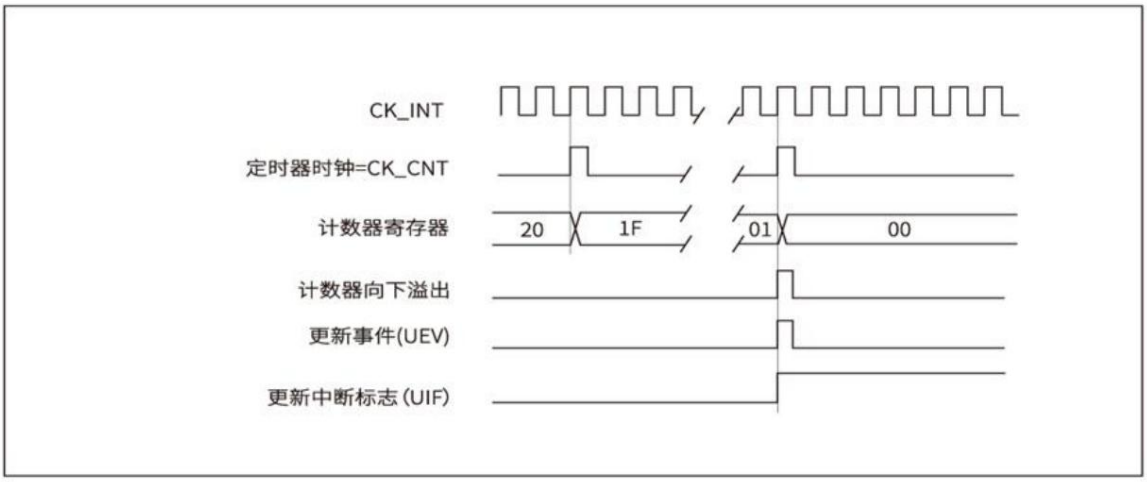


Figure 13-18 Counter timing diagram: internal clock frequency division factor is N

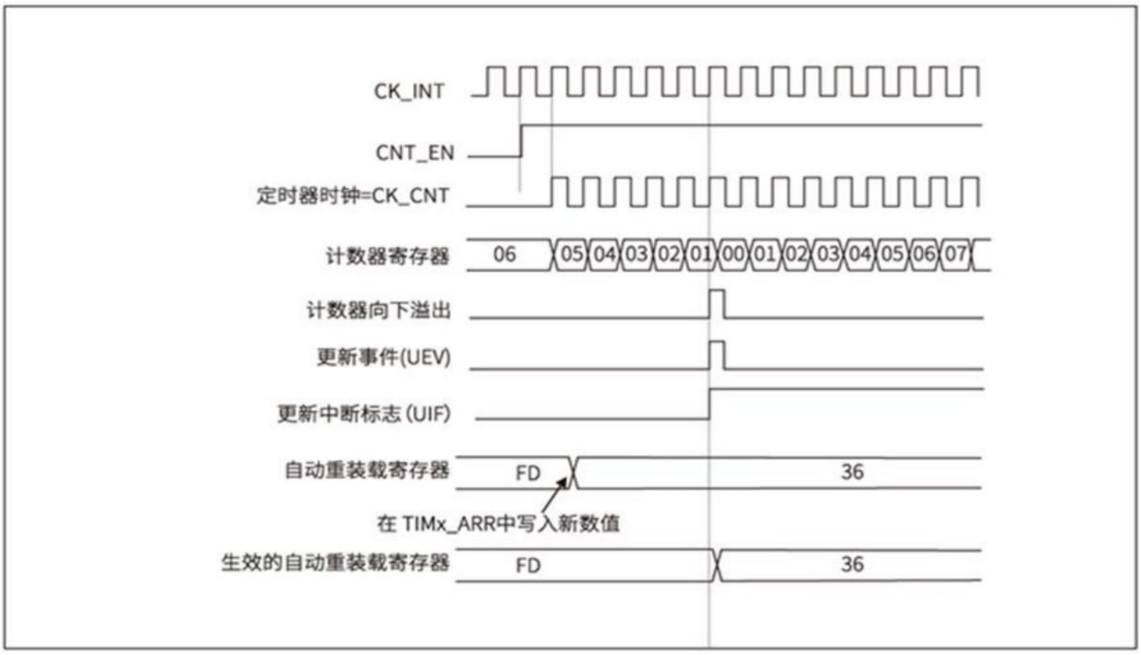


Figure 13-19 Counter timing diagram: update event when ARPE=1 (counter underflow)

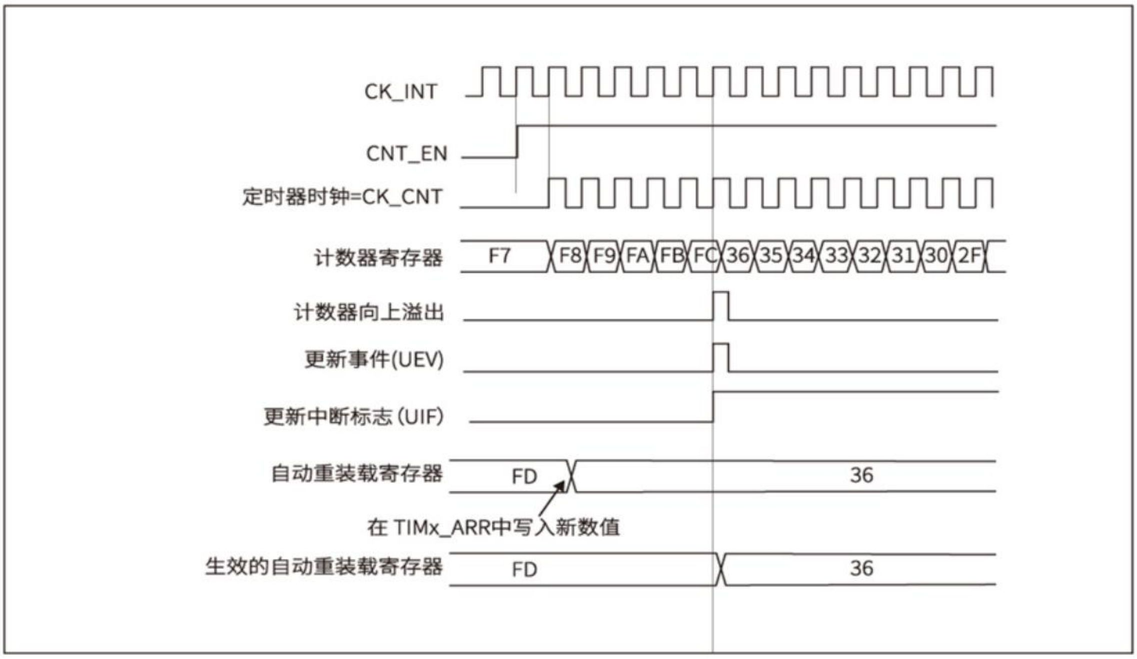


Figure 13-20 Counter timing diagram: update event when ARPE=1 (counter overflow)

### 13.3.3 Clock Selection

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

ÿInternal clock (CK_INT)

ÿExternal clock mode 1: External input pin (TIx)

ÿExternal clock mode 2: External trigger input (ETR)

ÿInternal trigger input (ITRx): Use one timer as the prescaler of another timer, such as a timer can be configured

Timer1 acts as a prescaler for another timer, Timer2.

13.3.3.1 Internal Clock Source (CK_INT)

If the slave mode controller is disabled (SMS=000 of TIM2_SMCR register), CEN, DIR (TIM2_CR1 register) and UG

bit (TIM2_EGR register) is the de facto control bit and can only be modified by software (the UG bit is still automatically cleared). As long as the CEN bit is written

becomes '1', the clock of the prescaler is provided by the internal clock CK_INT.

The figure below shows the operation of the control circuit and up-counter in normal mode without prescaler.

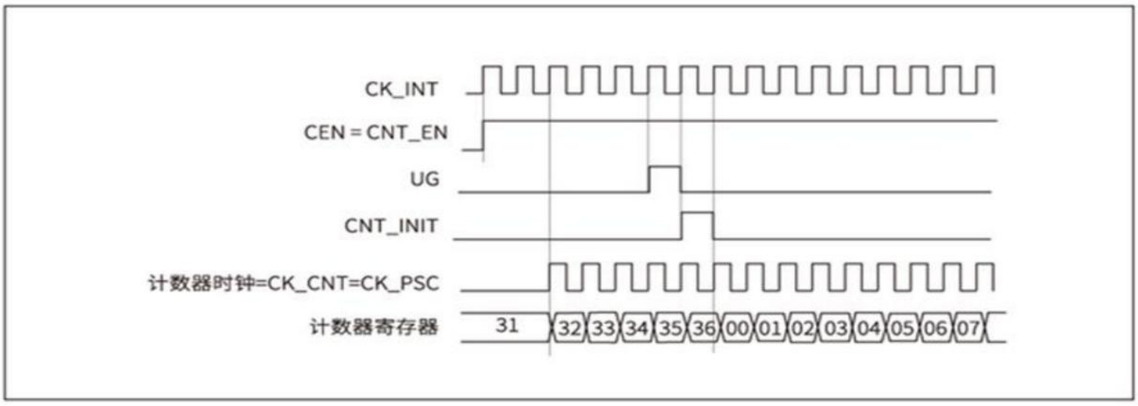


Figure 13-21 Control circuit in normal mode, internal clock frequency division factor is 1

13.3.3.2 EXTERNAL CLOCK SOURCE MODE 1

This mode is selected when SMS=111 in the TIM2_SMCR register. The counter can be activated on every rising or falling edge of the selected input

Falling edge count.

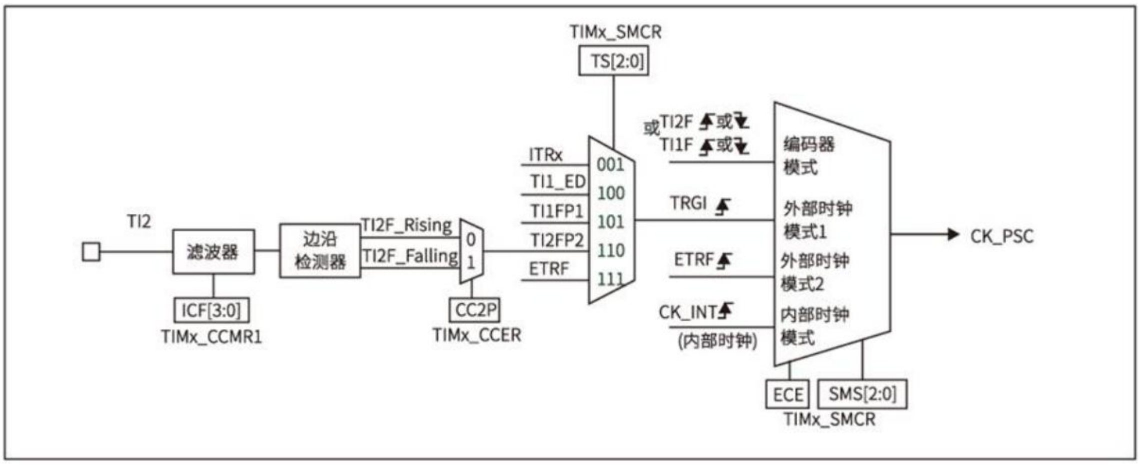


Figure 13-22 TI2 external clock connection example

For example, to configure an up counter to count on rising edges at the T12 input, use the following steps:

1. Configure TIM2_CCMR1 register CC2S='01', configure channel 2 to detect the rising edge of TI2 input
2. Configure IC2F[3:0] of the TIM2_CCMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000)

Note: The capture prescaler is not used as a trigger, so it does not need to be configured

3. Configure CC2P='0' of the TIM2_CCER register, select the rising edge polarity
4. Configure the SMS='111' of the TIM2_SMCR register to select the timer external clock mode 1
5. Configure TS='110' in the TIM2_SMCR register, and select TI2 as the trigger input source
6. Set CEN='1' of TIM2_CR1 register to start the counter

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge of TI2 and the actual clocking of the counter depends on the resynchronization circuit at the TI2 input.

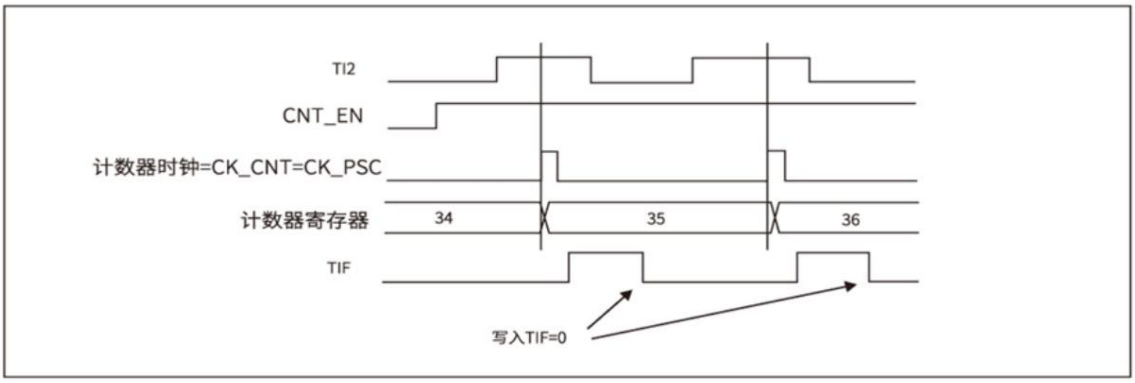


Figure 13-23 Control circuit in external clock mode 1

13.3.3.3 External Clock Source Mode 2

The method to select this mode is: make the ECE=1 counter in the TIM2_SMCR register can be triggered on each of the external trigger ETR

Rising or falling edge counting. The following figure is a block diagram of the external trigger input:

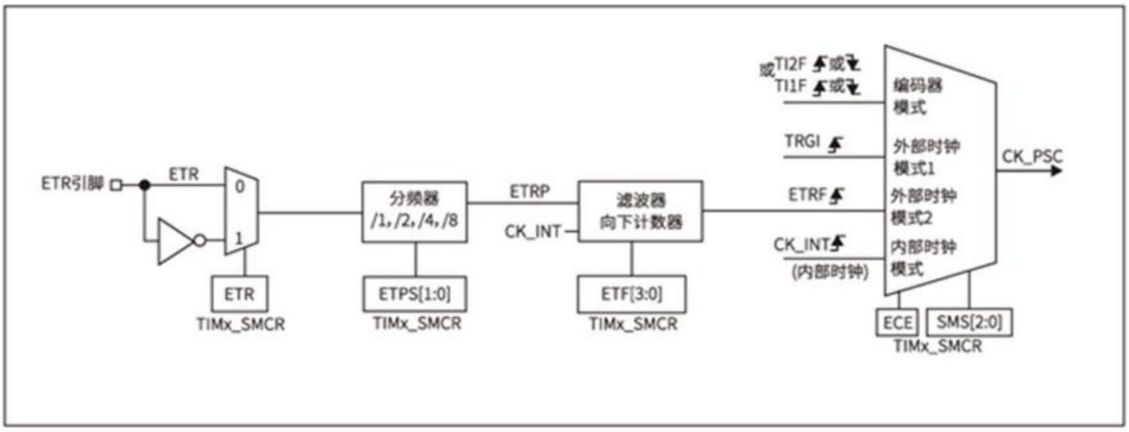


Figure 13-24 External trigger input block diagram

For example, to configure an up counter that counts every 2 rising edges at ETR, use the following steps:

1. No filter is needed in this example, set ETF[3:0]=0000 in the TIM2_SMCR register

2. Set the prescaler, set ETPS[1:0]=01 in the TIM2_SMCR register

3. Set the rising edge detection of ETR, set ETP=0 in the TIM2_SMCR register

4. Enable external clock mode 2, set ECE=1 in TIM2_SMCR register

5. Start the counter, set CEN=1 in the TIM2_CR1 register

The counter counts every 2 rising edges of ETR.

The delay between the rising edge of ETR and the actual clocking of the counter depends on the resynchronization circuitry on the ETRP signal.

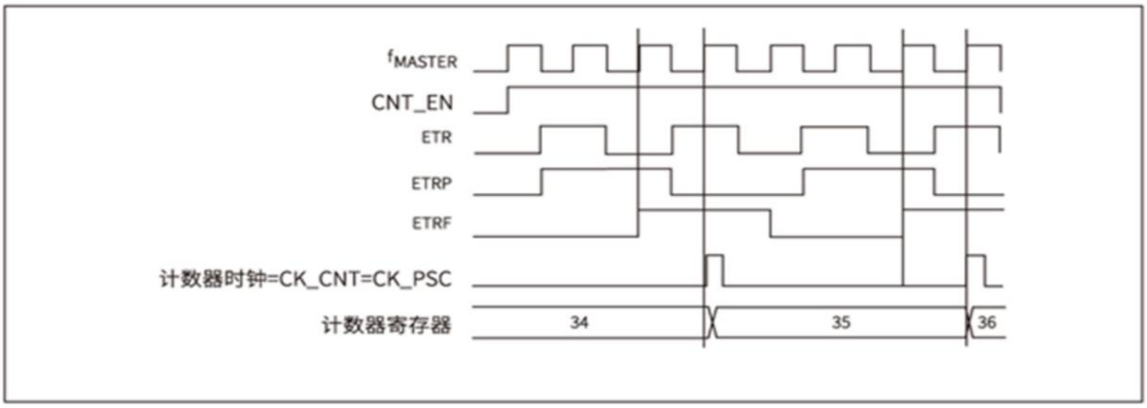


Figure 13-25 Control circuit in external clock mode 2

### 13.3.4 Capture/Compare Channels

Each capture/compare channel is built around a capture/compare register (including shadow registers), including the input section of the capture (digital filtering, multiplexing, and prescaler), and the output section (comparator and output control ).

The following figures are an overview of the capture/compare channel.

The input section samples the corresponding TIx input signal and produces a filtered signal TIxF. Then, an edge with polarity selection

The detector generates a signal (TIxFPx) which can be used as an input trigger for the slave mode controller or as a capture control. This signal is prescaled into the capture register (ICxPS).

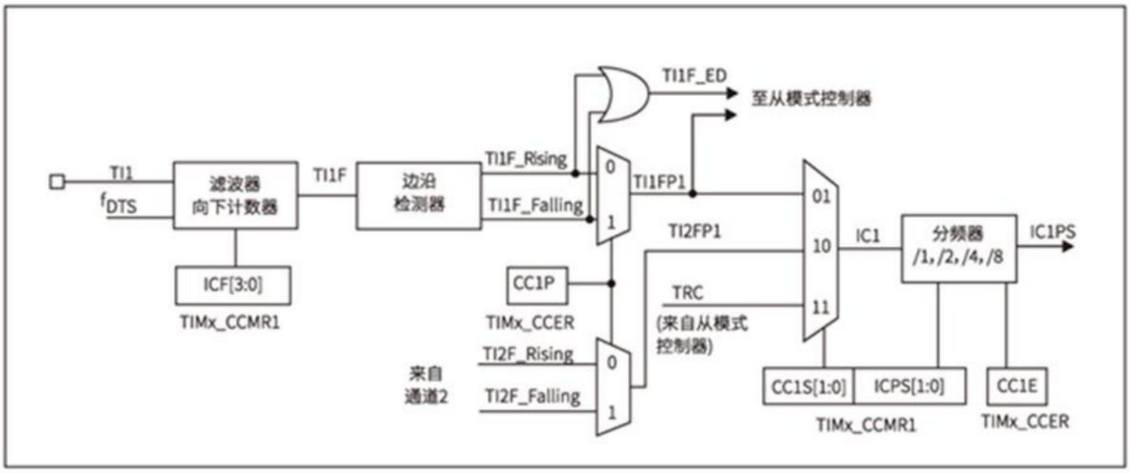


Figure 13-26 Capture/compare channel (eg: channel 1 input part)

The output section generates an intermediate waveform OCxRef (active high) as a reference, and the end of the chain determines the polarity of the final output signal.

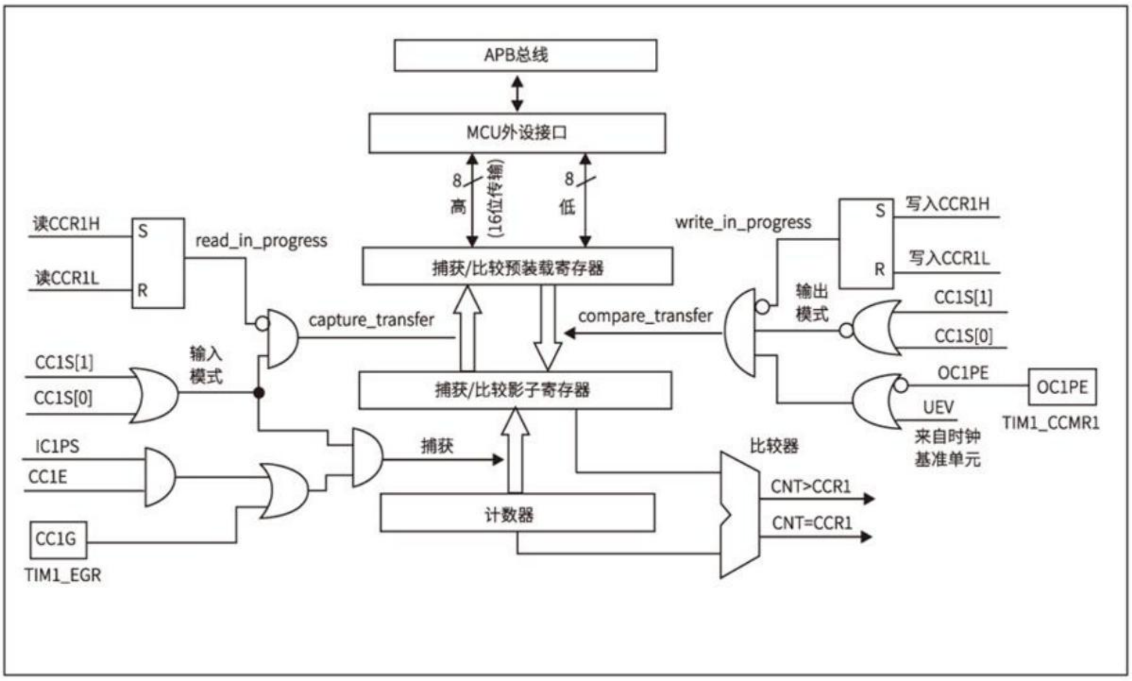


Figure 13-27 The main circuit of capture/compare channel 1

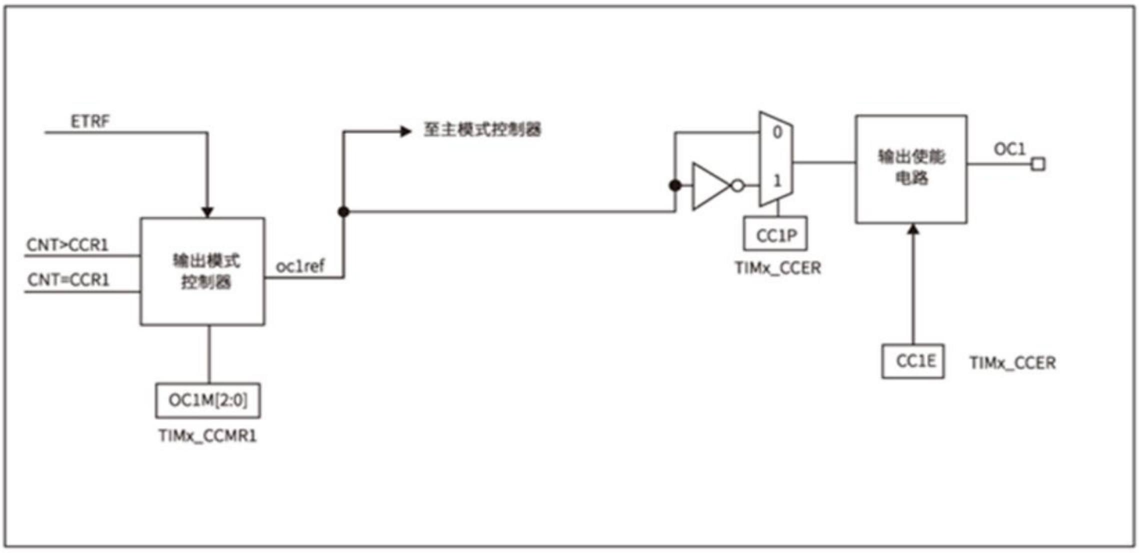


Figure 13-28 Capture/Compare Channel Output Section (Channel 1)

The capture/compare module consists of a preload register and a shadow register. The read and write process only operates the preload register.

In capture mode, the capture occurs on the shadow registers, which are then copied into the preload registers.

In compare mode, the content of the preload register is copied to the shadow register, and then the content of the shadow register is compared with the counter

Compare.

### 13.3.5 Input capture mode

In input capture mode, the current value of the counter is latched into the capture/compare register when the corresponding edge on the ICx signal is detected

(TIM2_CCRx). When a capture event occurs, the corresponding CCxIF flag (TIM2_SR register) is set to '1', if enabled

interrupt, an interrupt will be generated. If the CCxIF flag is already high when the capture event occurs, the repeated capture flag CCxOF (TIM2_SR register

device) is set to '1'. Write CCxIF=0 to clear CCxIF, or read capture data stored in TIM2_CCRx register to clear

CCxIF. Write CCxOF=0 to clear CCxOF.

The following example shows how to capture the value of the counter to the TIM2_CCR1 register on the rising edge of the TI1 input. The steps are as follows:

ÿSelect valid input: TIM2_CCR1 must be connected to TI1 input, so write to TIM2_CCR1 register

CC1S=01, as long as CC1S is not '00', the channel is configured as an input, and the TM1_CCR1 register becomes read-only.

ÿ According to the characteristics of the input signal, configure the input filter to the required bandwidth (that is, when the input is TIx, the input filter control bit is the ICxF bit in the TIM2_CCMRx register). Assuming that the input signal jitters within a period of at most 5 internal clock cycles, we have to configure the filter bandwidth to be longer than 5 clock cycles. We can therefore sample 8 consecutive times (at fDTS frequency) to confirm

On the last real edge transition of TI1, write IC1F=0011 in the TIM2_CCMR1 register.

ÿ Select the valid conversion edge of the TI1 channel, write CC1P=0 (rising edge) in the TIM2_CCER register.

ÿ Configure the input prescaler. In this example, we want the capture to occur at every valid level transition, so the prescaler

The register is disabled (write IC1PS=00 in TIM2_CCMR1 register).

ÿ Set CC1E=1 in the TIM2_CCER register to allow the value of the capture counter to be captured in the capture register.

ÿ If required, enable the associated interrupt request by setting the CC1IE bit in the TIM2_DIER register.

When an input capture occurs:

ÿWhen a valid level transition occurs, the counter value is transferred to the TIM2_CCR1 register.

ÿ CC1IF flag is set (interrupt flag). When at least 2 consecutive captures occur, and CC1IF has not been cleared, CC1OF is set to '1'.

ÿIf the CC1IE bit is set, an interrupt will be generated.

In order to deal with the capture overflow, it is recommended to read the data before reading the capture overflow flag, this is to avoid losing the capture overflow flag when reading Capture overflow information that may occur after and before reading data.

Note: An input capture interrupt can be generated by software by setting the corresponding CCxG bit in the TIM2_EGR register.

## 13.3.6 PWM Input Mode

This mode is a special case of input capture mode and operates the same as input capture mode except for the following differences:

ÿ Two ICx signals are mapped to the same TIx input.

ÿThe 2 ICx signals are edge active, but opposite in polarity.

ÿ One of the TIxFP signals is used as a trigger input signal, and the slave mode controller is configured in reset mode.

For example, you need to measure the length (TIM2_CCR1 register) and duty cycle (TIM2_CCR2 register) of the PWM signal input to TI1, the specific steps are as follows (depending on the frequency of CK_INT and the value of the prescaler) ÿSelect valid input of TIM2_CCR1: Set CC1S=01 in TIM2_CCMR1 register (select TI1).

ÿSelect the effective polarity of TI1FP1 (used to capture data into TIM2_CCR1 and clear the counter): set CC1P=0 (rising edge has effect).

ÿSelect valid input of TIM2_CCR2: set CC2S=10 in TIM2_CCMR1 register (select TI1).

ÿSelect the active polarity of TI1FP2 (capture data to TIM2_CCR2): set CC2P=1 (falling edge active).

ÿSelect a valid trigger input signal: set TS=101 in the TIM2_SMCR register (select TI1FP1).

ÿ Configure the slave mode controller as reset mode: set SMS=100 in TIM2_SMCR.

ÿEnable capture: set CC1E=1 and CC2E=1 in TIM2_CCER register.

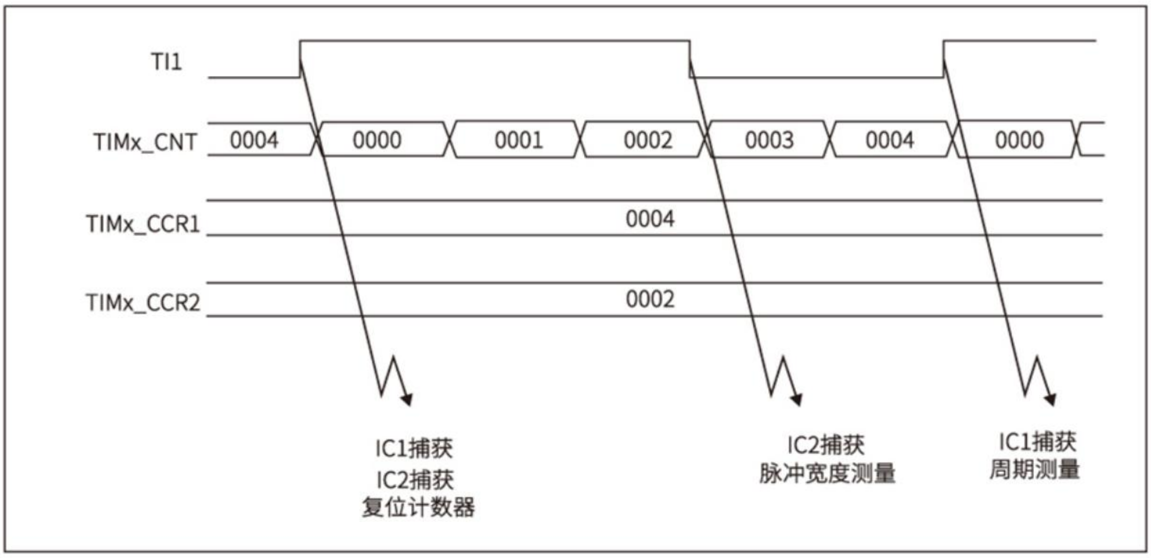


Figure 13-29 PWM input mode timing

Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can only be used TIM2_CH1/TIM2_CH2 signal.

## 13.3.7 Forced output mode

In output mode (CCxS=00 in TIM2_CCMRx register), the output compare signal (OCxREF and corresponding OCx) can be directly controlled by Software forces the active or inactive state independently of the comparison result between the output compare register and the counter.

Set the corresponding OCxM=101 in the TIM2_CCMRx register to force the output compare signal (OCxREF/OCx) to be valid. Thus OCxREF is forced high (OCxREF is always active high), and OCx gets the opposite value of the CCxP polarity bit.

For example: CCxP=0 (OCx active high), then OCx is forced to be high.

Set OCxM=100 in TIM2_CCMRx register to force OCxREF signal low. In this mode, the TIM2_CCRx shadow register The comparison between the register and the counter is still in progress, and the corresponding flag will be modified. So the corresponding interrupt will still be generated. this will Described in the Output Compare Mode section below.

### 13.3.8 Output Compare Mode

This function is used to control an output waveform, or to indicate that a given time has elapsed. when

When the contents of the counter and the capture/compare register are the same, the output compare

function does the following: ÿ Output the value defined by the output compare mode (OCxM bit in the TIM2_CCMRx register) and output polarity (CCxP bit

in the TIM2_CCER register) to the corresponding on the pin. On a compare match, the output pin can maintain its level

(OCxM=000), set to an active level (OCxM=001), set to an inactive level (OCxM=010), or toggle (OCxM=011).

ÿ Set the flag bit in the interrupt status register (CCxIF bit in the TIM2_SR register). ÿ An interrupt is generated

if the corresponding interrupt mask is set (CCxIE bit in the TIM2_DIER register).

The OCxPE bit in TIM2_CCMRx selects whether the TIM2_CCRx register needs to use the preload register. exist

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

The accuracy of the synchronization can reach one count cycle of the counter. The output compare mode (in single pulse mode) can also be used to output a single pulse

rush.

Configuration steps for output compare mode:

1. Select the counter clock (internal, external, prescaler)
2. Write the corresponding data into the TIM2_ARR and TIM2_CCRx registers
3. To generate an interrupt request, set the CCxIE bit.
4. Select the output mode, such as flipping the output pin of OCx when the counter CNT matches CCRx, CCRx preload is not used, and is enabled OCx output and active high, you must set OCxM='011', OCxPE='0', CCxP='0' and CCxE='1'.
5. Set the CEN bit of the TIM2_CR1 register to start the counter

The TIM2_CCRx register can be updated by software at any time to control the output waveform, provided that the preload register is not used (OCxPE='0',

otherwise the TIM2_CCRx shadow register can only be updated at the next update event). The figure below gives a

example.

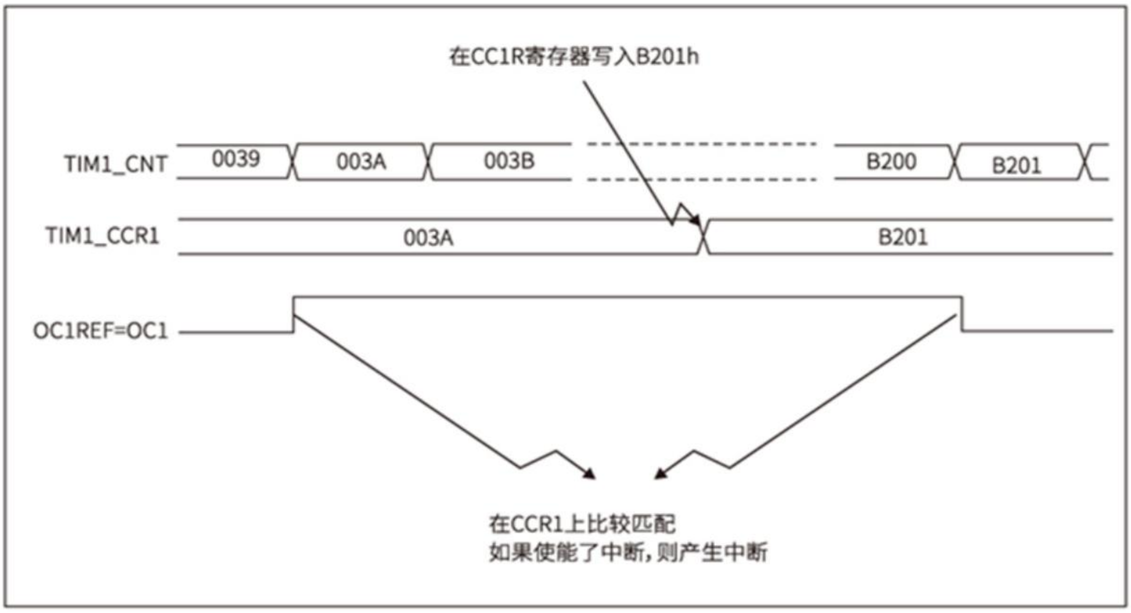


Figure 13-30 Output compare mode, flip OC1

## 13.3.9 PWM Mode

The Pulse Width Modulation mode can generate a frequency determined by the TIM2_ARR register and a duty cycle determined by the TIM2_CCRx register.

Signal.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bits in the TIM2_CCMRx register can independently

Set each OCx output channel to generate one PWM. The TIM2_CCMRx register OCxPE bit must be set to enable the corresponding preloaded

Load the register, and finally set the ARPE bit of the TIM2_CR1 register to enable auto-restart (in up-counting or centrosymmetric mode)

Loaded preload registers.

Only when an update event occurs, the preload register can be transferred to the shadow register, so before the counter starts counting, it must be

All registers must be initialized by setting the UG bit in the TIM2_EGR register. The polarity of OCx can be set by software in

CCxP bit setting in TIM2_CCER register, it can be set as active high or active low. TIM2_CCER register

The OCx output enable is controlled by the CCxE bit in . See the description of the IM2_CCERx register for details.

In PWM mode (mode 1 or mode 2), TIM2_CNT and TIM2_CCRx are always compared, (according to the counting direction of the counter) to determine whether

TIM2_CCRx ÿ TIM2_CNT or TIM2_CNT ÿ TIM2_CCRx. However, in order to be consistent with the function of OCREF_CLR (an external event on the ETR signal

can clear OCxREF before the next PWM cycle), the OCxREF signal can only be generated under the following conditions:

ÿ When the result of the comparison changes

or

ÿWhen output compare mode (OCxM bit in TIM2_CCMRx register) is switched from "frozen" (no compare, OCxM='000')

to a certain PWM mode (OCxM='110' or '111')

In this way, the PWM output can be forced by software during operation.

Depending on the state of the CMS bit in the TIM2_CR1 register, the timer can generate edge-aligned PWM signals or center-aligned PWM

Signal.

### 13.3.9.1 PWM Edge-Aligned Mode

#### 13.3.9.1.1 Count Up Configuration

Counting up is performed when the DIR bit in the TIM2_CR1 register is low.

Below is an example of PWM mode 1. PWM signal reference OCxREF is high when TIM2_CNT<TIM2_CCRx, otherwise it is

Low. If the compare value in TIM2_CCRx is greater than the auto-reload value (TIM2_ARR), OCxREF remains '1'. if compare

A value of 0 keeps OCxREF at '0'. The figure below is an example of edge-aligned PWM waveform when TIM2_ARR=8.

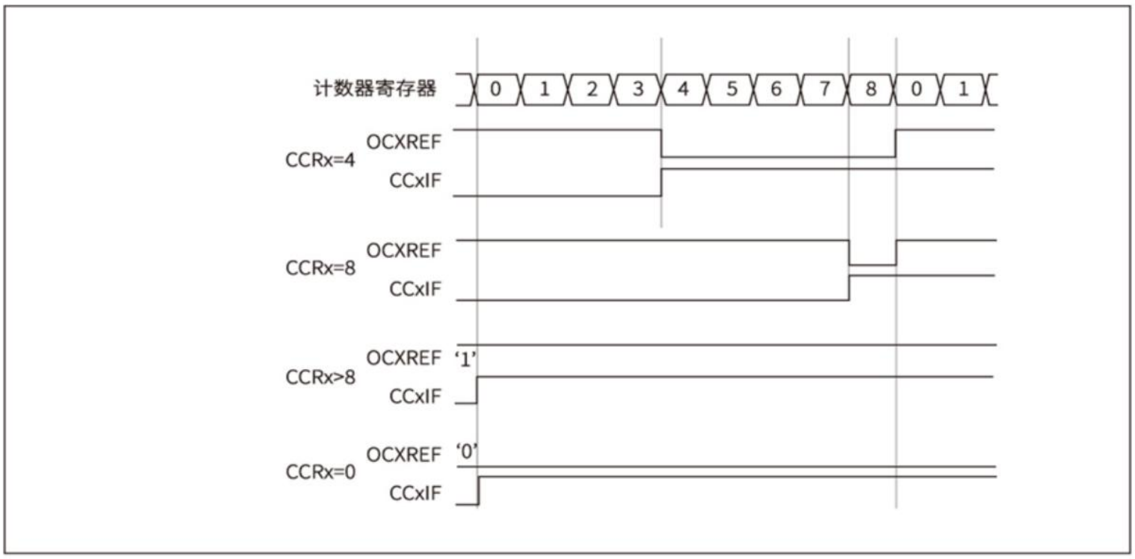


Figure 13-31 Edge-aligned PWM waveform (ARR=8)

13.3.9.1.2 Configuration of Count Down

Down counting is performed when the DIR bit of the TIM2_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when TIM2_CNT>TIM2_CCRx, otherwise it is high. If the ratio in TIM2_CCRx

If the compare value is greater than the auto-reload value in TIM2_ARR, OCxREF remains '1'. 0% PWM wave cannot be generated in this mode

shape.

#### 13.3.9.2 PWM Center-Aligned Mode

When the CMS bit in the TIM2_CR1 register is not '00', it is center-aligned mode (all other configurations are OCxREF/OCx signals have the same effect). Depending on the CMS bit setting, the compare flag can be set to '1' when the counter is counting up, Set to '1' when the counter is counting down, or set to '1' when the counter is counting up and down. Count in TIM2_CR1 register The direction bit (DIR) is updated by hardware, do not modify it by software.

The figure below gives some examples of center-aligned PWM waveforms

ÿ TIM2_ARR=8

ÿ PWM mode 1

ÿ CMS=01 in TIM2_CR1 register, in center-aligned mode 1, set compare flag when counter counts down.

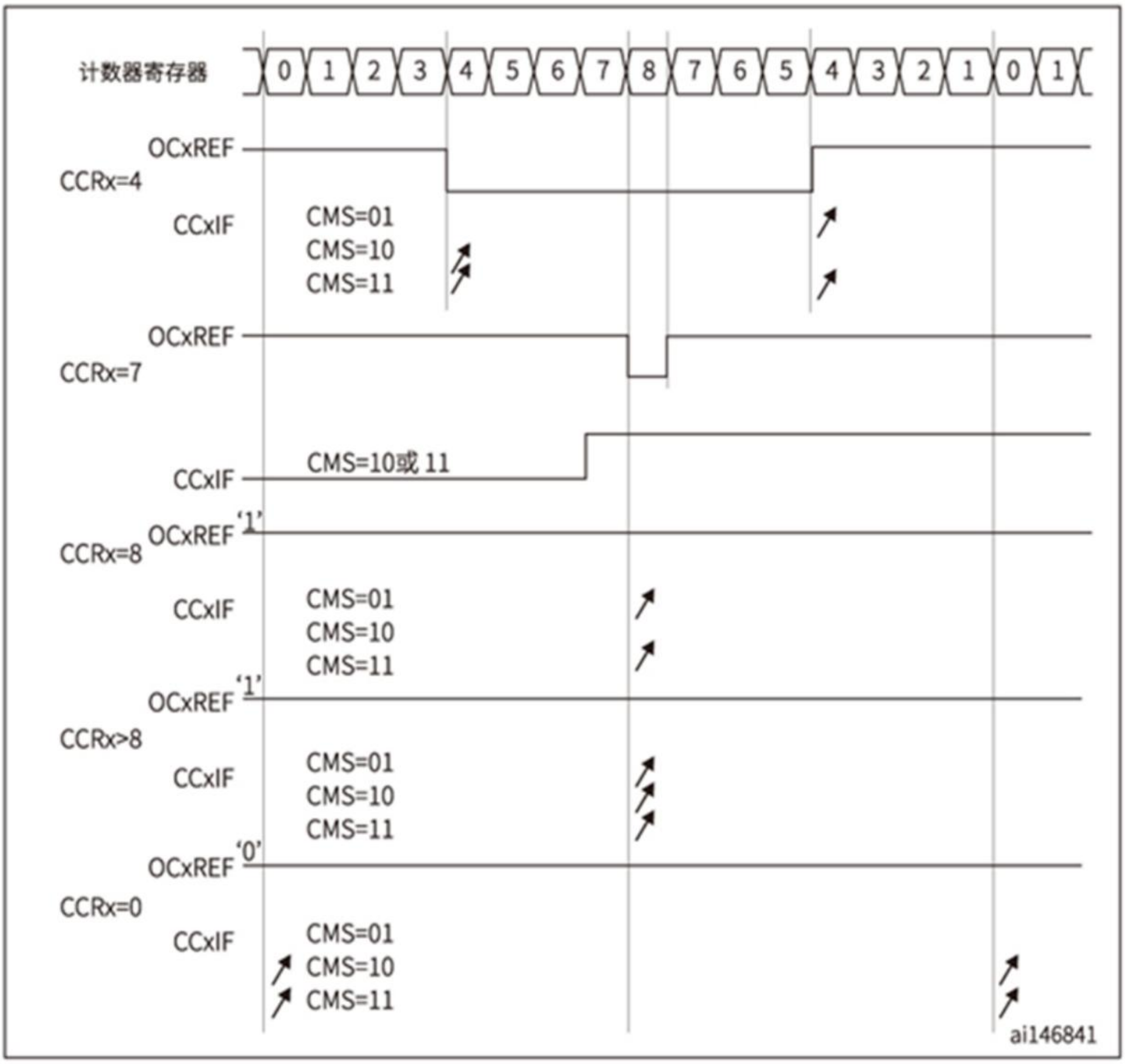


Figure 13-32 Center aligned PWM waveform (APR=8)

13.3.9.2.1 Tips for Using Center Alignment Mode

ÿWhen entering center-aligned mode, the current up/down counting configuration is used;

Depends on the current value of the DIR bit in the TIM2_CR1 register. Also, software cannot modify the DIR and CMS bits at the same time.

ÿ It is not recommended to overwrite counters when running in center-aligned mode, as this can produce unpredictable results. In particular:

- If the value written to the counter is greater than the auto-reload value (TIM2_CNT>TIM2_ARR), the direction will not be updated.

For example, if the counter is counting up, it continues to count up.

- If 0 or the value of TIM2_ARR is written to the counter, the direction is updated, but no update event UEV is generated.

ÿThe safest way to use center-aligned mode is to generate a software update before starting the counter (set

ÿ UG bit in TIM2_EGR bit), do not modify the counter value during counting.

## 13.3.10 Single Pulse Mode

One-Pulse Mode (OPM) is a special case of the aforementioned modes. This mode allows the counter to respond to a stimulus and, after a programmable delay, generate a pulse with a programmable pulse width.

The counter can be started from the mode controller to generate waveforms in output compare mode or PWM mode. Setting the OPM bit in the TIM2_CR1 register selects the single pulse mode, which allows the counter to automatically stop when the next update event UEV is generated.

A pulse is generated only when the comparison value differs from the counter's initial value. Before starting (while the timer is waiting to fire), the

Configure as follows:

Up counting method: CNT < CCRx ÿ ARR (especially, 0 < CCRx), to

Down counting method: CNT > CCRx.

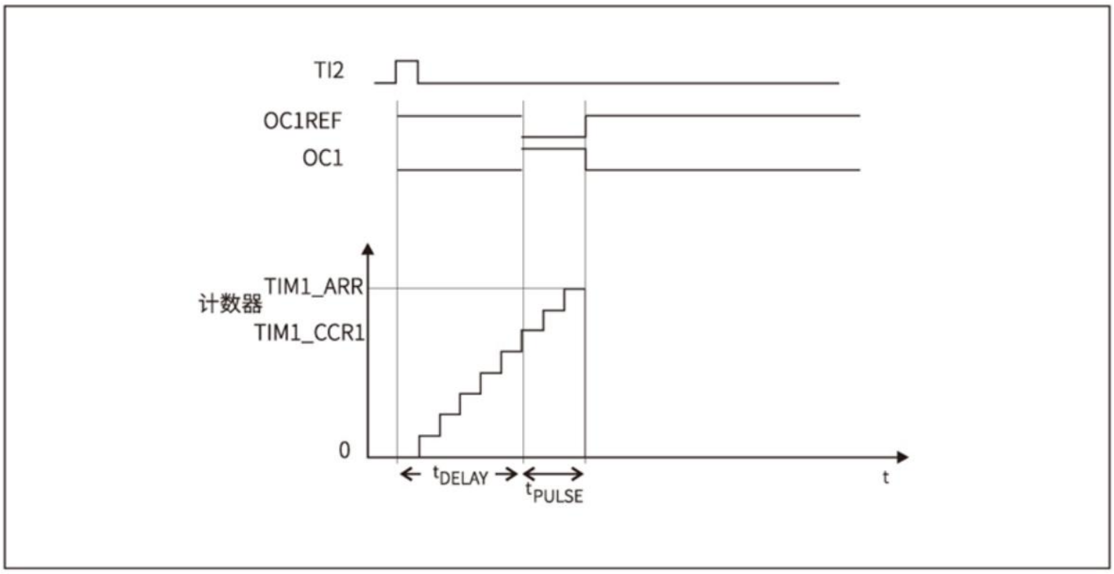


Figure 13-33 Example of single pulse mode

For example, you need to generate a length of

Positive pulse of tPULSE.

Assuming TI2FP2 as trigger 1:

ÿ Set CC2S='01' in the TIM2_CCMR1 register to map TI2FP2 to TI2. ÿ Set CC2P='0' in the TIM2_CCER register to enable TI2FP2 to detect rising edges. ÿSet TS='110' in the TIM2_SMCR register, TI2FP2 acts as the trigger (TRGI) of the slave mode controller. ÿ Set SMS='110' in TIM2_SMCR register (trigger mode), TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the compare register (considering clock frequency and counter prescaler)

ÿ tDELAY is defined by the value written in the TIM2_CCR1 register.

ÿ tPULSE is defined by the difference between the autoload value and the compare value (TIM2_ARR - TIM2_CCR1).

ÿAssume that a waveform from '0' to '1' is to be generated when a compare match occurs, and a waveform is to be generated when the counter reaches the preload value

Waveform from '1' to '0'; first set OC1M='111' in the TIM2_CCMR1 register to enter PWM mode 2; selectively enable the preload register as required: set OC1PE='1' in TIM2_CCMR1 and TIM2_CR1 register

ARPE; then fill in the comparison value in the TIM2_CCR1 register, fill in the autoload value in the TIM2_ARR register,

Modify the UG bit to generate an update event, then wait for an external trigger event on TI2. In this example, CC1P='0'.

In this example, the DIR and CMS bits in the TIM2_CR1 register should be low.

Since only one pulse is required, OPM='1' in the TIM2_CR1 register must be set to stop counting at the next update event (when the counter rolls over from the autoload value to 0).

Special case: OCx fast enable

In Single Pulse mode, the edge detection logic at the TIx input pin sets the CEN bit to start the counter. Then between the counter and the compare value The comparison operation produces an inversion of the output. But these operations require a certain clock cycle, so it limits the minimum delay that can be obtained tDELAY.

If you want to output the waveform with the minimum delay, you can set the OCxFE bit in the TIM2_CCMRx register; at this time, OCxREF (and OCx) are forced to respond to the stimulus and no longer depend on the comparison result, and the output waveform is the same as the waveform when the comparison matches. OCxFE is only available if the channel is configured as It works in PWM1 and PWM2 mode.

### 13.3.11 CLEARING THE OCxREF SIGNAL ON EXTERNAL EVENT

For a given channel, set the corresponding OCxCE bit in the TIM2_CCMRx register to '1' to enable the ETRF input The high level of the OCxREF signal will be pulled low, and the OCxREF signal will remain low until the next update event UEV occurs.

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

For example, the OCxREF signal can be tied to the output of a comparator for current control. At this time, ETR must be configured as follows:

1. The external trigger prescaler must be off: ETPS[1:0]='00' in the TIM2_SMCR register. 2. External clock mode 2 must be disabled: ECE='0' in TIM2_SMCR register.

3. External Trigger Polarity (ETP) and External Trigger Filter (ETF) can be configured as required.

The figure below shows the behavior of the OCxREF signal for different values of OCxCE when the ETRF input goes high. In this example, set Timer TIM2 is placed in PWM mode.

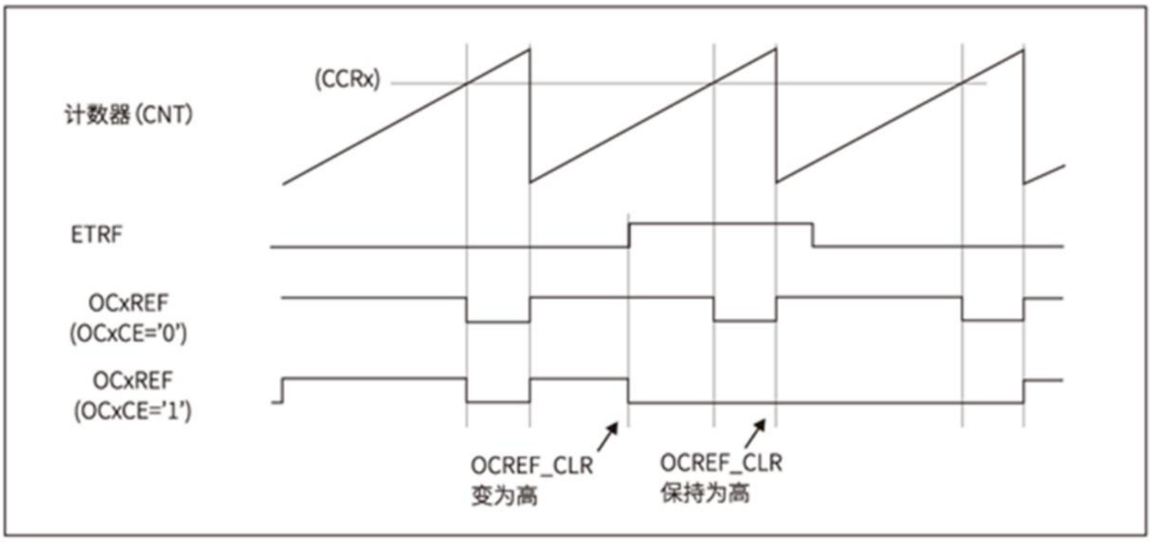


Figure 13-34 Clear OCxREF of TIM2

### 13.3.12 Encoder interface mode

The method to select the encoder interface mode is: if the counter only counts on the edge of TI2, set the TIM2_SMCR register SMS=001; if only counting on the edge of TI1, set SMS=010; if the counter counts on the edge of TI1 and TI2 at the same time, set SMS=011.

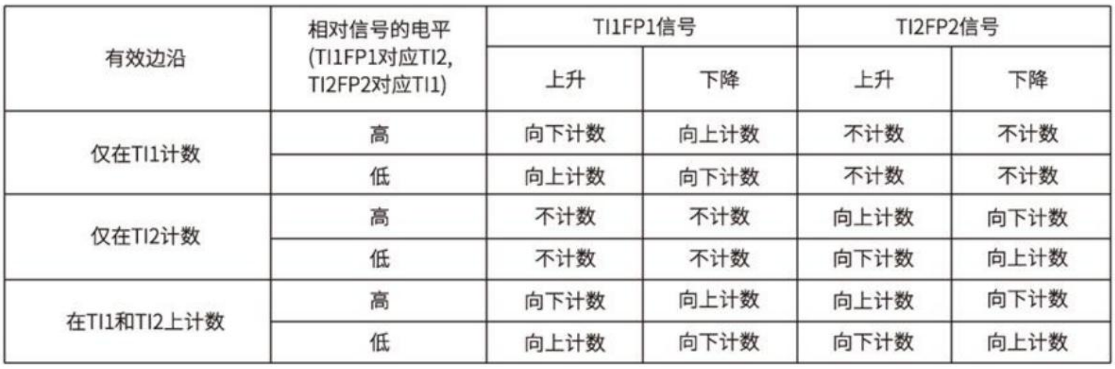
The TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIM2_CCER register; the input filter can also be programmed if required.

Two inputs TI1 and TI2 are used to interface the incremental encoder. Referring to Table 13-1, assuming the counter is enabled (CEN='1' in the TIM2_CR1 register), the counter is driven by every valid transition on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing through the input filter and polarity control; if there is no filter and phase change, then TI1FP1=TI1, TI2FP2=TI2. According to the transition sequence of the two input signals, count pulses and direction signals are generated. Depending on the transition sequence of the two input signals, the counter goes up Count down, and the hardware sets the DIR bit of the TIM2_CR1 register accordingly. Whether the counter is counted by TI1, by TI2 counts or counts against both TI1 and TI2. A transition on either input (TI1 or TI2) recalculates the DIR bit.

Encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter is only between 0 and Continuous count between autoload values of TIM2_ARR register (either 0 to ARR count or ARR to 0 count, depending on direction). Therefore, TIM2_ARR must be configured before counting; also, the capture, comparator, prescaler, trigger output characteristics, etc. are still working as usual.

In this mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so that the content of the counter always indicates the coded value. position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The table below lists all possible combinations, assuming TI1 and TI2 are different time conversion.

Table 13-1 Relationship between counting direction and encoder signal



| 有效边沿 | 相对信号的电平<br>(TI1FP1对应TI2,<br>TI2FP2对应TI1) | TI1FP1信号 | | TI2FP2信号 | |
|---|---|---|---|---|---|
| | | 上升 | 下降 | 上升 | 下降 |
| 仅在TI1计数 | 高 | 向下计数 | 向上计数 | 不计数 | 不计数 |
| | 低 | 向上计数 | 向下计数 | 不计数 | 不计数 |
| 仅在TI2计数 | 高 | 不计数 | 不计数 | 向上计数 | 向下计数 |
| | 低 | 不计数 | 不计数 | 向下计数 | 向上计数 |
| 在TI1和TI2上计数 | 高 | 向下计数 | 向上计数 | 向上计数 | 向下计数 |
| | 低 | 向上计数 | 向下计数 | 向下计数 | 向上计数 |

An external incremental encoder can be directly interfaced with the MCU without external interface logic. However, it is common to use a comparator to convert The differential output of the coder is converted to a digital signal, which greatly increases the ability to resist noise interference. The third signal output by the encoder indicates mechanical zero, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing count signal generation and direction control. It also shows that when both edges are selected, How input jitter is suppressed; jitter can occur when the sensor is positioned close to a transition point.

In this example, we assume the following configuration:

ÿ CC1S='01' (TIM2_CCMR1 register, IC1FP1 is mapped to TI1)

ÿ CC2S='01' (TIM2_CCMR2 register, IC2FP2 is mapped to TI2)

ÿ CC1P='0' (TIM2_CCER register, IC1FP1 is not inverted, IC1FP1=TI1)

ÿ CC2P='0' (TIM2_CCER register, IC2FP2 is not inverted, IC2FP2=TI2)

ÿ SMS='011' (TIM2_SMCR register, all inputs are valid on rising and falling edges)

ÿ CEN='1' (TIM2_CR1 register, counter enable)

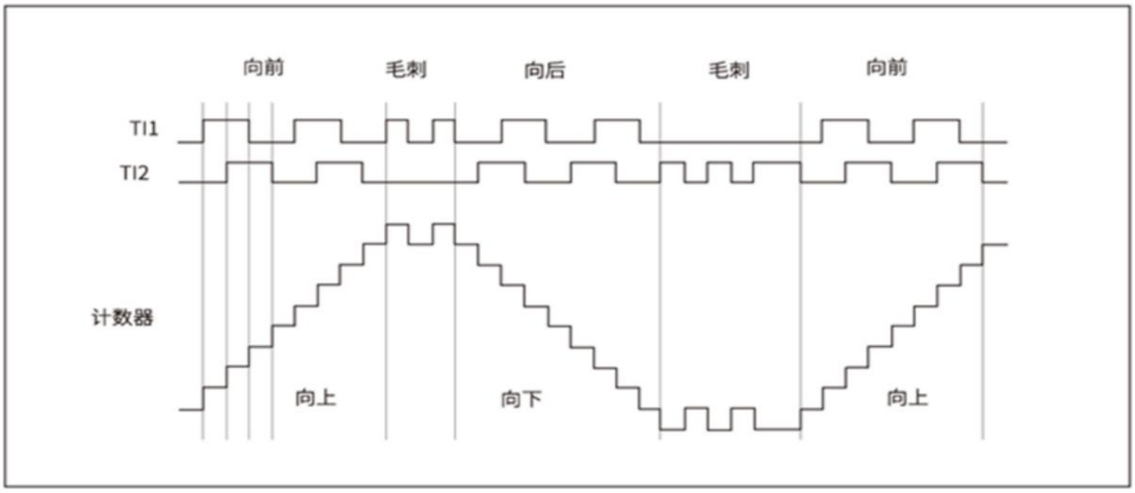


Figure 13-35 Example of counter operation in encoder mode

The figure below is an example of the operation of the counter when the polarity of IC1FP1 is reversed (CC1P='1', other configurations are the same as the above example)

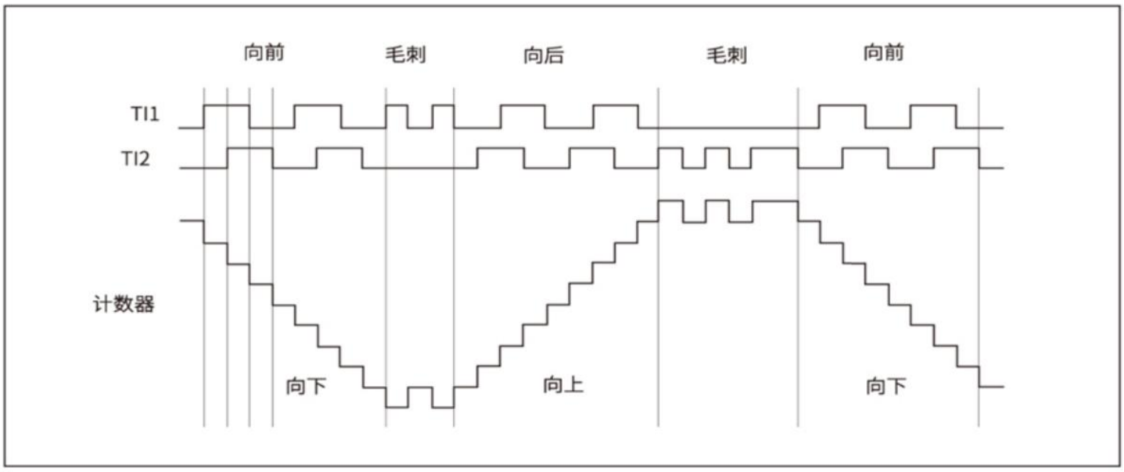


Figure 13-36 IC1FP1 Inverted Encoder Interface Mode Example

When the timer is configured in encoder interface mode, it provides information on the current position of the sensor. Timing in capture mode using the second configuration

The encoder can measure the interval between two encoder events and obtain dynamic information (velocity, acceleration, deceleration). Code indicating mechanical zero point

The encoder output can be used for this purpose. Depending on the interval between two events, the counter can be read out at a fixed time. if possible

If so, you can latch the counter value into a third input capture register (the capture signal must be periodic and can be determined by another

generated by the timer).

### 13.3.13 Timer input XOR function

The TI1S bit in the TIM2_CR2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, and the output of the XOR gate 3

The first inputs are TIM2_CH1, TIM2_CH2 and TIM2_CH3.

The XOR output can be used for all timer input functions such as trigger or input capture.

### 13.3.14 Synchronization of Timer and External Trigger

The TIM2 timer can be synchronized to an external trigger in several modes: reset mode, gated mode and trigger mode.

13.3.14.1 Slave Mode: Reset Mode

When a trigger input event occurs, the counter and its prescaler can be re-initialized; at the same time, if the URS bit of the TIM2_CR1 register is low, an update event UEV will be generated; then all preload registers (TIM2_ARR, TIM2_CCRx) will be updated.

In the following example, a rising edge at the TI1 input causes the up-counter to be cleared:

ÿ Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is needed, so keep IC1F=0000). The capture prescaler is not used in trigger operation, so there is no need to configure it. The CC1S bit only selects the input capture source, that is, CC1S=01 in the TIM2_CCMR1 register. Set CC1P=0 in TIM2_CCER register to determine the polarity (detect rising edge only).

ÿSet SMS=100 in the TIM2_SMCR register, configure the timer to reset mode; set it in the TIM2_SMCR register TS=101, select TI1 as the input source.

ÿ Set CEN=1 in the TIM2_CR1 register to start the counter.

The counter starts counting according to the internal clock, and then runs normally until a rising edge occurs on TI1; at this time, the counter is cleared and restarts counting from 0. At the same time, the trigger flag (TIF bit in the TIM2_SR register) is set, and an interrupt request is generated according to the TIE (interrupt enable) bit in the TIM2_DIER register. The

figure below shows the action when the auto-reload register TIM2_ARR=0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the TI1 input.

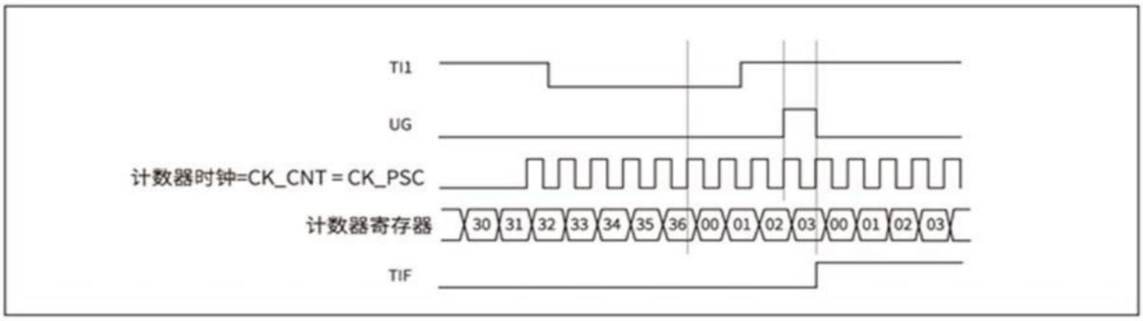


Figure 13-37 Control circuit in reset mode

13.3.14.2 Slave Mode: Gated Mode

Enables the counter according to the selected input level.

In the following example, the counter counts up only when TI1 is low:

ÿ Configure channel 1 to detect low level on TI1. Configure the input filter bandwidth (in this example, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, set CC1S=01 in the TIM2_CCMR1 register. Set CC1P=1 in the TIM2_CCER register to determine the polarity (only detect low level).

ÿSet SMS=101 in the TIM2_SMCR register, configure the timer as gated mode; set in the TIM2_SMCR register TS=101, select TI1 as the input source.

ÿ Set CEN=1 in the TIM2_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot be started active regardless of the trigger input level.

As long as TI1 is low, the counter starts counting according to the internal clock, and stops counting when TI1 goes high. When the counter starts or stops both Sets TIF flags in TIM2_SR.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the TI1 input.

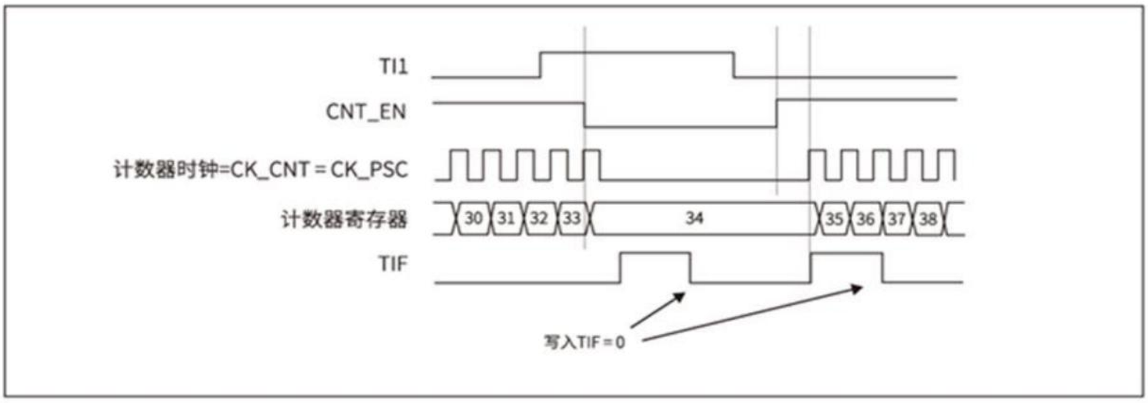


Figure 13-38 Control circuit in gated mode

13.3.14.3 Slave Mode: Trigger Mode

A selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

ÿ Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keep IC2F=0000). The capture prescaler is not used in trigger operation and does not require configuration. The CC2S bit is only used to select the input capture source, set CC2S=01 in the TIM2_CCMR1 register. Set CC2P=1 in the TIM2_CCER register to determine the polarity (only detect low level).

ÿSet SMS=110 in the TIM2_SMCR register, configure the timer as trigger mode; set in the TIM2_SMCR register TS=110, select TI2 as the input source.

When a rising edge occurs on TI2, the counter starts counting driven by the internal clock and sets the TIF flag at the same time.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

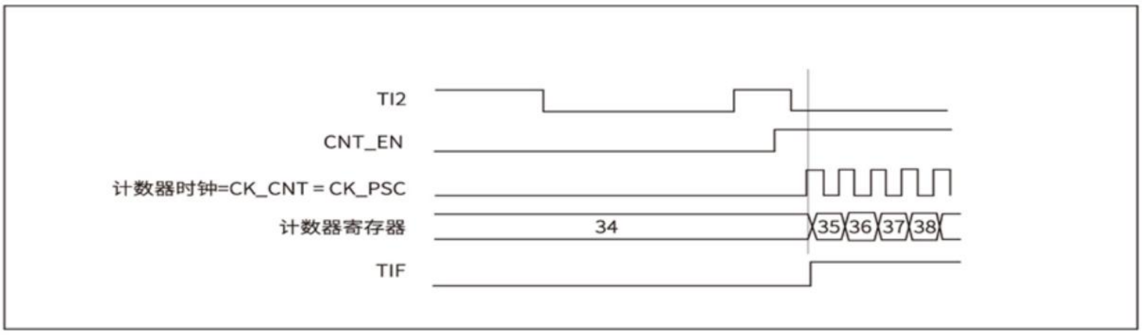


Figure 13-39 Control circuit in trigger mode

13.3.14.4 Slave Mode: External Clock Mode 2 + Trigger Mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). At this time, the ETR signal is Used as an input for an external clock, another input can be selected as a trigger input in reset mode, gated mode or trigger mode. do not build It is recommended to use the TS bit of the TIM2_SMCR register to select ETR as TRGI.

In the following example, after a rising edge on TI1, the counter counts up once on every rising edge of ETR:

1. Configure the external trigger input circuit through the TIM2_SMCR register:

   - ETF=0000: no filtering

   - ETPS=00: no prescaler

   - ETP=0: detect rising edge of ETR, set ECE=1 to enable external clock mode 2

2. Configure channel 1 as follows to detect the rising edge of TI:

   - IC1F=0000: no filtering

   - Capture prescaler is not used in trigger operation, no configuration required

   - Set CC1S=01 in the TIM2_CCMR1 register to select the input capture source - Set

   CC1P=0 in the TIM2_CCER register to determine the polarity (only detect rising edges)

3. Set SMS=110 in the TIM2_SMCR register to configure the timer as trigger mode. Set in TIM2_SMCR register

   TS=101, select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR.

The delay between the rising edge of the ETR signal and the actual reset of the counter depends on the resynchronization circuitry at the ETRP input.

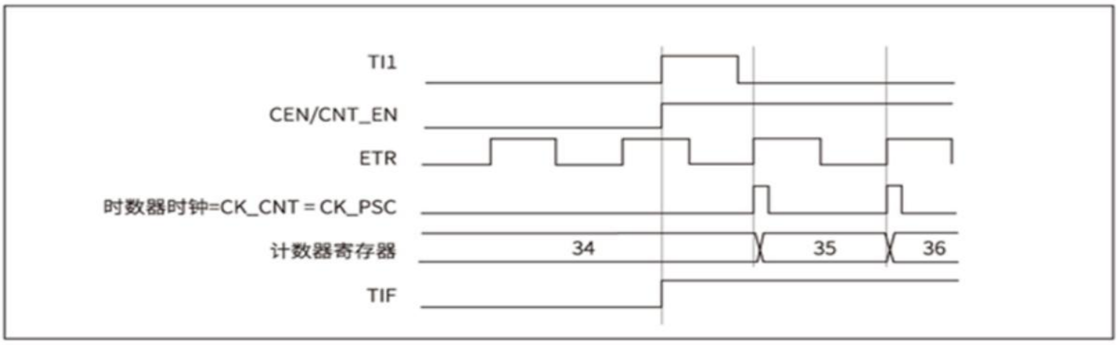


Figure 13-40 Control circuit in external clock mode 2 + trigger mode

## 13.3.15 Timer Synchronization

All TIM2 timers are connected internally for timer synchronization or chaining. When a timer is in master mode, it can reset, start, stop, or provide clocks to the counters of a timer in slave mode.

The figure below shows an overview of the trigger selection and master mode selection blocks.

13.3.15.1 USING ONE TIMER AS A PRESCALER FOR ANOTHER TIMER

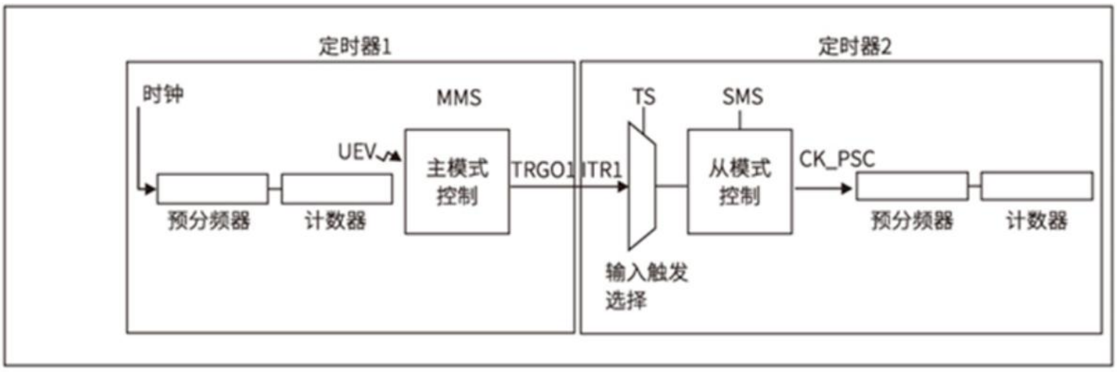


Figure 13-41 Example of a master/slave timer

For example: Timer 1 can be configured as the prescaler of Timer 2. Referring to Figure 13-41, perform the following operations:

ÿConfigure Timer 1 as master mode, it can output a periodic trigger signal every update event UEV. When MMS='010' of the TIM1_CR2 register, a rising edge is output on TRGO1 whenever an update event is generated

Signal.

ÿConnect TRGO1 output of timer 1 to timer 2, set TS='000' of TIM2_SMCR register, configure timer

2 is slave mode using ITR1 as internal trigger.

ÿThen put the slave mode controller in external clock mode 1 (SMS=111 in the TIM2_SMCR register); so that timer 2 can

Driven by the periodic rising edge of Timer 1 (that is, the timer 1 counter overflows) signal.

ÿFinally , the corresponding CEN bits (TIM2_CR1 register) must be set to start the two timers respectively.

Note: If OCx has been selected as the trigger output of Timer 1 (MMS=1xx), its rising edge is used to drive the counter of Timer 2.

13.3.15.2 Using One Timer to Enable Another Timer

In this example, the enable of Timer 2 is controlled by the output compare of Timer 1. Refer to Figure 13-41 for connections. Only when Timer 1's When OC1REF is high, Timer 2 will count the divided internal clock. The clock frequency of both timers is determined by the prescaler pair CK_INT is divided by 3 (fCK_CNT=fCK_INT/3).

ÿConfigure Timer 1 as master mode, send its output comparison reference signal (OC1REF) as trigger output (TIM1_CR2 register

MMS=100)

ÿConfigure Timer 1’s OC1REF waveform (TIM1_CCMR1 register) ÿConfigure Timer 2

to get input trigger from Timer 1 (TS=000 in TIM2_SMCR register) ÿConfigure Timer 2 in gated mode (SMS=101

in TIM2_SMCR register)

ÿSet CEN=1 in TIM2_CR1 register to enable Timer 2

ÿSet CEN=1 in TIM1_CR1 register to start Timer 1

Note: The clock of Timer 2 is not synchronized with the clock of Timer 1, this mode only affects the enable signal of Timer 2 counter.

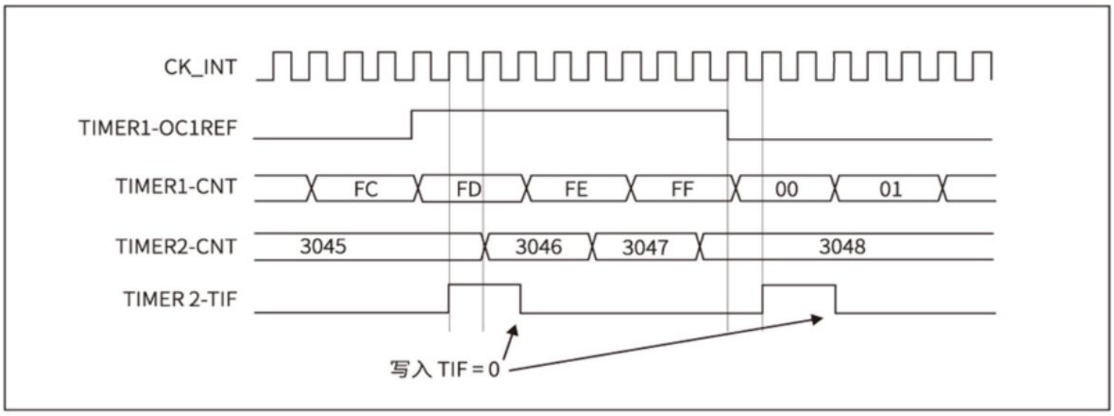


Figure 13-42 OC1REF of Timer 1 controls Timer 2

In the example in Figure 13-42, their counters and prescalers are not initialized before Timer 2 starts, so they start from the current value to start counting. It is possible to reset the 2 timers before starting timer 1 so that they start from a given value, i.e. Write any desired value into the counter. The timer can be reset by writing to the UG bit of the TIM2_EGR register.

In the next example, Timer1 and Timer2 need to be synchronized. Timer 1 is master mode and starts from 0, Timer 2 is slave mode and starts from 0xE7 start; both timers have the same prescaler coefficient. Writing '0' to CEN bit of TIM1_CR1 will disable Timer 1, Timer 2 stops immediately.

ÿConfigure Timer 1 as master mode, send out output compare 1 reference signal (OC1REF) as trigger output (TIM1_CR2 register MMS=100).

ÿ Configure the OC1REF waveform of Timer 1 (TIM1_CCMR1 register).

ÿConfigure Timer 2 to get input trigger from Timer 1 (TS=000 in TIM2_SMCR register) ÿConfigure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register)

ÿ Set UG='1' in the TIM1_EGR register to reset Timer 1.

ÿSet UG='1' in TIM2_EGR register to reset Timer 2.

ÿ Write '0xE7' to Timer 2 counter (TIM2_CNTL), initialize it to 0xE7.

ÿ Set CEN='1' in TIM2_CR1 register to enable Timer 2.

ÿSet CEN='1' in TIM1_CR1 register to start Timer 1.

ÿ Set CEN='0' in TIM1_CR1 register to stop Timer 1.

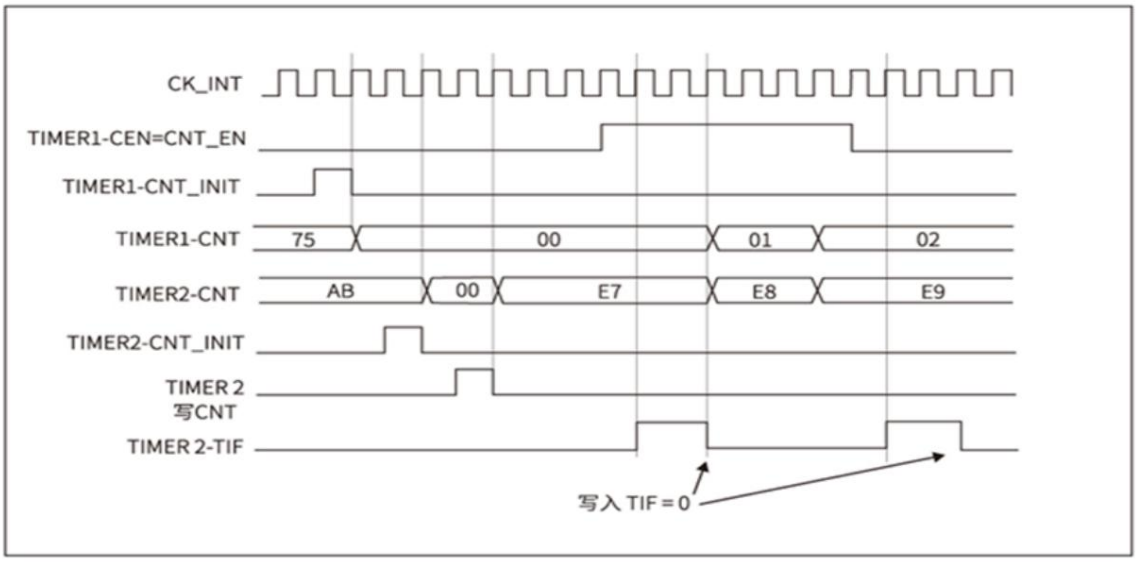


Figure 13-43 Timer 2 can be controlled by enabling Timer 1

13.3.15.3 Using a timer to start another timer

In this example, Timer 2 is enabled with an update event from Timer 1. Refer to Figure 13-41 for connections. Once Timer 1 generates more

For a new event, Timer 2 starts counting from its current value (can be non-zero) according to the frequency-divided internal clock. When a trigger signal is received,

The CEN bit of Timer 2 is automatically set to '1' and the counter starts counting until writing '0' to the CEN bit of the TIM2_CR1 register.

The clock frequency of both timers is divided by the prescaler pair CK_INT by 3 (fCK_CNT=fCK_INT/3).

ÿConfigure timer 1 as master mode, send its update event (UEV) as trigger output (TIM1_CR2 register

MMS=010).

ÿ Configure the period of Timer 1 (TIM1_ARR register).

ÿ Configure Timer 2 to get input trigger from Timer 1 (TS=000 in TIM2_SMCR register)

ÿ Configure Timer 2 as trigger mode (SMS=110 in TIM2_SMCR register)

ÿ Set CEN=1 in TIM1_CR1 register to start Timer 1.

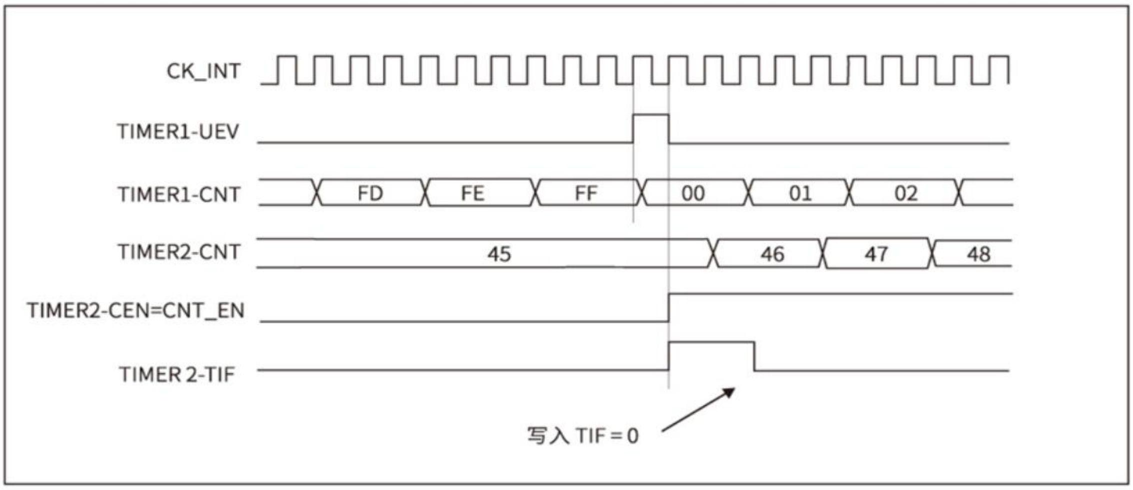


Figure 13-44 Using the update of timer 1 to trigger timer 2

In the previous example, two counters could be initialized before counting was started. Figure 13-45 shows that under the same configuration, using the touch

Transmit mode instead of gated mode (SMS=110 in TIM2_SMCR register).

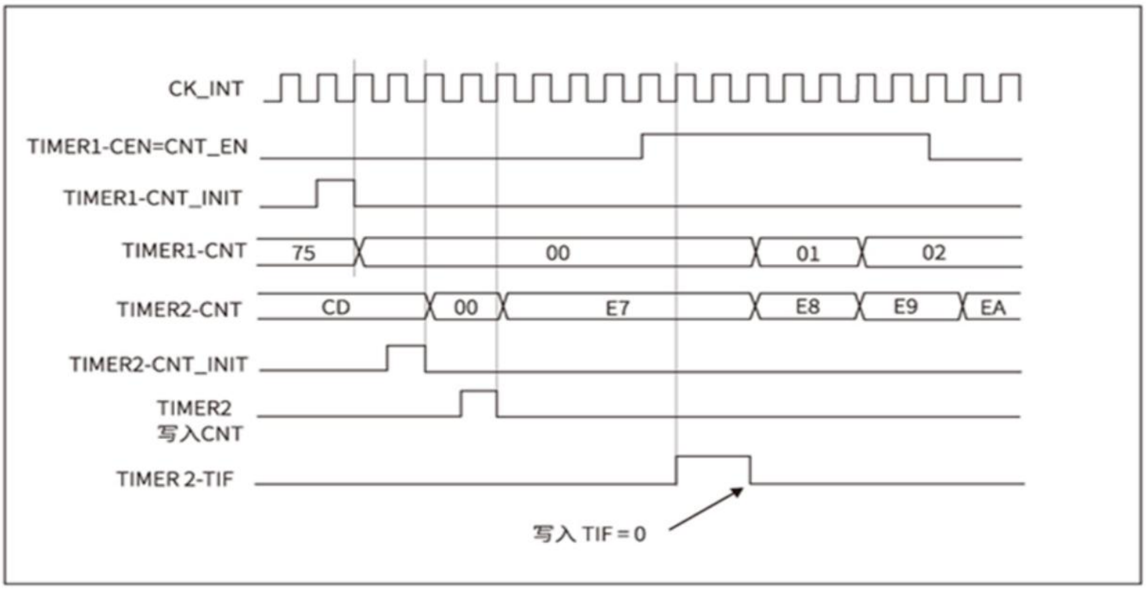


Figure 13-45 Timer 2 is triggered by enabling Timer 1

### 13.3.15.4 Using One Timer as a Prescaler for Another

This example uses Timer1 as the prescaler for Timer2. Referring to the connection in Figure 13-41, the configuration is as follows:

ÿ Configure Timer 1 as master mode, send its update event UEV as trigger output (TIM1_CR2 register

MMS='010'). Then output a periodic signal every time the counter overflows.

ÿ Configure the period of Timer 1 (TIM1_ARR register). ÿ

Configure Timer 2 to get input trigger from Timer 1 (TS=000 in TIM2_SMCR register) ÿ Configure

Timer 2 to use external clock mode (SMS=111 in TIM2_SMCR register)

ÿSet CEN=1 in TIM1_CR2 register to start Timer 2.

ÿ Set CEN=1 in TIM1_CR1 register to start Timer 1.

### 13.3.15.5 Starting 2 Timers Synchronously Using an External Trigger

In this example, Timer 1 is enabled when the TI1 input of Timer 1 rises, and Timer 2 is enabled while Timer 1 is enabled, see Figure 13-

41. In order to ensure the alignment of the counter, Timer 1 must be configured in master/slave mode (corresponding to TI1 as slave and corresponding to Timer 2 as master):

ÿConfigure timer 1 as master mode, send its enable as trigger output (MMS='001' in TIM1_CR2 register) ÿConfigure timer 1 as

slave mode, get input trigger from TI1 (TS='100 in TIM1_SMCR register ') ÿ Configure Timer 1 as trigger mode

(SMS='110' in TIM1_SMCR register) ÿ Configure Timer 1 in master/slave mode, MSM in

TIM1_SMCR register='1' ÿ Configure Timer 2 to get input from Timer 1 Trigger (TS=000

in TIM2_SMCR register) ÿConfigure timer 2 as trigger mode (SMS='110' in TIM2_SMCR register)

When a rising edge appears on TI1 of timer 1, the two timers start counting synchronously according to the internal clock, and the two TIF flags also is set.

Note: In this example, both timers are initialized (set the corresponding UG bit) before starting, both counters start from 0, but can be

To insert an offset between timers by writing to any of the counter registers (TIM2_CNT). In the figure below, you can see the master/slave mode

There is a delay between CNT_EN and CK_PSC of timer 1.

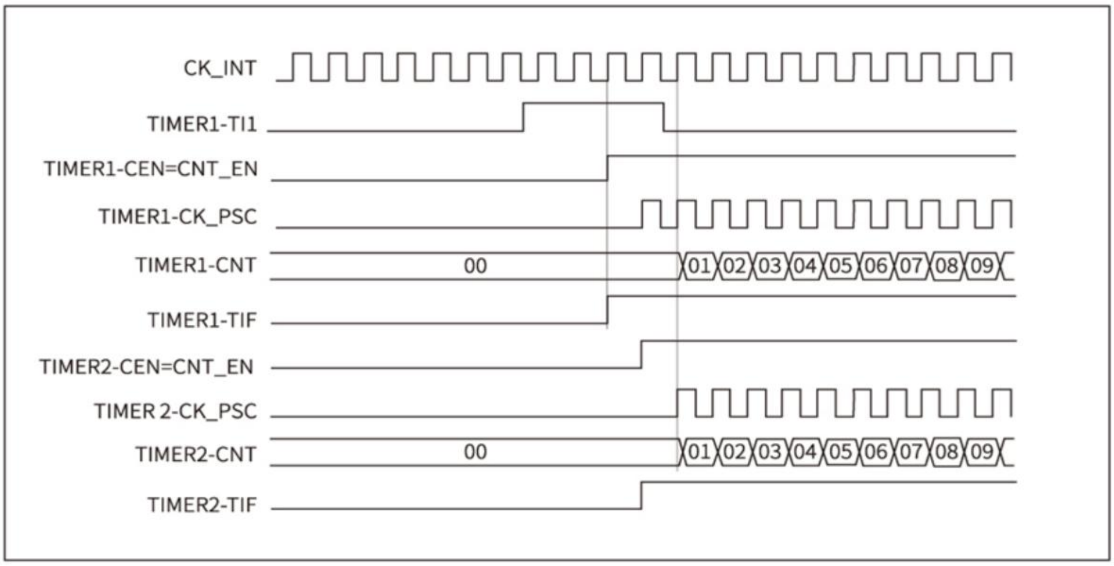


Figure 13-46 Use the TI1 input of timer 1 to trigger timer 1 and timer 2

## 13.3.16 Debug mode

When the microcontroller enters the debug mode (Cortex-M0+ core stops), according to the setting of DBG_TIM2_STOP in the DBG module, the TIM2 timer

The counter either continues to operate normally, or stops.

## 13.4 TIM2 register list

These peripheral registers can be manipulated in words (32 bits).

TIM2 base address 0x4000 3C00

Table 13-2 TIM2 register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | TIM2_CR1 | TIM2 Control Register 1 | 0x0000 0000 |
| 0x04 | TIM2_CR2 | TIM2 Control Register 2 | 0x0000 0000 |
| 0x08 | TIM2_SMCR | TIM2 slave mode control register | 0x0000 0000 |
| 0x0C | TIM2_DIER | TIM2 Interrupt Enable Register | 0x0000 0000 |
| 0x10 | TIM2_SR | TIM2 status register | 0x0000 0000 |
| 0x14 | TIM2_EGR | TIM2 event generation register | 0x0000 0000 |
| 0x18 | TIM2_CCMR1 | TIM2 capture/compare mode register 1 | 0x0000 0000 |
| 0x1C | TIM2_CCMR2 | TIM2 Capture/Compare Mode Register 2 | 0x0000 0000 |
| 0x20 | TIM2_CCER | TIM2 capture/compare enable register | 0x0000 0000 |
| 0x24 | TIM2_CNT | TIM2 counter | 0x0000 0000 |
| 0x28 | TIM2_PSC | TIM2 prescaler | 0x0000 0000 |
| 0x2C | TIM2_ARR | TIM2 auto-reload register | 0x0000 0000 |
| 0x30 Reserved | | | |
| 0x34 | TIM2_CCR1 | TIM2 capture/compare register 1 | 0x0000 0000 |
| 0x38 | TIM2_CCR2 | TIM2 capture/compare register 2 | 0x0000 0000 |
| 0x3C | TIM2_CCR3 | TIM2 capture/compare register 3 | 0x0000 0000 |
| 0x40 | TIM2_CCR4 | TIM2 capture/compare register 4 | 0x0000 0000 |

# 13.5 TIM2 register description

## 13.5.1 TIM2 Control Register 1 (TIM2_CR1)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | CKD[1:0] | | ARP E. | CMS[1:0] | | DIR | OPM | URS | UDIS | CEN |
| | | | | | | R/W | | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:10 | - | Reserved, always reads as 0. | 0 | - |
| 9:8 | CKD[1:0] | Clock division factor (Clock division)<br>These 2 bits define the timer clock (CK_INT) frequency, dead time and<br>The frequency division ratio between the sampling clocks used by the oscilloscopes (ETR, TIx).<br>00: tDTS = tCK_INT<br>01: tDTS = 2 x tCK_INT<br>10: tDTS = 4 x tCK_INT<br>11: Reserved, do not use this configuration | 0 | R/W |
| 7 | ARPE | Auto-reload preload enable bit (Auto-reload preload enable)<br>0: TIM2_ARR register is not buffered;<br>1: TIM2_ARR register is loaded into buffer. | 0 | R/W |
| 6:5 | CMS[1:0] | Select Center-aligned mode selection<br>00: Edge-aligned mode. The counter counts up or down according to the direction bit (DIR).<br>01: center alignment mode 1. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM2_CCMRx register) output comparison interrupt flag bit, only when the counter sends Set when counting down.<br>10: Center alignment mode 2. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM2_CCMRx register) output comparison interrupt flag bit, only when the counter sends Set when counting up.<br>11: Center alignment mode 3. The counter alternately counts up and down. Channel configured as output (CCxS=00 in the TIM2_CCMRx register) output comparison interrupt flag bit, when the counter is up Both are set when counting down.<br>Note: Transition from edge-aligned mode to center-aligned mode is not allowed while the counter is on (CEN=1). Mode. | 0 | R/W |
| 4 | DIR | Direction<br>0: The counter counts up;<br>1: The counter counts down.<br>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode. | 0 | R/W |
| 3 | OPM | Single pulse mode (One pulse mode)<br>0: the counter does not stop when an update event occurs;<br>1: The counter stops at the next update event (clearing the CEN bit). | 0 | R/W |
| 2 | URS | Update request source<br>Software selects the source of UEV events through this bit<br>0: If the update interrupt is enabled, any of the following events will generate an update interrupt:<br>– Counter overflow/underflow<br>– set the UG bit<br>– Updates generated from the mode controller<br>1: If update interrupt is enabled, only counter overflow/underflow will generate update interrupt. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 1 | UDIS | Update disable<br>The software enables/disables the generation of UEV events through this bit<br>0: Allow UEVs. Update (UEV) events are generated by any of the following events:<br>– Counter overflow/underflow<br>– set the UG bit<br>– Updates generated from the mode controller<br>Registers with cache are loaded with their preloaded values. (Annotation: Update shadow registers)<br>1: Disable UEVs. Update events are not generated, shadow registers (ARR, PSC, CCRx) keep their their value. If the UG bit is set or a hardware reset is issued from the mode controller, the count register and prescaler are reinitialized. | 0 | R/W |
| 0 | CEN | Counter enable<br>0: disable counter;<br>1: Enable counter.<br>Note: The external clock, gated mode and encoder mode can only work after the CEN bit is set by software. do. Trigger mode can automatically set the CEN bit by hardware. | 0 | R/W |

### 13.5.2 TIM2 Control Register 2 (TIM2_CR2)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | OIS4 | OIS3N | OIS3 | OIS2N | OIS2 | OIS1N | OIS1 | TI1S | MMS[1:0] | | | reserve | CCUS | reserve | CCPC |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | | | R/W | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:15 | - | Reserved, always reads as 0. | 0x0 | - |
| 14 | OIS4 | Output idle state 4 (OC4 output). See OIS1 bit. | 0 | R/W |
| 13 | OIS3N | Output idle state 3 (OC3N output). See OIS1N bit. | 0 | R/W |
| 12 | OIS3 | Output idle state 3 (OC3 output). See OIS1 bit. | 0 | R/W |
| 11 | OIS2N | Output idle state 2 (OC2N output). See OIS1N bit. | 0 | R/W |
| 10 | OIS2 | Output idle state 2 (OC2 output). See OIS1 bit. | 0 | R/W |
| 9 | OIS1N | Output idle state 1 (OC1N output) (Output Idle state 1)<br>0: When MOE=0, OC1N=0 after dead zone;<br>1: When MOE=0, OC1N=1 after dead time.<br>Note: After LOCK (TIM2_BKR register) level 1, 2 or 3 has been set, this bit cannot be Revise. | 0 | R/W |
| 8 | OIS1 | Output idle state 1 (OC1 output) (Output Idle state 1)<br>0: When MOE=0, if OC1N is realized, OC1=0 after dead zone;<br>1: When MOE=0, if OC1N is realized, then OC1=1 after dead time.<br>Note: After LOCK (TIM2_BKR register) level 1, 2 or 3 has been set, this bit is not can be modified. | 0 | R/W |
| 7 | TI1S | TI1 selection (TI1 selection)<br>0: TIM2_CH1 pin is connected to TI1 input;<br>1: TIM2_CH1, TIM2_CH2 and TIM2_CH3 pins are connected to TI1 input after XOR. | 0 | R/W |
| 6:4 | MMS[2:0] | Master mode selection<br>These 3 bits are used to select the synchronization message (TRGO) sent to the slave timer in master mode. possible group The combination is as follows: 000: Reset – The UG bit of the TIM2_EGR register is used as a trigger output (TRGO). If the reset is generated by a trigger input (slave mode controller is in reset mode), then The signal on TRGO has a delay relative to the actual reset.<br>001: Enable – The counter enable signal CNT_EN is used as trigger output (TRGO). have When it is necessary to start multiple timers at the same time or control to enable slave timers within a period of time device. The counter enable signal is through the CEN control bit and the trigger input signal in gated mode The logical OR generation of . When the counter enable signal is controlled by the trigger input, there will be a delay, unless master/slave mode is selected (see description of MSM bit in TIM2_SMCR register) described).<br>010: Update – The update event is selected as trigger input (TRGO). For example, a master timer The clock can be used as a prescaler for the slave timer.<br>011: Compare Pulse – When a capture or a compare is successful, when the CC1IF flag is to be set When the flag is active (even if it is already high), the trigger output sends a positive pulse (TRGO).<br>100: Compare – OC1REF signal is used as trigger output (TRGO).<br>101: Compare – OC2REF signal is used as trigger output (TRGO).<br>110: Compare - OC3REF signal is used as trigger output (TRGO). | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 111: Compare - OC4REF signal is used as trigger output (TRGO). | | |
| 3 | - | Reserved, always reads as 0. | 0x0 | - |
| 2 | CCUS | Capture/compare control update selection<br>0: If the capture/compare control bit is preloaded (CCPC=1), it can only be changed by setting the COM bit new them;<br>1: If the capture/compare control bit is preloaded (CCPC=1), it can be set by setting the COM bit or a rising edge on TRGI to update them.<br>Note: This bit only works on channels with complementary outputs. | 0 | R/W |
| 1 | - | Reserved, always reads as 0. | 0 | - |
| 0 | CCPC | Capture/compare preloaded control<br>0: CCxE, CCxNE and OCxM bits are not preloaded;<br>1: The CCxE, CCxNE and OCxM bits are preloaded; when this bit is set, they only is updated after setting the COM bit.<br>Note: This bit only works on channels with complementary outputs. | 0 | R/W |

## 13.5.3 TIM2 Slave Mode Control Register (TIM2_SMCR)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETP | ECE | ETPS[1:0] | | ETF[3:0] | | | | MSM | TS[2:0] | | | reserve | SMS[2:0] | | |
| R/W | R/W | R/W | | R/W | | | | R/W | R/W | | | | R/W | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0 | - |
| 15 | ETP | External trigger polarity (External trigger polarity) This bit selects whether to use ETR or ETR inversion to act as a trigger operation<br>0: ETR is not inverted, high level or rising edge is valid;<br>1: ETR is inverted, active low or falling edge. | 0 | R/W |
| 14 | ECE | External clock enable bit (External clock enable)<br>This bit enables external clock mode 2<br>0: disable external clock mode 2;<br>1: Enable External Clock Mode 2. The counter is driven by any active edge on the ETRF signal.<br>Note 1: Setting the ECE bit is the same as selecting the external clock mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111) have the same effect.<br>Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gated mode and triggered mode; however, TRGI cannot be connected to ETRF (TS bit cannot be '111').<br>Note 3: When external clock mode 1 and external clock mode 2 are enabled at the same time, the output of the external clock The entry is ETRF. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| 13:12 | ETPS[1:0] | External trigger prescaler<br>The frequency of the external trigger signal ETRP must be at most 1/4 of the frequency of TIM2CLK. When the input compares<br>When using a fast external clock, you can use prescaler to reduce the frequency of ETRP.<br>00: close the prescaler;<br>01: ETRP frequency divided by 2;<br>10: ETRP frequency divided by 4;<br>11: ETRP frequency divided by 8. | 0 | R/W |
| 11:8 | ETF[3:0] | External trigger filter<br>These bits define the frequency at which the ETRP signal is sampled and the bandwidth of the ETRP digital filter. Reality<br>In effect, the digital filter is an event counter that registers N events and generates a<br>output transitions.<br>0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6<br>0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8<br>0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5<br>0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6<br>0100: sampling frequency fSAMPLING=fDTS/2, N=6 1100: sampling frequency fSAMPLING= fDTS /16,<br>N=8<br>0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING= fDTS /32,<br>N=5<br>0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS /32, N=6<br>0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING= fDTS /32, N=8 | 0x0 | R/W |
| 7 | MSM | Master/slave mode (Master/slave mode)<br>0: no effect;<br>1: Events on the trigger input (TRGI) are delayed to allow<br>TRGO) with perfect synchronization between its slave timers. This pair requires synchronizing several timers into one<br>Useful when a single external event is present. | 0 | R/W |
| 6:4 | TS[2:0] | Trigger selection<br>These 3 bits select the trigger input for the synchronous counter.<br>000: Internal Trigger 0 (ITR0) 100: TI1 Edge Detector (TI1F_ED)<br>001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1)<br>010: Internal Trigger 2 (ITR2) 110: Filtered Timer Input 2 (TI2FP2)<br>011: Internal trigger 3 (ITR3) 111: External trigger input (ETRF)<br>See Table 12-1 for more details on ITRx.<br>Note: These bits can only be changed when they are not used (such as SMS=000) to avoid<br>False edge detection. | 0x0 | R/W |
| 3 | - | Reserved, always reads | 0 | - |
| 2:0 | SMS[2:0] | as 0. Slave mode selection<br>When the external signal is selected, the active edge of the trigger signal (TRGI) and the selected external input polarity<br>related (see description of Input Control Register and Control Register)<br>000: Disable Slave Mode – If CEN=1, the prescaler is directly driven by the internal clock.<br>001: Encoder Mode 1 – Counter clocks out on edge of TI2FP2 according to the level of TI1FP1<br>Up/down counting.<br>010: Encoder Mode 2 – The counter is clocked on the edge of TI1FP1 according to the level of TI2FP2.<br>Up/down counting.<br>011: Encoder Mode 3 - Depending on the input level of another signal, the counter is on TI1FP1<br>and TI2FP2 edge count up/down.<br>100: Reset Mode – A rising edge on the selected trigger input (TRGI) re-initializes the counter and<br>And generate a signal to update the register.<br>101: Gated Mode – When the trigger input (TRGI) is high, the counter's clock is on. Once touched<br>If the send input goes low, the counter is stopped (but not reset). The start and stop of the counter are controlled<br>of.<br>110: Trigger mode – the counter is started (but not reset) on a rising edge at the trigger input TRGI, only<br>Startup with counters is controlled.<br>111: External Clock Mode 1 - The rising edge of the selected trigger input (TRGI) drives the counter.<br>Note: Do not use gated mode if TI1F_EN is selected as trigger input (TS=100). This is<br>Because, TI1F_ED outputs a pulse every time TI1F changes, while the gated mode is<br>To check the level of the trigger input. | 0 | R/W |

Table 13-3 TIM2 internal trigger connection

| from the timer | ITR0(TS=000) | ITR1(TS=001) | ITR2(TS=010) | ITR3(TS=011) |
|---|---|---|---|---|
| TIM2 | tim1_trgo | irq_timer10 | irq_timer11 | irq_pca |

Note: If there is no corresponding timer in a certain product, the corresponding trigger signal ITRx does not exist either.

### 13.5.4 TIM2 Interrupt Enable Register (TIM2_DIER)

Offset address: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | TIE | reserve | CC4IE. | CC3IE. | CC2IE. | CC1IE. | UIE |
| | | | | | | | | | R/W | | R/W | R/W | R/W | R/W | R/W |

| bit | Mark | function description | Reset value | read and write |
|---|---|---|---|---|
| 31:7 | - | Reserved, always reads as 0. | 0 | - |
| 6 | TIE | Trigger interrupt enable (Triggerinterrupt enable)<br>0: Disable trigger interrupt;<br>1: Enable trigger interrupt. | 0 | R/W |
| 5 | - | Reserved, always reads as 0. | 0 | - |
| 4 | CC4IE | Enable capture/compare 4 interrupt (Capture/Compare 4 interrupt enable)<br>0: disable capture/compare 4 interrupt;<br>1: Enable capture/compare 4 interrupt. | 0 | R/W |
| 3 | CC3IE | Enable capture/compare 3 interrupt (Capture/Compare 3 interrupt enable)<br>0: disable capture/compare 3 interrupt;<br>1: Enable capture/compare 3 interrupt. | 0 | R/W |
| 2 | CC2IE | Enable capture/compare 2 interrupt (Capture/Compare 2 interrupt enable)<br>0: disable capture/compare 2 interrupt;<br>1: Capture/Compare 2 interrupt enabled. | 0 | R/W |
| 1 | CC1IE | Enable capture/compare 1 interrupt (Capture/Compare 1 interrupt enable)<br>0: disable capture/compare 1 interrupt;<br>1: Enable capture/compare 1 interrupt. | 0 | R/W |
| 0 | UIE | Update interrupt enable<br>0: disable update interruption;<br>1: Enable update interruption. | 0 | R/W |

### 13.5.5 TIM2 Status Register (TIM2_SR)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | CC4<br>OF | CC3<br>OF | CC2<br>OF | CC1<br>OF | reserve | | TIF | reserve | CC4I<br>f | CC3I<br>f | CC2I<br>f | CC1I<br>f | UIF |
| | | | RC<br>W0 | RC<br>W0 | RC<br>W0 | RC<br>W0 | | | RC<br>W0 | | RC<br>W0 | RC<br>W0 | RC<br>W0 | RC<br>W0 | RC<br>W0 |

| Bit | Flags | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:13 | - | Reserved, always reads as 0. | 0 | - |
| 12 | CC4OF | capture/compare 4 overcapture flag (Capture/Compare 4 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 11 | CC3OF | Capture/Compare 3 overcapture flag (Capture/Compare 3 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 10 | CC2OF | capture/compare 2 overcapture flag (Capture/Compare 2 overcapture flag)<br>See CC1OF description. | 0 | RC<br>W0 |
| 9 | CC1OF | Capture/Compare 1 overcapture flag<br>This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear the bit.<br>0: No duplicate capture occurs;<br>1: When the counter value is captured into the TIM2_CCR1 register, the state of CC1IF is already '1'. | 0 | RC<br>W0 |
| 8:7 | - | Reserved, always reads as 0. | 0 | - |
| 6 | TIF | Trigger interrupt flag<br>When a trigger event occurs (when the slave mode controller is in a mode other than gated mode, the TRGI The bit is set by hardware when a valid edge is detected at the input, or any edge in gated mode) set to '1'. It is cleared to '0' by software.<br>0: No trigger event is generated;<br>1: Trigger an interrupt and wait for a response. | 0 | RC<br>W0 |
| 5 | - | Reserved, always reads as 0. | 0 | - |
| 4 | CC4IF | Capture/Compare 4 interrupt flag (Capture/Compare 4 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 3 | CC3IF | Capture/Compare 3 interrupt flag (Capture/Compare 3 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 2 | CC2IF | Capture/Compare 2 interrupt flag (Capture/Compare 2 interrupt flag)<br>Refer to CC1IF description. | 0 | RC<br>W0 |
| 1 | CC1IF | Capture/Compare 1 interrupt flag<br>If channel CC1 is configured in output mode:<br>This bit is set by hardware when the counter value matches the compare value, except in centrosymmetric mode (refer to CMS bit of the TIM2_CR1 register). It is cleared to '0' by software.<br>0: no match occurs;<br>1: The value of TIM2_CNT matches the value of TIM2_CCR1.<br>When the content of TIM2_CCR1 is greater than the content of TIM2_APR, in up or up/down counting mode CC1IF bit goes high if<br>Channel CC1 configured as input mode:<br>This bit is set to '1' by hardware when a capture event occurs, it is cleared to '0' by software or by reading TIM2_CCR1<br>Clear '0'.<br>0: no input capture occurs;<br>1: Counter value has been captured (copied) to TIM2_CCR1 (same polarity as selected on IC1 detected edge). | 0 | RC<br>W0 |

| | | | | |
|---|---|---|---|---|
| 0 | UIF | Update interrupt flag<br>This bit is set to '1' by hardware when an update event occurs. It is cleared to '0' by software.<br>0: No update event is generated;<br>1: Update interrupt pending. This bit is set to '1' by hardware when the register is updated:<br>- If UDIS=0 of the TIM2_CR1 register, when the repeat counter value overflows or underflows (repeat count update event when register = 0).<br>- If URS=0 and UDIS=0 of TIM2_CR1 register, when setting TIM2_EGR register<br>An update event is generated when UG=1, when the counter CNT is reinitialized by software.<br>- If URS=0 and UDIS=0 of the TIM2_CR1 register, when the counter CNT is triggered and the event is reset When initializing. (Refer to 13.5.3: Slave Mode Control Register (TIM2_SMCR)). | 0 | RC<br>W0 |

### 13.5.6 TIM2 Event Generation Register (TIM2_EGR)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | TG | reserve | CC4G | CC3G | CC2G | CC1G | UG |
| | | | | | | | | | WO | | WO WO | WO WO WO | | | WO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:7 | - | Reserved, always reads as 0. | 0 | - |
| 6 | TG | Generate a trigger event (Trigger generation)<br>This bit is set to '1' by software to generate a trigger event, and is automatically cleared to '0' by hardware.<br>0: no action;<br>1: TIF=1 in the TIM2_SR register, if the corresponding interrupt is enabled, the corresponding interrupt will be generated. | 0 | WO |
| 5 | - | Reserved, always reads as 0. | 0 | - |
| 4 | CC4G | generates capture/compare 4 events (Capture/Compare 4 generation)<br>Refer to the CC1G description. | 0 | WO |
| 3 | CC3G | generates capture/compare 3 events (Capture/Compare 3 generation)<br>Refer to the CC1G description. | 0 | WO |
| 2 | CC2G | generates capture/compare 2 events (Capture/Compare 2 generation)<br>Refer to the CC1G description. | 0 | WO |
| 1 | CC1G | Generate capture/compare 1 event (Capture/Compare 1 generation)<br>This bit is set to '1' by software to generate a capture/compare event and automatically cleared to '0' by hardware.<br>0: no action;<br>1: Generate a capture/compare event on channel CC1:<br>If channel CC1 is configured as an output:<br>Set CC1IF=1, if the corresponding interrupt is enabled, the corresponding interrupt will be generated.<br>If channel CC1 is configured as an input:<br>The current counter value is captured to the TIM2_CCR1 register; set CC1IF=1, if it is enabled, the corresponding interrupt, a corresponding interrupt is generated. If CC1IF is already 1, set CC1OF=1. | 0 | WO |
| 0 | UG | Generate update event (Update generation) This bit is set to '1' by software, automatically by hardware Clear '0'.<br>0: no action;<br>1: Reinitializes the counter and generates an update event. Note that the prescaler counter is also Clear '0' (but the prescaler coefficient remains unchanged). If in centrosymmetric mode or DIR=0 (count up) then The counter is cleared to '0'; if DIR=1 (counting down), the counter takes the value of TIM2_ARR. | 0 | WO |

### 13.5.7 TIM2 capture/compare mode register 1 (TIM2_CCMR1)

Offset address: 0x18

Reset value: 0x0000 0000

A channel can be used as input (capture mode) or output (compare mode), the direction of the channel is defined by the corresponding CCxS bit. The other bits of this register are

The effect is different in input and output mode. OCxx describes the function of the channel in output mode, ICxx describes the function of the channel in input mode

function below. It must therefore be noted that the same bit functions differently in output mode than in input mode.

| 31 | 30 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | 22 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC2CE | OC2M[2:0] | | | OC2PE | OC2FE | CC2S[1:0] | | OC1CE | OC1M[2:0] | | | OC1PE | OC1FE | CC1S[1:0] | |
| IC2F[3:0] | | | | IC2PSC[1:0] | | | | IC1F[3:0] | | | | IC1PSC[1:0] | | | |
| R/W | | | | | | | | | | | | | | | |

Output compare mode:

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0 | - |
| 15 | OC2CE | Output Compare 2 clear enable (Output Compare 2 clear enable) | 0 | R/W |
| 14:12 | OC2M[2:0] | Output Compare 2 mode (Output Compare 2 mode) | 0 | R/W |
| 11 | OC2PE | Output Compare 2 preload enable (Output Compare 2 preload enable) | 0 | R/W |
| 10 | OC2FE | Output Compare 2 fast enable (Output Compare 2 fast enable) | 0 | R/W |
| 9:8 | CC2S[1:0] | Capture/Compare 2 selections. (Capture/Compare 2 selection) This bit defines the direction of the channel Direction (input/output), and input pin selection:<br>00: CC2 channel is configured as output;<br>01: CC2 channel is configured as input, IC2 is mapped on TI2;<br>10: CC2 channel is configured as input, IC2 is mapped on TI1;<br>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode only works internally When the trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC2S is writable only when the channel is closed (CC2E=0 of the TIM2_CCER register) of. | 0 | R/W |
| 7 | OC1CE | Output Compare 1 clear '0' enable (Output Compare 1 clear enable)<br>0: OC1REF is not affected by ETRF input;<br>1: Once ETRF input high level is detected, clear OC1REF=0. | 0 | R/W |
| 6:4 | OC1M[2:0] | Output Compare 1 mode<br>The 3 bits define the action of the output reference signal OC1REF, and OC1REF determines the OC1, The value of OC1N. OC1REF is active at high level, and the active level of OC1 and OC1N depends on In CC1P, CC1NP bits.<br>000: Freeze. The ratio between the output compare register TIM2_CCR1 and the counter TIM2_CNT Comparison does not work for OC1REF;<br>001: Set channel 1 to active level when matching. When the value of the counter TIM2_CNT and capture/ When compare register 1 (TIM2_CCR1) is the same, force OC1REF high.<br>010: Set channel 1 to invalid level when matching. When the value of the counter TIM2_CNT and capture/ When compare register 1 (TIM2_CCR1) is the same, force OC1REF low.<br>011: Flip. When TIM2_CCR1=TIM2_CNT, flip the level of OC1REF.<br>100: Forced to inactive level. Force OC1REF low.<br>101: Forced to active level. Force OC1REF high. | 0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 110: PWM mode 1 - when counting up, once TIM2_CNT<TIM2_CCR1 channel 1 is active level, otherwise it is inactive level; when counting down, once When TIM2_CNT>TIM2_CCR1, channel 1 is inactive level (OC1REF=0), otherwise it is active level (OC1REF=1).<br>111: PWM mode 2 - when counting up, once TIM2_CNT<TIM2_CCR1 channel 1 is inactive level, otherwise it is active level; when counting down, once When TIM2_CNT>TIM2_CCR1, channel 1 is active level, otherwise it is inactive level.<br>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIM2_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.<br>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in output compare mode. | | |
| 3 | OC1PE | Output Compare 1 preload enable (Output Compare 1 preload enable) 0: Disable the preload function of the TIM2_CCR1 register, the TIM2_CCR1 register can be written at any time, and the newly written value takes effect immediately.<br>1: Enable the preload function of the TIM2_CCR1 register, read and write operations only operate on the preload register, and the preload value of TIM2_CCR1 is loaded into the current register when the update event arrives. Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIM2_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified. Note 2: Only in single pulse mode (OPM=1 of TIM2_CR1 register), PWM mode can be used without confirming the preload register, | 0 | R/W |
| 2 | OC1FE | otherwise its action is undefined. Output Compare 1 fast enable This bit is used to speed up the CC output response to trigger input events. 0: According to the value of the counter and CCR1, CC1 operates normally even if the flip-flop is on. When the flip-flop input has an active edge, the minimum delay to activate the CC1 output is 5 clock cycles. 1: The active edge input to the flip-flop acts as a compare match. Therefore, OC is set as the comparison level regardless of the comparison result. The delay between the active edge of the sample flip-flop and the output of CC1 is shortened to 3 clock cycles.<br>OCFE works only when the channel is configured as PWM1 or PWM2 mode. | 0 | R/W |
| 1:0 | CC1S[1:0] | Capture/Compare 1 selection. (Capture/Compare 1 selection) These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIM2_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIM2_CCER register). | 0 | R/W |

Enter capture mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:12 | IC2F[3:0] | Input capture 2 filter (Input capture 2 filter) | 0x0 | R/W |
| 11:10 | IC2PSC[1:0] | Input/capture 2 prescaler (Input capture 2 prescaler) Capture/compare 2 selection | 0x0 | R/W |
| 9:8 | CC2S[1:0] | (Capture/Compare 2 selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC2 channel is configured as output;<br>01: CC2 channel is configured as input, IC2 is mapped on TI2;<br>10: CC2 channel is configured as input, IC2 is mapped on TI1;<br>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode only works within<br>When the internal trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC2S is writable only when the channel is closed (CC2E=0 of the TIM2_CCER register)<br>of. | 0x0 | R/W |
| 7:4 | IC1F[3:0] | Input capture 1 filter<br>These bits define the sampling frequency and digital filter length of the TI1 input. The digital filter consists of<br>It consists of an event counter, which will generate an output transition after recording N events:<br>0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6<br>0001: Sampling frequency fSAMPLING=fCK_INT , N=2 1001: Sampling frequency<br>fSAMPLING=fDTS/8, N=8<br>0010: Sampling frequency fSAMPLING=fCK_INT , N=4 1010: sampling frequency<br>fSAMPLING=fDTS/16, N=5<br>0011: Sampling frequency fSAMPLING=fCK_INT , N=8 1011: sampling frequency<br>fSAMPLING=fDTS/16, N=6<br>0100: Sampling frequency fSAMPLING=fDTS/2 , N=6 1100: sampling frequency<br>fSAMPLING=fDTS/16, N=8<br>0101: Sampling frequency fSAMPLING= fDTS /2, N=8 1101: Sampling frequency<br>fSAMPLING=fDTS/32, N=5<br>0110: sampling frequency fSAMPLING= fDTS /4, N=6 1110: sampling frequency<br>fSAMPLING=fDTS/32, N=6<br>0111: Sampling frequency fSAMPLING= fDTS /4, N=8 1111: sampling frequency<br>fSAMPLING=fDTS/32, N=8 | 0x0 | R/W |
| 3:2 | IC1PSC[1:0] | Input/capture 1 prescaler (Input capture 1 prescaler)<br>These 2 bits define the prescaler factor for CC1 input (IC1).<br>Once CC1E=0 (in TIM2_CCER register), the prescaler is reset.<br>00: No prescaler, each edge detected on the capture input triggers a capture;<br>01: trigger a capture every 2 events;<br>10: Every 4 events trigger a capture;<br>11: Capture is triggered every 8 events. | 0x0 | R/W |
| 1:0 | CC1S[1:0] | Capture/Compare 1 Selection (Capture/Compare 1 Selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC1 channel is configured as output;<br>01: CC1 channel is configured as input, IC1 is mapped on TI1;<br>10: CC1 channel is configured as input, IC1 is mapped on TI2;<br>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode only works within<br>When the internal trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC1S is writable only when the channel is closed (CC1E=0 of the TIM2_CCER register)<br>of. | 0x0 | R/W |

### 13.5.8 TIM2 Capture/Compare Mode Register 2 (TIM2_CCMR2)

Offset address: 0x1C

Reset value: 0x0000 0000

See the description of the CCMR1 register above.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OC4CE | OC4M[2:0] | | | OC4PE | OC4FE | CC4S[1:0] | | OC3CE | OC3M[2:0] | | | OC3PE | OC3FE | CC3S[1:0] | |
| IC4F[3:0] | | | | IC4PSC[1:0] | | | | IC3F[3:0] | | | | IC3PSC[1:0] | | | |
| R/W | | | | | | | | | | | | | | | |

Output compare mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0x0 | - |
| 15 | OC4CE | Output Compare 4 clear enable (Output Compare 4 clear enable) | 0 | R/W |
| 14:12 | OC4M[2:0] | Output Compare 4 mode (Output Compare 4 mode) | 0x0 | R/W |
| 11 | OC4PE | Output Compare 4 preload enable (Output Compare 4 preload enable) | 0 | R/W |
| 10 | OC4FE | Output Compare 4 fast enable (Output Compare 4 fast enable) | 0 | R/W |
| 9:8 | CC4S[1:0] | Capture/Compare 4 selections. (Capture/Compare 4 selection)<br>The 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC4 channel is configured as output;<br>01: CC4 channel is configured as input, IC4 is mapped on TI4;<br>10: CC4 channel is configured as input, IC4 is mapped on TI3;<br>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode only works internally When the trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC4S is writable only when the channel is closed (CC4E=0 of the TIM2_CCER register) of. | 0x0 | R/W |
| 7 | OC3CE | Output Compare 3 clear '0' enable (Output Compare 3 clear enable) | 0 | R/W |
| 6:4 | OC3M[2:0] | Output Compare 3 mode (Output Compare 3 mode) | 0x0 | R/W |
| 3 | OC3PE | Output Compare 3 preload enable (Output Compare 3 preload enable) | 0 | R/W |
| 2 | OC3FE | Output Compare 3 fast enable (Output Compare 3 fast enable) | 0 | R/W |
| 1:0 | CC3S[1:0] | Capture/Compare 3 selections. (Capture/Compare 3 selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC3 channel is configured as output;<br>01: CC3 channel is configured as input, IC3 is mapped on TI3;<br>10: CC3 channel is configured as input, IC3 is mapped on TI4;<br>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode only works internally When the trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIM2_CCER register) of. | 0x0 | R/W |

Enter capture mode:

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:12 | IC4F[3:0] | Input capture 4 filter (Input capture 4 filter) | 0 | R/W |
| 11:10 | IC4PSC[1:0] | Input/capture 4 prescaler (Input capture 4 prescaler) Capture/compare 4 selection | 0 | R/W |
| 9:8 | CC4S[1:0] | (Capture/Compare 4 selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC4 channel is configured as output;<br>01: CC4 channel is configured as input, IC4 is mapped on TI4;<br>10: CC4 channel is configured as input, IC4 is mapped on TI3;<br>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode only works within When the internal trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC4S is writable only when the channel is closed (CC4E=0 of the TIM2_CCER register) of. | 0 | R/W |
| 7:4 | IC3F[3:0] | Input capture 3 filter (Input capture 3 filter) | 0 | R/W |
| 3:2 | IC3PSC[1:0] | Input/capture 3 prescaler (Input capture 3 prescaler) Capture/compare 3 | 0 | R/W |
| 1:0 | CC3S[1:0] | selection (Capture/Compare 3 Selection)<br>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:<br>00: CC3 channel is configured as output;<br>01: CC3 channel is configured as input, IC3 is mapped on TI3;<br>10: CC3 channel is configured as input, IC3 is mapped on TI4;<br>11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode only works within When the internal trigger input is selected (selected by the TS bit of the TIM2_SMCR register).<br>Note: CC3S is writable only when the channel is closed (CC3E=0 of the TIM2_CCER register) of. | 0 | R/W |

### 13.5.9 TIM2 Capture/Compare Enable Register (TIM2_CCER)

Offset address: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | CC4P | CC4E | reserve | | CC3P | CC3E | reserve | | CC2P | CC2E | reserve | | CC1P | CC1E |
| | | R/W | R/W | | | R/W | R/W | | | R/WR/W | | | | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:14 | - | Reserved, always reads as 0. | 0x0 | - |
| 13 | CC4P | input/capture 4 output polarity (Capture/Compare 4 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 12 | CC4E | input/capture 4 output enable (Capture/Compare 4 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 11:10 | - | Reserved, always reads as 0. | 0x0 | - |
| 9 | CC3P | input/capture 3 output polarity (Capture/Compare 3 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 8 | CC3E | input/capture 3 output enable (Capture/Compare 3 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 7:6 | - | Reserved, always reads as 0. | 0x0 | - |
| 5 | CC2P | input/capture 2 output polarity (Capture/Compare 2 output polarity)<br>Refer to the description of CC1P. | 0 | R/W |
| 4 | CC2E | input/capture 2 output enable (Capture/Compare 2 output enable)<br>Refer to the description of CC1E. | 0 | R/W |
| 3:2 | - | Reserved, always reads as 0. | 0x0 | - |
| 1 | CC1P | Input/Capture 1 output polarity (Capture/Compare 1 output polarity)<br>CC1 channel configured as output:<br>0: OC1 active high;<br>1: OC1 active low.<br>CC1 channel configured as input:<br>This bit selects whether IC1 or the inverted signal of IC1 is used as trigger or capture signal.<br>0: No inversion: Capture occurs on the rising edge of IC1; when used as an external trigger, IC1 does not invert.<br>1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.<br>Note: Once the LOCK level (LOCK bit in the TIM2_BDTR register) is set to 3 or 2, the bit Cannot be modified. | 0 | R/W |
| 0 | CC1E | Input/Capture 1 output enable (Capture/Compare 1 output enable)<br>CC1 channel configured as output:<br>0: Off - OC1 disables the output, so the output level of OC1 depends on MOE, OSSI, The value of the OSSR, OIS1, OIS1N, and CC1NE bits.<br>1: On - OC1 signal is output to the corresponding output pin, and its output level depends on MOE, The value of the OSSI, OSSR, OIS1, OIS1N, and CC1NE bits.<br>CC1 channel configured as input:<br>This bit determines whether the counter value can be captured into the TIM2_CCR1 register.<br>0: capture disabled;<br>1: Capture enabled. | 0 | R/W |

Table 13-4 Output Control Bits for Standard OCx Channels

| CCxE bit | OCx output status |
|---|---|
| 0 | disable output (OCx=0, OCx_EN=0) |
| 1 | OCx=OCxREF + Polarity, OCx_EN=1 |

Note: The state of external I/O pins connected to standard OCx channels depends on the OCx channel state and GPIO and AFIO registers.

### 13.5.10 TIM2 counter (TIM2_CNT)

Offset address: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CNT[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:0 | CNT[15:0] Counter value (Counter value) | | 0x0 | R/W |

### 13.5.11 TIM2 Prescaler (TIM2_PSC)

Offset address: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PSC[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Bit Flag | Functional Description | | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | Reserved, always reads as 0. | 0x0 | - |
| 15:0 | PSC[15:0] | Prescaler value (Prescaler value)<br>The clock frequency of the counter (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1).<br>PSC contains the value loaded into the current prescaler register each time an update event occurs; more New events include the counter being cleared to '0' by the UG bit of TIM_EGR or being operated in reset mode Cleared to '0' from the controller. | 0x0 | R/W |

### 13.5.12 TIM2 Auto-Reload Register (TIM2_ARR)

Offset address: 0x2C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | ARR[15:0] | Auto-reload value (Prescaler value)<br>ARR contains the value to be loaded into the actual auto-reload register.<br>For details, refer to Section 13.3.1: Updates and actions related to ARR.<br>When the auto-reload value is empty, the counter does not work. | 0 | R/W |

### 13.5.13 TIM2 capture/compare register 1 (TIM2_CCR1)

Offset address: 0x34

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR1[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR1[15:0] | Capture/Compare 1 value (Capture/Compare 1 value)<br>If the CC1 channel is configured as an output:<br>CCR1 contains the value loaded into the current capture/compare 1 register (preload value).<br>If the preload function is not selected in the TIM2_CCMR1 register (OC1PE bit), write The value entered will be transferred to the current register immediately. Otherwise only when the update event occurs , this preload value is transferred to the current capture/compare 1 register.<br>The current capture/compare register participates in the comparison with the counter TIM2_CNT, and in OC1 An output signal is generated on the port.<br>If the CC1 channel is configured as an input:<br>CCR1 contains the counter value transmitted by the last Input Capture 1 event (IC1). | 0 | R/W |

### 13.5.14 TIM2 capture/compare register 2 (TIM2_CCR2)

Offset address: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR2[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR2[15:0] | Capture/Compare Channel 2 Value (Capture/Compare 2 value)<br>If the CC2 channel is configured as an output:<br>CCR2 contains the value loaded into the current capture/compare 2 register (preload value).<br>If the preload feature is not selected in the TIM2_CCMR2 register (OC2PE bit), write The value entered will be transferred to the current register immediately. Otherwise only when the update event occurs , the preload value is transferred to the current capture/compare 2 register.<br>The current capture/compare register participates in the comparison with the counter TIM2_CNT, and in OC2 An output signal is generated on the port.<br>If the CC2 channel is configured as an input:<br>CCR2 contains the counter value transmitted by the last Input Capture 2 event (IC2). | 0 | R/W |

### 13.5.15 TIM2 capture/compare register 3 (TIM2_CCR3)

Offset address: 0x3C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR3[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15:0 | CCR3[15:0] | Capture/Compare Channel 3 Value (Capture/Compare 3 value)<br>If the CC3 channel is configured as an output:<br>CCR3 contains the value loaded into the current capture/compare 3 register (preload value).<br>If the preload feature is not selected in the TIM2_CCMR3 register (OC3PE bit), write<br>The value entered will be transferred to the current register immediately. Otherwise only when the update event occurs<br>, the preload value is transferred to the current capture/compare 3 register.<br>The current capture/compare register participates in the comparison with the counter TIM2_CNT, and in OC3<br>An output signal is generated on the port.<br>If the CC3 channel is configured as an input:<br>CCR3 contains the counter value transmitted by the last Input Capture 3 event (IC3). | 0 | R/W |

### 13.5.16 TIM2 capture/compare register 4 (TIM2_CCR4)

Offset address: 0x40

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR4[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads | 0 | - |
| 15:0 | CCR4[15:0] | as 0. Capture/Compare 4 value (Capture/Compare 4 value)<br>If the CC4 channel is configured as an output:<br>CCR4 contains the value loaded into the current capture/compare 4 register (preload value).<br>If the preload feature is not selected in the TIM2_CCMR4 register (OC4PE bit), write<br>The value entered will be transferred to the current register immediately. Otherwise only when the update event occurs<br>, the preload value is transferred to the current capture/compare 4 register.<br>The current capture/compare register participates in the comparison with the counter TIM2_CNT, and in OC4<br>An output signal is generated on the port.<br>If the CC4 channel is configured as an input:<br>CCR4 contains the counter value transmitted by the last Input Capture 4 event (IC4). | 0 | R/W |

# 14 Programmable Count Array (PCA)

## 14.1 Introduction to PCA

PCA (ProgrammableCounterArray, programmable counter array) supports up to five 16-bit capture/comparison modules. the timing

The /counter can be used as a general-purpose clock count/capture/compare function of the event counter. Each module of PCA can be independently

can be programmed independently to provide input capture, output compare, or pulse-width modulation. In addition module 4 has additional watchdog timer mode.

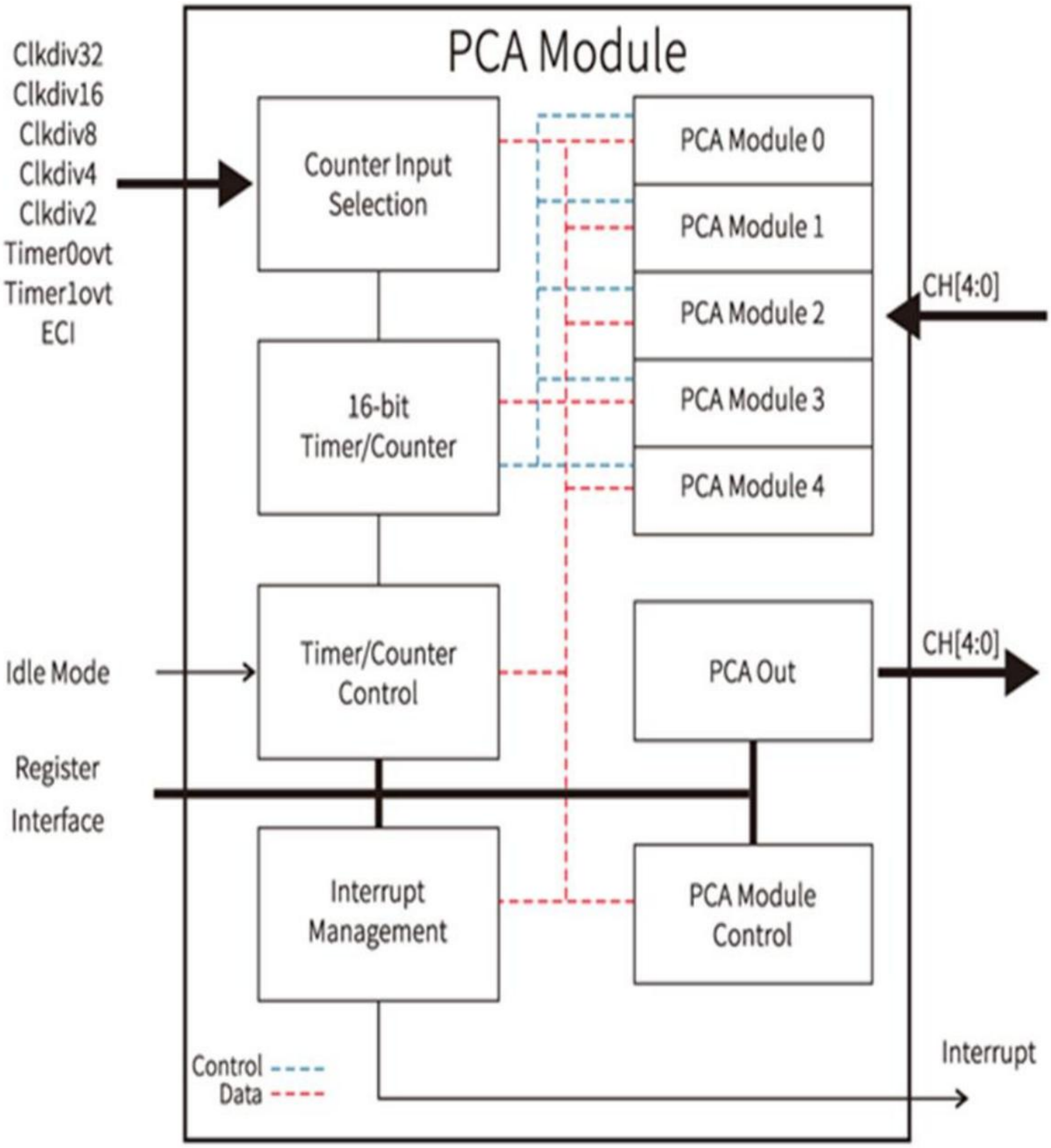


Figure 14-1 Overall block diagram of PCA

## 14.2 PCA Functional Description

Each module can be configured to work independently, and there are three working modes: edge-triggered capture, output compare, and 8-bit pulse width modulation. Each module has its own function registers in the system controller, and these registers are used to configure the working mode of the module and exchange data with the module. Each group of compare/capture modules is composed of a compare/capture register group (CCAPx), a 16-bit comparator and various logic gate controls. The register group is used to store time or times, for external trigger capture conditions, or internal trigger comparison conditions. In PWM mode, the register (CCAPxL) is used to control the duty cycle of the output waveform. Each module can be independently programmed to operate in any of the following modes:

ÿ Rising edge, falling edge or any edge trigger in 16-bit capture mode. ÿ
Compare mode: 16-bit software timer, 16-bit high-speed output or 8-bit pulse width modulation. ÿ Not
started.

The compare/capture module mode registers (CCAPMx) determine the corresponding operating mode. When programming the compare/capture modules, they are based on a common time count. The timer/counter is turned on and off through the CR.CR bit to control the operation of the PCA timer/counter. If the corresponding enable bit (CCAPMx.CCIE) is set, when a match or capture occurs, the compare/capture flag (CR.CCFx) is set and generates
PCA interrupt request. The CPU can read and write the CCAPx registers at any time.

### 14.2.1 PCA Timer/Counter

This group of special function registers of CNT can be used as a 16-bit timer/counter. This is a 16-bit up counter. If the MOD.CFIE bit is set to "1", the hardware will automatically set the PCA overflow flag (CR.CF) and generate a PCA interrupt request when CNT overflows. Three bits of MOD.CPS[2:0] select eight signals to be input to the timer/counter.

ÿ32 frequency division of system clock PCLK

ÿ16 frequency division of system clock PCLK

ÿThe system clock PCLK divided by 8

ÿThe frequency of the system clock PCLK is divided by 4

ÿThe system clock PCLK divided by 2

ÿTimer 0 overflow (overflow): Every time timer 0 overflows, CNT is incremented, which provides variable

Programmed frequency input.

ÿTimer 1 overflow (overflow): every time timer 1 overflows, CNT is incremented, which provides variable

Programmed frequency input.

ÿ ECI: The CPU samples PCA ECI every 4 PCLK clock cycles. When the sampling result changes from high to low each time,

CNT_L (CNT low 8 bit) is automatically increased by 1, so the highest ECI input frequency cannot be higher than 1/8 of the system clock PCLK, to

Satisfy sampling requirements.

Set the run controller (CR.CR) to start the PCA timer/counter. When MOD.CIDL is set to "1", the PCA timer/counter can continue to run in idle mode. The CPU can read the value of CNT at any time, but when the counting starts (CR.CR=1), in order to prevent the counting
Error, CNT is write-prohibited.

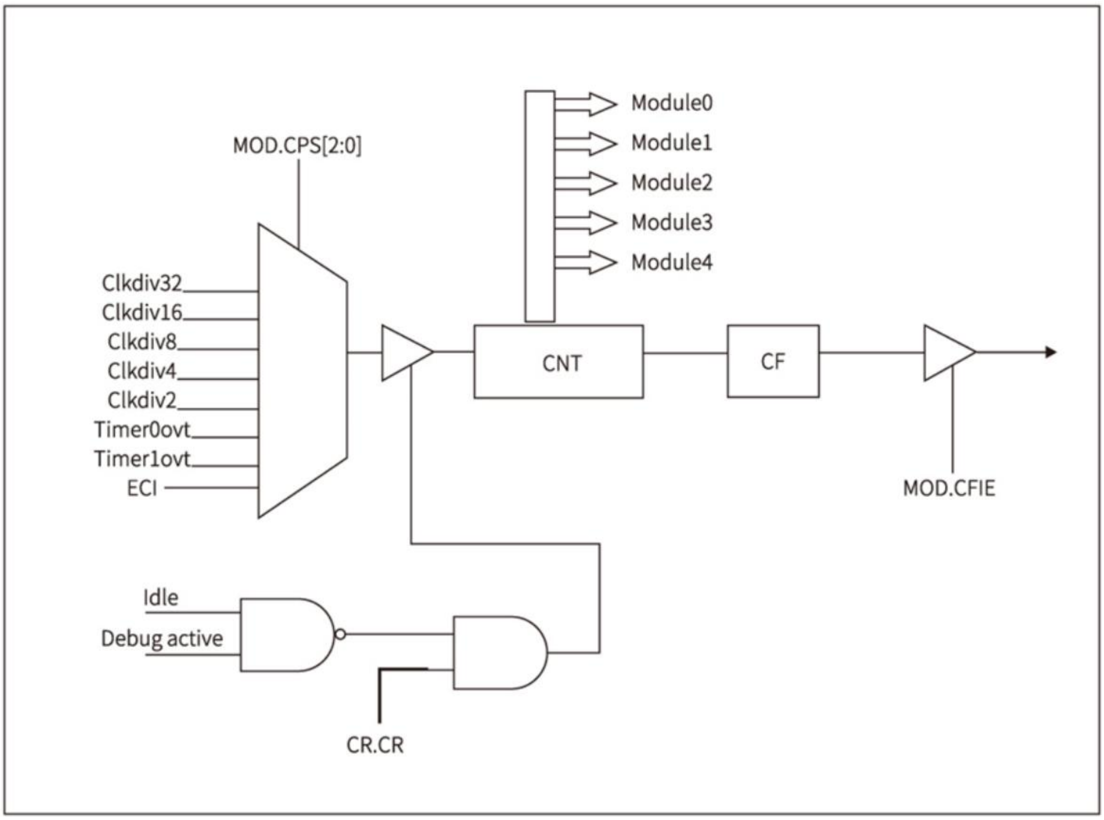


Figure 14-2 PCA counter block diagram

## 14.2.2 Capture function

PCA capture mode provides 5-way PCA to measure pulse period, pulse width, duty cycle and phase difference. A transition on the pin causes the PCA to capture the value of the PCA counter/timer and load it into the corresponding module's 16-bit capture/compare register (CCAPx).

The CCAPMx.CAPP and CCAPMx.CAPN bits are used to select the type of level change that triggers the capture: low to high (positive edge), high to low (negative edge), or any change (positive or negative edge) along). When a capture occurs, the capture/compare flag (CCFx) in CR is set to logic '1' and an interrupt request is generated (if CCF interrupt is enabled). When the CPU transfers to the interrupt service routine, the CR.CCFx bit cannot be automatically cleared by hardware, and the user software can write the INTCLR.CCFx register to clear this flag bit. If the CCPMx.CAPP and CCAPMx.CAPN bits are both set to logic '1', you can determine whether the capture is triggered by a rising edge or by directly reading the state of the corresponding port pin.

Triggered by falling edge. The resolution is equal to the clock of the timer/counter. The input signal must maintain at least 2 clock cycles during the high level or low level to ensure that the input signal can be recognized by the hardware.

The CPU can read or write the CCAPx registers at any time. Capture settings:

ÿ When capturing on an external rising edge is required, CCPMx.CAPP = "1" and CCAPMx.CAPN = "0"

ÿ When capturing on the external falling edge, CCPMx.CAPP = "0" and CCAPMx.CAPN = "1"

ÿ When capture is required on external rising and falling edges, CCPMx.CAPP = "1" and CCAPMx.CAPN = "1"

Notice:

Subsequent captures by the same module override existing captured values. To keep the captured value, in the interrupt service routine save it in In RAM, this operation must be completed before the next event occurs, otherwise the previous captured sample value will be lost.

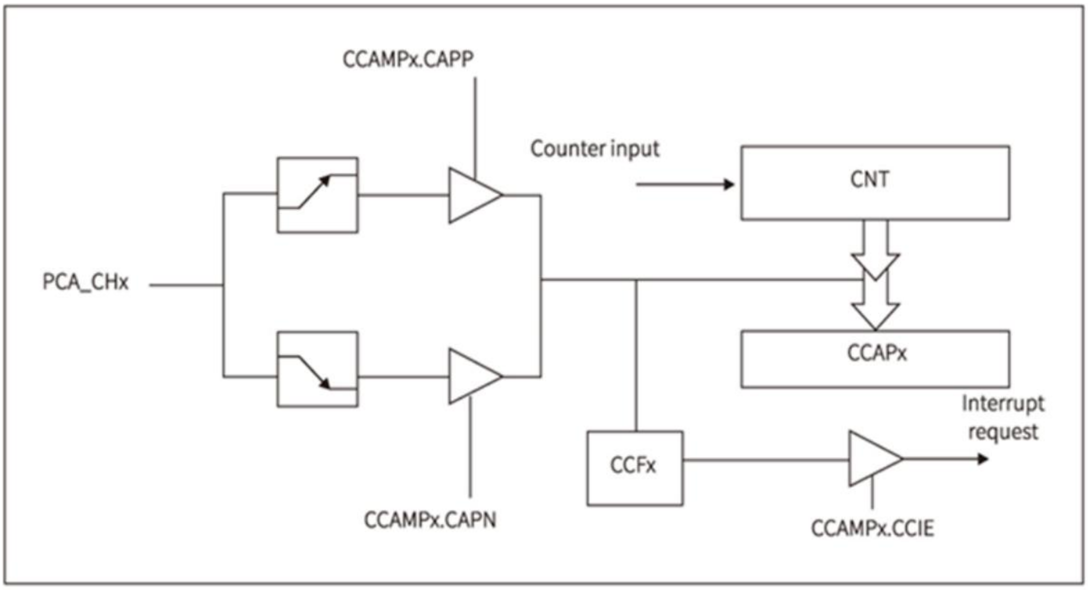


Figure 14-3 PCA capture functional block diagram

## 14.2.3 PCA comparison function

The PCA compare function provides the following functions: timer, event counter, pulse width modulation. The PCA comparison function can provide three models

Mode: 16-bit software timer mode, high-speed output mode, PWM mode. In the first two modes, the compare/capture module compares the 16-bit

The value of the PCA timer/counter and the 16-bit value preloaded into the module's CCAPx register. In PWM mode, the PCA mode

The block continuously compares the PCA Timer/Counter Low Register (CNT) with an 8-bit value in the CCAPxL register.

The comparison is done every 4 clock cycles, which matches the clock rate of the fastest PCA timer/counter. set up

The CCAPMx.ECOM bit selects the compare function of this module. To use modules in compare mode correctly, follow the general procedure below:

sequence:

ÿ Select the operating mode of the PCA module

ÿSelect the input signal of PCA timer/counter

ÿ Compare value is loaded into module compare/capture register pair

ÿ Set PCA timer/counter operation control bit

ÿ Generate an interrupt after a match, clear the comparison/capture flag of the module

### 14.2.3.1 16-bit software counter mode

To set a compare/capture module to work in 16-bit software timer mode, it is necessary to set CCAPMx.ECOM and

CCAPMx.MAT bits. Once a match occurs between the PCA timer/counter and the compare/capture register (CCAPx), this sets

Set the compare/capture flag (CR.CCFx) of the module. This will generate an interrupt request if the corresponding interrupt enable bit

(CCAPMx.CCIE) setting. Since the compare/capture flag (CR.CCFx) is not cleared by hardware, the user must clear the flag by software. on interruption

In the service routine, a new 16-bit compare value can be written to the compare/capture register (CCAPx). NOTE: Before updating these deposits

In order to prevent invalid matching, user software should write CCAPxL first, then CCAPxH. Once written to CCAPxL is cleared

In addition to disabling the ECOMx bit of the compare function, a write to CCAPxH will simultaneously set the ECOMx bit, re-enabling the compare function. Immediately

When writing a 16-bit value into the capture/compare register of PCA0, the low byte should be written first.

14.2.3.2 High Speed Output Mode

In high-speed output mode, whenever the value in the PCA counter matches the module's 16-bit capture/compare register (CCAPx), the module The value of the PCA_CCAPOx register (corresponding to the PCA_CHx pin) of the PCA will be toggled. This can provide more than switching IO outputs with Higher precision, because this high-speed output will not respond to interrupts and affect the output frequency. If the CPU is used to switch the IO output, power consumption, precision degrees are lacking.

To set the high-speed output mode of a compare/capture module, set the CCAPMx.ECOM, CCAPMx.MAT and CCAPMx.TOG bits. A match between the PCA timer/counter and the compare/capture register (CCAPx) flips the value of the PCA's PCA_CCAPOx register, And set the compare/capture flag (CR.CCFx) of the module.

The user can also choose to generate an interrupt request by setting the corresponding interrupt enable bit (CCAPMx.CCIE) when a match occurs, then An interrupt request is generated. Since the compare/capture flag cannot be cleared by hardware, the user must clear this flag in software. If the user is interrupted The comparison/capture register is not changed in the program, and the next rollover occurs after the PCA counter overflows and the count value matches the comparison value again. In the interrupt service routine, a new 16-bit compare value can be written to the compare/capture register (CCAPx).

Note: To prevent invalid matches while updating these registers, user software should write to CCAPxL first and then CCAPxH. Write Clearing the ECOM bit to CCAPxL disables the compare function, while writing to CCAPxH sets the ECOM bit and re-enables the compare function.

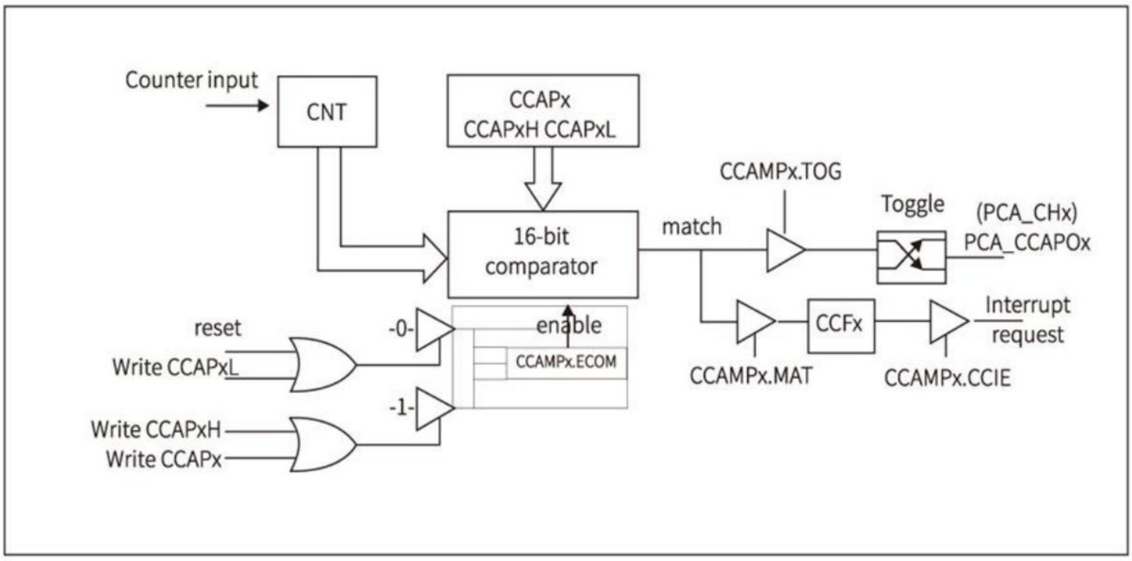


Figure 14-4 PCA comparison function block diagram

14.2.3.3 8-bit pulse width modulation (PWM) function

Pulse Width Modulation is a technique that uses a program to control the duty cycle, period, and phase of a waveform. Each of the five PCA modules can be used independently A pulse width modulated (PWM) output is generated at the PCA_CHx pin corresponding to the PCA, with a pulse width of 8-bit resolution. The frequency of the PWM output is taken as Depends on the time base of the PCA counter/timer. Use the module's capture/compare register CCAPxL to change the duty of the PWM output signal Compare. When the low byte (CNT_L) of the PCA counter/timer is equal to the value in CCAPxL, the output on the PCA_CHx pin of the PCA is set to "1"; when the count value in CNT_L overflows, the PCA_CHx output of the PCA is reset to "0". When the counter/timer's When the low byte CNT_L overflows (from 0xFF to 0x00), the value stored in CCAPxH is automatically loaded into CCAPxL without software intervention. Pre.

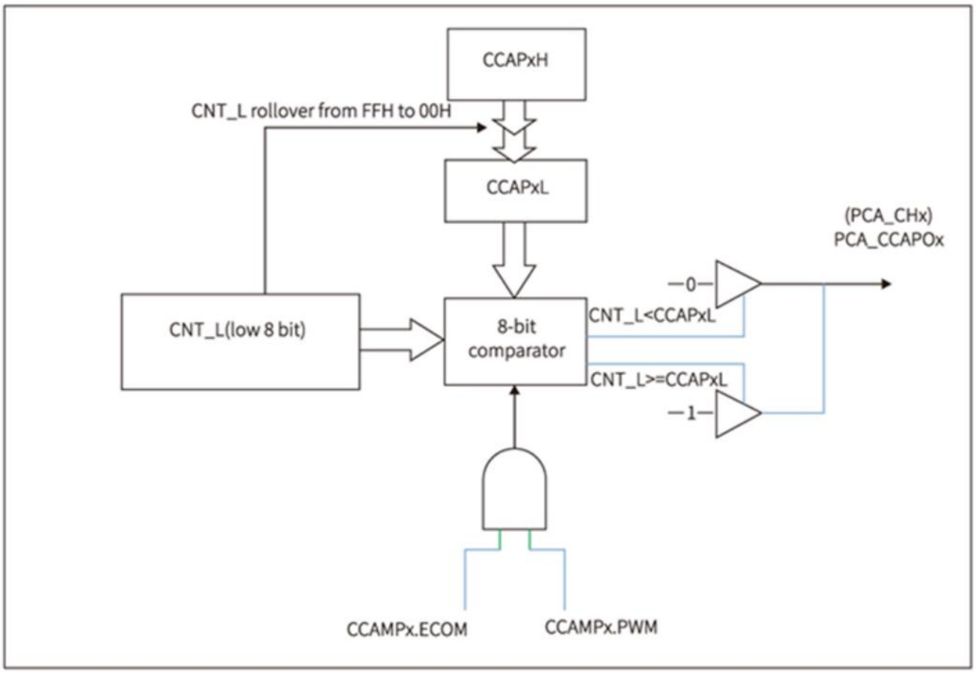


Figure 14-5 PCA PWM functional block diagram

In this mode, the PCA timer/counter CNT_L value is constantly compared with the value in the low byte compare/capture register (CCAPxL). When CNT_L < CCAPxL, the output waveform is low. When both match (CNT_L = CCAPxL), the output waveform goes high until CNT_L overflows from FFH to 00H and remains high during the end. On overflow, the value in CCAPxH is automatically loaded into CCAPxL, a the beginning of a new cycle.

The value in CCAPxL determines the duty cycle of the current waveform. The value in CCAPxH determines the duty cycle of the next waveform. Change the CCAPxL in Pulse Width Modulation where the value can be changed. As shown, the 8-bit value in CCAPxL can range from 0 (100% duty cycle), to 255 (0.4% duty cycle Compare). To change the CCAPxL value without glitching, a new value needs to be written to the high byte register (CCAPxH). Rollover when CL exceeds 0xFF Move to 0x00, this value is automatically loaded into CCAPxL by hardware.

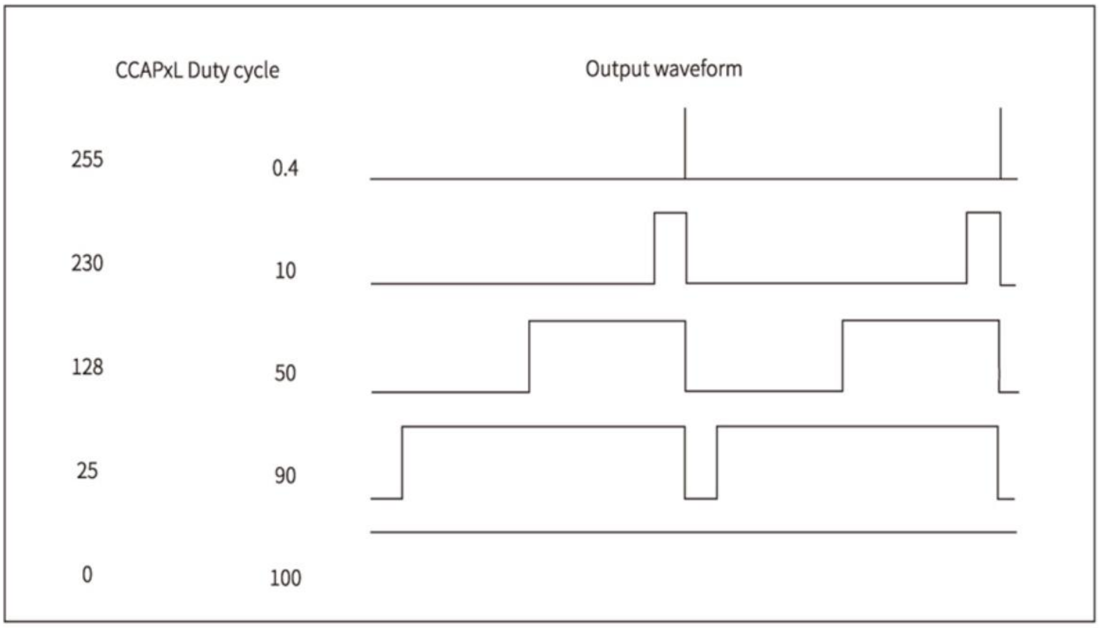


Figure 14-6 PCA PWM output waveform

To set a compare/capture module to work in PWM mode, the CCAPMx.ECOM and CCAPMx.PWM bits need to be set. in addition

PCA timer/counter can select the input count signal frequency by programming MOD.CSP[2:0]. An 8-bit value entered in CCAPxL refers to

Set the duty cycle of the first PWM waveform. Entering an 8-bit value in CCAPxH specifies the duty cycle of the second PWM waveform. set up

The timer/counter run control bit (CR.CR) starts the PCA timer/counter.

Table 14-1 PCA comparison/capture function module settings

| ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF | Way of working |
|---|---|---|---|---|---|---|---|
| x | 1 | 0 | 0 | 0 | 0 | x | Trigger capture with positive edge of CCPn |
| x | 0 | 1 | 0 | 0 | 0 | x | Capture with negative edge of CCPn |
| x | 1 | 1 | 0 | 0 | 0 | x | Trigger capture with transition of CCPn |
| 1 | 0 | 0 | 1 | 0 | 0 | x | software timer |
| 1 | 0 | 0 | 1 | 1 | 0 | x | high speed output |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 8-bit pulse width modulator |

## 14.3 Interconnection and control of PCA module and other modules

### 14.3.1 ECI interconnect

The ECI input can be an external input port selected through IO MUX, or it can be the compared filter output of the internal VC. VC output

The control registers are in the VC control block.

### 14.3.2 PCACAP0

The capture input for channel 0 can be:

ÿ External IO MUX input port, external UART RX MUX input

ÿComparison and filtered output of internal VC The UART selection is controlled in the PCA capture channel control register

In SYSCON_PCACR, the VC output control register is in the VC control block.

### 14.3.3 PCACAP1/2/3/4

The capture input for channels 1/2/3/4 can be:

ÿ External IO MUX input port, external UART RX MUX input

UART Selection Control The UART selection control is in the PCA capture channel control register, SYSCON_PCACR.

## 14.4 PCA register list

Base address 0x4000 1400

Table 14-2 PCA register list and reset value

| offset address name | | describe | reset value |
|---|---|---|---|
| 0x000 | PCA_CR | PCA Control Register | 0x0000 0000 |
| 0x004 | PCA_MOD | PCA Mode Register | 0x0000 0000 |
| 0x008 | PCA_CNT | PCA count register | 0x0000 0000 |
| 0x00C | PCA_INTCLR | PCA Interrupt Clear Register | 0x0000 009F |
| 0x010 | PCA_CCAPM0 | PCA Compare/Capture Module 0 Mode Register | 0x0000 0000 |
| 0x014 | PCA_CCAPM1 | PCA Compare/Capture Module 1 Mode Register | 0x0000 0000 |
| 0x018 | PCA_CCAPM2 | PCA Compare/Capture Module 2 Mode Register | 0x0000 0000 |
| 0x01C | PCA_CCAPM3 | PCA Compare/Capture Module 3 Mode Register | 0x0000 0000 |
| 0x020 | PCA_CCAPM4 | PCA Compare/Capture Module 4 Mode Register | 0x0000 0000 |
| 0x030 | PCA_CCAP0L | PCA compare/capture module 0 lower 8-bit register | 0x0000 0000 |
| 0x034 | PCA_CCAP0H | PCA compare/capture module 0 high 8-bit register | 0x0000 0000 |
| 0x038 | PCA_CCAP1L | PCA compare/capture module 1 lower 8-bit register | 0x0000 0000 |
| 0x03C | PCA_CCAP1H | PCA compare/capture module 1 high 8-bit register | 0x0000 0000 |
| 0x040 | PCA_CCAP2L | PCA compare/capture module 2 lower 8-bit register | 0x0000 0000 |
| 0x044 | PCA_CCAP2H | PCA compare/capture module 2 high 8-bit register | 0x0000 0000 |
| 0x048 | PCA_CCAP3L | PCA compare/capture module 3 lower 8-bit register | 0x0000 0000 |
| 0x04c | PCA_CCAP3H | PCA compare/capture module 3 high 8-bit register | 0x0000 0000 |
| 0x050 | PCA_CCAP4L | PCA compare/capture module 4 lower 8-bit register | 0x0000 0000 |
| 0x054 | PCA_CCAP4H | PCA compare/capture module 4 high 8-bit register | 0x0000 0000 |
| 0x058 | PCA_CCAPO | PCA PWM and High Speed Output Flag Register | 0x0000 0000 |
| 0x05C | PCA_POCR | PCA terminal output control register | 0x0000 0000 |
| 0x060 | PCA_CCAP0 | 16-bit register for PCA compare/capture module 0 | 0x0000 0000 |
| 0x064 | PCA_CCAP1 | 16-bit register for PCA compare/capture module 1 | 0x0000 0000 |
| 0x068 | PCA_CCAP2 | 16-bit register for PCA compare/capture module 2 | 0x0000 0000 |
| 0x06C | PCA_CCAP3 | 16-bit register for PCA compare/capture module 3 | 0x0000 0000 |
| 0x070 | PCA_CCAP4 | 16-bit register for PCA compare/capture module 4 | 0x0000 0000 |

## 14.5 Register description

## 14.5.1 Control Register (PCA_CR)

Offset address: 0x000

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | CF | CR | Save<br>Keep | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| | | | | | | | | RO | R/W | | RO | RO | RO | RO | RO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 - | | reserved bit | 0x00 | - |
| 7 | CF | PCA counter overflow flag:<br>0: no overflow<br>1: A counter overflow has occurred<br>When the PCA count overflows, CF is set by hardware<br>When the CFIE bit of the MOD register is 1, the CF flag can generate an interrupt | 0 | RO |
| 6 | CR | PCA counter run control bit<br>0: Disable PCA counter counting<br>1: Start PCA counter counting | 0 | R/W |
| 5 | - | reserved bit | 0 | - |
| 4 | CCF4 | PCA counter module 4 compare/capture flag:<br>This bit is set by hardware when a match or capture occurs<br>When CCAPM4.CCIE is set, this flag will generate a PCA interrupt<br>0: no match or capture<br>1: match or capture occurs | 0 | RO |
| 3 | CCF3 | PCA counter module 3 compare/capture flag:<br>This bit is set by hardware when a match or capture occurs<br>When CCAPM3.CCIE is set, this flag will generate a PCA interrupt<br>0: no match or capture<br>1: match or capture occurs | 0 | RO |
| 2 | CCF2 | PCA counter module 2 compare/capture flag:<br>This bit is set by hardware when a match or capture occurs<br>When CCAPM2.CCIE is set, this flag will generate a PCA interrupt<br>0: no match or capture<br>1: match or capture occurs | 0 | RO |
| 1 | CCF1 | PCA counter module 1 compare/capture flag:<br>This bit is set by hardware when a match or capture occurs<br>When CCAPM1.CCIE is set, this flag will generate a PCA interrupt<br>0: no match or capture<br>1: match or capture occurs | 0 | RO |
| 0 | CCF0 | PCA counter module 0 compare/capture flag:<br>This bit is set by hardware when a match or capture occurs<br>When CCAPM0.CCIE is set, this flag will generate a PCA interrupt<br>0: no match or capture<br>1: match or capture occurs | 0 | RO |

## 14.5.2 Mode Register (PCA_MOD)

Offset address: 0x004

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | CIDL | reserve | | | CPS | | | CFIE |
| | | | | | | | | R/W | | | | R/W | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | Reserved bits | 0x0 | - |
| 7 | CIDL | Does PCA stop working in idle mode IDLE?<br>0: In sleep mode (Sleep), PCA continues to work<br>1: In sleep mode (Sleep), PCA stops working | 0 | R/W |
| 6:4 | - | reserved bits | 0x0 | - |
| 3:1 | CPS | Clock frequency division selection and clock source selection<br>000: PCLK/32<br>001: PCLK/16<br>010: PCLK/8<br>011: PCLK/4<br>100: PCLK/2<br>101: Timer0 overflow<br>110: Timer1 overflow<br>111: ECI external clock, clock PCLK four-frequency sampling | 0 | R/W |
| 0 | CFIE | PCA counter interrupt enable control signal<br>0: disable interrupt<br>1: enable interrupt | 0 | R/W |

Machine Translated by Google

### 14.5.3 Count Register (PCA_CNT)

Offset address: 0x008

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CNT | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 - | | reserved bit | 0x0 | - |
| 15:0 CNT | | The value of the timer counter:<br>CNT can be written only when PCA is stopped, otherwise writing is invalid | 0x0 | R/W |

### 14.5.4 Interrupt Clear Register (PCA_INTCLR)

Offset address: 0x00C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | CF | reserve | | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| | | | | | | | | WO | | | WO | WO | WO | WO | WO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserved bit | 0x0 | - |
| 7 | CF | PCA counter overflow flag is cleared:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |
| 6:5 | - | reserved bit | 0x0 | - |
| 4 | CCF4 | PCA counter module 4 compare/capture flag clear:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |
| 3 | CCF3 | PCA counter module 3 compare/capture flag clear:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |
| 2 | CCF2 | PCA counter module 2 compare/capture flag clear:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |
| 1 | CCF1 | PCA counter module 1 compare/capture flag clear:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |
| 0 | CCF0 | PCA counter module 0 compare/capture flag clear:<br>0: Writing 0 is invalid<br>1: Software writes 1 to clear the corresponding flag | 0 | WO |

### 14.5.5 Compare Capture Mode Register (PCA_CCAPM0~4)

Offset address:

ÿ CCAPM0: 0x010

ÿ CCAPM1: 0x014

ÿ CCAPM2: 0x018

ÿ CCAPM3: 0x01C

ÿ CCAPM4: 0x020

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | ECOM | CAPP | CAPN | MAT | TOG | PWM | CCIE |
| | | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:7 | - | Reserved bits | 0x00 | - |
| 6 | ECOM | Enable comparator function control bits:<br>0: disable the stronger function<br>1: Comparator function enabled<br>When PCA is used for software counter, high-speed output and PWM mode, set ECOM;<br>When writing CCAMPHx or CCAMPx register will automatically set ECOM;<br>When writing the CCAMPLx register will automatically clear the ECOM bit; | 0 | R/W |
| 5 | CAPP | Positive edge capture control bit:<br>0: disable rising edge capture<br>1: Enable rising edge capture | 0 | R/W |
| 4 | CAPN | negative edge capture control bit<br>0: disable falling edge capture<br>1: Enable falling edge capture | 0 | R/W |
| 3 | MAT | Allow match control bits:<br>0: disable matching function;<br>1: Once the PCA count value matches the value of the compare/capture register of the module, the bit of the CR register will be set. Interrupt flag CCFx | 0 | R/W |
| 2 | TOG | Toggle control bits:<br>0: disable flip function<br>1: When working in the PCA high-speed output mode, the value of the PCA counter and the comparison/capture register of the module Once the value of the PCA_CHx pin is matched, the PCA_CHx pin is toggled | 0 | R/W |
| 1 | PWM | PWM control bits:<br>0: disable PWM pulse width modulation function<br>1: Enable PCA_CHx pin as PWM output<br>PWM function is valid only when CCAPMx[6:0]=100 0010'b | 0 | R/W |
| 0 | CCIE | PCA enable interrupt:<br>0: PCA compare/capture function interrupt disabled<br>1: Enable compare/capture interrupt | 0 | R/W |

### 14.5.6 Compare the lower 8 bits of the capture data register (PCA_CCAP0~4L)

Offset address:

ÿ CCAP0L: 0x030

ÿ CCAP1L: 0x038

ÿ CCAP2L: 0x040

ÿ CCAP3L: 0x048

ÿ CCAP4L: 0x050

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | CCAPx[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserved bit | 0x0 | - |
| 7:0 | CCAPx[7:0] | Compare/capture mode lower 8-bit register:<br>When PCA mode is used in compare/capture mode, it is used to save the lower 8 bits of the 16-bit capture count value bit; writing to the CCAPxH register automatically clears the ECOM bit in the CCAPMx register. when When PCA mode is used in PWM mode, it is used to control the output duty ratio compare register, in In PWM mode, the value of the lower 8 bits of the counter is less than the value of CCAPx[7:0] PWM output Output low level, otherwise PWM output high level. | 0x0 | R/W |

### 14.5.7 Compare the upper 8 bits of the capture data register (PCA_CCAP0~4H)

Offset address:

ÿ CCAP0H: 0x034

ÿ CCAP1H: 0x03C

ÿ CCAP2H: 0x044

ÿ CCAP3H: 0x04C

ÿ CCAP4H: 0x054

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | CCAPx[15:8] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 | - | reserved bit | 0x0 | - |
| 7:0 | CCAPx[15:8] | Compare/capture mode high 8-bit register:<br>When PCA mode is used in compare/capture mode, it is used to hold the high bit of the 16-bit capture count value 8 bits, writing to the CCAPxH register will automatically set the ECOM bit in the CCAPMx register.<br>When PCA mode is used in PWM mode, it is used to control the output duty cycle load register, When the lower 8 bits of the counter overflow, the load register will be automatically updated to the PWM compare register | 0x0 | R/W |

### 14.5.8 Compare High Speed Output Flag Register (PCA_CCAPO)

Offset address: 0x058

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | CCAPO4 | CCAPO3 | CCAPO2 | CCAPO1 | CCAPO0 |
| | | | | | | | | | | | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:5 | - | reserved bit | 0x0 | - |
| 4 | CCAPO4 | High-speed mode comparison module 4 output value<br>0: output 0<br>1: output 1 | 0 | R/W |
| 3 | CCAPO3 | High-speed mode compare module 3 output value<br>0: output 0<br>1: output 1 | 0 | R/W |
| 2 | CCAPO2 | High-speed mode compare module 2 output value<br>0: output 0<br>1: output 1 | 0 | R/W |
| 1 | CCAPO1 | High-speed mode compare module 1 output value<br>0: output 0<br>1: output 1 | 0 | R/W |
| 0 | CCAPO0 | High-speed mode compare module 0 output value<br>0: output 0<br>1: output 1 | 0 | R/W |

### 14.5.9 Terminal Output Control Register (PCA_POCR)

Offset address: 0x05C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | POIN V4 | POIN V3 | POIN V2 | POIN V1 | POIN V0 | reserve | | | POE4 | POE3 | POE2 | POE1 | POE0 |
| | | | R/W | R/W | R/W | R/W | R/W | | | | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:13 - | | reserved bit | 0x00 | - |
| 12 | POINV4 | Output polarity inversion of compare channel 4<br>0 = disable PWM0 output polarity inversion<br>1 = Enable PWM0 output polarity inversion | 0 | R/W |
| 11 | POINV3 | Comparator Channel 3 Output Polarity Inversion<br>0 = disable PWM0 output polarity inversion<br>1 = Enable PWM0 output polarity inversion | 0 | R/W |
| 10 | POINV2 | Comparator channel 2 output polarity inversion<br>0 = disable PWM0 output polarity inversion<br>1 = Enable PWM0 output polarity inversion | 0 | R/W |
| 9 | POINV1 | Output polarity inversion of compare channel 1<br>0 = disable PWM0 output polarity inversion<br>1 = Enable PWM0 output polarity inversion | 0 | R/W |
| 8 | POINV0 | Output polarity inversion of compare channel 0<br>0 = disable PWM0 output polarity inversion<br>1 = Enable PWM0 output polarity inversion | 0 | R/W |
| 7:5 | - | reserved bit | 0x0 | - |
| 4 | POE4 | Compare channel 4 output enable<br>0: output disabled<br>1: output enable | 0 | R/W |
| 3 | POE3 | Compare channel 3 output enable<br>0: output disabled<br>1: output enable | 0 | R/W |
| 2 | POE2 | Output enable for compare channel 2<br>0: output disabled<br>1: output enable | 0 | R/W |
| 1 | POE1 | Output enable for compare channel 1<br>0: output disabled<br>1: output enable | 0 | R/W |
| 0 | POE0 | Output enable for compare channel 0<br>0: output disabled<br>1: output enable | 0 | R/W |

### 14.5.10 Compare Capture 16-bit Register (PCA_CCAP0~4)

Offset address:

ÿ CCAP0: 0x060

ÿ CCAP1: 0x064

ÿ CCAP2: 0x068

ÿ CCAP3: 0x06C

ÿ CCAP4: 0x070

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | reserved bit | 0x0 | - |
| 15:0 | CCAPx | Compare/capture mode 16-bit register:<br>When PCA is used in compare/capture mode, it is used to save the 16-bit capture count value; write CCAPx register register will set the ECOM bit in register CCAPMx. Writing to the CCAPx register is equivalent to writing to the CCAPxL and CCAPxH are two 8-bit registers. In compare/capture mode, it is possible to directly read Write to this register; in PWM mode, use the CCAPxL and CCAPxH registers. | 0x0 | R/W |

# 15 base timer Base Timer (TIM10/TIM11)

## 15.1 Introduction to Base Timer

The base timer BaseTimer contains two timers TIM10/11. The TIM10/11 functions exactly the same. TIM10/11 is synchronous timing /counter, can be used as a 16/32-bit timer/counter with auto-reload function, or as a 16/32-bit timer without reload function /counter. TIM10/11 can count external pulses or implement system timing.

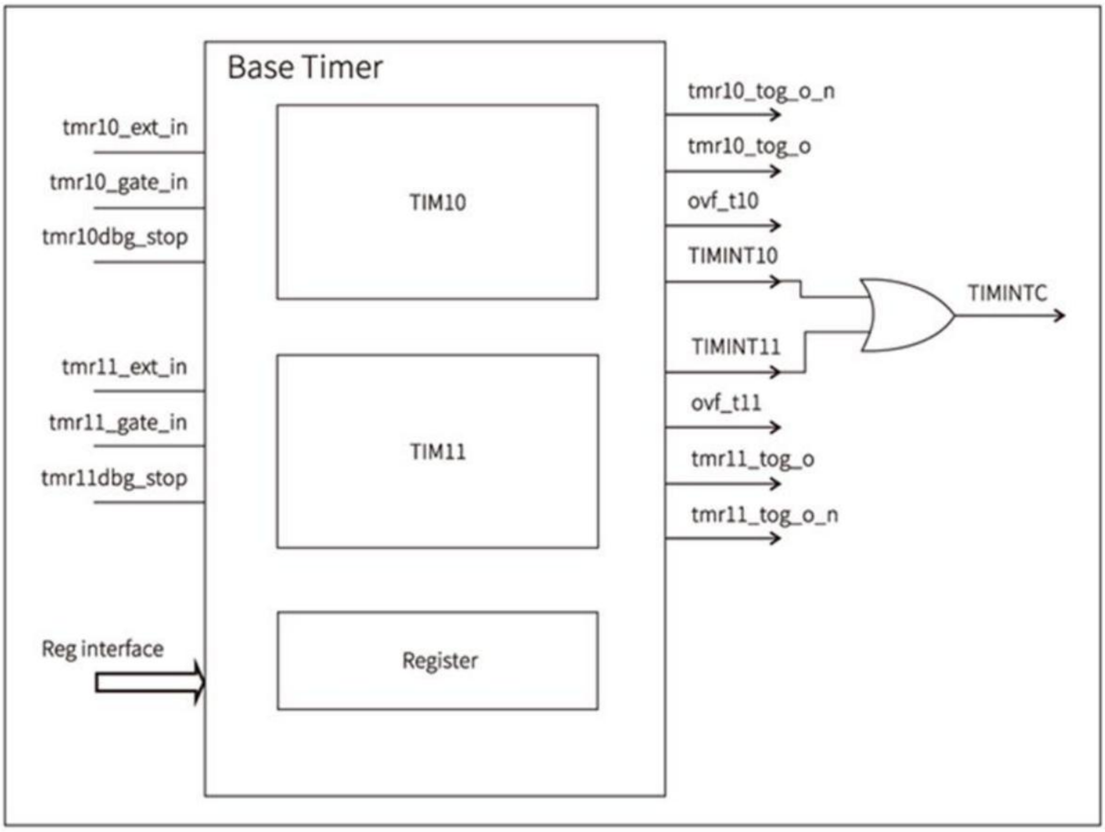


Figure 15-1 Base Timer block diagram

## 15.2 Base Timer Function Description

Each TIM10/11 timer/counter has an independent control start signal, external input clock and gating signal.

When TIM10/11 uses EXT and GATE for counting function, EXT is used for the external input clock signal of the counter, and GATE is used for Active level count enable signal. When the gating function is enabled, the counter will count only when the external input GATE level is valid. Otherwise the counter is held. Gate enable is controlled using TIMx_CR.GATE_EN. The default gating function is off. Gating level selection Select to use TIMx_CR.GATE_P control. The default high level is the gate active level; after setting TIMx_CR.GATE_P to 1, the gate is low The level is the active level.

When TIM10/11 uses PCLK and GATE for timing function, PCLK is used for the internal input clock signal of the timer, and GATE can Used for active level timing enable signals. When the gating function is enabled, the timer will count only when the external input GATE level is valid. Otherwise, the timer is in the stop state of the timing counter. Gate enable is controlled using TIMx_CR.GATE_EN. The default gating function is off. Gate level selection is controlled using TIMx_CR.GATE_P. The default high level is the effective level of gate control; when it is set to 1, the effective level of gate control is low level. Timing function can be configured with pre-divider. TIMx_CR.TMR_PRE controls the divider ratio.

| TMR_PRE [2:0] Frequency | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| division ratio | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |

When TIM10/11 uses PCLK and GATE for timing function, PCLK is used for the internal input clock signal of the timer, and GATE can

TIM10/11 supports both timer/counter functions, which can be configured by setting CT_SEL in the timer control register (TIMx_CR). Every

This function supports 2 modes, mode 1 is 16/32-bit free count mode, mode 2 is 16/32-bit reload mode.

In mode 1 free count mode

When counting to the maximum value (16-bit Max=0xFFFF, 32-bit maximum value is 0xFFFFFFFF), an interrupt will be generated after overflow, and the timer/counter will be cleared.

Then keep counting.

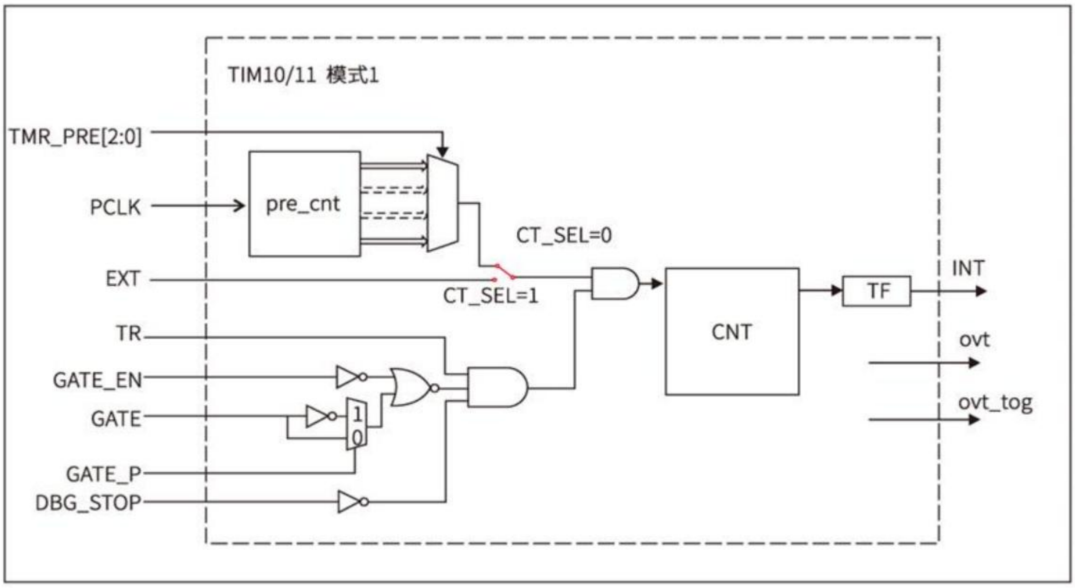


Figure 15-2 Timer mode 1 block diagram

Mode 2 reload mode

Reload mode, overflow after counting to the maximum value, an interrupt is generated, the value of the timer/counter is loaded as the value of BGLOAD, and then continues to

Count up. In the heavy load mode, the software processing speed needs to be considered when the timing time is set small, otherwise the interrupt will not be processed in time

cause the interrupt to be lost.

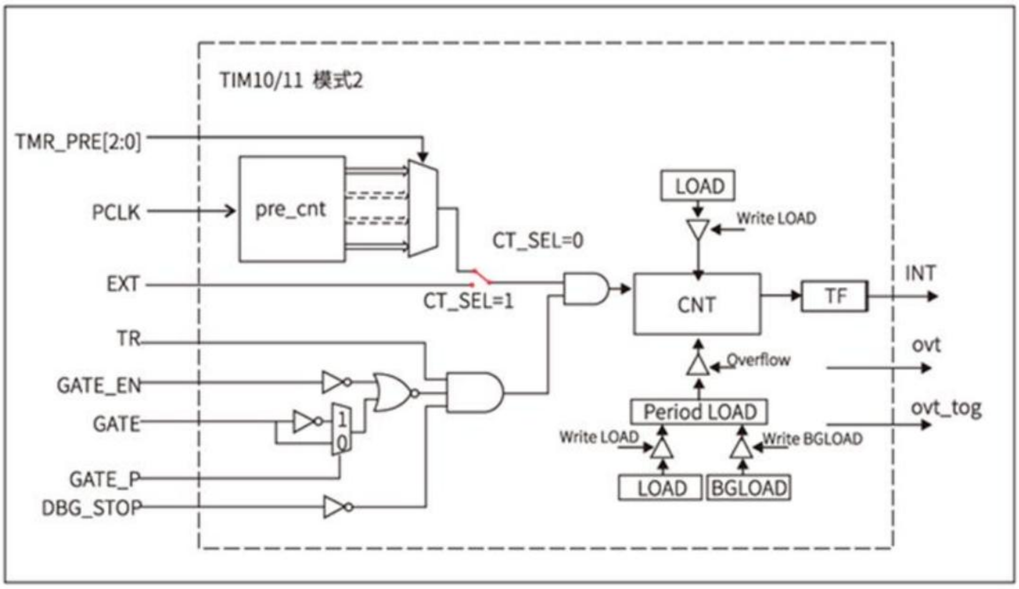


Figure 15-3 Timer mode 2 block diagram

When the corresponding timer TIMx_CR.TR is set to 1, the timer starts to run. Mode 1, start counting from the initial value set by the register after startup

After counting to the maximum, an overflow interrupt will be generated, and then continue counting from 0. Mode 2, count up from the initial value of the register CNT after startup

After counting to the maximum value, an interrupt is generated, and the value of the reload register BGLOAD is transferred to the counter CNT, and continues to count up. whether it is

In free counting mode or reload mode, as long as the LOAD value is written, the value of the timer/counter will be updated immediately, and then continue to go up

count.

### 15.2.1 Counting function

Counting functions are used to determine the number of times an event occurs. In the counting function, the counter falls on each corresponding input clock (EXT)

Accumulate along once. The input signal is sampled by the internal PCLK, so the external input clock frequency cannot exceed the system PCLK clock. count

Counting to the maximum value will overflow and generate an interrupt. The interrupt flag needs to be cleared by software.

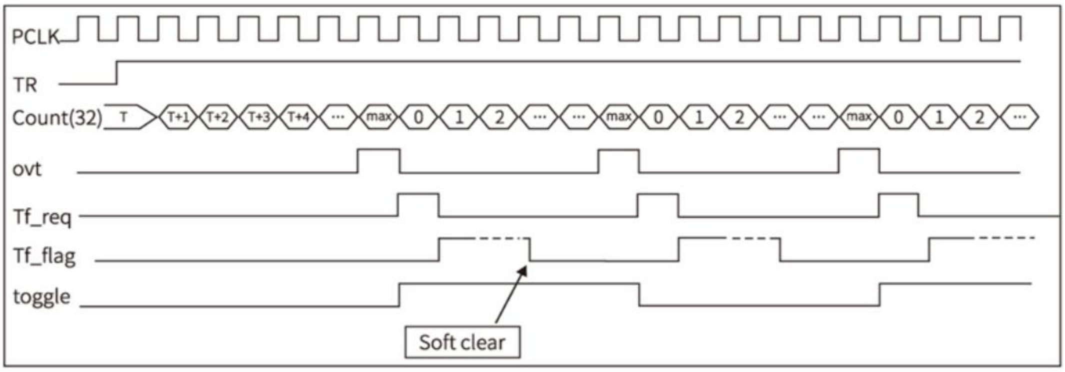


Figure 15-4 Timing diagram of 32-bit mode 1 (max=0xFFFF FFFF)

### 15.2.2 Timing function

The timing function is used to generate interval timing. In the timing function, the timer has a pre-divided frequency, and the timer accumulates a clock after each pre-divided frequency.

Add once, counting to the maximum value will overflow and generate an interrupt. The interrupt flag needs to be cleared by software.

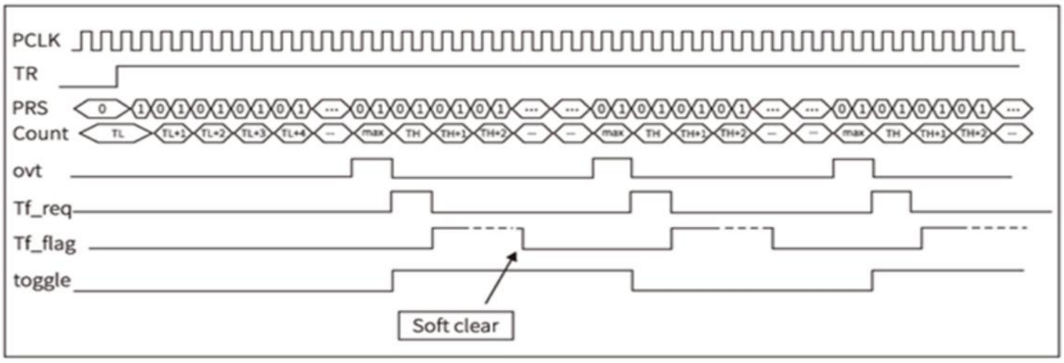


Figure 15-5 Timing diagram of 32-bit mode 2 (PCLK divided by two, max=0xFFFF FFFF)

### 15.2.3 Buzzer function

The function of driving the Buzzer can be realized through the flipping output function of the timer. When TIMx_CR.TOG_EN is 1, TOG, TOGN output

out reverse. Setting TIMx_CR.TOG_EN to 0 can set the port TOG and TOGN output to 0 at the same time. The counting clock is 4M

In this case, the Buzzer outputs the Timer reload mode with different frequencies. The configuration is as follows (16-bit Max=0xFFFF):

| Buzzer frequency | count period | count value | Overload value | CNT initial value | LOAD reload value |
|---|---|---|---|---|---|
| 1KHz | 0.5ms | 2000 | 63536 | 0xF830 | 0xF830 |
| 2KHz | 0.25ms | 1000 | 64536 | 0xFC18 | 0xFC18 |
| 4KHz | 0.125ms | 500 | 65036 | 0xFE0C | 0xFE0C |

## 15.3 Base Timer Interconnect

### 15.3.1 GATE interconnection

The GATE input can be directly input from the port, or the RX signal of UART/LPUART; it can also be configured as the output of VC as GATE signal. The GATE of TIM10/11 can be configured.

Through the internal interconnection configuration, the automatic identification of the UART baud rate can be realized, the pulse width of the VC comparison output can be measured, and the External control count.

The configuration selection RX input is controlled in the SYSCON_PORTCR register, and the VC control is controlled in the VC_OUTCFG register. port selection At this time, only one of UART/LPUART input selection and VC output selection can be selected as the gating input to be effective. VC's output selection priority Highest.

### 15.3.2 Toggle Output Interconnects

The flip output of TIM10 tmr10_tog_o is sent to the internal module UART0 to control the baud rate of UART0; the flip output of TIM11 tmr11_tog_o to the internal module UART1 to control the baud rate of UART1; the flip output of TIM10/11 is also output to the port, which can be To drive the Buzzer to realize the control of the buzzer.

## 15.4 Base Timer Register List

x=10 or 11;

Base Timer base address 0x 4000 1800

| | offset address | describe |
|---|---|---|
| TIM10 | 0x00 | TIM0 offset address |
| TIM11 | 0x100 | TIM1 offset address |

Table 15-1 Base Timer register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | TIMx_CR | control register | 0x0000 0000 |
| 0x04 | TIMx_LOAD | 32-bit immediate reload register | 0x0000 0000 |
| 0x08 | TIMx_CNT | Read counter register, read only | 0x0000 0000 |
| 0x0C | TIMx_RAWINTSR | read raw interrupt register, | 0x0000 0000 |
| 0x10 | TIMx_MSKINTSR | Read Interrupt Register | 0x0000 0000 |
| 0x14 | TIMx_INTCLR | Interrupt Clear Register | 0x0000 0000 |
| 0x18 | TIMx_BGLOAD | 32-bit period reload register | 0x0000 0000 |

## 15.5 Base Timer Register Description

### 15.5.1 Control Register (TIMx_CR)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | GATE _P | GATE _EN | TOG _EN | CT_S EL | TR | MOD E. | INTE N | TMR _SIZ E. | ONE SHO T | TMR_PRSC[2:0] | | |
| | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | |

| bit symbol | | describe | Reset value read and write | |
|---|---|---|---|---|
| 31:12 - | | Reserved bit, read as 0 | 0x0 | - |
| 11 | GATE_P | Port GATE polarity control, the default high level GATE is valid, set to 1 and then low level efficient<br>0: active high<br>1: Active low | 0 | R/W |
| 10 | GATE_EN | Timer Gating<br>0: No gate control, the timer works when TR=1;<br>1: Only work when the port GATE is valid and TR=1; | 0 | R/W |
| 9 | TOG_EN | TOG output enable<br>0: TOG, TOGN output 0 at the same time<br>1: TOG and TOGN output signals with opposite phases. Available for buzzers. | 0 | R/W |
| 8 | CT_SEL | Counter/timer function selection<br>0: Timer function, the timer is counted by PCLK.<br>1: Counter function, the counter counts by the falling edge of the external input. external input Sampled by PCLK, the external input clock frequency is lower than 1/2 sampling clock. | 0 | R/W |
| 7 | TR | Timer run control<br>0: timer stopped<br>1: Timer running | 0 | R/W |
| 6 | MODE | Timer working mode<br>0: mode 1 counter/timer<br>1: Mode 2 auto-reload counter/timer | 0 | R/W |
| 5 | INTEN | Interrupt enable control, enable interrupt after writing 1 | 0 | R/W |
| 4 | TMR_SIZE | TimerSize=0: max count value=0xFFFF;<br>TimerSize=1: max count value=0xFFFFFFFF; | 0 | R/W |
| 3 | ONE SHOT | Counter run once enable<br>0: repeat mode<br>1: oneshot mode | 0 | R/W |
| 2:0 | TMR_PRSC[2:0] | TIM prescaler selection.<br>000: frequency division number 1;<br>001: frequency division number 2;<br>010: frequency division number 4;<br>011: frequency division number 8;<br>100: frequency division number 16;<br>101: frequency division number 32;<br>110: frequency division number 64;<br>111: frequency division number 128 | 0x0 | R/W |

### 15.5.2 Immediate Reload Register (TIMx_LOAD)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOAD[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit symbol | | describe | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | LOAD[31:0] | reload registers immediately<br>Writing this register will immediately update the value of the counter register CNT<br>Note: Read TIMx_LOAD and TIMx_BGLOAD, you can read the latest updated LOAD or<br>or the value of the BGLOAD register | 0x0 | R/W |

### 15.5.3 Counter Register (TIMx_CNT)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT [31:16] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

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

| bit symbol | | describe | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | CNT[31:0] | Counter register | 0x0 | RO |

### 15.5.4 Raw Interrupt Status Register (TIMx_RAWINTSR)

Offset address: 0x0c

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit symbol | | describe | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | - | Reserved bit, read as 0 | 0x0 | - |
| 0 | RIS | Interrupt flag, set by hardware; no matter IntEnable=0 or 1, interrupt can be read | 0 | RO |

### 15.5.5 Interrupt Flag Register (TIMx_MSKINTSR)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit | symbol | description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | reserved | reserved bits, read as 0 | 0x0 | - |
| 0 | TF | INTEN=1, the interrupt register can be read | 0 | RO |

### 15.5.6 Interrupt Clear Register (TIMx_INTCLR)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit | symbol | description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | reserved | reserved bits, read as 0 | 0x0 | - |
| 0 | INTCLR | interrupt flag clear, write 1 to clear, write 0 to invalid | 0 | WO |

## 15.5.7 Period Reload Register (TIMx_BGLOAD)

Offset address: 0x018

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BGLOAD[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| bit | symbol | describe | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | BGLOAD[31:0] | BackGround cycle reload register, writing this register will not update the counter immediately The value of register CNT. Only when the CNT value overflows, the value of BGLOAD will be reloaded to the CNT register.<br>Note: read TIMx_LOAD and TIMx_BGLOAD, you can read the latest update LOAD or BGLOAD register value | 0x0 | R/W |

# 16 Low Power Timer (LPTIM)

LPTIM is an asynchronous 16-bit timer/counter that can still be clocked by an internal low-speed RC clock or an external low-speed crystal after the system clock is turned off.

Oscillating clock timing/counting, system wake-up in low power mode by interrupt.

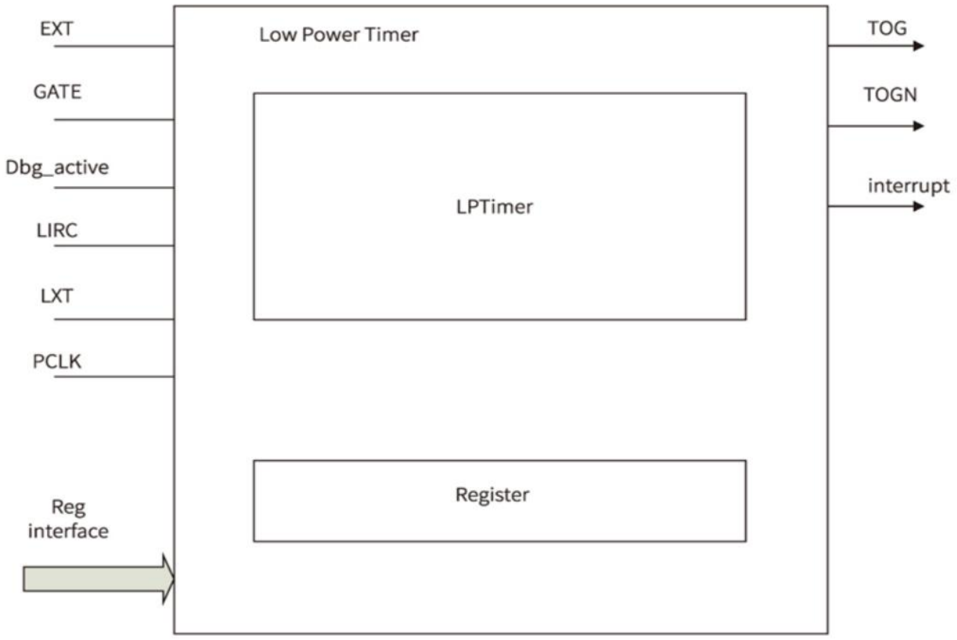


Figure 16-1 LPTIMER block diagram

## 16.1 LPTIM Functional Description

LPTIM has independent control start signal, external input clock and gating signal.

LPTIM uses EXT and GATE for counting function, EXT is used for the external input clock signal of the counter, GATE is the active level

Count enable signal.

The timer of LPTIM supports two working modes, and the working mode can be selected by setting the MODE in the timer control register (LPTIM_CR).

Mode.

ÿMode 1 is a 16-bit free counting mode, the counter starts counting from the value set by LPTIM_LOAD, and the counting value starts from

0x0000 Start counting again.

ÿ Mode 2 is a 16-bit reload mode. When LPTIM starts, it will automatically load the value of the reload register LPTIM_LOAD into the counter.

When it overflows, it will automatically load the value of Period LOAD into the counter. User writes LPTIM_LOAD or

LPTIM_BGLOAD will update the value of Period LOAD, and the value of Period LOAD is the user's last update

The value of LPTIM_LOAD or LPTIM_BGLOAD.

LPTIM can choose three kinds of clocks as the timer clock, which can be selected through the control register LPTIM_CR.TCK_SEL. default selection

PCLK. The clock selection is as shown in the table:

| TCK_SEL | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| timer clock | PCLK | PCLK | LXT | LIRC |
| Read timer count value | read synchronized | no sync | read synchronized | read synchronized |

After selecting the corresponding clock source, and then setting TCK_EN to 1, the counting clock source of the counter can be turned on.

After the clock source is selected and turned on, set the TIM_RUN of the corresponding timer to 1 and the timer starts running.

For mode 1 and mode 2, if the LOAD value is set, the count value will be updated to the LOAD value immediately at any time, and the counter starts from

The LOAD value starts counting again. Setting the LOAD value takes precedence over other counters being updated.

In mode 2, set the value of BGLOAD, and the set value will be updated to the counter only after the counter overflows.

Mode 1: If the LOAD value is not set, the counter starts counting from 0, and an overflow interrupt is generated after counting to a maximum of 0xFFFF. counter meter

After counting to a maximum of 0xFFFF, the counter starts counting from 0 again.

Mode 2: If the LOAD value is not set, the counter starts counting from 0, and an overflow interrupt is generated after counting to a maximum of 0xFFFF. counter meter

After counting to the maximum 0xFFFF, the count value will be updated to the value of Period LOAD, and then count up.

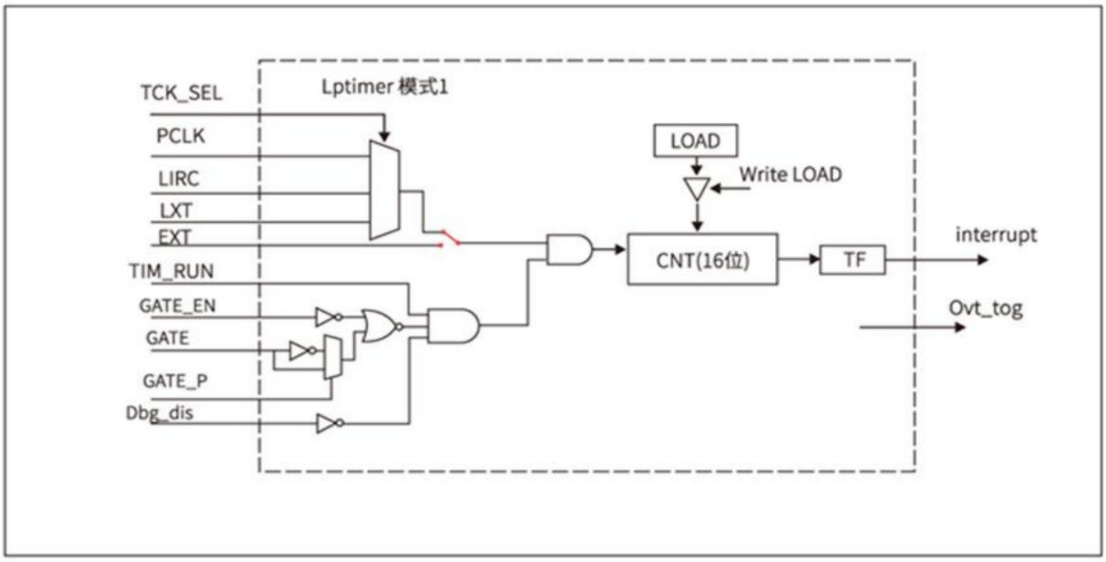


Figure 16-2 LPTIMER mode 1

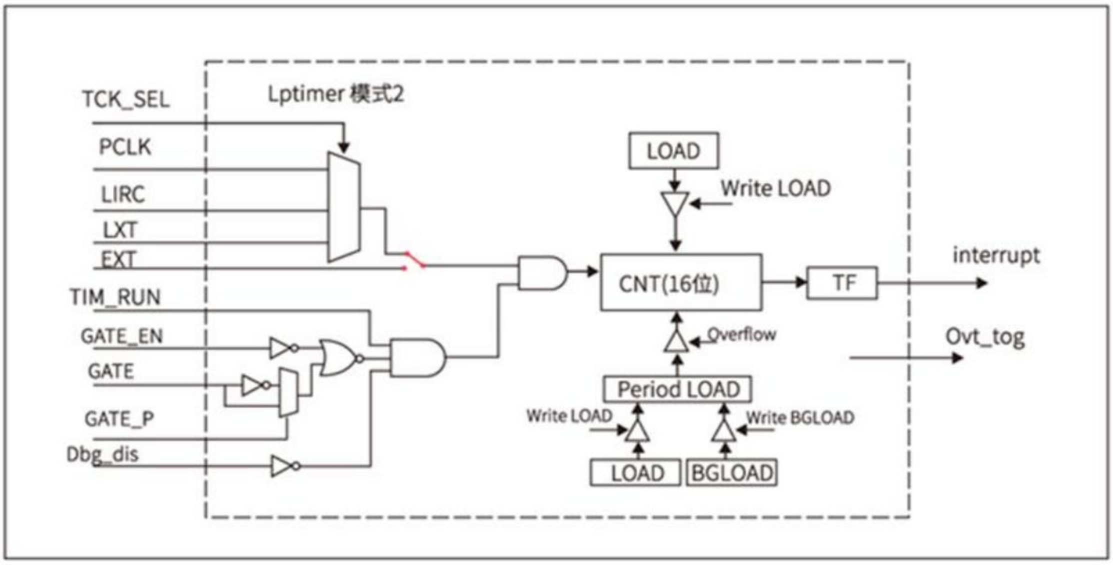


Figure 16-3 LPTIMER mode 2

### 16.1.1 Counting function

Counting functions are used to determine the number of times an event occurs. In the count function, the counter increments on each rising edge of the corresponding input clock once. The input signal is sampled by the internal count clock, so the external input clock frequency cannot exceed the system count clock. count up to Large values will overflow and generate an interrupt.

### 16.1.2 Timing function

The timing function is used to generate interval timing. In the timing function, the timer accumulates once per clock, and when it counts to the maximum value, it will overflow and generate Student interruption.

## 16.2 LPTIM Interconnect

### 16.2.1 GATE interconnection

The GATE input can be directly input from the port, or the RX signal of the UART can be input;

Through the internal interconnection configuration, the automatic identification of the UART baud rate can be realized, the pulse width of the VC comparison output can be measured, and the External control count.

The configuration selection RX input is controlled in the SYSCON_PORTCR register, and the VC control is controlled in the VC control register.

### 16.2.2 EXT interconnection

EXT input can be directly input from the port, and also configured as VC input as EXT signal.

With the interconnect configuration, VC pulse counts can be measured. The VC output control register is in the VC control block.

### 16.2.3 TOGGLE output interconnect

The inverted output of LPTIM is output to the port, which can drive BUZZER to realize buzzer control.

## 16.3 List of LPTIM registers

Base address: 0x4000 4400

Table 16-1 LPTIM register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | LPTIM_CNTVAL | LPTIM count value read-only register | 0x0000 0000 |
| 0x04 | LPTIM_CR | LPTIM Control Register | 0x0000 0000 |
| 0x08 | LPTIM_LOAD | LPTIM immediate reload register | 0x0000 0000 |
| 0x0C | LPTIM_INTSR | LPTIM Interrupt Register | 0x0000 0000 |
| 0x10 | LPTIM_INTCLR | LPTIM Interrupt Clear Register | 0x0000 0000 |
| 0x14 | LPTIM_BGLOAD | LPTIM Period Reload Register | 0x0000 0000 |

## 16.4 LPTIM register description

### 16.4.1 LPTIM count value read-only register (LPTIM_CNTVAL)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | – | reserve | 0x0 | – |
| 15:0 | LPT_CNT[15:0] | Count value read-only register | 0x0 | RO |

## 16.4.2 LPTIM Control Register (LPTIM_CR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | WT_<br>FLAG |
| | | | | | | | | | | | | | | | RO |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | TCK_<br>EN | INT_<br>EN | GATE<br>_POL<br>E. | GATE | TCK_SEL[1:0] | | TOG<br>_EN | CT_S<br>EL | MOD<br>E. | time<br>R_R<br>UN |
| | | | | | | R/W | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:17 | – | reserve | 0x0 | – |
| 16 | WT_FLAG | WT write synchronization flag<br>0: Synchronization is complete, you can change LOAD/BGLOAD at this time<br>1: Synchronizing, writing LOAD/BGLOAD is invalid at this time | 0x0 | RO |
| 15:10 | – | reserve | 0x0 | – |
| 9 | TCK_EN | LPTIM count clock enable<br>0: LPTIM count clock off<br>1: LPTIM count clock enable<br>LOAD/BGLOAD can be configured only after the count clock is enabled | 0 | R/W |
| 8 | INT_EN | Interrupt enable control, enable interrupt after writing 1 | 0 | R/W |
| 7 | GATE_P | The effective polarity of the input GATE<br>By default, the high level GATE is valid, and when it is set to 1, the low level | 0 | R/W |
| 6 | GATE_EN | active timer gates<br>0: no gating<br>1: gated | 0 | R/W |
| 5:4 | TCK_SEL[1:0] | LPTIM clock selection<br>00: PCLK;<br>10: LXT;<br>11: LIRC | 0x0 | R/W |
| 3 | TOG_EN | TOG output enable<br>0: TOG, TOGN output 0 at the same time<br>1: TOG, TOGN output signals with opposite phases. Available for BUZZER. | 0 | R/W |
| 2 | CT_SEL | Counter/timer function selection<br>0: Timer function, the timer uses the clock selected by TCK_SEL to count.<br>1: Counter function, the counter uses the falling edge of the external input to count. when sampling The clock uses the clock selected by TCK_SEL, and the external input clock is lower than 1/2 sampling time bell. | 0 | R/W |
| 1 | MODE | Timer working mode<br>0: Mode 1 No reload mode 16-bit counter/timer<br>1: Mode 2 auto-reload 16-bit counter/timer | 0 | R/W |
| 0 | TIM_RUN | Timer run control bit<br>0: timer stopped<br>1: Timer running | 0 | R/W |

### 16.4.3 LPTIM immediate reload register (LPTIM_LOAD)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| LOAD | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | reserve | 0x0 | – |
| 15:0 | LOAD[15:0] | immediate reload register<br>It has nothing to do with whether the Timer is running or not, and it has nothing to do with the MODE.<br>Need to read LPTIM_CR.WT_FLAG before writing LOAD, if and only if WT_FLAG is 0 to write data. WT_FLAG will go low after writing the LOAD register is completed. write the post The value of the counter is updated immediately when the memory is changed.<br>When reading this register, it returns the latest value updated to LOAD or BGLOAD | 0x0 | R/W |

### 16.4.4 LPTIM Interrupt Register (LPTIM_INTSR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | | | | | | | | INTF |
| | | | | | | | | | | | | | | | RO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | INTF | Interrupt flags:<br>0: No interrupt occurred<br>1: An overflow interrupt occurred | 0x0 | RO |

### 16.4.5 LPTIM Interrupt Register (LPTIM_INTCLR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | | | ICLR |
| | | | | | | | | | | | | | | | WO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | ICLR | Write 1 to clear the interrupt flag, write 0 to have no effect | 0x0 | WO |

### 16.4.6 LPTIM Period Reload Register (LPTIM_BGLOAD)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BGLOAD[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | – | reserve | 0x0 | – |
| 15:0 | BGLOAD[15:0] | BackGround cycle reload register<br>Need to read LPTIM_CR.WT_FLAG before writing BGLOAD, if and only if<br>Data can only be written when WT_FLAG is 0. After writing the BGLOAD register is complete<br>WT_FLAG will go low.<br>When writing this register, when the counter overflows, the set value will be updated to the counter.<br>Reading this register returns the value last updated to LOAD or BGLOAD. | 0x0 | R/W |

# 17 Self-wake-up timer (AWK)

CX32L003 has a dedicated self-wake-up timer (AWK), which provides a wake-up event reference for the chip in low-power mode. AWK

Keep count in Sleep or Deep Sleep mode. When AWK is used as a wake-up timer, AWK should be turned on before entering power-saving mode.

AWK can configure the internal low-speed clock source LIRC, the external low-speed clock source LXT, and the frequency-divided clock of the external high-speed clock HXT. attention system

The system clock frequency must be more than twice the AWK clock. If AWK starts counting, when the device enters Sleep or DeepSleep mode

, the selected clock source will also keep working. Note that the selected AWK clock source will not be automatically enabled together with the AWK configuration, the user should

This manually enables the selected clock source and waits for it to stabilize to ensure successful operation.

AWK is equipped with a simple 8-bit auto-reload count-up timer. Its prescaler can be selected from 1/2 to 1/65536, through

AWK_CR.DIVSEL[3:0] to set. User fills reload value into AWK_RLOAD register to determine its overflow rate.

After AWK_CR.AWKEN is set, when the CPU enters Sleep/Deep Sleep mode, load the value of AWK_RLOAD register to the internal

8-bit counter and start counting. When the counter overflows, AWK_SR.AWUF is set to 1 to wake up the CPU.

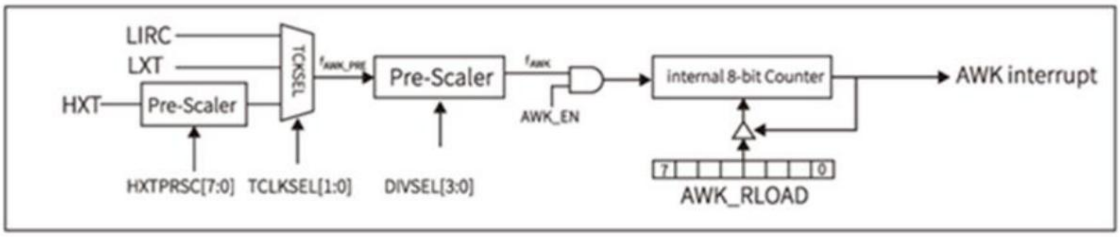


Figure 17-1 Self-wake-up timer structure diagram

## 17.1 Register List

Base address: 0x4000 1C00

Table 17-1 AWK register list and reset value

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x000 | AWK_CR | Self-wake-up timer control register | 0x0000BC00 |
| 0x004 | AWK_RLOAD | Self-wake-up timer reload data register | 0x0000 0000 |
| 0x008 | AWK_SR | Self-wake-up timer status register | 0x0000 0000 |
| 0x00C | AWK_INTCLR | Self-wake-up interrupt clear register | 0x0000 0000 |

## 17.2 Register description

### 17.2.1 Self-wake-up timer control register (AWK_CR)

Address offset: 0x00

Reset value: 0x0000 BC00

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserve</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">XTLPRSC[7:0]</td><td rowspan="2">reserve</td><td colspan="2">TCLKSEL[1:0]</td><td>AWK<br>EN</td><td colspan="4">DIVSEL[3:0]</td></tr>
<tr><td colspan="8">R/W</td><td colspan="2">R/W</td><td>R/W</td><td colspan="4">R/W</td></tr>
</table>

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 - | | reserve | 0x0 | – |
| 15:8 | HXTPRSC[7:0] | HXT clock frequency division factor: fHXT/(N+1) | 0xBC | R/W |
| 7 | - | reserve | 0x0 | – |
| 6:5 | TCLKSEL[1:0] | AWK counting clock source selection<br>00: stop<br>01: LIRC clock<br>10: Clock after HXT frequency division<br>11: LXT clock | 0x0 | R/W |
| 4 | AWKEN | AWK enabled<br>1: enable<br>0: disable | 0 | R/W |
| 3:0 | DIVSEL[3:0] | Counter clock source selection bit<br>0000: FAWK_PRE/2^1<br>0001: FAWK_PRE/2^2<br>0010: FAWK+PRE/2^3<br>…<br>1111: FAWK_PRE/2^16 | 0x0 | R/W |

### 17.2.2 Self-wake-up timer reload data register (AWK_RLOAD)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>twenty four</td><td>twenty three</td><td>twenty two</td><td>twenty one</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserve</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8" rowspan="2">reserve</td><td colspan="8">RLDVAL[7:0]</td></tr>
<tr><td colspan="8">R/W</td></tr>
</table>

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | RLDVAL[7:0] counter | | 0x0 | R/W |

### 17.2.3 Self-wake-up timer status register (AWK_SR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | AWUF | Automatic wake-up occurs, set to 1 by hardware, cleared by software<br>0: No automatic wakeup occurs<br>1: Counter overflow, automatic wake-up occurs | 0 | RO |

### 17.2.4 Self-wake-up interrupt clear register (AWK_INTCLR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | INTCLR | Automatic wake-up interrupt clear<br>0: no effect<br>1: Clear auto-wake-up interrupt | 0x0 | WO |

# 18 Buzzer (BEEP)

## 18.1 Introduction

Select LIRC clock, HXT clock or PCLK to generate buzzer signals of various frequencies through frequency division setting.

BEEP_CSR.CLKSEL[1:0] bits to select the fBEEP_PRE clock, set the buzzer clock by setting BEEPDIV[11:0]

fBEEP_PRE is divided to obtain fBEEP. By setting BEEPSEL[1:0], get fBEEP_O beep signal.

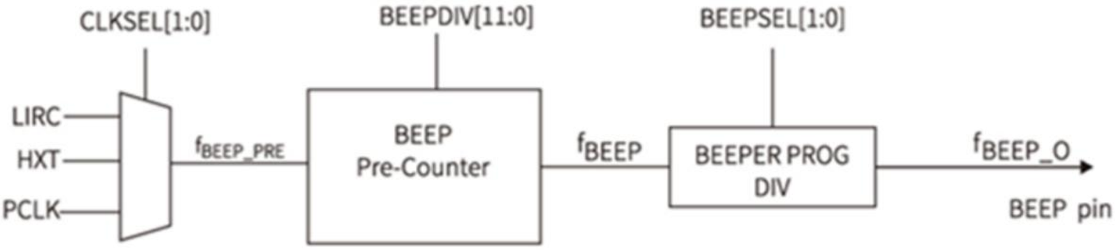


Figure 18-1 Buzzer function diagram

## 18.2 Functional Description

### 18.2.1 Buzzer Operation

In order to use the beep function, perform the following steps in order:

1. Determine the value of BEEPDIV[11:0] according to the required buzzer frequency output value;

2. Select the output frequency of fBEEP_O by writing the BEEPSEL[1:0] bits of BEEP_CSR ;

3. Set the BEEPEN bit of BEEP_CSR to enable the clock source;

Notice:

ÿThe prescaler counter starts to run only when the value of BEEPDIV[11:0] is different from the reset value 0x0FFF.

ÿ The value of BEEPDIV[11:0] should be kept unchanged during the buzzer operation.

### 18.2.2 Buzzer Calibration

This step can be used to calibrate the LIRC clock to achieve a more standard fBEEP_O (1KHz, 2KHz or 4KHz) frequency output.

Take the following steps:

1. TIM1, TIM2 or CLKTRIM module to measure the clock frequency of the internal low-speed clock (LIRC)

2. Calculate the value of BEEPDIV as follows, where A and x are the integer and fractional values of fBEEP_PRE/fBEEP: When x is small

When equal to or equal to A/(1+2*A), BEEPDIV = A-1; otherwise BEEPDIV = A

3. Write BEEPDIV value to BEEPDIV[11:0] bits of BEEP_CSR

## 18.3 Register List

BEEP base address: 0x4000 4800

Table 18-1 BEEP register list and reset value

| offset address name | | describe | reset value |
|---|---|---|---|
| 0x000 | BEEP_CSR | Buzzer Control Register | 0x0000 0FFF |

Machine Translated by Google

## 18.4 Register description

### 18.4.1 Beeper Control/Status Register (BEEP_CSR)

Address offset: 0x00

Reset value: 0x0000 0FFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | CLKSEL | | reserve | BEEPEN | BEEPSEL | |
| | | | | | | | | | | R/W | | | R/W | R/W | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | BEEPDIV | | | | | | | | | | | |
| | | | | R/W | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:22 | - | reserve | 0x0 | – |
| 21:20 | CLKSEL | Clock selection.<br>00: stop<br>01: LIRC<br>10: HXT<br>11: PCLK | 0x0 | R/W |
| 19 | - | reserve | | |
| 18 | BEEPEN | enable beep | 0 | R/W |
| 17:16 | BEEPSEL | Buzzer output BEEP_O frequency selection bit<br>00: fBEEP_PRE/8<br>01: fBEEP_PRE/4<br>1x: fBEEP_PRE/2 | 0x0 | R/W |
| 15:12 | - | reserve | 0x0 | – |
| 11:0 | BEEPDIV | The buzzer prescaler divides fBEEP_PRE to obtain a signal with a period of 8KHz<br>The division factor is BEEPDIV + 1<br>fBEEP = fBEEP_PRE/(BEEPDIV + 1) | 0xfff | R/W |

# 19 Independent Watchdog (IWDG)

## 19.1 Overview

The role of the IWDG is to prevent the software system from running abnormally, causing the system to work abnormally or crash. While the IWDG resets

Can help the system recover automatically. The working principle is that when the software system makes an error, a complex time is generated at a fixed time (this time can be configured).

Bit or interrupt, let the program re-execute or execute according to the interrupt service routine, so that the system will not crash. thereby increasing the software system

safety performance.

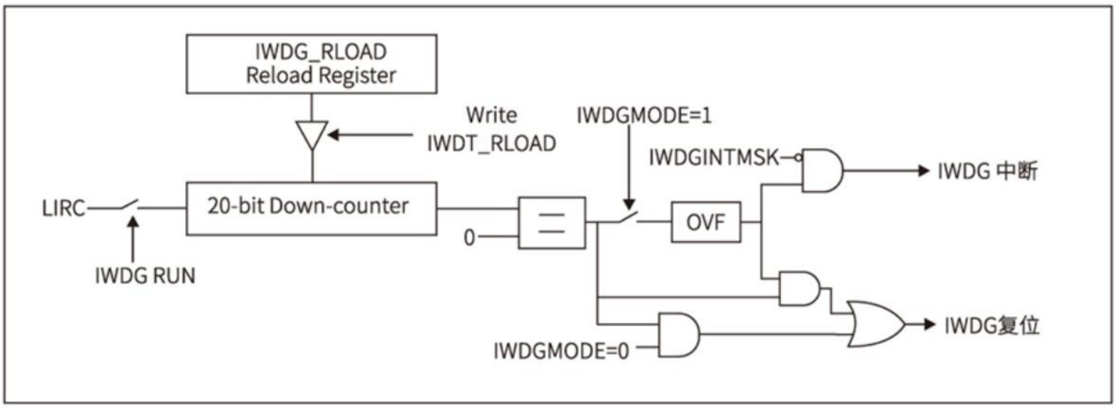


Figure 19-1 Overall block diagram of IWDG

## 19.2 Functions of the IWDG

ÿ The IWDG module is a 20-bit counter, each LIRC clock (assumed to be 38.4KHz) counts and accumulates once, and the counting time can be

The configuration is 26us-27s;

ÿ Internal low-speed clock (LIRC) counting;

ÿWhen the count overflows, it supports interrupt and reset;

ÿ Configurable count overflow time;

ÿStarting and clearing IWDG has operation sequence requirements to increase safety performance;

ÿ Partial register write protection function to prevent accidental operation of the program.

### 19.2.1 Timeout period

The watchdog timeout period is determined by the counter value, the following table lists their values

Table 18-1 Watchdog timeout period (assuming LIRC is 38.4KHz)

| IWDG_RLOAD[19:0] register configuration value | timeout period |
|---|---|
| 0x00000 | 26us |
| … | … |
| 0x003FF | 26.6ms |
| … | … |
| 0xFFFFF | 27s |

### 19.2.2 Interrupt after IWDG overflow

In this mode, IWDG will periodically generate interrupts according to the set time. Need to clear the IWDG overflow flag in the interrupt service routine Chi.

The configuration method is as follows:

1. Write 0x55AA6699 to IWDG_UNLOCK to release IWDG register write protection. If 2. IWDG_UNLOCK.IWDGREN is 1, this
step can be omitted. 3. Set IWDG_CFGR.IWDGMODE to 1 to select the
interrupt mode. 4. If IWDG_CFGR.IWDGINTMSK is set to 0, the CPU responds to the
interrupt signal of IWDG; if it is set to 1, the interrupt is
Shielded, the CPU can only judge whether the count overflows by reading IWDG_SR.IWDGOVF.
5. Configure the IWDG_RLOAD register. Select the IWDG count overflow time. 6. Write any
value other than 0x55AA6699 to IWDG_UNLOCK to enable IWDG register write protection. 7. Write 0x55 to IWDG_CMDCR
to start IWDG. 8. If an interrupt occurs, first release the IWDG
register protection in the interrupt service routine, and then write 1 to IWDG_INTCLR to clear it
break mark.

### 19.2.3 Reset after IWDG overflow

In this mode, after the IWDG counter overflows, a Reset signal will be generated, which will reset the MCU. Users need to Reloads the IWDG counter to avoid an IWDG reset.

The configuration method is as follows:

1. Write 0x55AA6699 to IWDG_UNLOCK to release IWDG register write protection. If 2. IWDG_UNLOCK.IWDGREN is 1, this
step can be omitted. 3. Set IWDG_CFGR.IWDGMODE to 0 to select the reset
mode. 4. Configure the IWDG_RLOAD register. Select the IWDG count overflow time.
5. Write any value other than 0x55AA6699 to IWDG_UNLOCK to enable IWDG register write
protection. 6. Write 0x55 to IWDG_CMDCR to start IWDG. 7. Write 0xAA to IWDG_CMDCR before the counter overflows to
refresh the IWDG counter.

## 19.3 Register List

Base address: 0x4000 2400

Table 19-1 AWK register list and reset value

| offset address | name | describe | Defaults |
|---|---|---|---|
| 0x00 | IWDG_CMDCR | IWDG Control Command Register | 0x0000 0000 |
| 0x04 | IWDG_CFGR | IWDG configuration register | 0x0000 0000 |
| 0x08 | IWDG_RLOAD | IWDG Counter Reload Register | 0x000F FFFF |
| 0x0c | IWDG_CNTVAL | IWDG counter value | 0x000F FFFF |
| 0x10 | IWDG_SR | IWDG Interrupt Status Register | 0x0000 0000 |
| 0x14 | IWDG_INTCLR | IWDG Interrupt Clear Register | 0x0000 0000 |
| 0x18 | IWDG_UNLOCK | IWDG register access protection | 0x0000 0000 |

## 19.4 Register description

### 19.4.1 IWDG Control Command Register (IWDG_CMDCR)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 – | | | 0x0 | – |
| 7:0 | CMD[7:0] | Reserved 0x55: IWDG start command<br>0xAA: IWDG reload refresh command | 0x0 | WO |

Note: The reload command can only be written when IWDG is running

### 19.4.2 IWDG Configuration Register (IWDG_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | IWDGRUNF | IWDGINTMS | IWDGMOsDE |
| | | | | | | | | | | | | | RO | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:3 – | | Keep | 0x0 | – |
| 2 | IWDGRUNF | IWDG running flag<br>0: IWDG stop<br>1: IWDG is running (i.e. counting) | 0 | RO |
| 1 | IWDGINTMSK | IWDG interrupt mask<br>0: Interrupts are not masked (notified to the CPU)<br>1: Interrupts are masked | 0 | R/W |
| 0 | IWDGMODE | IWDG count overflow mode selection bit<br>0: reset mode<br>1: Interrupt mode, generate an interrupt signal, and then restart the counter, if the interrupt does not If it is cleared before the second timeout occurs, a system reset signal is generated.<br>Note: IWDG will reset the whole system after generating a reset | 0 | R/W |

Note: protected by IWDG_UNLOCK

### 19.4.3 IWDG Counter Reload Register (IWDG_RLOAD)

Address offset: 0x08

Reset value: 0x000F FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | IWDGRLOAD[19:16] | | | |
| | | | | | | | | | | | | R/W | | | |

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

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:20 | - | reserved bit | | |
| 19:0 | IWDGRLOAD[19:0] | IWDG reload register | 0xFFFFF | R/W |

Note: protected by IWDG_UNLOCK

### 19.4.4 IWDG Counter Value Register (IWDG_CNTVAL)

Address offset: 0x0C

Reset value: 0x000F FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | IWDGCNT [19:16] | | | |
| | | | | | | | | | | | | R/W | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IWDGCNT[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:20 | - | reserved bit | 0x0 | - |
| 19:0 | IWDGCNT[19:0] | IWDG count value register, | 0xFFFFF | RO |

### 19.4.5 IWDG Interrupt Status Register (IWDG_SR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | | | IWD GOV f |
| | | | | | | | | | | | | | | | RO |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 – | | reserve | 0x0 | – |
| 0 | IWDGOVF | IWDG overflow interrupt flag<br>0: IWDG no overflow interrupt occurs<br>1: IWDG overflow interrupt occurs<br>Notice:<br>1. When IWDG is configured as reset mode, regardless of whether the IWDG counter Overflow, this bit will not be set high.<br>2. When IWDG is configured as interrupt mode, regardless of the IWDGINTMSK bit If not set high, this bit will be set high whenever the IWDG counter overflows. | 0 | RO |

### 19.4.6 IWDG Interrupt Clear Register (IWDG_INTCLR)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | | | IWD GINT CLR |
| | | | | | | | | | | | | | | | WO |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 – | | reserve | 0x0 | – |
| 0 | IWDGINTCLR | IWDG Interrupt Clear<br>0: write 0, no action<br>1: Write 1 to clear the IWDG interrupt flag | 0x0 WO | |

Note: protected by IWDG_UNLOCK

### 19.4.7 IWDG Protection Register (IWDG_UNLOCK)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | IWDGREN |
| | | | | | | | | | | | | | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 - | | reserve | 0x0 | – |
| 0 | IWDGREN | IWDG Interrupt Clear<br>0: can not change the related registers of IWDG<br>1: You can change the related registers of IWDG | 0 | R/W |

Note: Write 0x55AA6699 to IWDG_UNLOCK to release IWDG register write protection, write any value other than 0x55AA6699 To IWDG_UNLOCK can open the IWDG register write protection (IWDG_CFGR, IWDG_RLOAD, IWDG_INTCLR).

## 19.5 Attention

ÿ A delay of two watchdog count clock source clocks is required between the feed dog instruction and the watchdog timer being updated.

ÿWhen the system operates the watchdog, at least three watchdog clocks need to be allowed between two dog feeding intervals.

# 20 System Window Watchdog (WWDG)

## 20.1 Overview

The purpose of the Window Watchdog Timer (WWDG) is to perform a system reset during a specified window period, preventing the software from

into an uncontrollable state under known conditions.

## 20.2 Features

ÿ An 8-bit down counter (WWDG_CNT) and an 8-bit comparison value (WINCMP) make the WWDG timeout window period adjustable

ÿSupport 20-bit value (PRSC) to select the watchdog prescaler value

ÿSupport window count value comparison interrupt and count overflow, load count value error reset

### 20.3 Structural block diagram

The block diagram of the window watchdog timer is as follows:

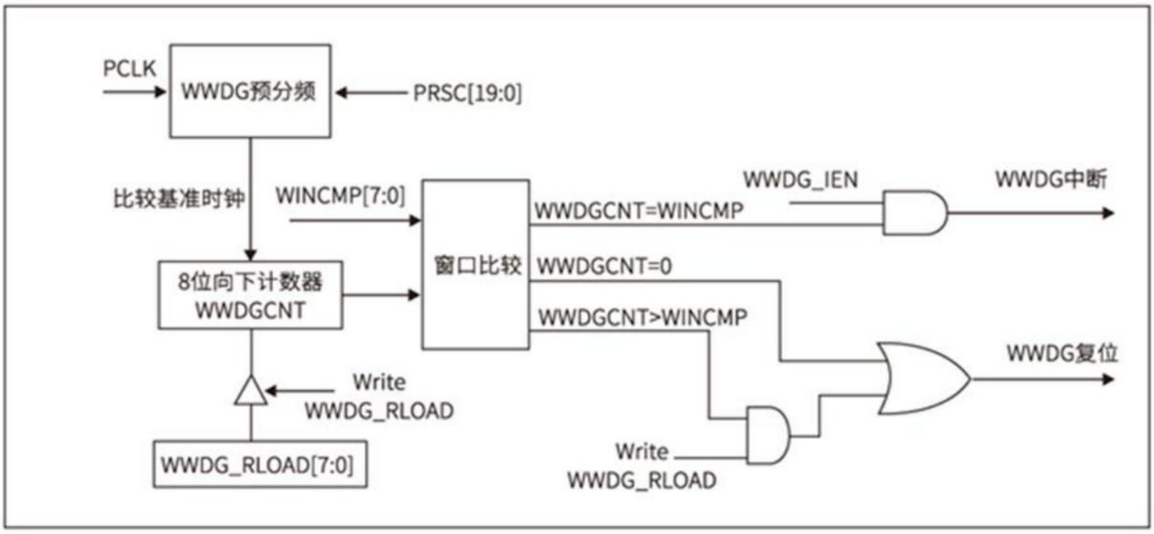


Figure 20-1 WWDG structure block diagram

## 20.4 Basic configuration

The WWDG peripheral clock source is enabled through RCC_PCLKEN.WWDGCKEN.

1. Configure the window watchdog count initial value through WWDG_RLOAD[19:0]
2. Configure the count clock prescaler through WWDG_CR.PRSC[19:0]
3. Configure the window comparison value through WWDG_CR.WINCMP[7:0]
4. Configure WWDG_INTEN.WWDGIEN according to whether interrupts need to be enabled
5. Write 1 to WWDG_CR.WWDGEN to enable window watchdog

## 20.5 Functional Description

The window watchdog timer (WWDG) is an 8-bit down counter with a selectable prescaler value, different prescaler values

Corresponding to different watchdog timer overflow time. The clock source of the 8-bit window watchdog timer is the clock after the frequency division of the PCLK clock, see

The clock source of the watchdog has an optional 20-bit prescaler value, which can be set and selected by the WWDG_CR.PRSC[19:0] bits, corresponding to

The prescaler value is shown in the table below.

Table 20-1 Window watchdog timer prescale value selection

| PRSC[19:0] | Prescale value | Timeout period | timeout interval<br>PCLK=24MHz |
|---|---|---|---|
| 0x00000 | 1 | TPCLK * 1 | 41.7ns |
| 0x00001 | 2 | TPCLK * 2 | 83.4ns |
| 0x00002 | 3 | TPCLK * 3 | 125.1ns |
| 0x00003 | 4 | TPCLK * 4 | 166.8ns |
| 0x00004 | 5 | TPCLK * 5 | 208.5ns |
| … | … | … | … |
| 0x80000 | 524289 | TPCLK* 524289 | 21.9ms |
| … | … | … | … |
| 0xFFFFF | 1048576 | TPCLK * 1048576 | 43.8ms |

## 20.5.1 Window Watchdog Timer Counting

When the WWDG_CR.WWDGEN bit is enabled, the window watchdog down counter will count down from WWDG_CNT[7:0] to

0, and cannot be turned off by software. In order to prevent the program from closing the window watchdog timer at a position not specified by the user, the window watchdog timer

The WWDGEN of the timer control register can only be written once after the chip is powered on or reset. When the WWDG_CR.WWDGEN bit is enabled by software

After that, the user cannot disable the window watchdog timer WWDG_CR.WWDGEN, and modify the counter prescaler period

WWDG_CR.PRSC[19:0], or modify the window comparison value WWDG_CR.WINCMP[7:0], unless the chip is reset. window watchdog

The timer will stop counting when the CPU enters Sleep mode or Deep Sleep mode, and resumes normal work after the CPU wakes up.

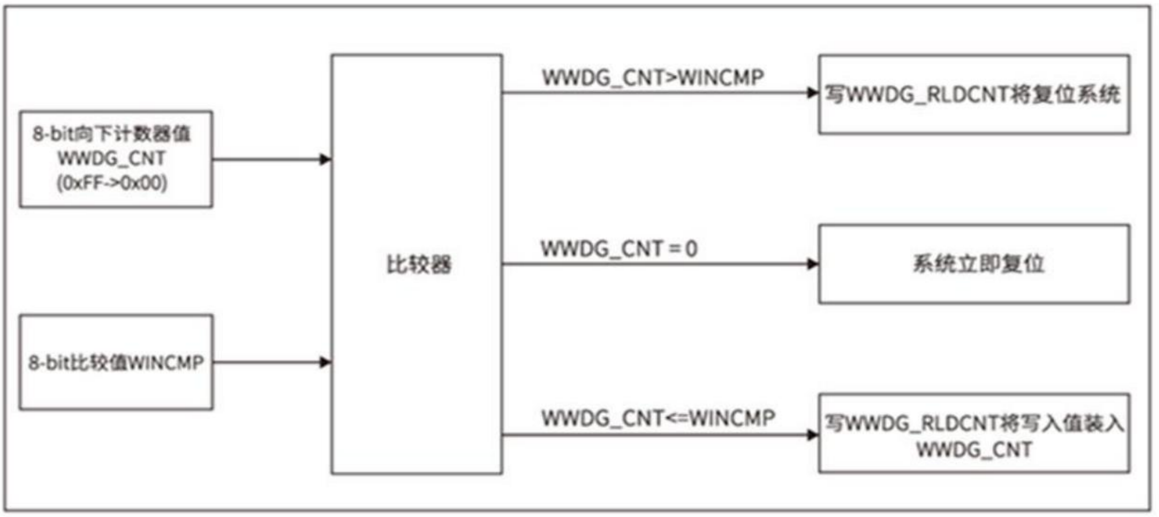


Figure 20-2 WWDG reset and reload process

### 20.5.2 Windowed Watchdog Timer Compare Interrupt

During the down counting process of the window watchdog timer, when the window watchdog timer count value WWDG_CNT[7:0] is equal to the window comparison value When WWDG_CR.WINCMP[7:0], WWDG_SR.WWDGIF will be set to 1 and WWDG_SR.WWDGIF can be cleared by software zero. If the WWDG_INTEN.WWDGIEN bit is enabled, when the WWDG_SR.WWDGIF bit is set to 1 by hardware, a window will be generated Watchdog compare match interrupt.

### 20.5.3 Window Watchdog Timer Reset System

RCC_RSTSR.WWDGRST will be set to 1 when the value of WWDG counter reaches 0. Before the WWDG counter counts down to 0, the user It must be reloaded by writing a value to WWDG_RLOAD to prevent a WWDG reset from occurring. Overloaded actions can only be performed on counters if the value is less than or equal to the WINCMP value. If the current value of the WWDG counter is greater than the value of WINCMP, the user will WWDG_RLOAD register write, window watchdog timer reset system signal will be generated immediately, and cause chip reset.

### 20.5.4 Window Setting Restrictions for Windowed Watchdog Timer

When the user writes the reload WWDG value to the WWDG_RLOAD register,

TPCLK = THCLK * (2 * RCC_PCLKDIV.APBCKDIV[7:0])

Set time interval:

T = TPCLK * (WWDG_CR.WWDG_PRSC[19:0] + 1) * (WWDG_RLOAD.WWDG_RLOAD[7:0] + 1)

The user can configure the values of the frequency division registers WWDG_PRSC[19:0] and WWDG_RLOAD[7:0] to achieve the desired time Interval.

In order to ensure normal operation, the value of WWDG_RLOAD[7:0] must be greater than or equal to 1.

## 20.6 Comparison with Independent Watchdog Timer (IWDG)

### 20.6.1 Reset Condition and Reset Delay

IWDG and WWDG are usually used to reset the system after the system runs into an uncontrollable state. The IWDG has only one condition that can trigger a relapse bit signal, WWDG has two conditions that can trigger WWDG to generate a reset signal:

WWDGCNT = 0;

Write to WWDG_RLOAD when WWDGCNT is greater than WINCMP.

Once WWDGRST is set to 1, WWDG will reset the system immediately.

### 20.6.2 Wakeup function

IWDG supports wake-up function and continues to work in Deep Sleep mode. In contrast, WWDG does not support wakeup and WWDG's counter will stop counting in Deep Sleep mode.

## 20.7 Register List

WWDG: Base Address: 0x4000 2000

| Offset Address Name Description | | | Default |
|---|---|---|---|
| | WWDG_RLOAD Window watchdog timer reload count register 0x00 0x04 | | value 0x0000 00FF |
| | WWDG_CR | Windowed Watchdog Timer Control Register | 0x0800 00FF |
| 0x08 | WWDG_INTEN Windowed Watchdog Timer Interrupt Enable Register | | 0x0000 0000 |
| 0x0C | WWDG_SR | Windowed Watchdog Timer Status Register | 0x0000 0000 |
| 0x10 | WWDG_INTCLR Windowed Watchdog Timer Interrupt Clear Register | | 0x0000 0000 |
| 0x14 | WWDG_CNTVAL Window Watchdog Timer Counter Value Register | | 0x0000 00FF |

## 20.8 Register description

### 20.8.1 Window Watchdog Timer Reload Count Register (WWDG_RLOAD)

Address offset: 0x00

Reset value: 0x0000 00FF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit | flag | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | WWDG_RLOAD[7:0] | Window watchdog timer reload count register. write value greater than 0 | 0xFF | WO |

### 20.8.2 Window Watchdog Timer Control Register (WWDG_CR)

Address offset: 0x04

Reset value: 0x0800 00FF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | WWD GEN | PRSC[19:8] | | | | | | | | | | | |
| | | | R/W | R/W | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRSC[7:0] | | | | | | | | WINCMP[7:0] | | | | | | | |
| R/W | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:29 – | | reserve | 0x0 | – |
| 28 | WWDGEN | Window watchdog enable bit<br>Setting this bit enables the window watchdog timer<br>0: disable window watchdog timer function<br>1: Enable window watchdog timer function | 0 | R/W |
| 27:8 | PRSC[19:0] | WWDG Prescaler<br>FPCLK/(PRSC+1) | 0x80000 R/W | |
| 7:0 | WINCMP[7:0] | WWDG Window Compare Register<br>Setting this register adjusts the effective reload window.<br>Note: Software can only write when the WWDG counter value is between 0 and WINCMP WWDG_RLOAD. When the WWDG counter value is greater than WINCMP, if the soft Write WWDG_RLOAD, WWDG will generate a reset signal | 0xFF | R/W |

Note: When WWDGEN is set to 1, the software configuration of this register will be disabled.

### 20.8.3 Window Watchdog Timer Interrupt Enable Register (WWDG_INTEN)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | WWD GIEN |
| | | | | | | | | | | | | | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 – | | reserve | 0x0 | – |
| 0 | WWDGIEN | WWDG interrupt enable bit<br>Setting this bit enables the windowed watchdog timer interrupt function.<br>0: Disable window watchdog timer interrupt function<br>1: Enable window watchdog timer interrupt function | 0 | R/W |

### 20.8.4 Window Watchdog Timer Status Register (WWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | WWD GIFs |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | WWDGIF | WWDG compare match interrupt flag<br>0: No window watchdog timer interrupt<br>1: Window watchdog timer interrupt<br>When the WINCMP and WWDG counters match, this bit is set to 1, and software WWDG_INTCLR.INTCLR Write 1 to clear this bit. | 0 | RO |

### 20.8.5 Window Watchdog Timer Interrupt Clear Register (WWDG_INTCLR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | INTC LR |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | INTCLR | WWDG compare match interrupt flag clear<br>Software writes 1 to clear 0 to the corresponding interrupt flag WWDG_SR.WWDGIF. | 0 | WO |

Machine Translated by Google

### 20.8.6 Window Watchdog Timer Counter Value Register (WWDG_CNTVAL)

Address offset: 0x14

Reset value: 0x0000 00FF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | WWDGCNT[7:0] | WWDG counter value<br>This register represents the current value of the window watchdog counter, this register is read-only | 0xFF | RO |

# twenty one Universal Asynchronous Transceiver (UART0/UART1)

## 21.1 Overview

This product has two general-purpose UART modules (UART0/1), which support half-duplex and full-duplex transmission; support 8bit and 9bit data formats; support

Support Mode0/1/2/3 four different transmission modes; UART0 baud rate can be generated by TIM10 or automatic baud rate generator,

The baud rate of UART1 can be generated by TIM11 or automatic baud rate generator; support multi-machine communication mode; support automatic address recognition

Don't; support given address and broadcast address.

The general-purpose UART (UART0/1) has only one clock input PCLK, and the register configuration logic and data transceiver logic all work in this clock domain.

## 21.2 Structural block diagram

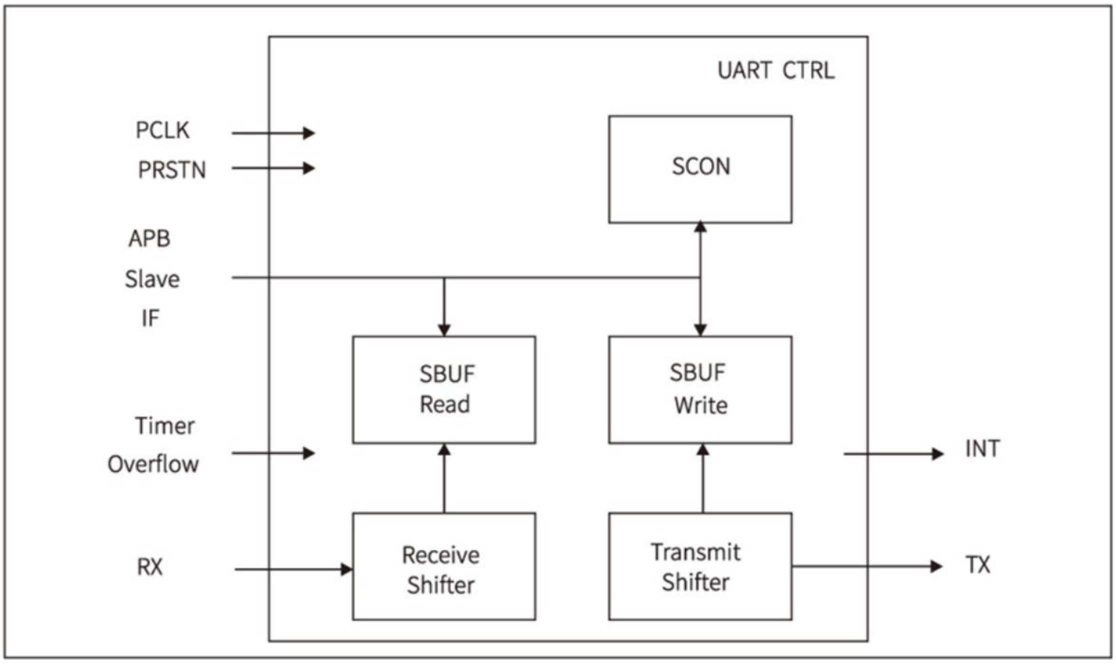


Figure 21-1 UART structure diagram

## 21.3 Working Mode

### 21.3.1 Mode 0 (synchronous mode, half-duplex)

When working in Mode0, UART works in synchronous mode, and its baud rate is 1/12 of the fixed PCLK clock. UART receives data It is input by RXD, and the data sent by UART is output by RXD. RXD is an input and output port at this time. UART synchronous shift clock is input by TXD output, TXD is the output port at this time. Note that this mode can only be used as a master to send a synchronous shift clock, and cannot be used as a slave from an external Receive shift clock. In this mode, the transmitted data bit width can only be 8 bits, without start bit and stop bit.

Clear UARTx_SCON.SM0 and UARTx_SCON.SM1 to enter Mode0 working mode.

#### 21.3.1.1 Sending Data

When sending data, clear the UARTx_SCON.REN bit and write the data to the UARTx_SBUF register. At this point, the send data will be from RXD output (low bit first, high bit later), synchronous shift clock output from TXD.

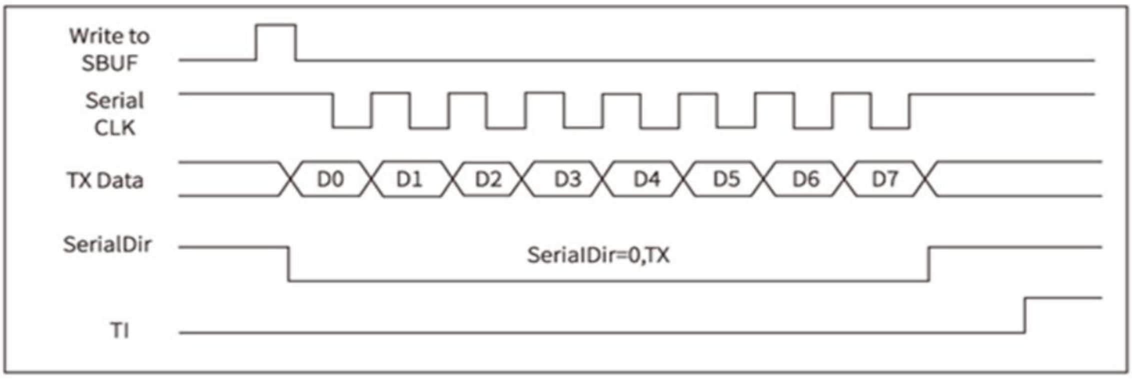


Figure 21-2 Mode0 sending data

#### 21.3.1.2 Receiving Data

When receiving data, set the UARTx_SCON.REN bit and clear the UARTx_INTSR.RI bit. When reception is complete, data is available from the UARTx_SBUF register read. At this time, the received data is input from RXD (low bit first, high bit later), and the synchronous shift clock is from TXD output.

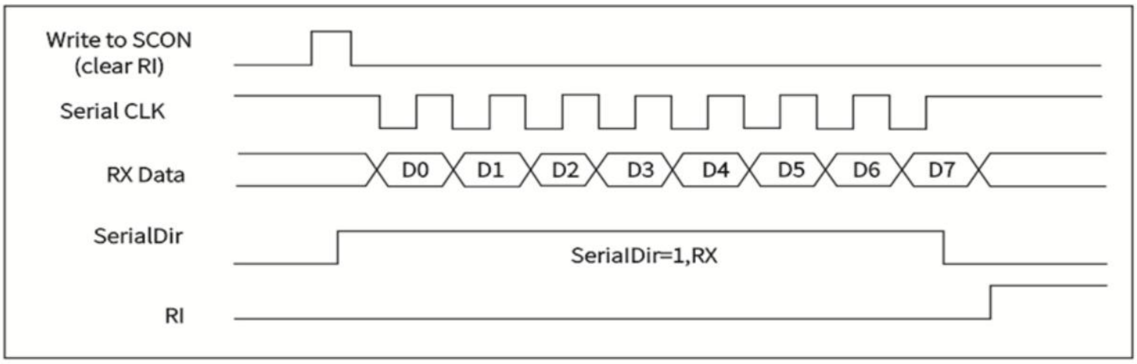


Figure 21-3 Mode0 receiving data

### 21.3.2 Mode 1 (asynchronous mode, full duplex)

When working in Mode1, the sending data is sent through TXD, and the receiving data is received through RXD. The data consists of 10 bits: start bit

Start with "0", followed by 8 data bits (lower bit first, higher bit later), and finally the end bit "1". In this mode, the baud rate can be

Generated by the programmable timer module, it can also be generated by the automatic baud rate generator inside the module. When UARTx_BAUDCR.SELF_BRG

When it is 0, when the baud rate is selected to be generated by the timer, the baud rate of UART0 is generated by TIM10, and the baud rate of UART1 is generated by TIM11

When UARTx_BAUDCR.SELF_BRG is set to 1, the baud rates of UART0 and UART1 are sent by their own internal auto-baud rate

Generator produces. Please refer to Section 21.3.5.2 Mode1/3 for the baud rate generating formula.

Clear UARTx_SCON.SM0 to 0 and set UARTx_SCON.SM1 to 1 to enter Mode1 working mode.

#### 21.3.2.1 Send Data

When sending data, it has nothing to do with the value of UARTx_SCON.REN, write the sent data into the UARTx_SBUF register, and the data will be

Shift out from TXD (low bit first, high bit last).

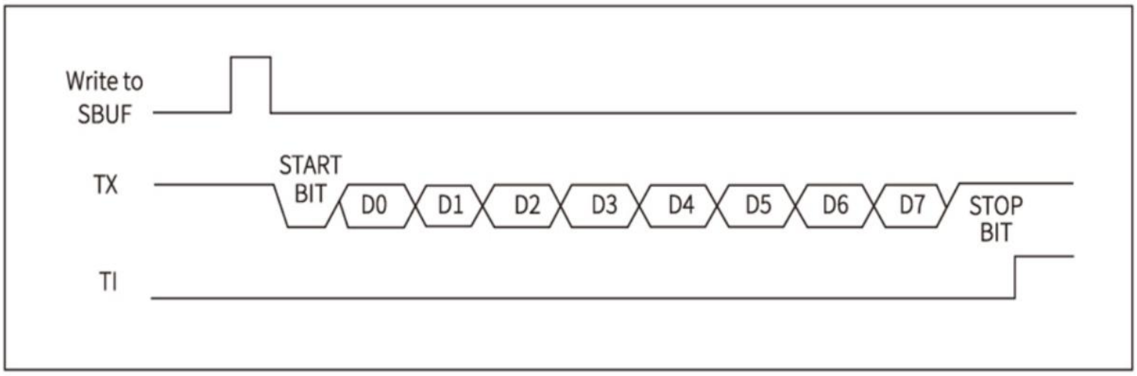


Figure 21-4 Mode1 sending data

#### 21.3.2.2 Receiving Data

When receiving data, the UARTx_SCON.REN bit should be set to 1, and the UARTx_INTSR.RI bit should be cleared to 0. Start to receive data on RXD (low

bit first, high bit later), when the reception is completed, it can be read from the UARTx_SBUF register.

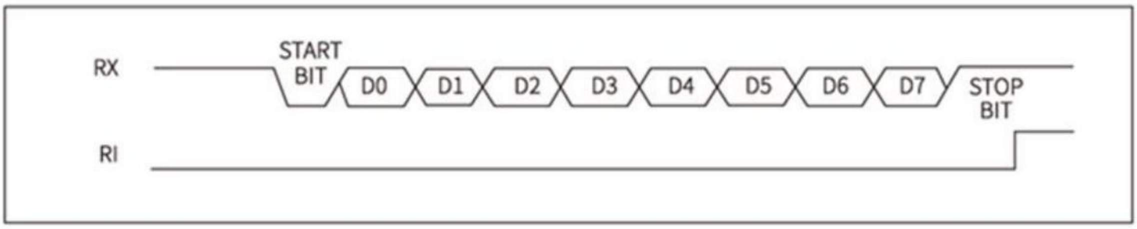


Figure 21-5 Mode1 receiving data

### 21.3.3 Mode 2 (asynchronous mode, full duplex)

When working in Mode2, the sending data is sent through TXD, and the receiving data is received through RXD. The data consists of 11 bits: start bit

"0" starts, followed by 8 data bits, 1 TB8 bit and stop bit. The extra TB8 bit is used in multi-machine communication environment,

When TB8=1, it indicates that what is received is an address frame; when TB8=0, it indicates that what is received is a data frame. When multi-machine communication is not required, this bit can also be

to be used as a parity bit. In this mode, the baud rate can be generated independently without an external timer module.

Set UARTx_SCON.SM0 to 1 and UARTx_SCON.SM1 to 0 to enter Mode2 working mode.

#### 21.3.3.1 Send Data

When sending data, it has nothing to do with the value of UARTx_SCON.REN, and write the sent data to UARTx_SBUF

register, the data will be shifted out from TXD (low bit first, high bit later).

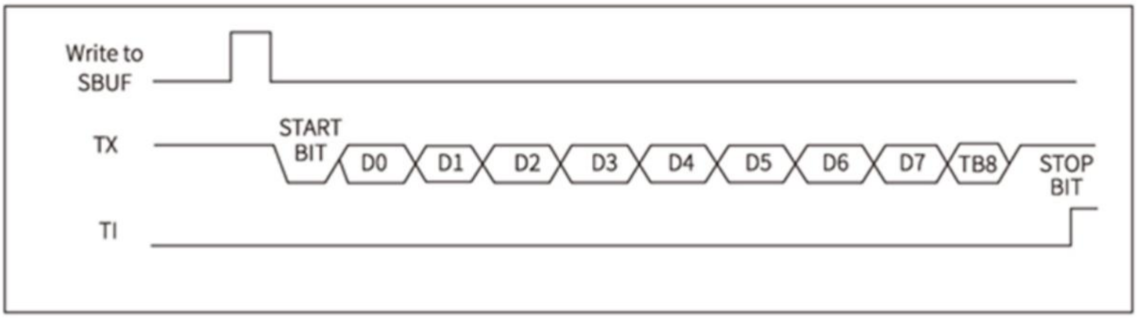


Figure 21-6 Mode2 sending data

#### 21.3.3.2 Receiving Data

When receiving data, the UARTx_SCON.REN bit should be set to 1, and the UARTx_INTSR.RI bit should be cleared to 0. Start to receive data on RXD (low bit first, high bit later), when the reception is completed, it can be read from the UARTx_SBUF register.

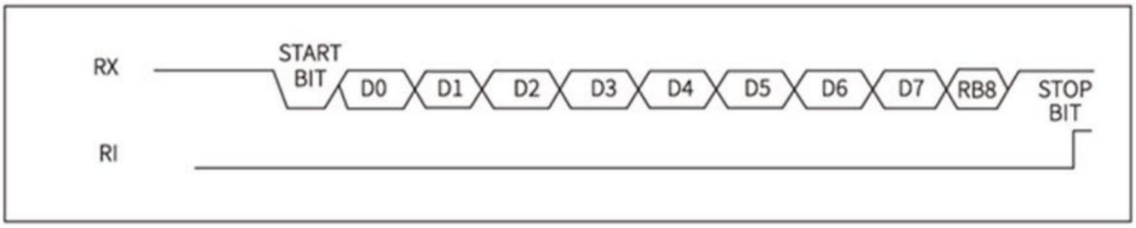


Figure 21-7 Mode2 receiving data

### 21.3.4 Mode 3 (asynchronous mode, full duplex)

The data format, transmission timing and operation mode of Mode3 are the same as Mode2, the only difference is that the baud rate of Mode3 is selected by

Generated by a programmable timer or an internal auto-baud rate generator, instead of being independently generated by the device itself like Mode2.

The baud rate of Mode3 is programmable, and the baud rate generation method is the same as that of Mode1.

Set UARTx_SCON.SM0 to 1 and UARTx_SCON.SM1 to 1 to enter Mode3 working mode.

### 21.3.5 Baud Rate Programming

#### 21.3.5.1 Mode 0

When working in Mode0, the baud rate is fixed at 1/12 of PCLK, without the support of programmable timer (Timer).

### 21.3.5.2 Mode 1/3

When working in Mode1 or Mode3, the baud rate generation formula is shown in the figure below:

UARTx_BAUDCR.SELF_BRG=0, use programmable timer (Timer) baud rate mode:

$$\text{BaudRate} = \frac{(\text{UARTx\_SCON. DBAUD} + 1)\,\text{ÿFPCLK}}{32\ \text{ÿ}\ (216\ \text{ÿ}\ \text{TIMx\_BGLOAD}[15{:}\ 0])}$$

UARTx_BAUDCR.SELF_BRG=1, use its own baud rate generation mode:

$$\text{BaudRate} = \frac{(\text{UARTx\_SCON. DBAUD} + 1)\ \text{ÿ}\ \text{FPCLK}}{32\ \text{ÿ}\ (\text{UARTx\_BAUDCR. BRG}[15{:}\ 0] + 1)}$$

Among them, UARTX_SCON.DBAUD means double baud rate, FPCLK is PCLK clock frequency, TIMx_BGLOAD is Timer Cycle load count value. Note that the Timer must be configured in 16-bit auto-reload mode, and the immediate reload register (TIMx_LOAD) and period reload register (TIMx_BGLOAD) must be written with the same initial value.

### 21.3.5.3 Mode 2

When working in Mode 2, the baud rate is fixed at the value obtained by the following formula:

$$\text{BaudRate} = \frac{(\text{UARTX\_SCON. DBAUD} + 1)\,\text{ÿFPCLK}}{64}$$

Among them, UARTX_SCON.DBAUD means double baud rate, FPCLK is PCLK clock frequency.

### 21.3.6 Framing Error Detection

Mode 1/2/3 has a frame error detection function, and the hardware will automatically detect whether the received frame data has a valid Stop bit. UARTx_INTSR.FE is set if no valid Stop bit is received. The UARTx_INTSR.FE bit is set to 1 by hardware and cleared to 0 by software. If the software does not clear it to 0 in time, the UARTx_INTSR.FE flag will not be cleared to 0 even if the subsequent received data has a valid Stop bit.

### 21.3.7 Multi-machine communication

Mode 2/3 has multi-machine communication function, so a bit TB8/RB8 is added to its frame format. Set UARTx_SCON.SM2 to "1" to enable the multi-device communication bit. When the multi-machine communication bit is turned on, when sending data, the host can use UARTx_SCON.TB8 to distinguish whether the current frame is an address frame (UARTx_SCON.TB8=1) or a data frame (UARTx_SCON.TB8=0). When receiving data, the slave will ignore the current received frame whose RB8 bit (9th bit) is "0". When the RB8 bit (9th bit) of the received frame is "1" indicating that it is an address frame, the slave will continue to judge whether the received address is equal to its own address. If there is a match, the slave will set UARTx_SCON.RB8 to “1” and UARTx_INTSR.RI to indicate that the frame is an address frame and the address has been matched. After the slave software sees UARTx_SCON.RB8=1 and UARTx_INTSR.RI=1, it first clears the UARTx_SCON.SM2 bit to “0”, and then prepares to accept the data frame for it. If the addresses are not equal, it means that the master is not addressing the slave, the hardware of the slave keeps UARTx_SCON.RB8 and UARTx_INTSR.RI as "0", the software keeps the bit of UARTx_SCON.SM2 as "1", and the slave continues to be in the address monitoring state.

### 21.3.8 Automatic Address Recognition

When the multi-machine communication bit is turned on (UARTx_SCON.SM2 is set to "1"), the automatic address recognition function will also be turned on. This function is implemented by hardware, so that the slave can detect each address frame received, and if the address matches the address of the slave, the receiver will give a UARTx_INTSR.RI reception flag. If the addresses do not match, the receiving end will not give any acceptance flag.

If necessary, the multi-device communication bit can also be turned on in Mode1, at this time the TB8 bit is replaced by the Stop bit. When the slave receives a matching address frame and a valid Stop bit, UARTx_INTSR.RI will be set to "1". In order to support automatic address recognition, the broadcast address address and the concept of a given address.

## 21.3.9 Given address

The UARTx_SADDR register of the UART device is used to indicate the given address of its own device, and the UARTx_SADEN register is the address mask code that can be used to define don't care bits in an address. When a certain bit of UARTx_SADEN is "0", it means that the address of this bit is irrelevant, that is, It means that in the process of address matching, this bit address does not participate in address matching. These don't care bits add addressing flexibility, allowing the host to Address one or more slave devices simultaneously. Note that if a unique matching address needs to be given, the UARTx_SADEN register must be set is 0xFF.

GivenAddr = UARTx_SADDR & UARTx_SADEN

### 21.3.10 Broadcast address

The broadcast address is used to address all slave devices at the same time, and the general broadcast address is 0xFF.

BoardCastAddr = UARTx_SADDR | UARTx_SADEN

Examples of given address and broadcast address

Suppose the UARTx_SADDR and UARTx_SADEN of a slave are configured as follows:

UARTx_SADDR: 0b01101001

UARTx_SADEN: 0b11111011

Then its given address and broadcast address are as follows:

Given: 0b01101x01

Broadcast: 0b11111x11

It can be seen that the master can use four addresses to address the slave, namely:

0b01101001 and 0b01101101 (given address)

0b11111011 and 0b11111111 (broadcast address)

### 21.3.11 Transceiver buffer

#### 21.3.11.1 Receive Buffer

The universal UART (UART0/1) receiver has a frame length (8/9bits) receiving buffer, that is to say, when a frame of data is received,

The data in the receiving buffer will be kept until the Stop bit of the next frame of data is received, and the receiving buffer will be updated with a new one.

frame data.

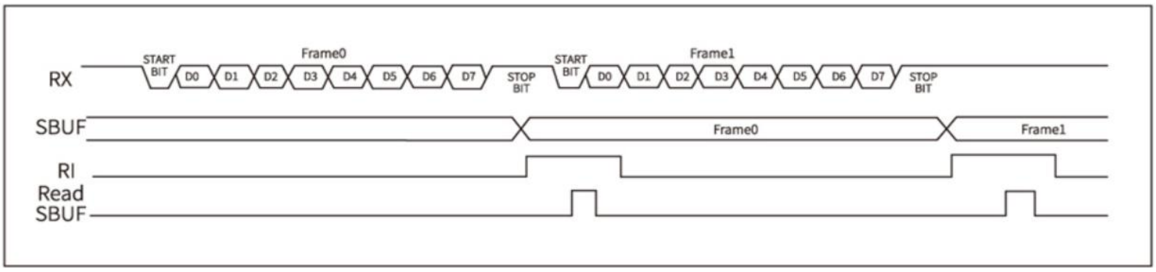


Figure 21-8 Receive buffer

#### 21.3.11.2 Send Buffer

Universal UART (UART0/1) transmitters do not support transmit buffering. If in the process of sending data, fill in UARTx_SBUF

register, will destroy the data currently being sent. Software should avoid this operation.

## 21.4 IrDA infrared function

The IrDA SIR physical layer specifies the use of an inverted return-to-zero modulation scheme (RZI), which uses a pulse of infrared light to represent a logic '0', see Figure 21-9 IrDA block diagram. The SIR transmit encoder modulates the NRZ (non-return-to-zero) bit stream output from the UART. The output pulse stream is transmitted to an external output driver and IR LED. For SIRENDEC application, UART only supports up to 115.2Kbps rate. exist

In normal mode, the pulse width is specified as 3/16 of a bit period. SIR Receiver Decoder to Zero Bit from IR Receiver

The bit stream is demodulated and the received NRZ serial bit stream is output to UART. In the idle state, the decoder input is normally high (flag status). The polarity of the output of the sending encoder is opposite to that of the input of the decoder. A start bit is detected when the decoder input is low.

ÿ IrDA is a half-duplex communication protocol. If the transmitter is busy (that is, the UART is sending data to the IrDA encoder), the IrDA Any data on the receive line will be ignored by the IrDA decoder. If the receiver is busy (that is, UART is receiving decoded data from IrDA decoder), the data from UART TX to IrDA will not be encoded by IrDA. When receiving data, sending should be avoided because the data to be sent may be corrupted.

ÿ SIR sending logic sends '0' as a high pulse, sends '1' as a low level; or inverts and sends. pulse width It is specified as 3/16 of the bit period in normal mode, see Figure 21-10 IrDA send and receive pulses.

ÿ SIR decoder converts the received IrDA signal into bit stream and sends it to UART.

ÿ The SIR receiving logic interprets the high level state as '0' and the low pulse as '1'; or inverse reception.

ÿ Send encoder output with opposite polarity to decoder input. When idle, the SIR output is in a low state.

ÿ IrDA specification requires pulse width to be greater than 1.41us. Pulse width is programmable. The spike detection circuit at the receiver will over Filter operation is performed on pulses whose width is less than 2 periods of PSC (PSC is the prescaler value programmed in UARTx_IRDACR). Pulses with a width less than 1 PSC period must be filtered out, but those pulses with a width greater than 1 and less than 2 PSC periods may be received or filtered, and those with a width greater than 2 periods will be regarded as a valid pulse .

ÿ An IrDA receiver can communicate with another IrDA low power transmitter.

### 21.4.1 IrDA Low Power Mode

IrDA can work in normal mode as well as in low power mode. Selecting a low power mode requires setting the UART_IRDACR.IRLPMODE register to 1.

Transmitter

In low-power mode, the pulse width no longer lasts 3/16 bit periods. Instead, the pulse width is three times the period of the low-power baud rate clock, which can be as low as 1.42MHz. Usually this value is 1.8432MHz (1.42MHz < PSC <2.12MHz)

A low-power mode programmable divider divides the system clock to achieve this value.

receiver

Reception in low power mode is similar to reception in normal mode. In order to filter out the spike interference pulse, UART should filter out the width shorter than 1 cycle period pulse. Only the low level of the IrDA low-power baud rate clock (PSC in UART_IRDACR) that lasts longer than 2 cycles signal is accepted as a valid signal.

Notice:

1. Pulses with a width less than 2 periods greater than 1 PSC period may or may not be filtered.
2. The settling time of the receiver should be managed by software. The IrDA physical layer specification specifies a minimum 10ms delay (IrDA is a half-duplex protocol).

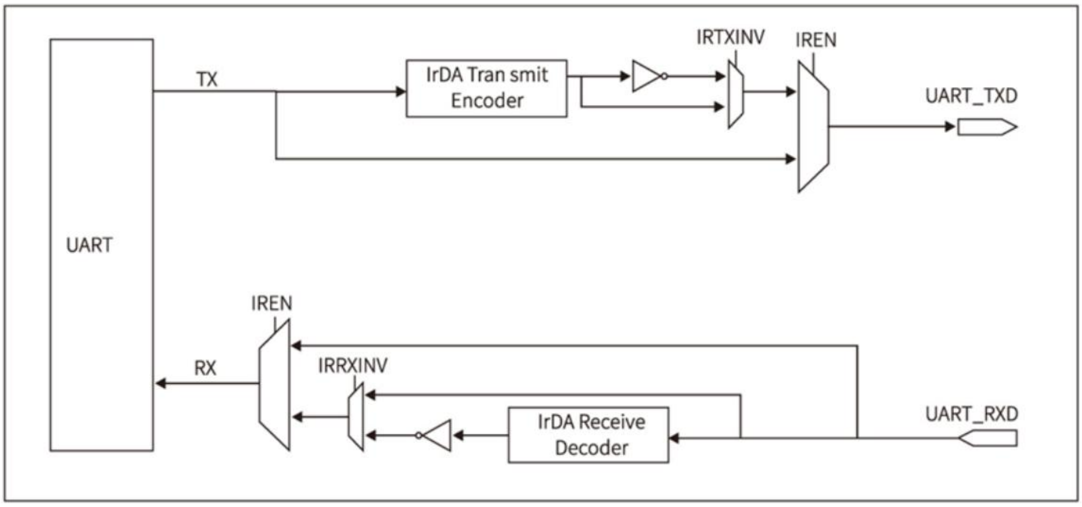


Figure 21-9 IrDA block diagram

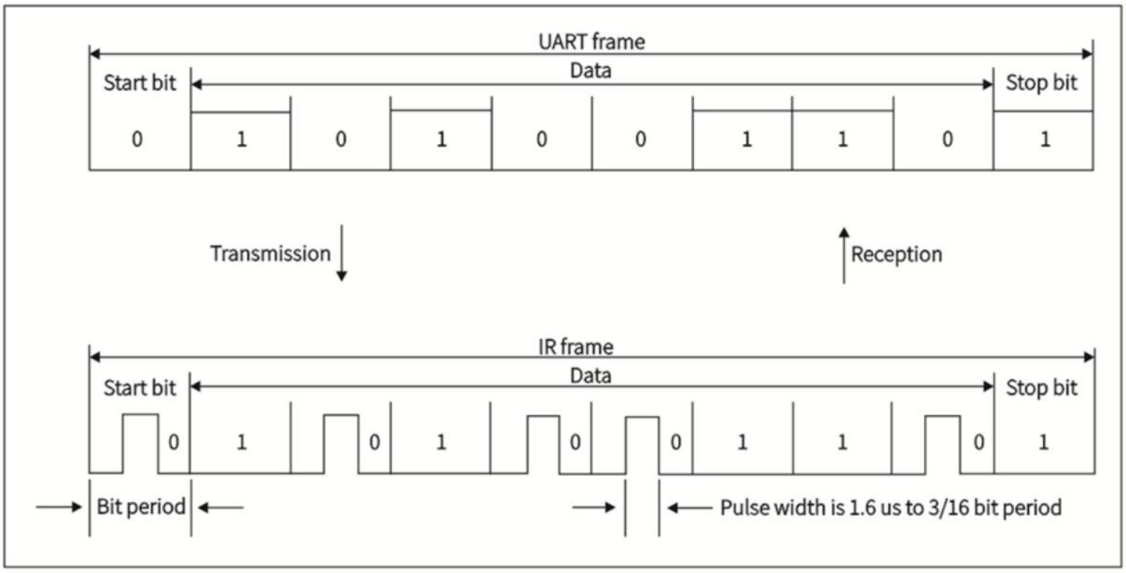


Figure 21-10 IrDA sending and receiving pulses

21.5 Frequency division settings for different baud rates

| baud rate | PCLK = 1 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 26 | 2403.85 | 0.16% | 13 | 2403.85 | 0.16% |
| 4800 | 13 | 4807.69 | 0.16% | 7 | 4464.29 | -6.99% |
| 9600 | 7 | 8928.57 | -6.99% | 3 | 10416.67 | 8.51% |
| 19200 | 3 | 20833.33 | 8.51% | 2 | 15625.00 | -18.62% |
| 38400 | 2 | 31250.00 | -18.62% | 1 | 31250.00 | -18.62% |
| 57600 | 1 | 62500.00 | 8.51% | 1 | 31250.00 | -45.75% |
| 76800 | 1 | 62500.00 | -18.62% | 0 | - | - |
| 115200 | 1 | 62500.00 | -45.75% | 0 | - | - |

| baud rate | PCLK = 4MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 104 | 2403.85 | 0.16% | 52 | 2403.85 | 0.16% |
| 4800 | 52 | 4807.69 | 0.16% | 26 | 4807.69 | 0.16% |
| 9600 | 26 | 9615.38 | 0.16% | 13 | 9615.38 | 0.16% |
| 19200 | 13 | 19230.77 | 0.16% | 7 | 17857.14 | -6.99% |
| 38400 | 7 | 35714.29 | -6.99% | 3 | 41666.67 | 8.51% |
| 57600 | 4 | 62500.00 | 8.51% | 2 | 62500.00 | 8.51% |
| 76800 | 3 | 83333.33 | 8.51% | 2 | 62500.00 | -18.62% |
| 115200 | 2 | 125000.00 | 8.51% | 1 | 125000.00 | 8.51% |

| baud rate | PCLK = 10 MHz | | | | | |
|---|---|---|---|---|---|---|
| | Dual baud | | | single baud rate | | |
| | CNT | rate actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 260 | 2403.85 | 0.16% | 130 | 2403.85 | 0.16% |
| 4800 | 130 | 4807.69 | 0.16% | 65 | 4807.69 | 0.16% |
| 9600 | 65 | 9615.38 | 0.16% | 33 | 9469.70 | -1.36% |
| 19200 | 33 | 18939.39 | -1.36% | 16 | 19531.25 | 1.73% |
| 38400 | 16 | 39062.50 | 1.73% | 8 | 39062.50 | 1.73% |
| 57600 | 11 | 56818.18 | -1.36% | 5 | 62500.00 | 8.51% |
| 76800 | 8 | 78125.00 | 1.73% | 4 | 78125.00 | 1.73% |
| 115200 | 5 | 125000.00 | 8.51% | 3 | 104166.67 | -9.58% |

| baud rate | PCLK = 14 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 365 | 2397.26 | -0.11% | 182 | 2403.85 | 0.16% |
| 4800 | 182 | 4807.69 | 0.16% | 91 | 4807.69 | 0.16% |
| 9600 | 91 | 9615.38 | 0.16% | 46 | 9510.87 | -0.93% |
| 19200 | 46 | 19021.74 | -0.93% | twenty three | 19021.74 | -0.93% |
| 38400 | twenty three | 38043.48 | -0.93% | 11 | 39772.73 | 3.57% |
| 57600 | 15 | 58333.33 | 1.27% | 8 | 54687.50 | -5.06% |
| 76800 | 11 | 79545.45 | 3.57% | 6 | 72916.67 | -5.06% |
| 115200 | 8 | 109375.00 | -5.06% | 4 | 109375.00 | -5.06% |

| baud rate | PCLK = 20 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 521 | 2399.23 | -0.03% | 260 | 2403.85 | 0.16% |
| 4800 | 260 | 4807.69 | 0.16% | 130 | 4807.69 | 0.16% |
| 9600 | 130 | 9615.38 | 0.16% | 65 | 9615.38 | 0.16% |
| 19200 | 65 | 19230.77 | 0.16% | 33 | 18939.39 | -1.36% |
| 38400 | 33 | 37878.79 | -1.36% | 16 | 39062.50 | 1.73% |
| 57600 | twenty two | 56818.18 | -1.36% | 11 | 56818.18 | -1.36% |
| 76800 | 16 | 78125.00 | 1.73% | 8 | 78125.00 | 1.73% |
| 115200 | 11 | 113636.36 | -1.36% | 5 | 125000.00 | 8.51% |

| baud rate | PCLK = 24 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 625 | 2400.00 | 0.00% | 313 | 2396.17 | -0.16% |
| 4800 | 313 | 4792.33 | -0.16% | 156 | 4807.69 | 0.16% |
| 9600 | 156 | 9615.38 | 0.16% | 78 | 9615.38 | 0.16% |
| 19200 | 78 | 19230.77 | 0.16% | 39 | 19230.77 | 0.16% |
| 38400 | 39 | 38461.54 | 0.16% | 20 | 37500.00 | -2.34% |
| 57600 | 26 | 57692.31 | 0.16% | 13 | 57692.31 | 0.16% |
| 76800 | 20 | 75000.00 | -2.34% | 10 | 75000.00 | -2.34% |
| 115200 | 13 | 115384.62 | 0.16% | 7 | 107142.86 | -6.99% |

| baud rate | PCLK = 2 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 52 | 2403.85 | 0.16% | 26 | 2403.85 | 0.16% |
| 4800 | 26 | 4807.69 | 0.16% | 13 | 4807.69 | 0.16% |
| 9600 | 13 | 9615.38 | 0.16% | 7 | 8928.57 | -6.99% |
| 19200 | 7 | 17857.14 | -6.99% | 3 | 20833.33 | 8.51% |
| 38400 | 3 | 41666.67 | 8.51% | 2 | 31250.00 | -18.62% |
| 57600 | 2 | 62500.00 | 8.51% | 1 | 62500.00 | 8.51% |
| 76800 | 2 | 62500.00 | -18.62% | 1 | 62500.00 | -18.62% |
| 115200 | 1 | 125000.00 | 8.51% | 1 | 62500.00 | -45.75% |

| baud rate | PCLK = 8 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 208 | 2403.85 | 0.16% | 104 | 2403.85 | 0.16% |
| 4800 | 104 | 4807.69 | 0.16% | 52 | 4807.69 | 0.16% |
| 9600 | 52 | 9615.38 | 0.16% | 26 | 9615.38 | 0.16% |
| 19200 | 26 | 19230.77 | 0.16% | 13 | 19230.77 | 0.16% |
| 38400 | 13 | 38461.54 | 0.16% | 7 | 35714.29 | -6.99% |
| 57600 | 9 | 55555.56 | -3.55% | 4 | 62500.00 | 8.51% |
| 76800 | 7 | 71428.57 | -6.99% | 3 | 83333.33 | 8.51% |
| 115200 | 4 | 125000.00 | 8.51% | 2 | 125000.00 | 8.51% |

| baud rate | PCLK = 11.0592 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 288 | 2400.00 | 0.00% | 144 | 2400.00 | 0.00% |
| 4800 | 144 | 4800.00 | 0.00% | 72 | 4800.00 | 0.00% |
| 9600 | 72 | 9600.00 | 0.00% | 36 | 9600.00 | 0.00% |
| 19200 | 36 | 19200.00 | 0.00% | 18 | 19200.00 | 0.00% |
| 38400 | 18 | 38400.00 | 0.00% | 9 | 38400.00 | 0.00% |
| 57600 | 12 | 57600.00 | 0.00% | 6 | 57600.00 | 0.00% |
| 76800 | 9 | 76800.00 | 0.00% | 5 | 69120.00 | -10.00% |
| 115200 | 6 | 115200.00 | 0.00% | 3 | 115200.00 | 0.00% |

| baud rate | PCLK = 16MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 417 | 2398.08 | -0.08% | 208 | 2403.85 | 0.16% |
| 4800 | 208 | 4807.69 | 0.16% | 104 | 4807.69 | 0.16% |
| 9600 | 104 | 9615.38 | 0.16% | 52 | 9615.38 | 0.16% |
| 19200 | 52 | 19230.77 | 0.16% | 26 | 19230.77 | 0.16% |
| 38400 | 26 | 38461.54 | 0.16% | 13 | 38461.54 | 0.16% |
| 57600 | 17 | 58823.53 | 2.12% | 9 | 55555.56 | -3.55% |
| 76800 | 13 | 76923.08 | 0.16% | 7 | 71428.57 | -6.99% |
| 115200 | 9 | 111111.11 | -3.55% | 4 | 125000.00 | 8.51% |

| baud rate | PCLK = 22.12 MHz | | | | | |
|---|---|---|---|---|---|---|
| | dual baud rate | | | single baud rate | | |
| | CNT | actual baud rate | Error % | CNT | Actual baud rate | error% |
| 2400 | 576 | 2400.17 | 0.01% | 288 | 2400.17 | 0.01% |
| 4800 | 288 | 4800.35 | 0.01% | 144 | 4800.35 | 0.01% |
| 9600 | 144 | 9600.69 | 0.01% | 72 | 9600.69 | 0.01% |
| 19200 | 72 | 19201.39 | 0.01% | 36 | 19201.39 | 0.01% |
| 38400 | 36 | 38402.78 | 0.01% | 18 | 38402.78 | 0.01% |
| 57600 | twenty four | 57604.17 | 0.01% | 12 | 57604.17 | 0.01% |
| 76800 | 18 | 76805.56 | 0.01% | 9 | 76805.56 | 0.01% |
| 115200 | 12 | 115208.33 | 0.01% | 6 | 115208.33 | 0.01% |

## 21.6 List of UART Registers

UART0 base address: 0x4000 0000

UART1 base address: 0x4000 0400

| Offset Address Name Description | | | Reset |
|---|---|---|---|
| | UARTx_SCON Control Register 0x00 | | value 0x0000 0000 |
| | UARTx_SBUF Data Register 0x04 | | 0x0000 0000 |
| | UARTx_SADDR Address Register 0x08 | | 0x0000 0000 |
| 0x0C | UARTx_SADEN Address Mask Register | | 0x0000 0000 |
| 0x10 | UARTx_INTSR Interrupt Flag Bit Register | | 0x0000 0000 |
| 0x14 | UARTx_INTCLR interrupt flag bit clear register | | 0x0000 0000 |
| 0x18 | UARTx_BAUDCR Baud Rate Control Register | | 0x0000 0000 |
| 0x1C | UARTx_IRDACR | IrDA Control Register | 0x0000 0000 |

## 21.7 UART register description

### 21.7.1 UART Control Register (UARTx_SCON)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | DBAUD | FEEN | SM01[1:0] | | SM2 | REN | TB8 | RB8 | TIEN | RIEN |
| | | | | | | R/W | R/W | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | complex<br>Bit | read<br>Write |
|---|---|---|---|---|
| 31:10 - | | Reserve | value 0x0 - | |
| 9 | DBAUD | double baud rate<br>0: Single baud rate<br>1: double baud rate | 0 | R/W |
| 8 | FEEN | Receive frame error interrupt enable<br>0: Disable<br>1: Enable | 0 | R/W |
| 7:6 | SM0:SM1 | Operating mode<br>00: Mode 0; 01: Mode 1; 10: Mode 2; 11: Mode 3<br>SM0 SM1 MODE describes the baud rate<br>0 0 0 shift register CLK/12<br>0 1 1 8-bit serial transmission with variable baud rate<br>1 0 2 9-bit serial port transmission CLK/32, CLK/64<br>1 1 3 9-bit serial transmission with variable baud rate | 0x0 R/W | |
| 5 | SM2 | Multi-host communication<br>0: Disable<br>1: Enable<br>SM2: Software configuration multi-machine communication and automatic address matching mode<br>1: Start multi-slave communication and automatic address matching<br>0: Disable multi-slave communication and automatic address matching<br>In mode 2 and mode 3:<br>ÿ If SM2=1, and REN=1, the receiver is in address frame monitoring mode and can use The 9th bit RB8 received is used for address screening. RB8=1 is address frame, communication data can be To enter SBUF, set RI, and enter the interrupt service routine for address comparison; RB8=0 For data frames, the receiver ignores these data frames and keeps RI=0.<br>ÿ If SM2=0, and REN=1, the receiver does not use The received RB8 is 0 or 1, both are directly received and enter SUBF, set RI, RB8 is here In this mode, it is a check digit. | 0 | R/W |
| 4 | REN | receive enable<br>Mode 0: 0: send, 1: receive<br>Others: 0: send, 1: receive/send | 0 | R/W |
| 3 | TB8 | Send TB8 bit | 0 | R/W |
| 2 | RB8 | receive RB8 bit | 0 | R/W |
| 1 | TIEN | send complete interrupt enable<br>0: Disable<br>1: Enable | 0 | R/W |
| 0 | RIEN | Receive complete interrupt enable<br>0: Disable<br>1: Enable | 0 | R/W |

### 21.7.2 UART Data Register (UARTx_SBUF)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SBUF[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:0 | SBUF[7:0] | When sending data, when the sending data is written into this register; when receiving data, after the data is received, read from this register.<br>Note: The value read to this register is actually the value in RXBuffer, and the value written to this register is actually It is written into TXShifter. | 0x0 | R/W |

### 21.7.3 UART Address Register (UARTx_SADDR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SADDR[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:0 | SADDR[7:0] | Slave Device Address Register | 0x0 | R/W |

### 21.7.4 UART Address Mask Register (UARTx_SADEN)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SADEN[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:0 | SADEN[7:0] | Slave Device Address Mask Register | 0x0 | R/W |

### 21.7.5 UART Flag Register (UARTx_INTSR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | FE | Ti | RI |
| | | | | | | | | | | | | | RO | RO | RO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:3 | - | reserve | 0x0 | - |
| 2 | FE | Receive frame error flag, set by hardware, cleared by software<br>0: FE interrupt is invalid<br>1: FE interrupt valid | 0 | RO |
| 1 | Ti | transmission complete interrupt flag bit, set by hardware, cleared by software<br>0: TI interrupt is invalid<br>1: TI interrupt valid | 0 | RO |
| 0 | RI | reception completion interrupt flag bit, set by hardware, cleared by software<br>0: RI interrupt is invalid<br>1: RI interrupt is valid | 0 | RO |

### 21.7.6 UART flag bit clear register (UARTx_INTCLR)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | FECLR | TICLR | RICLR |
| | | | | | | | | | | | | | WO | WO WO | |

| Bit Flag | Functional Description | | Reset value | read and write |
|---|---|---|---|---|
| 31:3 | - | reserve | 0x0 | - |
| 2 | FECLR | clear receive frame error flag; write 1 to clear, write 0 to be invalid | 0x0 | WO |
| 1 | TICLR | Clear transmission completion interrupt flag bit; write 1 to clear, write 0 to be invalid | 0x0 | WO |
| 0 | RICLR | clears the receive completion interrupt flag; write 1 to clear, write 0 to be invalid | 0x0 | WO |

### 21.7.7 UART Baud Rate Control Register (UARTx_BAUDCR)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | SELF_BRG |
| | | | | | | | | | | | | | | | R/W |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:17 | - | reserve | 0x0 | - |
| 16 | SELF_BRG | UART baud rate selection bits:<br>0: UART baud rate is generated by timer<br>1: The baud rate of UART is generated by (DBAUD+1)*FPCLK/(32*(BRG[15:0]+1)) | 0 | R/W |
| 15:0 | BRG[15:0] | UART auto-baud generation configuration bits:<br>Baud rate=(DBAUD+1)*FPCLK/(32*(BRG[15:0]+1)) | 0x0 | R/W |

### 21.7.8 IrDA Control Register (UARTx_IRDACR)

Address offset: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | IRLP MOD | IR RXD | IR TXD | IR DA | PSC[7:0] | | | | | | | |
| | | | | R/W | R/W | R/W | R/W | R/W | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:12 - | | reserve | 0x0 | - |
| 11 IRLPMODE | | Ir low power mode<br>0: Ir normal mode<br>1: Ir low power consumption mode | 0 | R/W |
| 10 | IRRXINV | IrRXD data inversion bit<br>0: no inversion<br>1: Invert output | 0 | R/W |
| 9 | IRTXINV | IrTXD Data Toggle Bit<br>0: no inversion<br>1: Invert output | 0 | R/W |
| 8 | IREN | IrDA enable bit<br>0: invalid<br>1: enable | 0 | R/W |
| 7:0PSC | | PSC[7:0]<br>Infrared mode sending, receiving mode filter frequency division<br>Divide the system clock frequency to achieve low power consumption | 0x0 | R/W |

# twenty two Low Power Universal Asynchronous Transceiver (LPUART)

## 22.1 Overview

This product has 1 LPUART module, supports half-duplex and full-duplex transmission; supports 8BIT, 9BIT data format; supports Mode 0/1/2/3 four different transmission modes; the baud rate of LPUART is generated by LPTIM, and can also be generated by the internal automatic baud rate generator Support multi-machine communication mode; support automatic address recognition; support given address and broadcast address, support low power consumption mode.

In order to support low power consumption applications, LPUART adds a SCLK clock in addition to the original PCLK clock, and can control LPUART working status. The internal register configuration logic of the LPUART module works in the PCLK clock domain, and the data sending and receiving logic works in the SCLK clock domain. When the system enters the low power consumption mode and is in the LPUART working state, turn off the high frequency PCLK clock and turn on the low If the frequency of SCLK clock is low, LPUART can still send and receive data normally. Turn off the working state to stop the generation of baud rate.

SCLK clock source can be selected: PCLK, external low-speed clock (LXT), internal low-speed clock (LIRC). When LPMODE=1, The SCLK clock also supports 1/2/4/8/16/32/64/128 times prescaler.

Note that when LPMODE=0, LPUART receives the TOGGLE output signal of the LPTIM clock instead of OVERFLOW signal, so the TOGGLE output of LPTIM must be enabled.

## 22.2 Structural block diagram

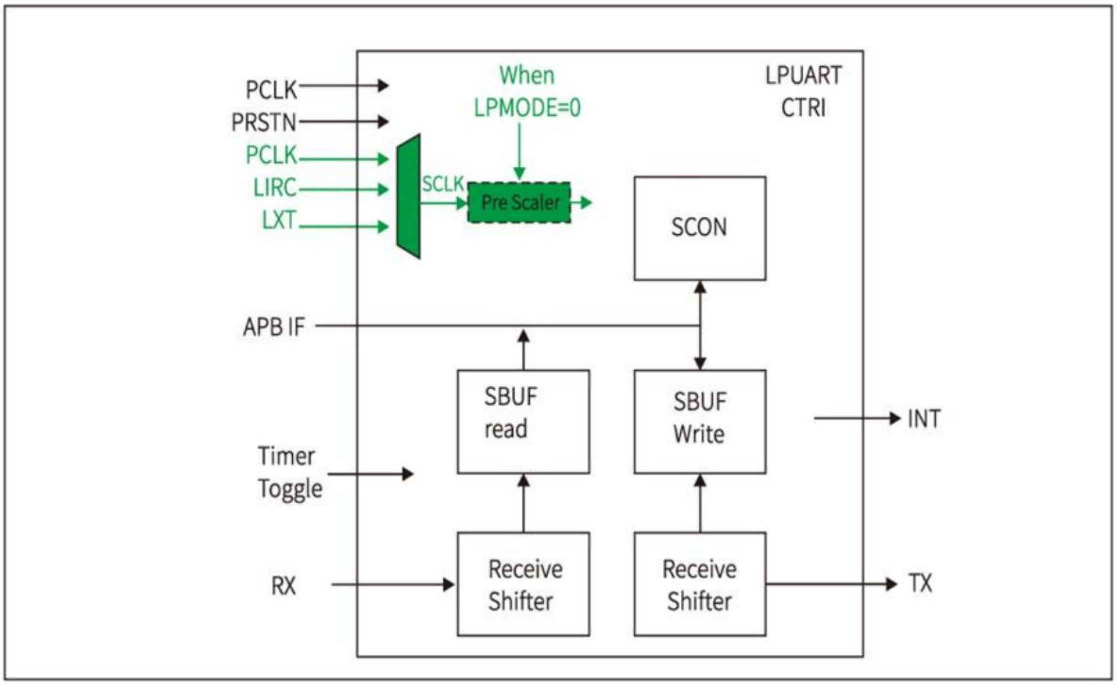


Figure 22-1 LPUART block diagram

## 22.3 Working Mode

Compared with general UART (UART0/1), LPUART adds a LPMODE control bit. When this bit is "1", only supports

Mode 1/3 working mode, and the baud rate generation method will also change. For detailed description, please refer to the following chapters.

### 22.3.1 Mode 0 (synchronous mode, half-duplex)

When working in Mode 0, UART works in synchronous mode, and its baud rate is 1/12 of the fixed SCLK clock. UART receive count

According to the input by RXD, the data sent by UART has RXD output, and RXD is the input and output port at this time. The UART synchronous shift clock is given by

TXD output, TXD is the output port at this time. Note that this mode can only be used as a master to send a synchronous shift clock, not as a slave

The machine receives the clock from the outside. In this mode, the transmitted data bit width can only be 8 bits, without start bit and stop bit.

Clear LPUARAT_SCON.SM0 and LPUART_SCON.SM1 to enter Mode 0. when

When LPMODE=1, Mode 0 work mode is not supported.

#### 22.3.1.1 Sending Data

When sending data, clear the LPUART_SCON.REN bit and write the data to the LPUART_SBUF register. At this time, the number of sending

The data will be output from RXD (low bit first, high bit later), and the synchronous shift clock is output from TXD.

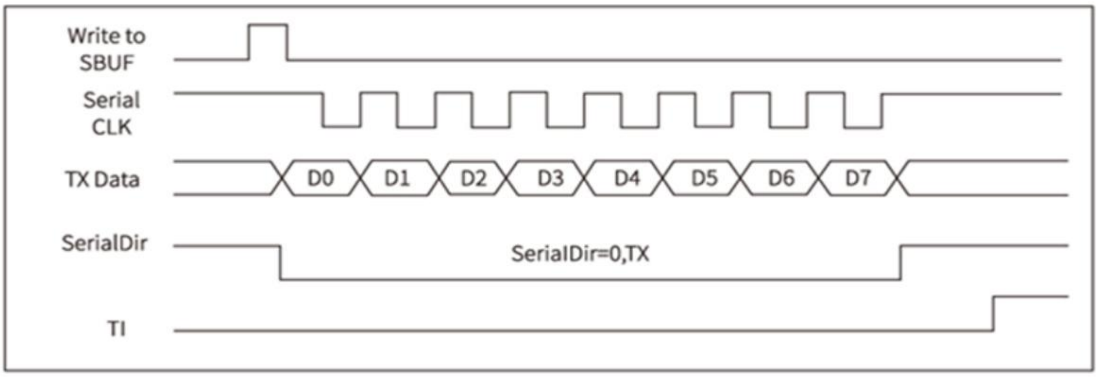


Figure 22-2 Mode 0 sending data

#### 22.3.1.2 Receiving Data

When receiving data, set the LPUART_SCON.REN bit and clear the LPUART_INTSR.RI bit. When reception ends, the data

Can be read from the LPUART_SBUF register. At this time, the received data is input from RXD (lower bit first, higher bit later), when synchronous shift

The clock is output from TXD.

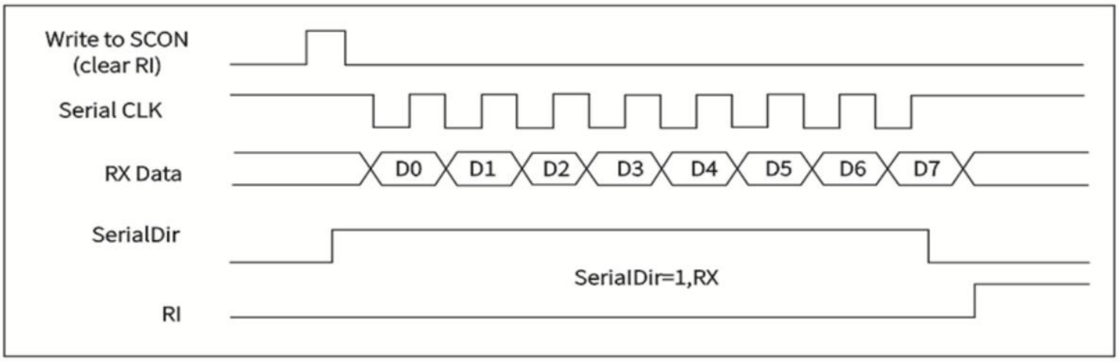


Figure 22-3 Mode 0 receiving data

### 22.3.2 Mode 1 (asynchronous mode, full duplex)

When working in Mode 1, the sending data is sent through TXD, and the receiving data is received through RXD. The data consists of 10 bits: start Bit "0" starts, followed by 8 data bits (lower bit first, upper bit later), and finally the end bit "1".

Clear LPUART_SCON.SM0 to 0 and set LPUART_SCON.SM1 to 1 to enter Mode 1 working mode.

In this mode, when LPMODE=0, the baud rate of LPUART can be selected by the automatic baud rate generator or timer LPTIM Modules are generated and are programmable.

When LPMODE=1, the baud rate calculation method will change, please refer to the baud rate programming chapter for details.

#### 22.3.2.1 Send Data

When sending data, regardless of the value of LPUART_SCON.REN, write the sent data into the LPUART_SBUF register, and the data It will be shifted out from TXD (low bit first, high bit later).

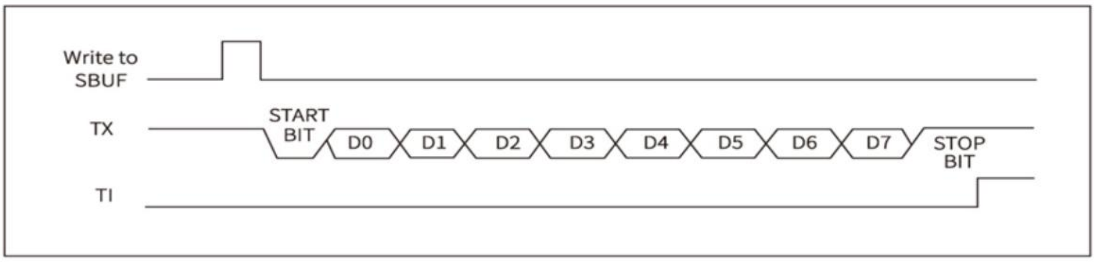


Figure 22-4 Mode 1 sending data

#### 22.3.2.2 Receiving Data

When receiving data, the LPUART_SCON.REN bit should be set to 1, and the LPUART_INTSR.RI bit should be cleared to 0. Start receiving RXD The upper data (low bit first, high bit later), can be read from the LPUART_SBUF register when receiving is completed.

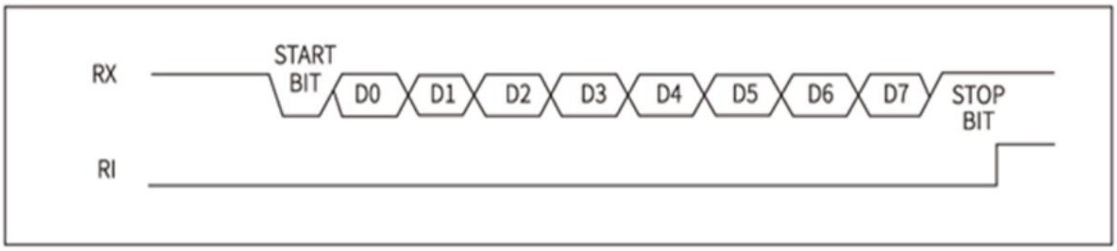


Figure 22-5 Mode 1 receiving data

### 22.3.3 Mode 2 (asynchronous mode, full duplex)

When working in Mode 2, the sending data is sent through TXD, and the receiving data is received through RXD. The data consists of 11 bits: from Start bit "0", followed by 8 data bits, 1 TB8 bit and stop bit. The extra TB8 bit is used in multi-machine communication environment When used below, when TB8=1, it indicates that the address frame is received; when TB8=0, it indicates that the received data frame. When multi-device communication is not required This bit can also be used as a parity check bit.

Set LPUART_SCON.SM0 to 1, and clear LPUART_SCON.SM1 to 0 to enter Mode 2 working mode. the mod In this mode, the baud rate can be generated independently without an external Timer.

When LPMODE=1, Mode 2 work mode is not supported.

22.3.3.1 Send Data

When sending data, it has nothing to do with the value of LPUART_SCON.REN, and write the sent data into the LPUART_SBUF register,

The data will be shifted out from TXD (low bit first, high bit later).

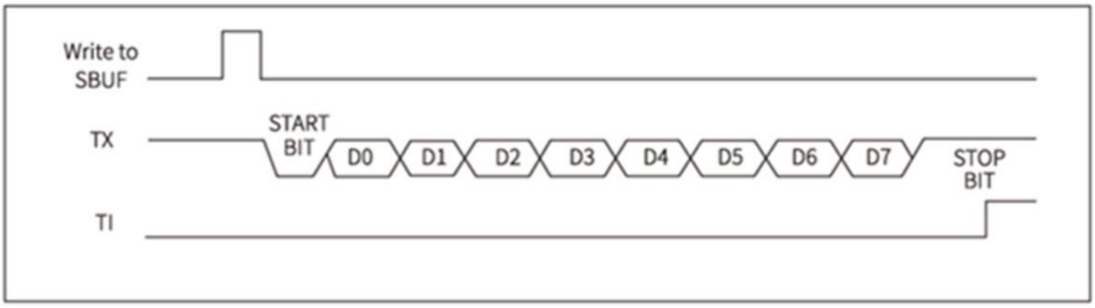


Figure 22-6 Mode 2 sending data

22.3.3.2 Receiving Data

When receiving data, the LPUART_SCON.REN bit should be set to 1, and the LPUART_INTSR.RI bit should be cleared to 0. Start receiving RXD

The upper data (low bit first, high bit later), can be read from the LPUART_SBUF register when receiving is completed.

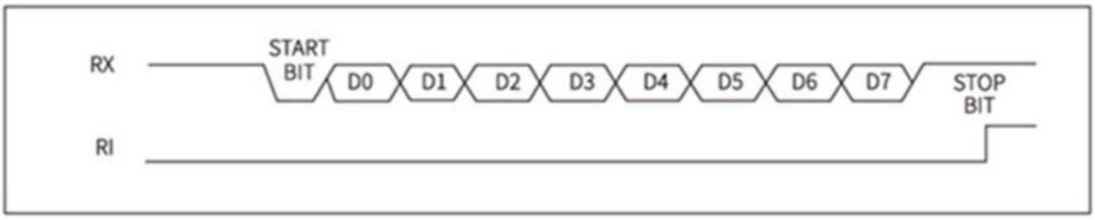


Figure 22-7 Mode 2 receiving data

## 22.3.4 Mode 3 (asynchronous mode, full duplex)

The data format, transmission timing and operation mode of Mode 3 are the same as Mode 2, the only difference is that the baud rate of Mode 3 is determined by

Generated by LPTIM or internal automatic baud rate generator, not independently generated by the device itself like Mode 2. Waves of Mode 3

The baud rate is programmable, and the baud rate generation method is the same as Mode 1.

Set LPUART_SCON.SM0 to 1 and LPUART_SCON.SM1 to 1 to enter Mode 3 working mode.

When LPMODE=1, it supports Mode 3 working mode. However, the calculation method of baud rate has changed, please refer to baud rate programming for details

chapter. Baud rate programming

## 22.3.5 Mode 0

LPMODE=0

When working in Mode 0, the baud rate is fixed at 1/12 of PCLK, and the support of LPTIM is not needed.

LPMODE=1

This mode is not supported when LPMODE=1.

### 22.3.6 Mode 1/3

LPMODE=0

When working in Mode 1 or Mode 3, the baud rate can be determined by the overflow time of LPTIM. The specific formula is shown in the figure below:

$$\text{BaudRate} = \frac{(\text{LPUART\_SCON. DBAUD} + 1)\ \text{ÿFSCLK}}{32\ \text{ÿ}\ (216\ \text{ÿ}\ \text{LPTIM\_BGLOAD}[15:\ 0])}$$

Among them, LPUART_SCON.DBAUD means double baud rate, FSCLK is SCLK clock frequency, LPTIM_BGLOAD is Period load count value for LPTIM.

Note that LPTIM must be configured as 16-bit auto-reload mode, immediate reload register (LPTIM_LOAD) and period reload register (LPTIM_BGLOAD) to write the same initial value.

It is also possible to use the own baud rate generation mode:

$$\text{BAUDRATE=} \frac{(\text{LPUART\_SCON.DBAUD} + 1)\ \text{ÿ}\ \text{FSCLK}}{32\ \text{ÿ}\ (\text{LPUART\_BAUDCR.BRG} + 1)}$$

Among them, UARTX_SCON.DBAUD means double baud rate, FSCLK is SCLK clock frequency.

LPMODE=1

When LPMODE is set to "1", the Baud calculation formula is different from the above formula, which is simplified as:

$$\text{BAUDRATE=} \frac{\text{FSCLK}}{4\ \text{ÿ}\ \text{LPUART\_SCON.PRSC}}$$

Among them, FSCLK is the SCLK clock frequency, and LPUART_SCON.PRSC is the prescaler coefficient.

### 22.3.7 Mode 2

LPMODE=0

When working in Mode 2, the transmission clock can only choose PCLK, and the baud rate is fixed at the value obtained by the following formula:

$$\text{BAUDRATE=}\frac{(\text{LPUART\_SCON.DBAUD + 1) ÿ FPCLK}}{64}$$

Among them, LPUART_SCON.DBAUD means double baud rate, FPCLK is PCLK clock frequency.

LPMODE=1

This mode is not supported when LPMODE=1.

## 22.4 Framing Error Detection

Mode 1/2/3 has a frame error detection function, and the hardware will automatically detect whether the received frame data has a valid STOP bit. if LPUART_INTSR.FE is set to 1 if no valid STOP bit is received. The LPUART_INTSR.FE bit is set to 1 by hardware and cleared to 0 by software. If the software does not clear it to 0 in time, the subsequent received data will not clear the LPUART_INTSR.FE flag even if it has a valid STOP bit.

## 22.5 Multi-machine communication

Mode 2/3 has multi-machine communication function, so a bit TB8/RB8 is added to its frame format. Set LPUART_SCON.SM2 to "1", the multi-machine communication bit can be turned on. When the multi-machine communication bit is turned on, when sending data, the host can communicate with LPUART_SCON.TB8 To distinguish whether the current frame is an address frame (LPUART_SCON.TB8=1) or a data frame (LPUART_SCON.TB8=0). When receiving data, The slave will ignore the current received frame whose RB8 bit (9th bit) is "0". When the RB8 bit (9th bit) of the received frame is "1", it indicates that it is an address frame, the slave will continue to judge whether the received address is equal to its own address. If it matches, the slave will pair the LPUART_SCON.RB8 is set to "1" and LPUART_INTSR.RI is set to "1" to indicate that the frame is an address frame and the address is been matched. After the slave software sees LPUART_SCON.RB8=1 and LPUART_INTSR.RI=1, first set The LPUART_SCON.SM2 bit is cleared to "0", and then it is ready to accept data frames addressed to it. If the addresses are not equal, it indicates that the host is not seeking address the slave, the slave hardware keeps LPUART_SCON.RB8 and LPUART_INTSR.RI as "0", and the software keeps The LPUART_SCON.SM2 bit is "1", and the slave continues to be in the address monitoring state.

## 22.6 Automatic Address Recognition

When the multi-device communication bit is enabled (LPUART_SCON.SM2 is set to "1"), the automatic address recognition function will also be enabled. This function is provided by the hardware implemented so that the slave can detect each address frame received, and if the address matches the slave address, the receiver will give LPUART_INTSR.RI Receive flag. If the addresses do not match, the receiving end will not give any acceptance flag.

If necessary, the multi-machine communication bit can also be turned on in Mode 1, and the TB8 bit is replaced by the STOP bit. When the slave receives LPUART_INTSR.RI will be set to "1" when there is a matching address frame and a valid STOP bit. To support automatic address recognition, set Defines the concepts of broadcast address and given address.

## 22.7 Given address

The LPUART_SADDR register of the LPUART device is used to indicate the given address of its own device, and the LPUART_SADEN register is an address mask that can be used to define don't care bits in an address. When a bit of LPUART_SADEN is "0", it means The address is a don't care bit, that is to say, the address of this bit does not participate in the address matching during the address matching process. These don't care bits increase the flexibility of addressing Active, so that the master can address one or more slave devices at the same time. Note that if you need to give a unique matching address, The LPUART_SADEN register must be set to 0xFF.

GIVENADDR = SADDR & SADEN

## 22.8 Broadcast address

The broadcast address is used to address all slave devices at the same time, and the general broadcast address is 0xHFF.

BoardCastAddr = SADDR | SADEN

Examples of given address and broadcast address

Suppose the configuration of LPUART_SADDR and LPUART_SADEN of a slave is as follows:

SADDR: 0b01101001

SADEN: 0b11111011

Then its given address and broadcast address are as follows:

Given Address: 0b01101x01

Broadcast Address: 0b11111x11

It can be seen that the master can use four addresses to address the slave, namely:

0b01101001 and 0b01101101(given address)

0b11111011 and 0b11111111 (broadcast address).

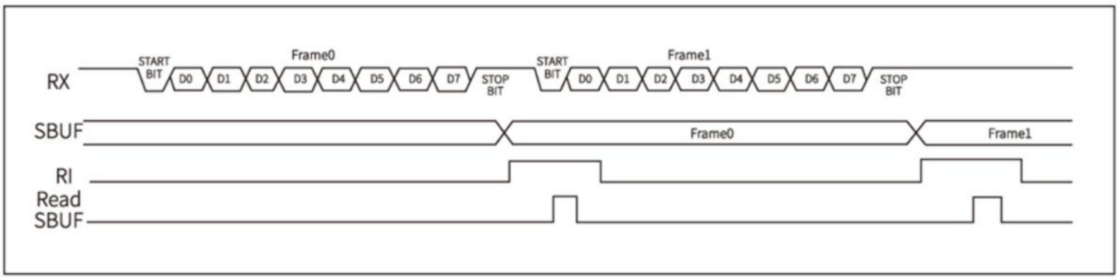
## 22.9 Transceiver Buffer

### 22.9.1 Receive buffer

The LPUART receiving end has a receiving buffer with a frame length (8/9BITS), that is to say, when a frame of data is received, the receiving buffer

The data in will be kept until the STOP bit of the next frame of data is received, and the receiving buffer will be updated with the new frame number

according to.



### 22.9.2 Send Buffer

The LPUART transmitter does not support transmit buffering. If the LPUART_SBUF register is filled in during sending data, it will be blocked

the write operation. Software should avoid this operation.

## 22.10 Register List

LPUART base address: 0x4000 5000

| Offset Address | Name Description Data Register 0x00 | | Reset |
|---|---|---|---|
| | LPUART_SBUF | | value 0x0000 0000 |
| | LPUART_SCON Control Register 0x04 | | 0x0000 E000 |
| | LPUART_SADDR Address Register 0x08 0x0C | | 0x0000 0000 |
| | LPUART_SADEN Address Mask Register 0x10 | | 0x0000 0000 |
| | LPUART_INTSR interrupt flag bit register | | 0x0000 0000 |
| 0x14 | LPUART_INTCLR interrupt flag bit clear register | | 0x0000 0000 |
| 0x18 | LPUART_BAUDCR Baud Rate Control Register | | 0x0000 0000 |

## 22.11 Register Description

### 22.11.1 LPUART Data Register (LPUART_SBUF)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SBUF[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | SBUF[7:0] | When sending data, when the sending data is written into this register; when receiving data, the data receiving When finished, read from this register. | 0x0 | R/W |

### 22.11.2 LPUART Control Register (LPUART_SCON)

Address offset: 0x04

Reset value: 0x0000E000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | EN |
| | | | | | | | | | | | | | | | R/W |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRSC[2:0] | | | SCLKSEL | | LPMODE | DBAUD | TEEN | SM0:SM1 | | SM2 | REN | TB8 | RB8 | TIEN | RIEN |
| R/W | | | R/W | | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |

<table>
<tr><th>bit flag</th><th></th><th>Functional description</th><th>Reset value</th><th>read and write</th></tr>
<tr><td>31:17</td><td>–</td><td>reserve</td><td>0x0</td><td>–</td></tr>
<tr><td>16</td><td>EN</td><td>Low-Power UART work enable<br>0: Low-Power UART off, no receiving/sending data<br>1: Low-Power UART is enabled, this bit must be set to 1 before data transmission</td><td>0</td><td>R/W</td></tr>
<tr><td>15:13</td><td>PRSC[2:0]</td><td>Transmission clock SCLK prescaler selection<br>000: DIV128; 001: DIV64; 010: DIV32; 011:16; 100:8; 101:4;<br>110: DIV2; 111: DIV1.<br>PRSC[2:0] is valid only when LPMODE=1; when LPMODE=0, PRS[2:0] SCLK is not prescaled.</td><td>0x7</td><td>R/W</td></tr>
<tr><td>12:11</td><td>SCLKSEL[1:0]</td><td>Transmission clock SCLK selection<br>00/01: PCLK<br>10: LXT<br>11: LIRC</td><td>0x0</td><td>R/W</td></tr>
<tr><td>10</td><td>LPMODE</td><td>low power mode<br>0: normal working mode<br>1: Double baud rate in low power</td><td>0</td><td>R/W</td></tr>
<tr><td>9</td><td>DBAUD</td><td>consumption mode<br>0: Single baud rate<br>1: Double baud rate</td><td>0</td><td>R/W</td></tr>
<tr><td>8</td><td>TEEN</td><td>transmit buffer empty interrupt enable<br>0: DISABLE<br>1: ENABLE</td><td>0</td><td>R/W</td></tr>
<tr><td>7:6</td><td>SM0:SM1</td><td>Working mode: 00: Mode 0; 01: Mode 1; 10: Mode 2; 11: Mode 3
<table>
<tr><td>SM0</td><td>SM1</td><td>MODE</td><td>describes</td><td>the baud rate</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register</td><td>PCLK/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8-bit serial port transmission</td><td>variable baud rate</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9-bit serial port transmission</td><td>PCLK/32, PCLK/64</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9-bit serial transmission</td><td>with variable baud rate</td></tr>
</table></td><td>0x0</td><td>R/W</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 5 | SM2 | Multi-host communication; 0: DISABLE, 1: ENABLE<br>SM2: Software configuration multi-machine communication and automatic address matching mode<br>1: Start multi-slave communication and automatic address matching<br>0: Disable multi-slave communication and automatic address matching<br>In mode 2 and mode 3:<br>ÿ If SM2=1 and REN=1, the receiver is in address frame monitoring mode and can<br>Use the received 9th bit RB8 for address screening. RB8=1 is address frame, communication<br>Data can enter SBUF, set RI, and enter the interrupt service routine for address<br>Compare; RB8=0 is a data frame, the receiver ignores these data frames and keeps RI=0<br>ÿ If SM2=0, and REN=1, the receiver does not use address monitoring mode, no matter<br>The received RB8 is 0 or 1, both are received directly and enter SUBF, set RI, RB8<br>In this mode it is the check digit. | 0 | R/W |
| 4 | REN | receive enable<br>Mode 0: 0: send, 1: receive<br>Others: 0: send, 1: receive/send | 0 | R/W |
| 3 | TB8 | Send TB8 bits | 0 | R/W |
| 2 | RB8 | Receive RB8 bit | 0 | R/W |
| 1 | TIEN | Receive complete interrupt enable<br>0: DISABLE<br>1: ENABLE | 0 | R/W |
| 0 | RIEN | Receive complete interrupt enable<br>0: DISABLE<br>1: ENABLE | 0 | R/W |

### 22.11.3 LPUART Address Register (LPUART_SADDR)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SADDR[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:8 – | | reserve | 0x0 | – |
| 7:0 | SADDR[7:0] | Slave Device Address Register | 0x0 | R/W |

### 22.11.4 LPUART Address Mask Register (LPUART_SADEN)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SADEN[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 – | | reserve | 0x0 | – |
| 7:0 | SADEN[7:0] | Slave Device Address Mask Register | 0x0 | R/W |

## 22.11.5 LPUART Flag Register (LPUART_INTSR)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | FE | Ti | RI |
| | | | | | | | | | | | | | RO | RO | RO |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:3 | – | reserve | 0x0 | – |
| 2 | FE | Receive frame error flag, set by hardware, cleared by software<br>1: FE interrupt is valid<br>0: FE interrupt is invalid | 0 | RO |
| 1 | Ti | Transmit complete interrupt flag, set by hardware, cleared by software<br>1: TI interrupt is valid<br>0: TI interrupt is invalid | 0 | RO |
| 0 | RI | Receive completion interrupt flag, set by hardware, cleared by software | 0 | RO |

## 22.11.6 LPUART flag clear register (LPUART_INTCLR)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | | | | | FECLR | TICLR | RICLR |
| | | | | | | | | | | | | | WO | WO WO | |

| bit | Mark | function description | Reset value | read and write |
|---|---|---|---|---|
| 31:3 | – | reserve | 0x0 | – |
| 2 | FECLR | clear receive frame error flag; write 1 to clear, write 0 to be invalid | 0 | WO |
| 1 | TICLR | Clear transmission completion interrupt flag bit; write 1 to clear, write 0 to be invalid | 0 | WO |
| 0 | RICLR | clears the receive completion interrupt flag; write 1 to clear, write 0 to be invalid | 0 | WO |

### 22.11.7 LPUART Baud Rate Control Register (LPUART_BAUDCR)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | SELF_BRG | |
| | | | | | | | | | | | | | | R/W | |

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

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:17 | – | reserve | 0x0 | – |
| 16 | SELF_BRG | LPUART baud rate select bit<br>0: The baud rate of LPUART is generated by LPTIMER<br>1: The baud rate of LPUART is generated by (DBAUD+1)*FSCLK/(32*(BRG[15:0]+1)) | 0 | R/W |
| 15:0 | BRG[15:0] | LPUART auto-baud generation configuration bits:<br>Baud rate=(BAUD+1)*FSCLK/(32*(BRG[15:0]+1)) | 0x0 | R/W |

# twenty three I2C interface (I2C)

## 23.1 Introduction to I2C

I2C is a two-wire bidirectional serial bus that provides a simple and efficient method for data exchange between devices. The I2C standard is a A true multi-master bus with a collision detection mechanism and an arbitration mechanism. It prevents two or more hosts from requesting A data collision occurred while requesting control of the bus.

The I2C bus controller can meet various specifications of the I2C bus and support all transmission modes communicating with the I2C bus.

The I2C bus uses "SCL" (serial clock bus) and "SDA" (serial data bus) to connect devices to transfer information. data on the host with The slaves are controlled by the SCL clock line to realize synchronous transmission of one byte and one byte on the SDA data line, each byte is 8 Bit length, one SCL clock pulse transmits one data bit, the data is transmitted from the highest bit MSB, and each transmission byte is followed by An acknowledge bit, each bit sampled while SCL is high; therefore, the SDA line can only change when SCL is low, and when SCL is high When SDA remains stable. When SCL is high, a transition on the SDA line is regarded as a command interrupt (START or STOP), and the I2C logic can operate independently Handle the transfer of bytes. It keeps track of serial transfers, and there is a status register (I2C_SR) that reflects the I2C bus control state of the controller and the I2C bus.

## 23.2 I2C main features

The I2C controller supports the following features:

ÿSupport master send/receive, slave send/receive four working modes

ÿ Support standard (100Kbps) / fast (400Kbps) / high-speed (1Mbps) three working rates

ÿSupport 7-bit addressing function

ÿ Support noise filtering function

ÿ Support broadcast address

ÿSupport interrupt status query function

## 23.3 I2C protocol description

Usually the standard I2C transmission protocol consists of four parts:

1. Start signal or repeated start signal

2. Slave address transmission and R/W bit transmission

3. Data transmission

4. Stop signal

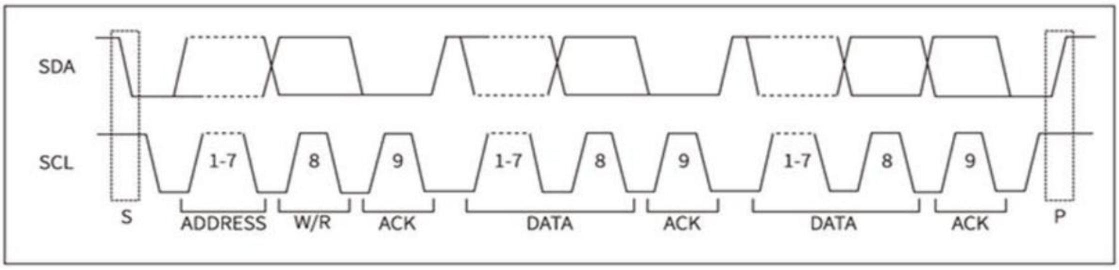


Figure 23-1 I2C transmission protocol

### 23.3.1 Data transfer on the I2C bus

The master sends a slave to receive a 7-bit address (one byte), and the transfer direction remains unchanged.

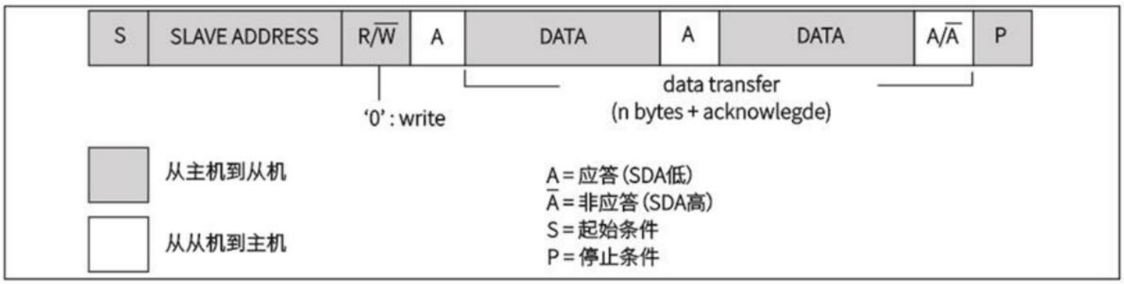


Figure 23-2 Host transmits data to slave

After the first byte, the master reads the data from the slave (the content is the address of the slave), and the transmission direction changes.

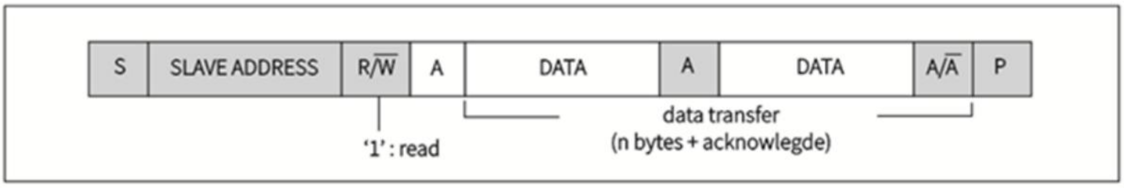


Figure 23-3 The master reads the address from the slave

### 23.3.2 Start Bit or Repeated Start Signal

When the bus is in an idle state, it means that there is no host to initiate a transfer request to the bus (the SCL and SDA lines are high at the same time), and the host can pass

A transfer request is initiated by sending a START signal.

Start signal: usually expressed as S-bit, when the SCL line is high, the signal on the SDA line goes from high to low, indicating that the start signal is generated on the bus

number, a new transmission starts.

Repeated start signal (Sr): ie there is no STOP signal between two START signals. The master adopts this method to communicate with another slave

Or the same slave communicates in a different transfer direction (for example: from writing to the device to reading from the device) without releasing the bus.

STOP signal: The host sends a stop signal to the bus to end data transmission. Stop signal, usually represented by P-bit, when the SCL line is high

, a signal from low to high appears on the SDA line, which is defined as a stop signal.

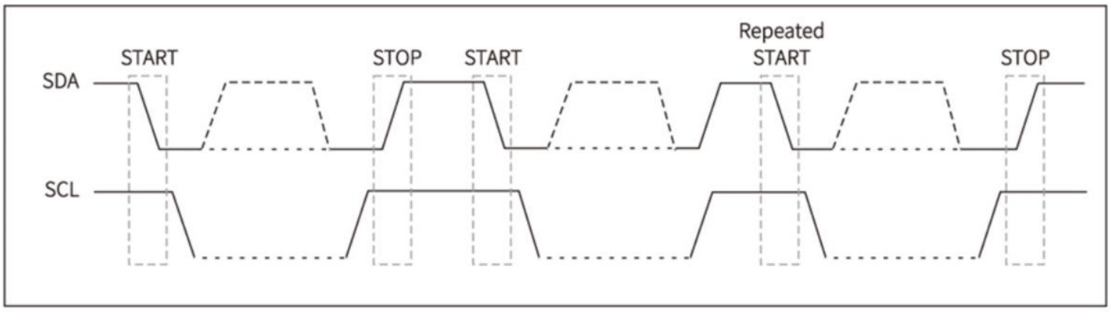


Figure 23-4 START and STOP conditions

### 23.3.3 Slave Address Transfer

When the START signal is the slave address, the master immediately transmits the first bit of data. This is a 7-bit callout followed by an RW bit

address, the RW bit controls the signal transmission direction of the slave. No two slaves in the system have the same address, only slaves addressed by the master

Acknowledges by taking SDA low on the 9th SCL clock cycle.

### 23.3.4 Data Transmission

When the slave address is successfully identified, the data transmission can be started byte by byte according to the direction determined by RW, and each transmission

At the end of the byte output, there is a response signal on the 9th clock cycle. If the slave generates a no response signal (NACK), the master can generate

Generate a stop signal to exit data transmission, or generate a repeated start signal to start a new round of data transmission.

When the host is used as a receiving device, if there is no response signal (NACK), the slave releases the SDA line, causing the host to generate a stop signal or repeat the start.

start signal.

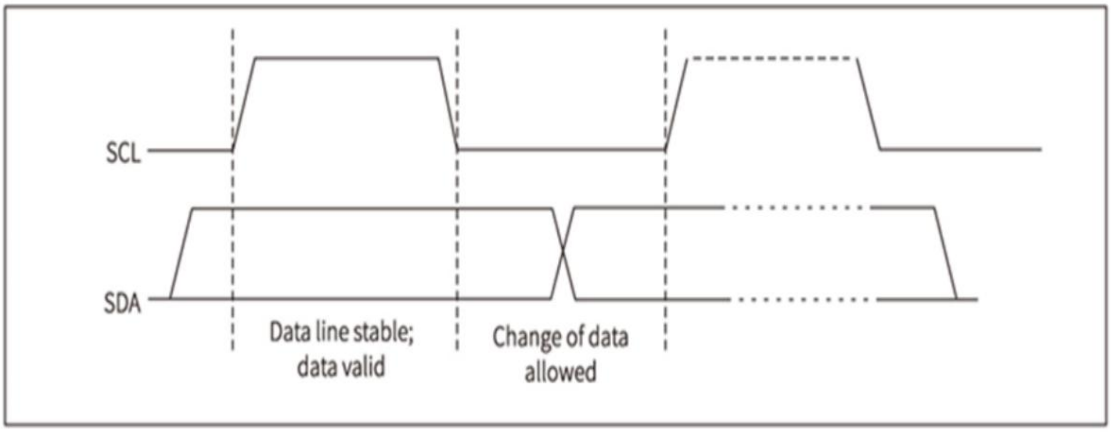


Figure 23-5 Bit transmission on the I2C bus

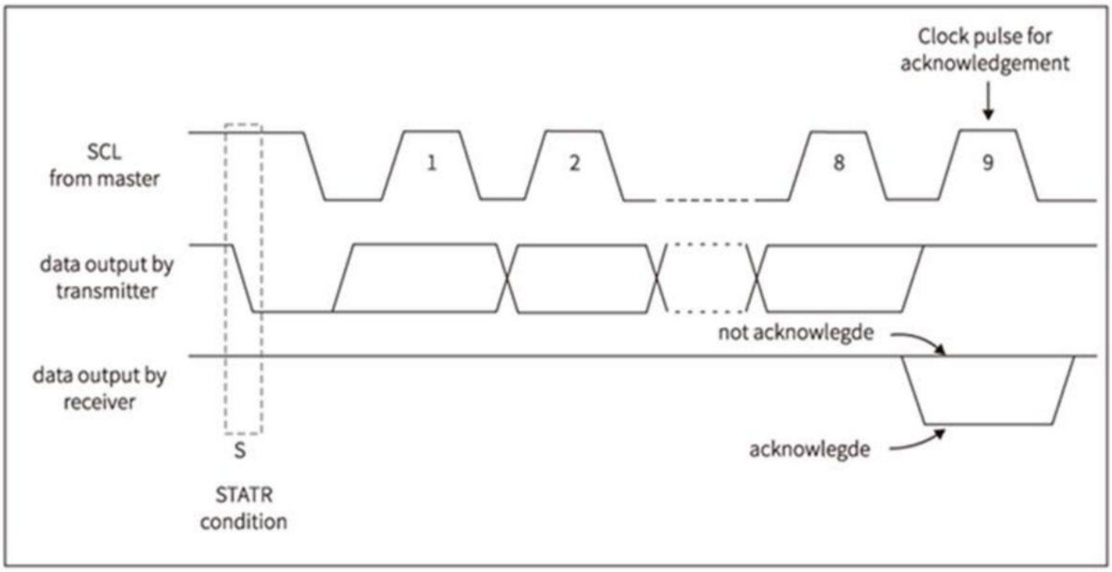


Figure 23-6 Acknowledgment signal on the I2C bus

## 23.4 I2C Functional Description

The I2C bus uses two wires to transfer information between devices connected to the bus "SCL" (Serial Clock Line) and "SDA" (Serial Data Line).

Since there are only non-directional ports, the I2C component requires the use of open-drain buffers to the pins. Every device connected to the bus can use the Addressed by software with a specific address. The I2C standard is a true multi-master bus with a conflict detection mechanism and an arbitration mechanism. Wire. It prevents data collisions when two or more hosts start transmitting data at the same time. Filtering logic can filter the data bus glitches on to protect data integrity.

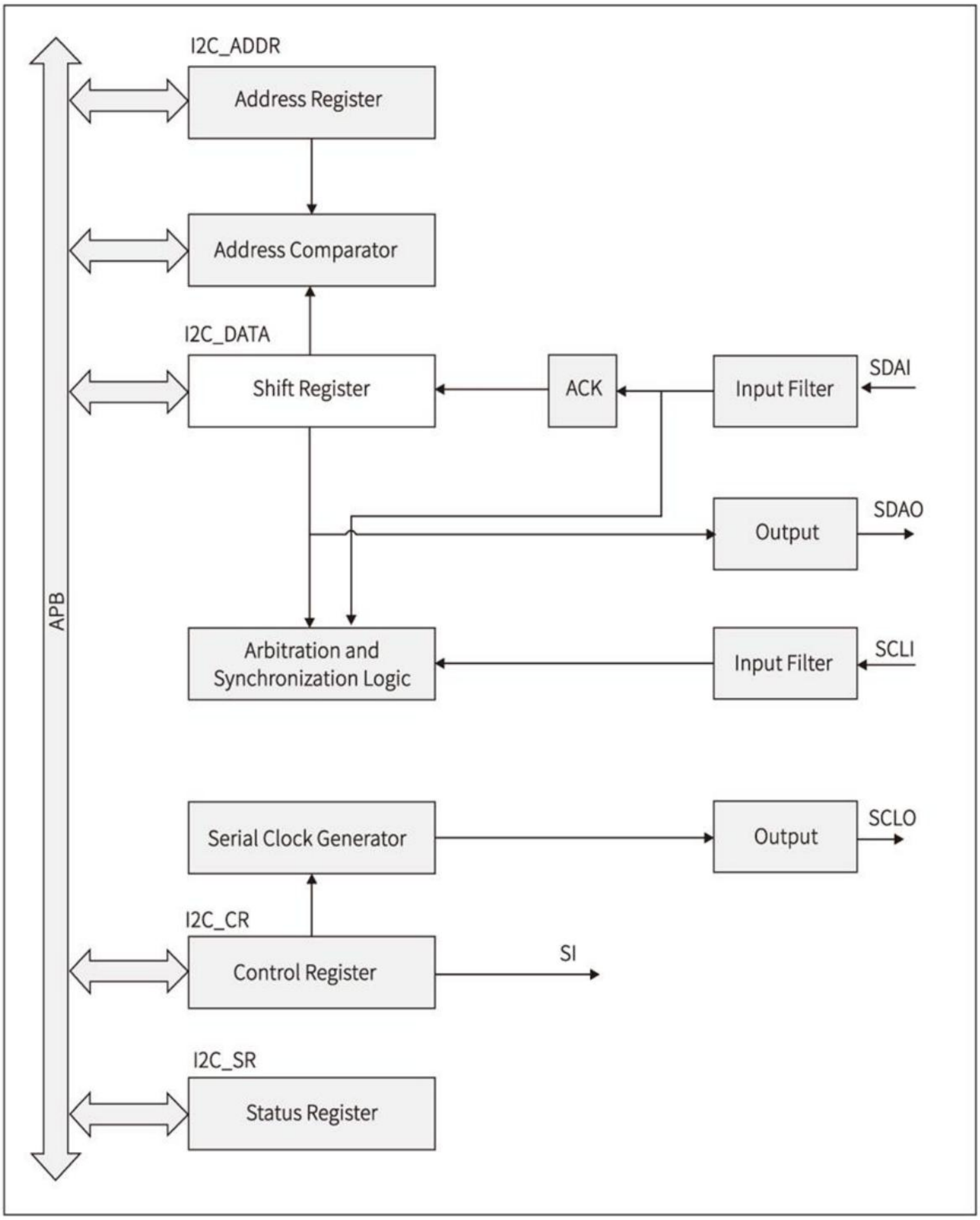


Figure 23-7 I2C function block diagram

## 23.5 I2C working mode

The I2C module can realize 8-bit bidirectional data transmission, and the transmission rate can reach 100Kbits/s in the standard mode and 100Kbits/s in the high-speed mode.

Up to 400Kbits/s, up to 1Mbits/s in ultra-high speed mode, and can work in the following four modes:

1. Host sending mode: when "SCL" outputs serial clock signal, "SDA" outputs serial data.

2. Host receiving mode: Serial data is received through “SDA” when “SCL” outputs serial clock signal.

3. Slave Receive Mode: Serial data and serial clock are received through “SDA” and “SCL” respectively.

4. Slave sending mode: When the serial clock is input from the "SCL" port, the serial data is sent through the "SDA" port.

### 23.5.1 Arbitration and Synchronization Logic

In master transmit mode, the arbitration logic checks that each transmitted logic 1 is actually present on the bus. If another device on the bus deasserts a logic 1 and pulls the SDA line low, arbitration is lost and the I2C module immediately changes from a master transmitter to a slave receiver. The I2C block will continue to output clock pulses (on SCL) until the current serial byte is sent.

Arbitration may also be lost in master receive mode. This is the case only if the I2C module is returning a "Not Acknowledge (logic 1)" to the bus appears. Arbitration is lost when another device on the bus pulls the signal low. Since it only occurs at the end of a serial byte, the I2C module No further clock pulses are generated.

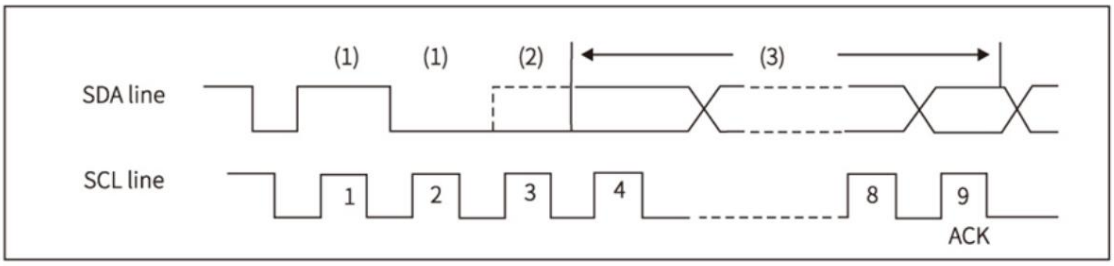


Figure 23-8 Arbitration on the I2C bus

1. Another device sends serial data;

2. Another device first negates a logic 1 sent by the I2C master by pulling SDA low (dotted line). Arbitration lost, I2C enters slave receive model;

3. At this time, I2C is in the slave receiving mode, but still generates clock pulses until the current byte is sent. I2C will not transfer the next byte Generate clock pulses. Once arbitration is won, the data transfer on SDA is initiated by the new master.

Synchronization logic synchronizes the serial clock generator to clock pulses on the SCL line of another device. If 2 or more master devices generate clock pulses, the high period is determined by the device producing the shortest high time; the low period is determined by the device producing the longest low time.

### 23.5.2 Serial Clock Generator

The serial clock generator uses an 8-bit counter as the baud rate generator, and the frequency relationship between the SCL signal and the PCLK signal is

FSCL=FPCLK/8*(N+1)

The table below shows the frequency value of SCL signal when PCLK is various frequencies and the frequency division factor is 1-7.

Table 23-1 I2C clock signal baud rate

| Frequency | | | | 4 | | | |
|---|---|---|---|---|---|---|---|
| (KHz) | 1 | 2 | 3 | 25 | 5 | 6 | 7 15 |
| 1000 | 62 | 41 | 31 | 50 | 20 41 | 17 | 31 |
| 2000 | 125 | 83 | 62 | 100 | 83 | 35 | 62 |
| 4000 | 250 | 166 | 125 | 150 | 125 | 71 | 93 |
| 6000 | 375 | 250 | 187 | 200 | 166 | 107 | 125 |
| 8000 | 500 | 333 | 250 | 250 | 208 | 142 | 156 |
| 10000 | 625 | 416 | 312 | 300 | 250 | 178 | 187 |
| 12000 | 750 | 500 | 375 | 350 | 291 | 214 | 218 |
| 14000 16000 | 875 1000 | 583 666 | 437 500 | 400 | 333 | 250 285 | 250 |

### 23.5.3 Input Filters

The input signal is synchronized with the clock signal (clk), and the spike pulse signal lower than 3 clock cycles will be filtered out. Each filter is triggered by 3

device composition. The first flip-flop is used to directly latch the input signal and load the data into the shift register formed by the other two.

When the states of the second and third flip-flops are "11" or "00", the internal filter signal is set to 1 or 0 respectively.

### 23.5.4 Address Comparator

The I2C comparator compares its own slave address with the received 7-bit slave address. It can use the "I2C_ADDR" register for its own The slave address for programming. And it will match the first received octet or Compare with general call address (0x00). If any of them are the same, the "SI" bit of the "I2C_CR" register will be set to 1 and a interrupt request.

### 23.5.5 Interrupt Generators

When all four modes of the I2C block are used, there are 26 possible bus states. When I2C enters 25 states of 26 states In this state, the "SI" flag in the "I2C_CR" register will be set to 1 by hardware. The only state in which the "SI" bit will not be set is 0xF8, which indicates that there is no There is valid associated status information. The "SI" flag must be cleared by software. In order to clear the "SI" bit, 0 must be written to this bit. Writing 1 in "SI" will not change the value of "SI". In order to determine the actual interrupt source of the interrupt, before the interrupt service routine clears the "SI" flag bit, The I2C status register is queried.

### 23.5.6 I2C Master Transmit Mode

ENS must be set to "1" to enable the I2C module. If the AA bit is reset, the I2C module Will not acknowledge its own slave address or general call address. In other words, if the AA bit is reset, the I2C interface cannot enter the slave machine mode. STA, STO and SI must be reset.

At this point, the master transmit mode can be entered by setting the STA bit. Once the bus is free, the I2C logic will test the I2C bus and generate a an initial condition. When a start condition is sent, the serial interrupt flag (SI) is set, and the status code in the status register (I2C_SR) is 0x08. The interrupt service program uses the status code to enter the corresponding status service program, and loads the slave address and data direction bit (SLA+W) into I2C_DATA. The SI bit of I2C_CR must be reset before the serial transfer can continue. When the slave address and direction bits have been sent and the An acknowledge bit, the serial interrupt flag (SI) is set again, and a series of different status codes may be in the I2C_SR. host mode 0x18, 0x20 or 0x38 in slave mode (AA=1), 0x68, 0x78 or 0xB0 in slave mode (AA=1). The operation corresponding to each status code is as follows detailed in the table. After sending the Repeated START condition (status 0x10), the I2C module switches off by loading SLA+R into I2C_DATA Switch to master receive mode.

Machine Translated by Google

Table 23-2 I2C Master Transmit Mode Status Table

| state<br>generation<br>code | I2C bus and hard file status | Application software | | | | | The next action performed by the I2C hardware |
|---|---|---|---|---|---|---|---|
| | | read/write<br>I2C_DATA | responds to write I2C_CR | | | | |
| | | | STA | STO | SI | AA | |
| 08H | Start bar sent<br>pieces | Load SLA+WX | | 0 | 0X | | Will send SLA+W, receive ACK |
| 10H | Sent to repeat<br>initial condition | Load SLA+WX | | 0 | 0X | | ditto |
| | | Load SLA+RX | | 0 | 0X | | Will send SLA+R, I2C automatically switches to master receive mode |
| 18H | Has been sent<br>SLA+W accepted<br>Receive ACK | Load Data Byte 0 No | | 0 | 0 X | | Data byte will be sent, ACK will be received |
| | | I2C_DATA | 1 | 0 | 0X | | A repeated START condition will be sent |
| | | action no I2C_DATA | 0 | 1 | 0 X | | will send stop condition, STO flag reset |
| | | action no I2C_DATA | 1 | 1 | 0X | | A STOP condition will be sent first, followed by a START condition, the STO flag is reset |
| 20H | Has been sent<br>SLA+W accepted<br>Receive non-ACK | Action Load Data Byte | | 0 | 0 X | | Data byte will be sent, ACK will be received |
| | | 0 None I2C_DATA | 1 | 0 | 0X | | A repeated START condition will be sent |
| | | action no I2C_DATA | 0 | 1 | 0 X | | will send stop condition, STO flag reset |
| | | action no I2C_DATA | 1 | 1 | 0X | | A STOP condition will be sent first, followed by a START condition, the STO flag is reset |
| 28H | Has been sent<br>I2C_DATA<br>data, received<br>Receive ACK | Action Load Data Byte | | 0 | 0 | X | will send data bytes, will receive ACK |
| | | 0 None I2C_DATA | 1 | 0 | 0 | x | A repeated START condition will be sent |
| | | action no I2C_DATA | 0 | 1 | 0 | X | will send a STOP condition and the STO flag will be reset |
| | | action no I2C_DATA | 1 | 1 | 0 | x | A STOP condition will be sent first, followed by a START condition, the STO flag is reset |
| 30H | Has been sent<br>I2C_DATA<br>The data | Action Load Data Byte | | 0 | 0 | X | will send data bytes, will receive ACK |
| | | 0 None I2C_DATA | 1 | 0 | 0 | x | A repeated START condition will be sent |
| | | action no I2C_DATA<br>action | 0 | 1 | 0 | X | will send a STOP condition and the STO flag will be reset |
| | | no I2C_DATA | 1 | 1 | 0 | x | A STOP condition will be sent first, followed by a START condition, the STO flag is reset |
| 38H | In SLA+R/W<br>or write data bytes<br>loss of arbitration | action no I2C_DATA | 0 | 0 | 0 | x | The I2C bus is released and enters the non-addressable slave mode<br>Mode |
| | | action no I2C_DATA<br>action | 1 | 0 | 0 | X | Send a START condition when the I2C bus is free |

Machine Translated by Google

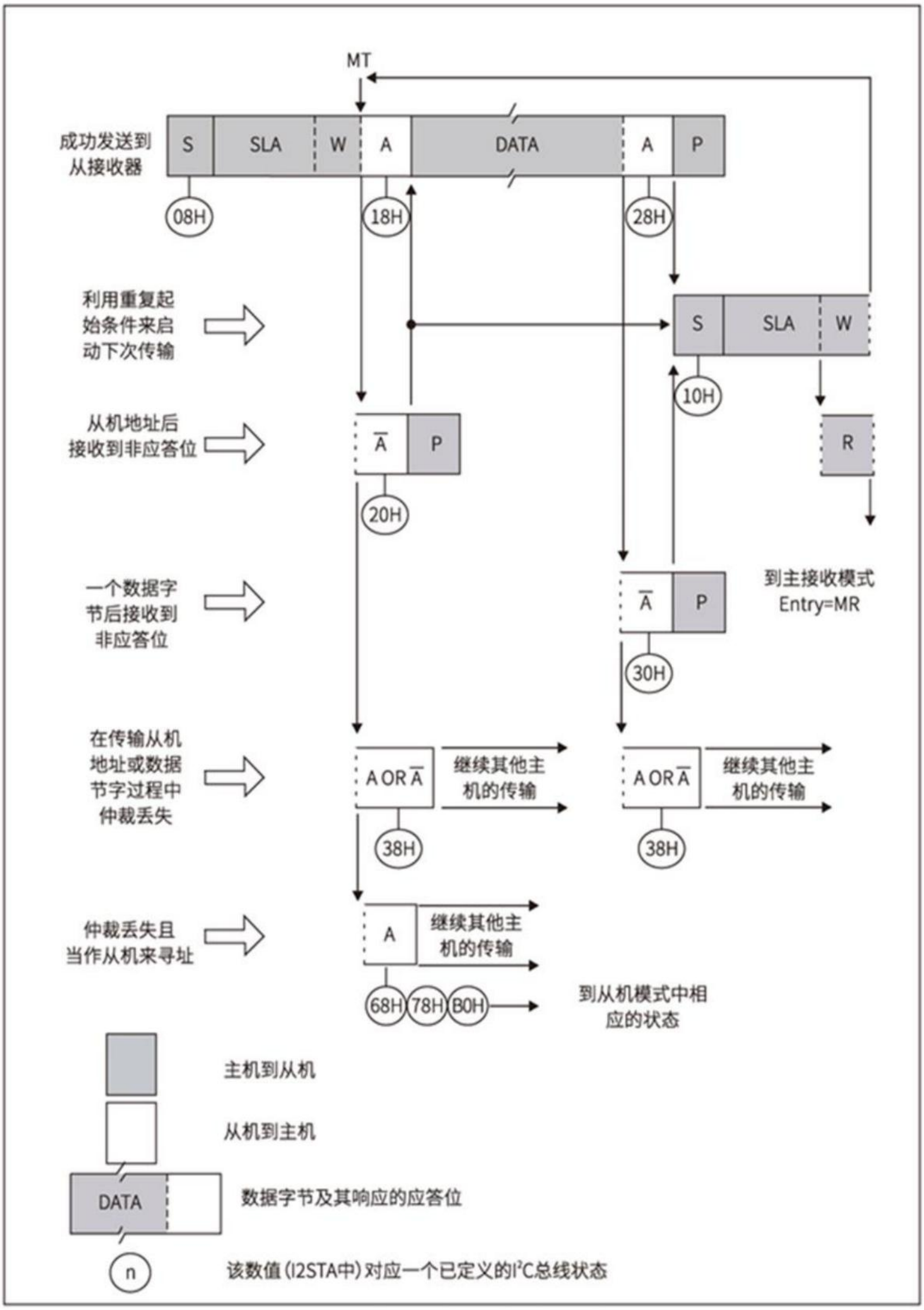


Figure 23-9 I2C host sending status diagram

### 23.5.7 I2C Master Receive Mode

In master receive mode, the data bytes received by the master come from the slave transmitter. Initiate transfer as in master send mode.

After sending the start condition, the interrupt service routine must load the 7-bit slave address and data direction bit (SLA+R) into I2C_DATA. must

The SI bit in I2C_CR must be cleared before proceeding with the serial transfer. When the slave address and data direction bits have been sent and a

When acknowledging the bit, the serial interrupt flag SI is set again. At this time, there may be a series of different status codes in I2C_SR. host

It is 0x40, 0x48 or 0x38 in slave mode, and 0x68, 0x78 or 0xB0 in slave mode (AA=1). Actions for each status code

See the table below for details. After sending a repeated START condition (state 0x10), the I2C module switches to

Master sender mode.

Table 2 3-3 I 2 C Master Receive Mode Status Table

| shape<br>state<br>the<br>code | I2C bus and<br>hardware status | Application software | | | | | The next action performed by the I2C hardware |
|---|---|---|---|---|---|---|---|
| | | read/write<br>I2C_DATA | responds to write I2C_CR | | | | |
| | | | STA | STO | SI | AA | |
| 08H | Start bar sent<br>pieces | Load SLA+R | x | 0 | 0X | | Will send SLA+R, receive ACK |
| 10H | Sent to repeat<br>initial condition | Load SLA+R | x | 0 | 0X | | ditto |
| | | Load SLA+WX without | | 0 | 0X | | Will send SLA+W, I2C automatically switches to master sending mode |
| 38H | in non-ACK<br>lost arbitration | I2C_DATA<br>action | 0 | 0 | 0 X | | I2C bus will be released; enter slave mode |
| | | no I2C_DATA | 1 | 0 | 0X | | Initiate a START condition when the bus is free |
| 40H | Has been sent<br>SLA+R accepted<br>Receive ACK | action no I2C_DATA<br>action | 0 | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | no I2C_DATA | 0 | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 48H | Has been sent<br>SLA+R accepted<br>Receive non-ACK | action no I2C_DATA | 1 | 0 | 0X | | A repeated START condition will be sent |
| | | action no I2C_DATA<br>action | 0 | 1 | 0X | | will send a STOP condition, the STO flag resets bit |
| | | no I2C_DATA | 1 | 1 | 0X | | A STOP condition will be sent first, followed by a START Start condition, STO flag reset |
| 50H | received data word section, ACK has been return | Action read data byte | 0 | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | read data bytes | 0 | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 58H | received data word section, not ACK returned | read data byte 1 | | 0 | 0X | | A repeated START condition will be sent |
| | | read data bytes | 0 | 1 | 0X | | will send a STOP condition, the STO flag resets bit |

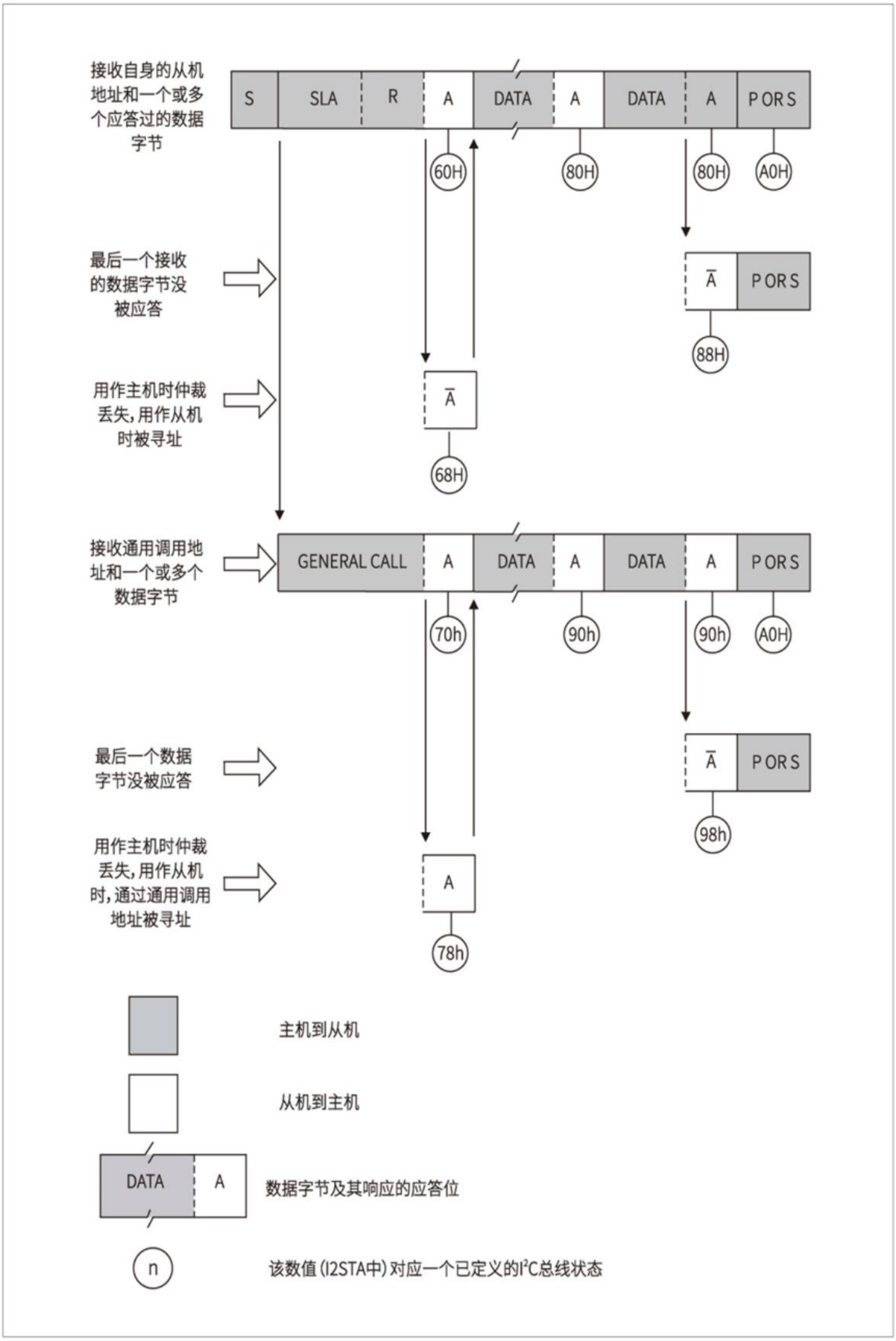

接收自身的从机地址和一个或多个应答过的数据字节
S
SLA
R
A
DATA
A
DATA
A
P OR S
60H
80H
80H
A0H
最后一个接收的数据字节没被应答
Ā
P OR S
88H
用作主机时仲裁丢失，用作从机时被寻址
Ā
68H
接收通用调用地址和一个或多个数据字节
GENERAL CALL
A
DATA
A
DATA
A
P OR S
70h
90h
90h
A0H
最后一个数据字节没被应答
Ā
P OR S
98h
用作主机时仲裁丢失，用作从机时，通过通用调用地址被寻址
A
78h
主机到从机
从机到主机
DATA
A
数据字节及其响应的应答位
n
该数值（I2STA中）对应一个已定义的I²C总线状态

Figure 23-10 I2C host receiving state diagram

### 23.5.8 I2C Slave Receive Mode

In slave receive mode, the slave receives data bytes from the master transmitter. The upper 7 bits are the response address of the I2C module when the host addresses site. If the LSB (GC) is set, the I2C module will respond to the general call address (0x00); otherwise ignore the general call address.

The setting of the I2C bus speed does not affect the I2C block in slave mode. ENS must be set to enable the I2C block. AA bit must be set to enable the I2C module to acknowledge its own slave address or general call address. STA, STO and SI must be reset.

After I2C_ADDR and I2C_CR are initialized, the I2C module waits until it is addressed by the slave address, followed by the data block For bit addressing, the data direction bit must be "0" (W) in order to work in slave receive mode. has finished receiving its own slave address and the W After the bit, the serial interrupt flag (SI) is set, and a valid status code can be read from I2C_SR. This status code is used as a status service A vector of programs. The corresponding action for each status code is shown in the table below. If the I2C module loses arbitration in master mode, it can also enter the slave interface. receive mode (please refer to the description of status 0x68 and 0x78).

If the AA bit is reset during transmission, the I2C module will return a non-acknowledgement to SDA after receiving the next data byte (logic Series 1). When AA is reset, the I2C block does not respond to its own slave address or general call address. However, the I2C bus is still monitored, Also, address recognition can be resumed at any time by setting bit AA. This means that the AA bit can temporarily detach the I2C module from the I2C bus come out.

Table 23-4 I2C slave receiver mode status table

| shape<br>state<br>the<br>code | I2C bus and<br>hardware status | application response | | | | | The next action performed by the I2C hardware |
|---|---|---|---|---|---|---|---|
| | | Read/write I2C_DATA | Write to I2C_CR | | | | |
| | | | STA | STO | SI | AA | |
| 60H | received itself<br>SLA+W;<br>Received ACK | No I2C_DATA active<br>do | x | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | No I2C_DATA active<br>do | x | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 68H | when mastering<br>SLA+R/W<br>loss of arbitration;<br>received itself<br>SLA+W;<br>return ACK; | No I2C_DATA active<br>do | x | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | No I2C_DATA active<br>do | x | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 70H | received generic<br>calling address<br>(0x00); has returned<br>return ACK; | No I2C_DATA active<br>do | x | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | No I2C_DATA active<br>No | x | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 78H | when mastering<br>SLA+R/W<br>lost arbitration;<br>Received general call<br>with address;<br>ACK has been returned; | I2C_DATA action<br>do | x | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | No I2C_DATA active<br>do | x | 0 | 0 | 1 | Data byte will be received, ACK will be returned |
| 80H | previous addressing<br>use itself from<br>address; received<br>data bytes;<br>return ACK; | No I2C_DATA active<br>do | x | 0 | 0 | 0 | Data byte will be received, will return NOT ACK |
| | | No I2C_DATA active<br>do | x | 1 | 0 | 1 | Data byte will be received, ACK will be returned |
| | previous addressing<br>use itself from<br>address; received | read data bytes | 0 | 0 | 0 | 0 | Switch to non-addressable slave mode; does not recognize self origin or general address; |
| | | read data bytes | 0 | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self origin or general address; |
| | | read data bytes | 1 | 0 | 0 | 0 | Switch to non-addressable slave mode; does not recognize self origin or general address; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 88H | data bytes;<br>return NOT ACK; read data bytes | | 1 | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self slave address or general address; when the bus is free After sending the start condition; |
| 90H | previous addressing Use common with address; received received data; return ACK; | read data byte 1 | | 0 | 0 | X | will receive data bytes, will return NOT ACK |
| | | read data byte 0 | | 1 | 0 | X | will receive data bytes, will return ACK |
| 98H | The last read data byte 0 | | | 0 | 0 | 0 | Switch to non-addressable slave mode; no self-identification origin or general address; |
| | common calling address; received data; return not ACK; | | | | | | |
| | | read data bytes | 0 | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self origin or general address; |
| | | read data byte 1 | | 0 | 0 | 0 | Switch to non-addressable slave mode; does not recognize self slave address or general address; when the bus is free After sending the start condition; |
| | | read data byte 1 | | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self slave address or general address; when the bus is free After sending the start condition; |
| A0 h | When using the slave receive/send from static in mode When addressing, receive to the stop condition or repeat start condition | No I2C_DATA active do | 0 | 0 | 0 | 0 | Switch to non-addressable slave mode; does not recognize self origin or general address; |
| | | No I2C_DATA active do | 0 | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self origin or general address; |
| | | No I2C_DATA active do | 1 | 0 | 0 | 0 | Switch to non-addressable slave mode; does not recognize self slave address or general address; when the bus is free After sending the start condition; |
| | | No I2C_DATA active do | 1 | 0 | 0 | 1 | Switch to non-addressable slave mode; does not recognize self slave address or general address; when the bus is free After sending the start condition; |

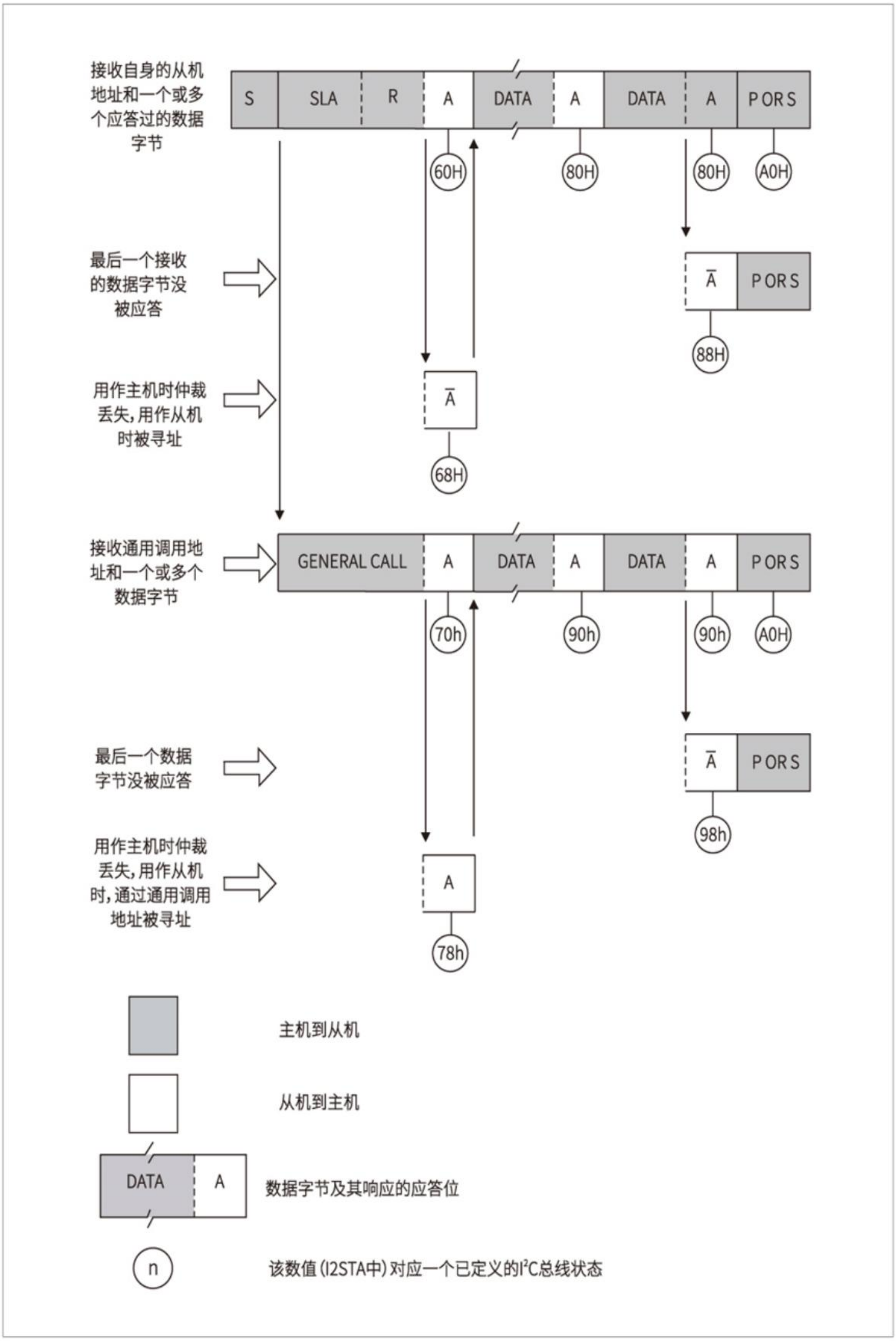

接收自身的从机地址和一个或多个应答过的数据字节
S
SLA
R
A
DATA
A
DATA
A
P OR S
60H
80H
80H
A0H
最后一个接收的数据字节没被应答
Ā
P OR S
88H
用作主机时仲裁丢失,用作从机时被寻址
Ā
68H
接收通用调用地址和一个或多个数据字节
GENERAL CALL
A
DATA
A
DATA
A
P OR S
70h
90h
90h
A0H
最后一个数据字节没被应答
Ā
P OR S
98h
用作主机时仲裁丢失,用作从机时,通过通用调用地址被寻址
A
78h
主机到从机
从机到主机
DATA
A
数据字节及其响应的应答位
n
该数值(I2STA中)对应一个已定义的I²C总线状态

Figure 23-11 I2C slave receiving state diagram

### 23.5.9 I2C Slave Transmitter Mode

In slave transmit mode, data bytes are transmitted to the master receiver. Data transfers are initiated as in slave receive mode. at first After initializing I2C_ADDR and I2C_CR, the I2C module waits until it is addressed by its own slave address, followed by the data direction bit, the data direction bit must be "1" (R) in order for the I2C block to work in slave transmit mode. After receiving its own slave land After the address and the R bit, the serial interrupt flag (SI) is set and a valid status code can be read from the I2C_SR. This status code is used As the vector of the status service program, the corresponding operation of each status code is shown in the table below. If the I2C module is in master mode If the trimming is lost, it can enter the slave transmission mode (see status 0xB0).

If the AA bit is reset during a transfer, the I2C block will send the last byte and go to state 0xC0 or 0xC8. I2C module switch Switching to non-addressed slave mode, it will ignore the master receiver if it continues to transmit. So the master receiver receives all 1's as serial data. When AA is reset, the I2C block does not respond to its own slave address or general call address. However, the I2C bus is still monitored and the Also, address recognition can be resumed at any time by setting bit AA. This means that the AA bit can be used to temporarily disconnect the I2C module from the I2C bus. get out.

Table 23-5 Slave sending mode status table

| state<br>the code | I2C bus and hardware status | application response | | | | | The next action performed by the I2C hardware |
|---|---|---|---|---|---|---|---|
| | | read/write<br>I2C_DATA | Write to I2C_CR | | | | |
| | | | STA | STO | SI | AA | |
| A8H | has received its own SLA+R; ACK has been returned | Load data byte X | | 0 | 0 | 0 | The last data byte will be sent; ACK will be received; |
| | | Load data byte X | | 0 | 0 | 1 | will send a data byte; will receive Receive ACK; |
| B0H | In SLA+R/W when master Arbitration lost in ; Received Own SLA+R; returned ACK; | Load data byte X | | 0 | 0 | 0 | The last data byte will be sent; ACK will be received; |
| | | Load data byte X | | 0 | 0 | 1 | will send a data byte; will receive Receive ACK; |
| B8H | Data sent; received ACK; | Load data byte X | | 0 | 0 | 0 | will send the last data byte will; receive ACK; |
| | | Load data byte X | | 0 | 0 | 1 | will send a data byte; will receive Receive ACK; |
| C0H | data bytes sent; received Receive non-ACK; | no I2C_DATA action | 0 | 0 | 0 | 0 | switch to non-addressable slave mode; no Identify itself from the address or general ground site; |
| | | no I2C_DATA action | 0 | 0 | 0 | 1 | switch to non-addressable slave mode; no Identify itself from the address or general ground site; |
| | | no I2C_DATA action | 1 | 0 | 0 | 0 | switch to non-addressable slave mode; no Identify itself from the address or general ground address; send a start when the bus is free condition; |
| | | no I2C_DATA action | 1 | 0 | 0 | 1 | switch to non-addressable slave mode; no Identify itself from the address or general ground address; send a start when the bus is free condition; |
| C8H | Loaded data bytes have been sent Sent; Received ACK; | no I2C_DATA action | 0 | 0 | 0 | 0 | switch to non-addressable slave mode; no Identify itself from the address or general ground site; |
| | | no I2C_DATA action | 0 | 0 | 0 | 1 | switch to non-addressable slave mode; no Identify itself from the address or general ground site; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | No I2C_DATA action | 1 | 0 | 0 | 0 | Switches to non-addressable slave mode; does not recognize own slave address or general address; sends START condition when bus is free; |
| | | No I2C_DATA action | 1 | 0 | 0 | 1 | switches to non-addressable slave mode; does not recognize own slave address or general address; sends START condition when bus is free condition; |

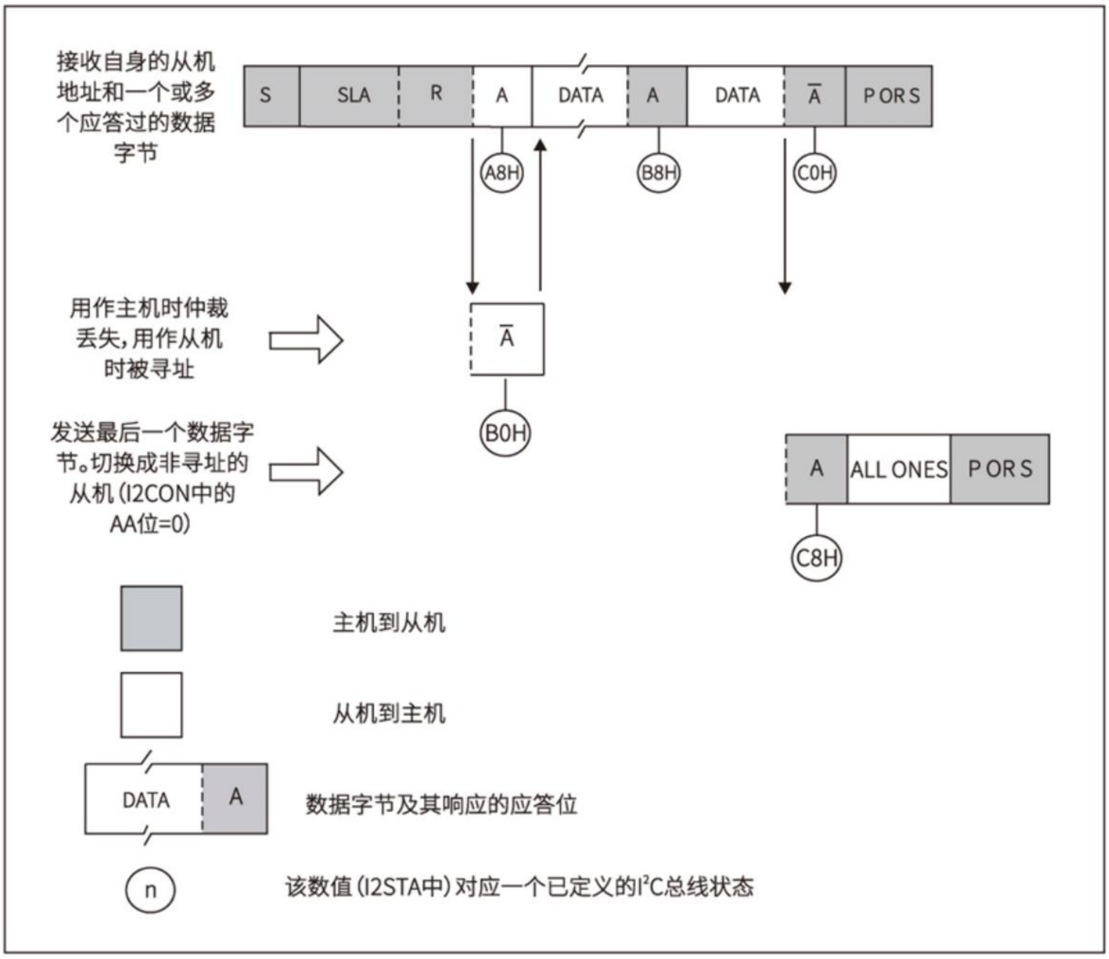

接收自身的从机地址和一个或多个应答过的数据字节
S
SLA
R
A
DATA
A
DATA
Ā
P OR S
A8H
B8H
C0H
用作主机时仲裁丢失，用作从机时被寻址
Ā
B0H
发送最后一个数据字节。切换成非寻址的从机（I2CON中的AA位=0）
A
ALL ONES
P OR S
C8H
主机到从机
从机到主机
DATA
A
数据字节及其响应的应答位
n
该数值（I2STA中）对应一个已定义的I²C总线状态

Figure 23-12 I2C slave sending state diagram

### 23.5.10 I2C Miscellaneous Status

I2C_SR = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag SI has not been set. This is the case in other state and the I2C module has not started to perform a serial transfer.

I2C_SR = 0x00

This status code indicates that a bus error occurred during an I2C serial transfer. When a start or stop occurs at an illegal position in a format frame condition a bus error occurs. These illegal locations refer to address bytes, data bytes, or acknowledge bits during serial transfers. When outside Bus errors can also occur when external disturbances affect the internal I2C block signals. SI is set when a bus error occurs. To recover from a bus error reset, the STO flag must be set and SI must be cleared. This puts the I2C block into "unaddressed" slave mode (defined state) and clears the STO flag (other bits in I2C_CR are not affected). SDA and SCL lines are released (stop condition is not sent).

Table 23-6 Other Miscellaneous Status Table

| shape<br>state<br>generation<br>code | I2C bus and hardware status | application response | | | | | The I2C hardware executes the next action |
|---|---|---|---|---|---|---|---|
| | | Read/write I2C_DATA | Write to I2C_CR | | | | |
| | | | STA | STO | SI | AA | |
| F8H | No relevant status letter available Information; SI=0; | No I2C_DATA action do | No I2C_DATA action | | | | wait or execute current transfer |
| 00H | due to illegal start or stop stop condition occurs, the host or the selected slave will output A bus error is present; when an external Glitch puts I2C into pending righteous state will also appear 0x00 state | No I2C_DATA action do | 0 | 1 | 0 | x | are addressed only if the host or In slave mode, the internal Hardware is affected. general situation case, the bus is released, I2C module switches to non-addressed slave mode. STO complex bit. |

## 23.6 I2C mode of operation

### 23.6.1 Initialization procedure

Example of using the I2C interface initialization as a slave and/or master.

1. Load its own slave address into I2C_ADDR, enable general call identification (if necessary);
2. Enable I 2 C interrupt;
3. Write 0x44 to the register I2C_CR to set the ENS and AA bits and enable the slave function. For master functions, register the I2C_CR writes 0x40.

### 23.6.2 Port configuration program

Example of I2C interface signals SCL, SDA mapped to chip pins PB4, PB5.

1. Configure PB4 and PB5 as open-drain output mode: P2OD[6], P2OD[5] are configured as 0x1
2. Configure the function configuration registers of PB4 and PB5: PBAFR4 and PBAFR5 are configured as 0x4
3. Configure the pull-up enable configuration registers of PB4 and PB5: PBPPUPD4 and PBPPUD5 are configured as 0x1

### 23.6.3 Start Host Send Function

The master send operation is performed by setting up the buffer, pointer and data count and then asserting the START condition.

. Initialize the host data counter;

2. Establish the slave address to which the data will be sent and add the write bit;

3. To I 2 C _ Write 0 x 2 0 to CR to set the STA bit;

4 . Establish the data to be sent in the main send buffer;

5. Initialize the host data counter to match the length of the message being sent;

6. Exit.

### 23.6.4 Start host receiving function

A master receive operation is performed by setting up buffers, pointers, and data counts and then asserting a start condition.

1 . Initialize the host data counter;

2 . Establish the slave address to which the data will be sent and add the read bit;

3. To I 2 C _ Write 0 x 2 0 to CR to set the STA bit;

4 . Establish the data to be sent in the main receive buffer;

5. Initialize the host data counter to match the length of the message being sent;

6. Exit.

### 23.6.5 I2C Interrupt Routine

Determines the status of the I2C and the status routine that handles that status.

1 . From I 2 C _ Read the status of I 2 C from SR;

2 . Use the state value to jump to one of 2 6 possible state programs.

### 23.6.6 Unspecified Mode Status

1. Status: 0x00 Bus error. Enters non-addressed slave mode and releases the bus.

   a) Write 0x14 to I2C_CR to set the STO and AA bits;

   b) Write 0xF7 to I2C_CR to clear the SI flag;

   c) exit

2. Host Status

States 08 and 10 apply to master transmit mode and master receive mode.

The R/W bit determines whether the next state is in master transmit mode or in master receive mode.

3. Status: 0x08

A start condition has been sent. About to send slave address + R/W bit and receive ACK bit.

   a) Write the slave address and R/W bit to I2C_DATA;

   b) Write 0x04 to I2C_CR to set AA bit;

   c) Write 0xF7 to I2C_CR to clear the SI flag;

   d) Establish the data buffer of the main sending mode;

   e) Establish the main receive mode data buffer;

   f) Initialize the host data counter;

   g) Exit.

4. Status: 0x10

A repeated start condition has been sent. About to send slave address + R/W bit and receive ACK bit.

a) Write the slave address and R/W bit to I2C_DATA;

b) Write 0x04 to I2C_CR to set AA bit;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Establish the data buffer of the main sending mode;

e) Establish the main receive mode data buffer;

f) Initialize the host data counter;

g) Exit.

### 23.6.7 Master Transmit Status

1. Status: 0x18

A previous state of 8 or 10 indicates that the slave address and write bits were sent and an acknowledgment was received. The first data byte is about to be sent and the ACK bit is received.

a) Load the first data byte of the main send buffer into I2C_DATA;

b) Write 0x04 to I2C_CR to set the AA bit;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Add 1 to the main send buffer pointer;

e) Exit.

2. Status: 0x20

Slave address and write operation bits have been sent and no acknowledgment received. A STOP condition is about to be sent.

a) Write 0x14 to I2C_CR to set the STO and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) exit

3. Status: 0x28

Data has been sent and ACK has been received. If the data sent is the last data byte send a STOP condition, otherwise send next data byte.

a) The host data counter is decremented by 1, if what is sent is not the last data byte, just skip to the e) step;

b) Write 0x14 to I2C_CR to set the STO and AA bits;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) exit;

e) Load the next data byte of the main send buffer into I2C_DATA;

f) Write 0x04 to I2C_CR to set the AA bit;

g) Write 0xF7 to I2C_CR to clear the SI flag;

h) The host sends the buffer pointer plus 1;

i) Exit.

4. Status: 0x30 Data has been sent and a non-acknowledgement has been received. The stop condition is about to be sent;

a) Write 0x14 to I2C_CR to set the STO and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

5. Status: 0x38 Arbitration has been lost during the transmission of the slave address and write operation bits or data. The bus has been released and entered the non-addressed slave mode. A new START condition will be sent when the bus becomes free again.

a) Write 0x24 to I2C_CR to set the STA and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

### 23.6.8 Master Receive Status

1. Status: 0x40

A previous state of 08 or 10 indicates that the slave address and read operation bits have been sent, and an ACK has been received. Will receive data and return ACK.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

2. Status: 0x48

Slave Address and Read Action bit sent and Not Acknowledged received. A STOP condition will be sent.

a) Write 0x14 to I2C_CR to set the STO and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

3. Status: 0x50

Data has been received and ACK is returned. Data will be read from I2C_DATA. Additional data will be received. if this is the last number Byte, return non-acknowledgement, otherwise return ACK.

a) Read the data bytes in I2C_DATA and store them in the host receiving buffer;

b) The host data counter is decremented by 1, and if it is not the last data byte, skip to step e);

c) Write 0xF3 to I2C_CR to clear SI flag and AA bit;

d) exit;

e) Write 0x04 to I2C_CR to set the AA bit;

f) Write 0xF7 to I2C_CR to clear the SI flag;

g) The host receive buffer pointer plus 1;

h) Exit.

4. Status: 0x58

Data has been received, a non-acknowledgement has been returned. Will read data from I2C_DATA and send stop condition.

a) Read the data bytes in I2C_DATA and store them in the host receiving buffer;

b) I2C_CR writes 0x14 to set the STO and AA bits;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Exit.

### 23.6.9 Slave Receive Status

1. Status: 0x60

Own slave address and write operation bits have been received, ACK has been returned. Will receive data and return ACK.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Create a slave receive mode data buffer;

d) Initialize the slave data counter;

e) Exit.

2. Status: 0x68

Arbitration has been lost while transmitting the slave address and R/W bit when acting as a bus master. has received its own slave address and write bit, and ACK has been returned. Will receive data and return ACK. Set STA to restart master mode when the bus becomes free again.

a) Write 0x24 to I2C_CR to set the STA and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Create a slave receive mode data buffer;

d) Initialize the slave data counter;

e) Exit.

3. Status: 0x70

A generic call has been received and an ACK is returned. Will receive data and return ACK.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Create a slave receive mode data buffer;

d) Initialize the slave data counter;

e) Exit.

4. Status: 0x78

Arbitration has been lost while transmitting the slave address and R/W bit when acting as a bus master. Received generic call and returns ACK. Will receive data and return ACK. Set STA to restart master mode when the bus becomes free again.

a) Write 0x24 to I2C_CR to set the STA and AA bits;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Create a slave receive mode data buffer;

d) Initialize the slave data counter;

e) exit

5. Status: 0x80

Previously addressed its own slave address. Data has been received and ACK is returned. Additional data will be read.

a) Read the data bytes of I2C_DATA and store them in the slave receive buffer.

b) Subtract 1 from the data counter of the slave machine, if it is not the last data byte, skip to step e);

c) Write 0xF3 to I2C_CR to clear SI flag and AA bit;

d) exit;

e) Write 0x04 to I2C_CR to set the AA bit;

f) Write 0xF7 to I2C_CR to clear the SI flag;

g) Add 1 to the receive buffer pointer of the slave;

h) Exit.

6. Status: 0x88

Previously addressed its own slave address. Data has been received and a non-acknowledgement has been returned. Received data will not be saved. Enter non-addressed slave mode.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

7. Status: 0x90

Before addressing the general call address. Data has been received and ACK is returned. The received data will be saved. Only the first data word is received section and return ACK. A non-acknowledgement is returned after receiving additional data bytes.

a) Read the data bytes of I2C_DATA and put them into the slave receiving buffer;

b) Write 0xF3 to I2C_CR to clear SI flag and AA bit;

c) Exit.

8. Status: 0x98

Before addressing the general call address. Data has been received and a non-acknowledgement has been returned. Received data will not be saved. Access to non-addressed slave model.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

9. Status: 0xA0

A STOP condition or repeated START condition has been received, but is still being addressed as a slave. Received data is not saved. Entering non-addressed slave mode Mode.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

### 23.6.10 Slave Transmit Status

1. Status: 0xA8

Received own slave address and read operation bit and returned ACK. Data will be sent and ACK bit will be received.

a) Load the first data byte of the slave transmit buffer into I2C_DATA;

b) Write 0x04 to I2C_CR to set the AA bit;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Create a data buffer in the slave sending mode;

e) Add 1 to the slave sending buffer pointer;

f) Exit.

2. Status: 0xB0

When acting as a bus master, arbitration is lost during the transfer of the slave address and the R/W bit. Received own slave address and read operation bit and returned ACK. Data will be sent and ACK bit will be received. Set STA to restart master mode when the bus becomes free again.

a) Load the first data byte of the slave transmit buffer into I2C_DATA;

b) Write 0x24 to I2C_CR to set the STA and AA bits;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Create a data buffer in the slave sending mode;

e) Add 1 to the slave sending buffer pointer;

f) Exit.

3. Status: 0xB8

Data has been sent and ACK has been received. Data will be sent and ACK bit will be received.

a) Load the data bytes of the slave send buffer into I2C_DATA;

b) Write 0x04 to I2C_CR to set the AA bit;

c) Write 0xF7 to I2C_CR to clear the SI flag;

d) Add 1 to the slave sending buffer pointer;

e) Exit.

4. Status: 0xC0

Data was sent and a non-acknowledge was received. Enter non-addressed slave mode.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

5. Status: 0xC8

The last data byte has been sent and an ACK has been received. Enter non-addressed slave mode.

a) Write 0x04 to I2C_CR to set the AA bit;

b) Write 0xF7 to I2C_CR to clear the SI flag;

c) Exit.

## 23.7 I2C Register List

I2C base address: 0x40000C00

Table 23-7 I2C Register List

| offset address | name | Register description | reset value |
|---|---|---|---|
| 0x00 | I2C_CR | I2C configuration register. | 0x00000000 |
| 0x04 | I2C_DATA | I2C data register. | 0x00000000 |
| 0x08 | I2C_ADDR | I2C address register. | 0x00000000 |
| 0x0c | I2C_SR | I2C status register. | 0x000000F8 |
| 0x10 | I2C_TIMRUN | I2C baud rate counter enable register. | 0x00000000 |
| 0x14 | I2C_BAUDCR | I2C Baud Rate Counter Configuration Register. | 0x00000000 |

## 23.8 I2C Register Description

### 23.8.1 I2C Configuration Register (I2C_CR)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 30 29 28 27 26 25 24 23 | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| reserve | | | | | | | |

| 15 14 13 12 11 10 | 9 | 8 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| reserve | | | ENS | STA | STO | Si | AAA | reserve | H1M |
| | | | R/W | R/W | R/W | R/W | R/W | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:7 | - | reserve | 0x0 | - |
| 6 | ENS | The I2C module is enabled.<br>0: disabled<br>1: Enable | 0 | R/W |
| 5 | STA | start flag enable.<br>0: disabled<br>1: Enable | 0 | R/W |
| 4 | STO | stop flag enable.<br>0: disabled<br>1: enable | 0 | R/W |
| 3 | Si | I2C interrupt flag bit. The | 0 | R/W0C |
| 2 | AAA | acknowledge flag is enabled.<br>0: disabled<br>1: enable | 0 | R/W |
| 1 | - | reserve | 0 | - |
| 0 | H1M | I2C high speed 1Mbps mode enable.<br>0: disabled<br>1: enable | 0 | R/W |

### 23.8.2 I2C Data Register (I2C_DATA)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | I2CDAT[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 - | | reserve | 0x0 | - |
| 7:0 | I2CDAT[7:0] | I2C data register.<br>In I2C transmit mode, write transmit data to this register. In I2C receive mode, read Receive data from this register. | 0x00 | R/W |

### 23.8.3 I2C Address Register (I2C_ADDR)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | I2CADR[6:0] | | | | | | | GC |
| | | | | | | | | R/W | | | | | | | R/W |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:1 | I2CADR[6:0] | I2C slave mode address. Broadcast | 0x0 | R/W |
| 0 | GC | address acknowledge enable.<br>0: disabled<br>1: enable | 0 | R/W |

### 23.8.4 I2C Status Register (I2C_SR)

Offset address: 0x0C

Reset value: 0x0000 00F8

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | I2CSTA[7:0] | | | | | | | |
| | | | | | | | | RO | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:0 | I2CSTA[7:0] | I2C status register. | 0xF8 | R/W |

### 23.8.5 I2C Baud Rate Counter Enable Register (I2C_TIMRUN)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | | | | | | | | TME |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | - | reserve | 0x0 | - |
| 0 | TME | Baud Rate Counter Enable Register.<br>0: disabled<br>1: enable | 0 | R/W |

### 23.8.6 I2C Baud Rate Counter Configuration Register (I2C_BAUDCR)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | TM[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| Bit Flag | Functional | Description | reset value | read<br>and write |
|---|---|---|---|---|
| 31:8 - | | reserve | 0x0 | - |
| 7:0TM | | TM: Baud rate counter configuration value.<br>Fo=Fi/8*(N+1), N=TM, N>0 | 0x0 | R/W |

# twenty four Serial Peripheral Interface (SPI)

## 24.1 Introduction to SPIs

The SPI (Serial Peripheral Interface) bus is a synchronous serial peripheral interface, which enables the MCU to communicate with various peripheral devices in serial

The direction of the line is exchanged. The SPI interface uses 4 lines: serial clock line (SCLK), master output/slave input line (MOSI), master

Input/Slave Output Line (MISO), Active Low Slave Select Line (SSN).

## 24.2 SPI main features

The SPI controller supports the following features:

ÿCan be configured as master or slave by programming

ÿFull -duplex communication capability

ÿ 7 baud rates are configurable.

ÿ 4-wire transmission mode

ÿMaximum baud rate in master mode is 1/2 system clock

ÿThe maximum baud rate of the slave mode is 1/4 of the system clock

ÿConfigurable serial clock polarity and phase

ÿ Support interrupt mode

ÿ 8-bit data transmission first transmits the high bit and then the low bit

## 24.3 Functional Description

### 24.3.1 SPI master mode

All data transfers on the SPI bus are initiated by the SPI master, which is set by setting the master/slave control bit SPI_CR.MSTR to "1".

SPI is placed in host mode. When SPI is in master mode, enable SPI (set SPI_CR.SPEN to "1") and register data to SPI at the same time

Data transfer begins when a byte is written to SPI_DATA. The SPI master immediately shifts out data serially on the MOSI line, while

The serial clock is provided on SCK and the SPI_SR.SPIF interrupt flag is set to "1" after the end of the transfer. If interrupts are enabled, will generate

Generate an interrupt request. In full-duplex applications, when an SPI master sends data to a slave, the addressed SPI slave can be

Sends data to the master device on the MISO line. Therefore, the SPIF flag is used both as a sending completion flag and as a ready flag for receiving data. place

The processor gets the received data by reading the SPI_DATA register.

#### 24.3.1.1 Operation process

1. Port configuration: configure the port controller, map the SCK, MISO, MOSI signals to the correct pins, and set them as correct input/output status.

2. In host mode, the level of the chip select signal SPI_CS is determined by the value of the register SPI_SSN.SSN

3. SPI baud rate configuration: set SPI_CR.SPR2, SPI_CR.SPR1, SPI_CR.SPR0

4. Serial clock configuration: set SPI clock polarity SPI_CR.CPOL, clock phase SPI_CR.CPHA. See SPI_CR register for details

5. Host mode configuration: SPI_CR.MSTR=1

6. SPI enable is on: SPI_CR.SPEN=1

7. Slave selection: configure SPI_SSN.SSN=0;

8. Start sending data: the data to be sent to the slave is written to the SPI data register SPI_DATA

9. Wait for the completion of sending/receiving data, and prepare to send/receive the next data

The following is the interrupt service routine:

10. Read the SPI_DATA register, that is, receive the data sent from the slave device.

Notice:

ÿ After step 10 is completed, the SPIF interrupt is cleared to "0"; if you want to send/receive data continuously, repeat steps 8 and 9.

ÿ In multi-machine communication, the SSN pin can be replaced by GPIO.

#### 24.3.1.2 Timing

Using the clock polarity CPOL and phase CPHA in the SPI control register SPI_CR, the serial clock can be selected from 4 combinations1 kind. SPI_CR.CPOL is whether SCK is high or low when SPI is idle. SPI_CR.CPHA is when choosing two

Clock phase (edge used to sample data) One. Master and slave must be configured to use the same clock phase and polarity. baud rate

The setting is only valid for the master, and the baud rate setting for the slave will be ignored. The main mode data/clock timing is as follows:

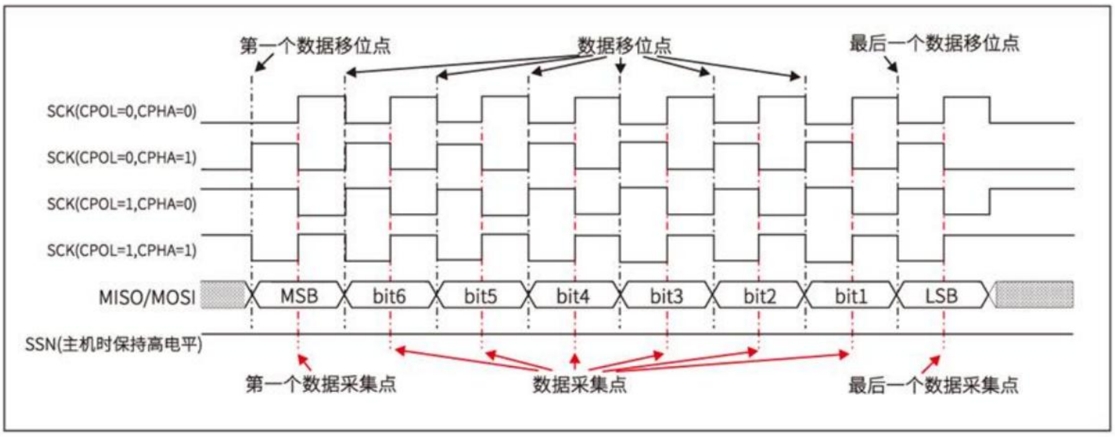


Figure 24-1 Data/clock timing diagram in host mode

## 24.3.2 SPI slave mode

When the SPI is enabled and not configured as a master, it operates in SPI slave mode. By setting the master/slave control bit

Setting SPI_CR.MSTR to "0" puts the SPI in slave mode. When the SPI is in slave mode, the serial clock (SCK) is controlled by the master device,

Move in data from MOSI. The counter in the SPI logic counts the SCK edges, and when the 8-bit data shift is complete, the SPIF flag is set

"1". Received data is obtained by reading SPI_DATA. The slave device cannot start the data sending data function, by writing SPI_DATA to

The data to be sent to the master device is preloaded, and under the action of the master device SCK, one by one is moved to the MISO line to send to the master device.

### 24.3.2.1 Operation process

1. Port configuration: configure the port controller, map the SCK, MISO, MOSI signals to the correct pins, and set them as correct

   input/output status.

2. In slave mode, a GPIO is selected by the port control register as the source of the chip select signal, see 7.2.3 Terminal control register

   device (SYSCON_PORTCR)

3. SPI baud rate configuration: set SPI_CR.SPR2, SPI_CR.SPR1, SPI_CR.SPR0

4. Serial clock configuration: set SPI clock polarity SPI_CR.CPOL, clock phase SPI_CR.CPHA. See SPI_CR register for details

5. Slave mode configuration: SPI_CR.MSTR=0

6. SPI enable is on: SPI_CR.SPEN=1

7. The data to be sent to the host is written to the SPI data register SPI_DATA

8. Wait for the completion of sending/receiving data, and prepare to send/receive the next data

The following is the interrupt service routine:

9. Read the SPI_DATA register, that is, receive the data sent by the master device

Notice:

ÿ SPIF interrupt is cleared to "0" after step 9 is completed.

ÿWhen the slave clock phase SPI_CR.CPHA is configured as 0, each time the master pulls down the SPI_CS signal, only one

bytes of data. If the host pulls down the SPI_CS signal every time, when it wants to continuously transmit data to the slave, the transmission complete interrupt

It needs to wait at least 1/2 SCK cycle after the occurrence to perform the next SPI_DATA write operation.

ÿIf you want to send/receive data continuously, repeat steps 7 and 8.

The slave mode data/clock timing is as follows:

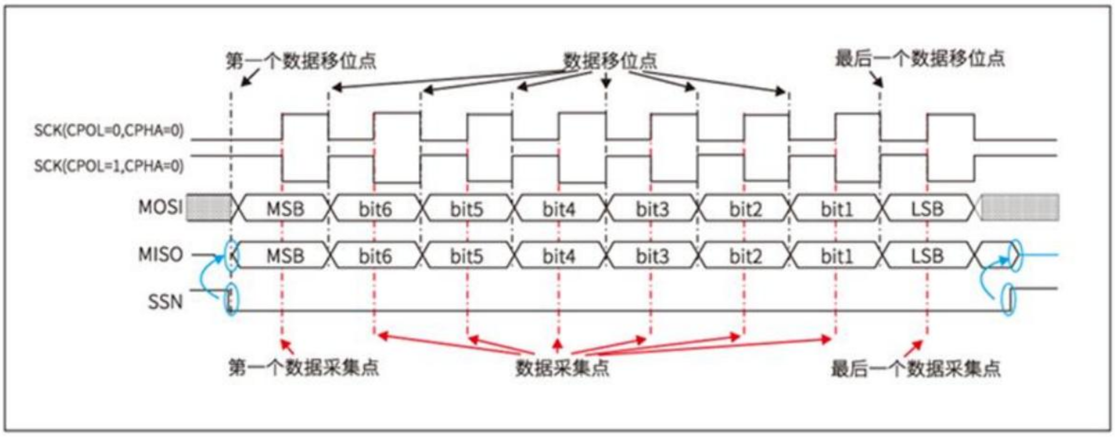


Figure 24-2 Slave mode data/clock timing diagram (CPHA=0)

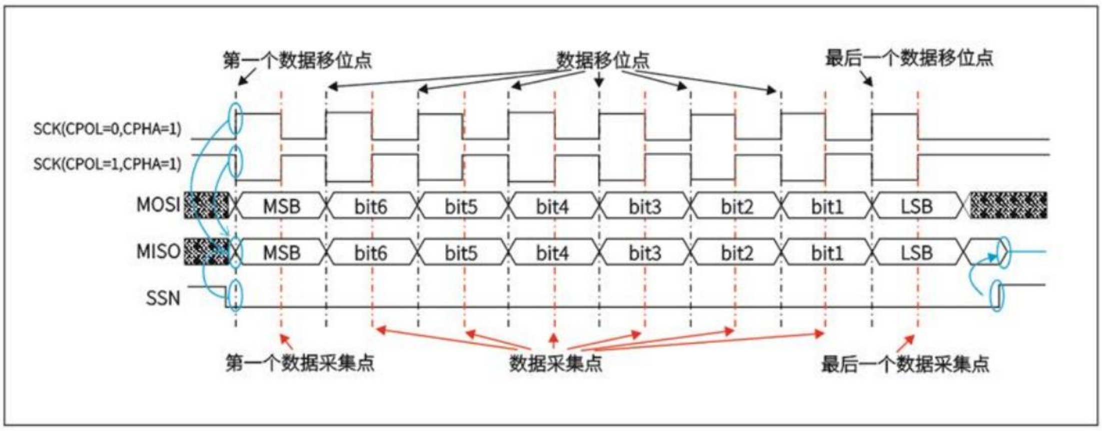


Figure 24-3 Slave mode data/clock timing diagram (CPHA=1)

## 24.4 SPI interrupt

If the SPI interrupt is enabled, an interrupt will be generated when the SPI transfer completion interrupt flag bit SPI_SR.SPIF is set to "1"; or when the SPI

The host mode error interrupt flag bit SPI_SR.MDF is set to "1" will also generate an interrupt.

At the end of each byte transfer, the SPI transfer completion interrupt flag bit SPI_SR.SPIF will be automatically set to "1" by hardware.

When the SPI is configured as the host mode, the external SSN input is low level. At this time, the SSN input level conflicts with the SPI working mode.

SPI host mode error interrupt flag bit SPI_SR.MDF is automatically set to "1" by hardware.

## 24.5 Multi-Master/Multi-Slave Mode

When this product is used as the host in the SPI single-master single-slave system, the SPI chip select configuration register SPI_SSN can be configured to output high/low level signal to the SPI_CS pin. As a slave, you can configure the 7.2.3 terminal control register (SYSCON_PORTCR) register

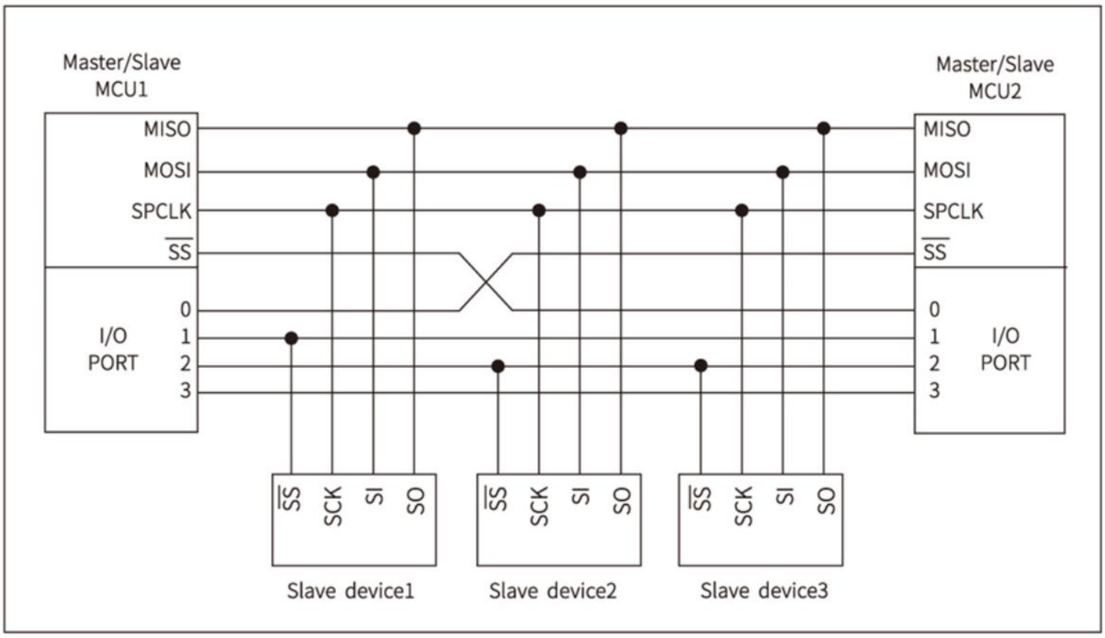
Select a GPIO pin as the source of SPI_SSN. However, in a multi-master and multi-slave system, as shown in the figure below, you must follow the

corresponding process configuration.

When the system is a single master with multiple slaves, you can use the SPI_CS pin as the chip select signal of slave 1, and the chip select signals of other slaves pass through

connected via GPIO pins. When the system is multi-master and multi-slave, all slave chip select signals are connected through GPIO pins, and the master also

The GPIO pin must be connected to the SPI_CS signal of other hosts to monitor whether the bus is occupied.



## 24.6 SPI register list

Base address: 0x4000 0800

Table 24-1 Register list

| offset address | name | describe | reset value |
|---|---|---|---|
| 0x00 | SPI_CR | SPI configuration register. | 0x0000 0014 |
| 0x04 | SPI_SSN | SPI chip select configuration register. | 0x0000 0001 |
| 0x08 | SPI_SR | SPI status register. | 0x0000 0000 |
| 0x0C | SPI_DATA | SPI data register. | 0x0000 0000 |

## 24.7 SPI register description

### 24.7.1 SPI Configuration Register (SPI_CR)

Offset address: 0x00

Reset value: 0x0000 0014

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SPR2 | SPEN | reserve | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| | | | | | | | | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | SPR2 | Baud rate selection bit 2, refer to Table 24-2 Baud rate configuration table | 0 | R/W |
| 6 | SPEN | SPI Module Enable Register<br>0: disabled<br>1: enable | 0 | R/W |
| 5 | – | reserve | 0 | – |
| 4 | MSTR | Master/slave mode selection<br>0: slave<br>1: Host | 1 | R/W |
| 3 | CPOL | Clock Polarity Select Register<br>0: low<br>1: High | 0 | R/W |
| 2 | CPHA | Clock Phase Select Register<br>0: first edge<br>1: second edge | 1 | R/W |
| 1 | SPR1 | Baud rate selection bit 1, refer to Table 24-2 Baud rate configuration table | 0 | R/W |
| 0 | SPR0 | Baud rate selection bit 0, refer to Table 24-2 Baud rate configuration table | 0 | R/W |

Table 24-2 Baud rate configuration table

| SPR2 0 | SPR1 0 | SPR0 0 | SPI_CLK Rate |
|---|---|---|---|
| 0 | 0 | 1 | Fsys/2 |
| 0 | 1 | | Fsys/4 |
| 0 | 1 | 0 | Fsys/8 |
| 1 | | 1 | Fsys/16 |
| 1 | 0 | 0 | Fsys/32 |
| 1 | 0 | 1 | Fsys/64 |
| | 1 | 0 | Fsys/128 |

### 24.7.2 SPI Chip Select Configuration Register (SPI_SSN)

Offset address: 0x04

Reset value: 0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | SSN |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserved bit | 0x0 | – |
| 0 | SSN | SSN output value, in host mode, the software configures the SSN value to control the level of the SPI_CS port | 1 | R/W |

### 24.7.3 SPI Status Register (SPI_SR)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SPIF | WCOL | SSERR | MDF | reserve | | | |
| | | | | | | | | RO | RO | RO | RO | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7 | SPIF | transfer end interrupt flag | 0 | RO |
| 6 | WCOL | write conflict flag | 0 | RO |
| 5 | SSERR | Slave mode SSN error flag | 0 | RO |
| 4 | MDF | Host Mode Error Flags | 0 | RO |
| 3:0 | – | reserve | 0x0 | – |

### 24.7.4 SPI Data Register (SPI_DATA)

Offset address: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | SPIDATA[7:0] | | | | | | | |
| | | | | | | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserved bit | 0x0 | – |
| 7:0 | SPIDATA[7:0] | Data register in transmit mode, write transmit value to this register in receive mode, from This register reads the received value. | 0x0 | R/W |

# 25 One-Wire Interface (OWIRE)

## 25.1 Single bus protocol (One-Wire)

The host computer and the slave computer communicate through one wire, and the number of slave devices that can be attached to one bus is almost unlimited.

### 25.1.1 Features

It uses a single signal line, which can transmit both clock and data, and the data transmission is bidirectional.

### 25.1.2 Advantages

Single-bus technology has the advantages of simple wiring, less hardware overhead, low cost, and easy bus expansion and maintenance.

## 25.2 Single bus communication process

### 25.2.1 Initialization

Initialization process = reset pulse + slave acknowledge pulse.

The host generates a reset pulse by pulling down the single bus for 480~960 us, and then releases the bus to enter the receiving mode. When the master releases the bus, a Generate a rising edge that transitions from a low level to a high level. After the single-bus device detects the rising edge, there is a delay of 15~60 us, and the single-bus device pulls down Bus 60~240 us to generate the response pulse. The host receives the response pulse from the slave, indicating that the single-bus device is ready, and the initialization process Finish.

The initialization timing diagram is shown in Figure 25-1 reset and response pulse during initialization:

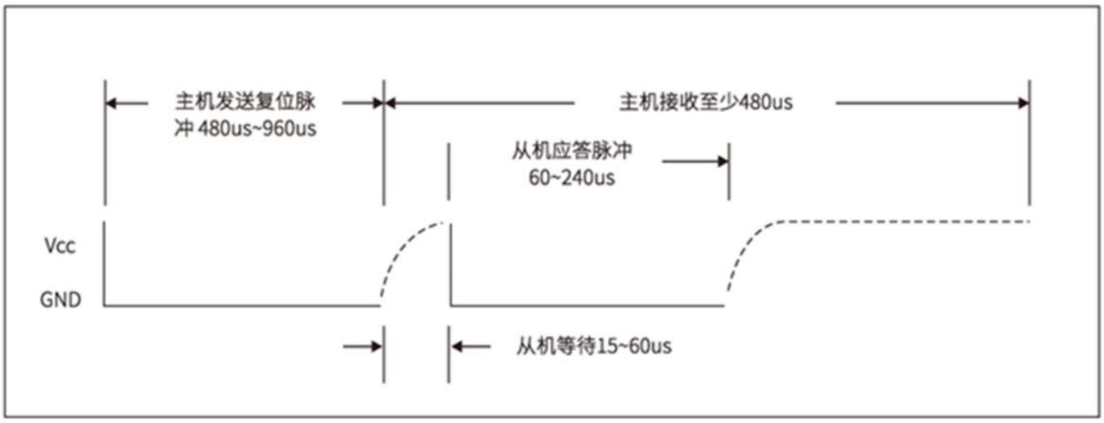


Figure 25-1 Reset and acknowledge pulse during initialization

### 25.2.2 Write Time Gap

There are two types of write time slots, including the time slot for writing 0 and the time slot for writing 1.

When the data line is pulled low, the data line is sampled within the time window of 15~60us. If the data line is low, it is to write 0, such as

If the data line is high, write 1. If the host wants to generate a time gap of writing 1, it must pull the data line low and open the time gap of writing

The data line is allowed to be pulled high within 15us after the start. To generate a write 0 time gap, the host must pull the data line low and keep it for 60us.

The timing diagram of the writing gap is shown in Figure 25-2 The timing diagram of the writing gap in the single-bus communication protocol:

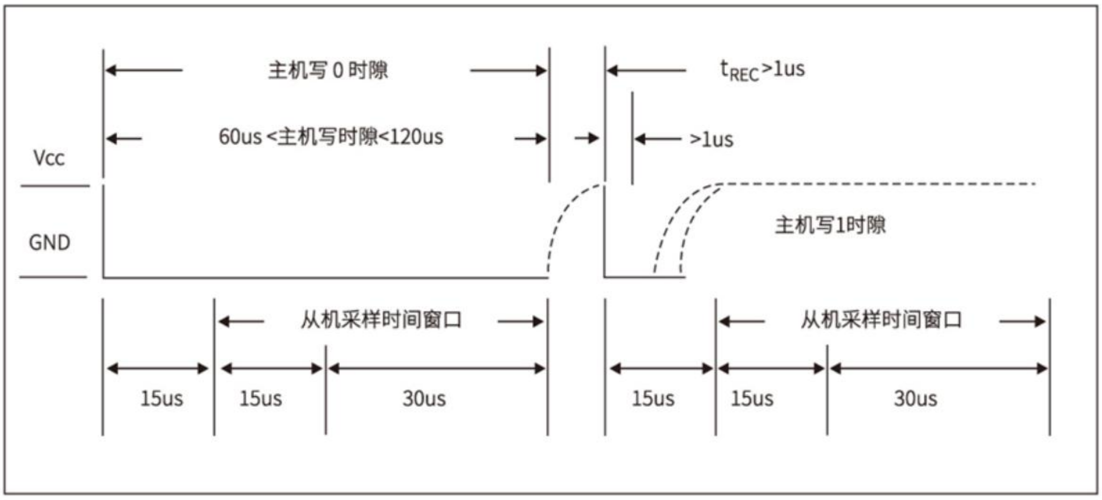


Figure 25-2 Sequence diagram of write time gap in single-bus communication protocol

### 25.2.3 Read Time Gap

When the host pulls the bus low and keeps it for at least 1us and then releases the bus, the data must be read within 15us. read

The time slot timing diagram is shown in Figure 25-3 Read time slot timing diagram in the single-bus communication protocol:

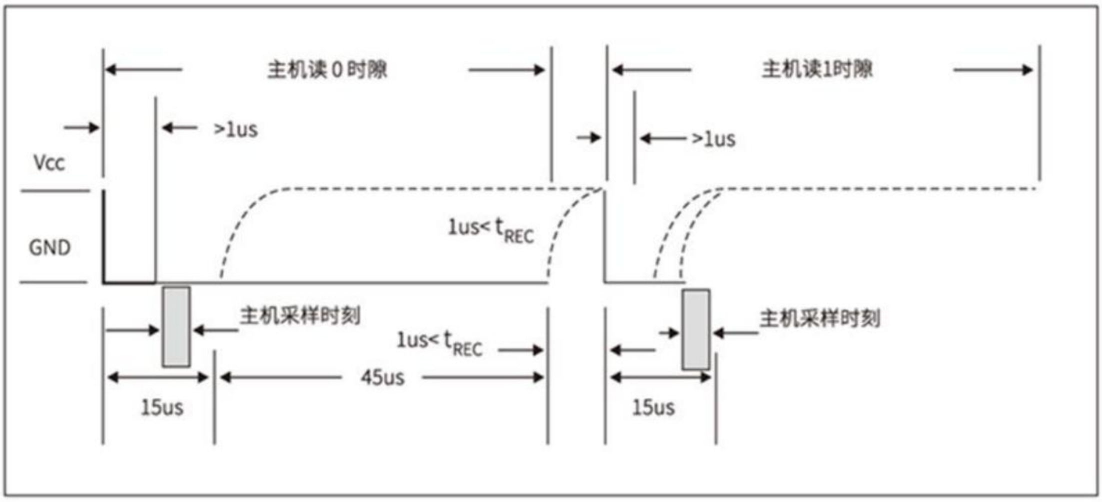


Figure 25-3 Sequence diagram of read time gap in single-bus communication protocol

## 25.3 Configuration instructions

### 25.3.1 Initial Configuration Instructions

1. Configure the corresponding GPIO pins to reuse One-Wire pins;

2. Configure the OWIRE_CR.CLKDIV register to set the One-Wire mode clock selection;

3. Configure the One-Wire Reset width control register OWIRE_RSTCNT to set the One-Wire master send reset time

   (480us~960us);

4. Configure the One-Wire response width count register OWIRE_PRESCNT, and set the One-Wire slave response setting count value

   (60us~240us);

5. Configure the One-Wire interrupt enable register OWIRE_INTEN.INITEN to enable the initialization completion interrupt;

6. Configure the OWIRE_CR.EN register to enable the One-Wire module;

7. Configure the One-Wire bus operation command register OWIRE_CMD, and set the Initial command;

8. The system enters the interrupt subroutine, configure the state clear register OWIRE_INTCLR in the interrupt subroutine, and clear the corresponding interrupt flag.

   Chi;

### 25.3.2 Read Data Configuration Instructions

1. Configure One-Wire Bit rate counter OWIRE_BITRATECNT, set 1 Bit data width (15us~60us);
2. Configure the One-Wire master read/write PULL0 drive time OWIRE_DRVCNT, set the drive time width (0us~15us);
3. Configure the One-Wire master read sampling time setting OWIRE_RDSMPCNT, set the read sampling time (1us~15us);
4. Configure the 1-Wire Recover Time count interval value OWIRE_RECCNT, and set the RECOVER time to (TREC>1us);
5. Configure the One-Wire interrupt enable register OWIRE_INTEN.RXDONEEN to enable the receive completion interrupt;
6. Configure the One-W ire bus operation command register OWIRE _ CMD, set RX command;
7. The system enters the interrupt subroutine, the status clear register OWIRE_INTCLR is configured in the interrupt subroutine, and the receiving completion interrupt is cleared. FLAG, and read the One-Wire data register OWIRE_DATA;

### 25.3.3 Write Data Configuration Instructions

1. Configure One-Wire Bit rate counter OWIRE_BITRATECNT, set 1 Bit data width (15us~60us);
2. Configure the One-Wire master read/write PULL0 drive time OWIRE_DRVCNT, set the drive time width (0us~15us);
3. Configure the 1-Wire Recover Time count interval value OWIRE_RECCNT, and set the RECOVER time to (TREC>1us);
4. Configure the One-Wire interrupt enable register OWIRE_INTEN.TXDONEEN to enable the transmit completion interrupt;
5. Write data to the One-Wire data register OWIRE_DATA;
6. Configure the One-Wire bus operation command register OWIRE_CMD, and set the TX command;
7. After the data is sent, the system enters the interrupt subroutine, and the status clear register OWIRE_INTCLR is configured in the interrupt subroutine to clear TX complete interrupt FLAG;

## 25.4 Register List

Base address: 0x4000 3800

| Offset Address | Name | Description 1-wire Module Control Register | Defaults |
|---|---|---|---|
| 0x00 | OWIRE_CR | | 0x0000 0000 |
| 0x04 | OWIRE_NFCR 0x0000 0000 | 1-Wire Input Terminal Filter Control Register | |
| 0x08 | OWRIE_RSTCNT 0x0000 0000 | 1-Wire Master Reset pulse Width Count Register | |
| 0x0C | OWIRE_PRESCNT | 1-Wire Device Presence Pulse Width Count Register 0x0000 0000 | |
| 0x10 | OWIRE_BITRATECNT 1-Wire Bit | rate design counter 0x0000 0000 | |
| 0x14 | OWIRE_DRVCNT | 1-Wire master read/write PULL0 drive time 0x0000 0000 | |
| 0x18 | OWIRE_RDSMPCNT 1-Wire mas | ter read sample time setting | 0x0000 0000 |
| 0x1C | OWIRE_RECCNT | 1-Wire Recover Time count interval value | 0x0000 0000 |
| 0x20 | OWIRE_DATA | 1-Wire Data Register | 0x0000 0000 |
| 0x24 | OWIRE_CMD 1- | 1-Wire Bus Operation Command Register | 0x0000 0000 |
| 0x28 | OWIRE_INTEN wire interrupt ena | ble register | 0x0000 0000 |
| 0x2C | OWIRE_SR 1-wire status register | 1-wire interrupt status clear | 0x0000 0000 |
| 0x30 | OWIRE_INTCLR | register | 0x0000 0000 |

## 25.5 Register description

### 25.5.1 1-Wire Module Control Register (OWIRE_CR)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | RDM<br>ODEs | MSB<br>f<br>IRST | EN | SIZE | reserve | | CLKDIV[1:0] | |
| | | | | | | | | R/W | R/W | R/W | R/W | | | R/W | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:8 – | | reserve | 0x0 | – |
| 7 | RDMODE | 0: normal mode<br>1: Write 0/read 0 bit pattern | 0 | R/W |
| 6 | MSB FIRST | setting for gap equal byte transmission<br>0: LSB(bit0) send/receive first<br>1: MSB(bit7) send/receive first<br>When OWIRE_CR.SIZE=0, this bit should be set to 0 | 0 | R/W |
| 5 | EN | 1-wire module enable control bit<br>0: 1-wire module stopped<br>1: 1-wire module enables | 0 | R/W |
| 4 | SIZE | data processing bit control bit<br>0: Single processing 1 bit (Bit mode)<br>1: Single processing 8 bit (Byte mode) | 0 | R/W |
| 3:2 | - | reserve | 0x0 | – |
| 1:0 | CLKDIV[1:0] | Counter clock source selection bit<br>00: FPCLK<br>01: FPCLK/2<br>10: FPCLK/4<br>11: FPCLK/16 | 0 | R/W |

### 25.5.2 1-Wire Input Terminal Filter Control Register (OWIRE_NFCR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | NFEN | reserve | | NFDIV[1:0] | |
| | | | | | | | | | | | R/W | | | R/W | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:5 – | | reserve | 0x0 | – |
| 4 | NFEN | Input terminal filter enable control bit<br>0: filter function is invalid<br>1: The filtering function is valid | 0 | R/W |
| 3:2 | - | reserve | 0x0 | – |
| 1:0 | NFDIV[1:0] | Input terminal filter clock source selection bit<br>00: FPCLK<br>01: FPCLK/2<br>10: FPCLK/4<br>11: FPCLK/8 | 0x0 | R/W |

### 25.5.3 1-Wire RESET Width Control Register (OWIRE_RSTCNT)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| RSTCNT[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:16 | – | reserve | 0x0 | – |
| 15:0 | RSTCNT[15:0] | Master sends reset time setting count value | 0x0 | R/W |

### 25.5.4 1-Wire Presence Pulse Width Count Register (OWIRE_PRESCNT)

Address offset: 0x00C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | PRESCNT[12:0] | | | | | | | | | | | | |
| | | | R/W | | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:13 | – | reserve | 0x0 | – |
| 12:0 | PRESCNT[12:0] | Set count value from response time | 0 | R/W |

### 25.5.5 1-Wire Bit rate design counter (OWIRE_BITRATECNT)

Address offset: 0x010

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | BITRATE CNT[11:0] | | | | | | | | | | | |
| | | | | R/W | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:12 – | | reserve | 0x0 | – |
| 11:0 | BITRATE CNT[11:0] | Bit Rate time setting count value | 0 | R/W |

### 25.5.6 1-Wire Master Read/Write PULL0 Drive Time (OWIRE_DRVCNT)

Address offset: 0x014

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | DRVCNT[8:0] | | | | | | | | |
| | | | | | | | R/W | | | | | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:9 | – | reserve | 0x0 | – |
| 8:0 | DRVCNT[8:0] | Master device reads/writes PULL0 drive time setting count value | 0x0 | R/W |

### 25.5.7 1-Wire Master Read Sample Time Set (OWIRE_RDSMPCNT)

Address offset: 0x018

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | RDSMPCNT[8:0] | | | | | | | | |
| | | | | | | | R/W | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:9 | – | reserve | 0x0 | – |
| 8:0 | RDSMPCNT[8:0] | The master reads the sampling time to set the count value | 0x0 | R/W |

### 25.5.8 1-Wire Recover Time count interval value (OWIRE_RECCNT)

Address offset: 0x01C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | RECCNT[10:0] | | | | | | | | | | |
| | | | | | R/W | | | | | | | | | | |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:11 | – | reserve | 0x0 | – |
| 10:0 | RECCNT[10:0] | Recover Time count interval value | 0x0 | R/W |

### 25.5.9 1-Wire Data Register (OWIRE_DATA)

Address offset: 0x020

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | DRVCNT[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | – | reserve | 0x0 | – |
| 7:0 | DATA[7:0] | 1bit mode (Bit mode): only send and receive bit0<br>8bit mode (Byte mode):<br>Can send and receive all 8 bits | 0x0 | R/W |

### 25.5.10 1-Wire Bus Operation Command Register (OWIRE_CMD)

Address offset: 0x024

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | CMD [1:0] | |
| | | | | | | | | | | | | | | R/W | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:2 | – | reserve | 0x0 | – |
| 1:0 | cmd | 00: reserved<br>01: Initial<br>10:TX<br>11: RX | 0x0 | R/W |

Machine Translated by Google

### 25.5.11 1-wire interrupt enable register (OWIRE_INTEN)

Address offset: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| reserve | | | | | | | | | | | | RXD<br>ONE<br>EN | TXD<br>ONE<br>EN | INIT<br>DON<br>E. | ACK<br>ERR<br>EN |
| | | | | | | | | | | | | R/W | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:4 | – | reserve | 0x0 | – |
| 3 | RXDONEEN | Receive complete interrupt enable<br>0: disabled<br>1: Enable | 0 | R/W |
| 2 | TXDONEEN | transmit complete interrupt enable<br>0: disabled<br>1: enable | 0 | R/W |
| 1 | INITEN | Initialization complete interrupt enable<br>0: disabled<br>1: Enable | 0 | R/W |
| 0 | ACKERREN | slave acknowledge error interrupt enable<br>0: disabled<br>1: enable | 0 | R/W |

## 25.5.12 1-wire status register (OWIRE_SR)

Address offset: 0x2c

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | RXD ONE | TXD ONE | INITD ONE | ACK ERR |
| | | | | | | | | | | | | RO | RO | RO | RO |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:4 | – | reserve | 0x0 | – |
| 3 | RXDONE | Receive complete interrupt status<br>0: Receive completion did not occur<br>1: receiving complete | 0 | RO |
| 2 | TXDONE | sending complete interrupt flag<br>0: Send completion did not occur<br>1: send complete | 0 | RO |
| 1 | INIT DONE | initialization complete flag<br>0: initialization completion did not occur<br>1: The initialization is completed | 0 | RO |
| 0 | ACKERR | and the slave responds to the error interrupt flag<br>0: Slave response error does not occur<br>1: Slave response error occurs | 0 | RO |

### 25.5.13 1-wire status clear register (OWIRE_INTCLR)

Address offset: 0x2c

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | RXD<br>ONE<br>CLR | TXD<br>ONE<br>CLR | INITD<br>ONE<br>CLR | ACK<br>ERR<br>CLR |
| | | | | | | | | | | | | WO WO WO WO WO | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:4 | – | reserve | 0x0 | – |
| 3 | RXDONECLR | Receive complete interrupt clear<br>Write 0: no effect<br>Write 1: clear send complete interrupt | 0 | WO |
| 2 | TXDONECLR | send complete interrupt flag clear<br>Write 0: no effect<br>Write 1: clear the receiving completion | 0 | WO |
| 1 | INIT DONE CLR | interrupt initialization completion interrupt flag clear<br>Write 0: no effect<br>Write 1: Clear initialization complete | 0 | WO |
| 0 | ACKERRCLR | interrupt slave response error interrupt flag clear<br>Write 0: no effect<br>Write 1: Clear slave acknowledge error interrupt | 0 | WO |

# 26 Clock Trim/Monitor Module (CLKTRIM)

## 26.1 Introduction

The CLKTRIM (Clock Trimming) module is a circuit specially used to calibrate/monitor the clock. Select the precise Use the clock source to calibrate the inaccurate clock source, repeat the calibration, and adjust the parameters of the inaccurate clock source until the frequency of the calibrated clock source reaches to the accuracy requirements. In the calibration mode, the count value will have a certain error, but it is within the allowable precision error range. In monitor mode select Select a stable clock source to monitor the system working clock, and monitor whether the system working clock is invalid under the set monitoring period occurs and generates an interrupt. In calibration mode and monitoring mode, the required clock source must be initialized and enabled, the specific configuration process Please refer to Chapter 6 System Reset and Clock (RCC).

## 26.2 Main features

CLKTRIM supports the following features:

ÿ Calibration mode

ÿMonitoring mode

ÿ 32-bit reference clock counter can load initial value

ÿ 32-bit clock counter to be calibrated with configurable overflow value

ÿ 6 reference clock sources

ÿ 4 clock sources to be calibrated

ÿ Support interrupt mode

## 26.3 CLKTRIM function description

### 26.3.1 CLKTRIM calibration mode

The calibration mode is mainly used to select an accurate clock source as a reference clock to calibrate an inaccurate clock source to be calibrated. by soft The software is calibrated repeatedly according to the following operation process, and the parameters of the clock source to be calibrated are adjusted until the clock source to be calibrated meets the frequency accuracy requirements.

#### 26.3.1.1 Operation process

1. Set the CLKTRIM_CR.REFCLK_SEL register to select the reference clock.

2. Set the CLKTRIM_CR.CALCLK_SEL register to select the clock to be calibrated.

3. Set CLKTRIM_CR.CLKEN to enable trim and reference clock.

4. Set the CLKTRIM_REFCON.RCNTVAL register to the calibration time.

5. Set the CLKTRIM_CR.IE register to enable the interrupt.

6. Set the CLKTRIM_CR.TRIM_START register to start the trim.

7. The reference clock counter and the clock to be calibrated counter start counting.

8. When the reference clock counter counts down from the initial value to 0, CLKTRIM_IFR.STOP is set to 1 and an interrupt is triggered.

9. The interrupt service subroutine judges that CLKTRIM_IFR.STOP is 1, and reads the register CLKTRIM_REFCNT and CLKTRIM_CALCNT value, clear the CLKTRIM_CR.TRIM_START register to end the calibration.

Note that in the calibration mode, it may happen that the clock counter to be calibrated is in the

In case of overflow before CLKTRIM_IFR.STOP is set to 1, CLKTRIM_IFR.CALCNT_OVF is set to 1, triggering an interrupt. interrupt service

When the subroutine finds that CLKTRIM_IFR.CALCNT_OVF is set to 1, clear the CLKTRIM_CR.TRIM_START register to end the calibration.

In this case, the calibration cannot be performed correctly, and the calibration time must be adjusted and re-calibrated. The specific steps are:

ÿSet the CLKTRIM_REFCON.RCNTVAL register to adjust the calibration time.

ÿSet the CLKTRIM_CR.TRIM_START register to restart the calibration.

### 26.3.2 CLKTRIM monitoring mode

The monitoring mode is mainly used to select a stable clock source as the reference clock, and monitor the system working clock in the set time period

abnormal state. In monitoring mode, only external HXT clock or external LXT clock can be selected as the monitored clock.

#### 26.3.2.1 Operation process

1. Set the CLKTRIM_CR.REFCLK_SEL register to select the reference clock.

2. Set the CLKTRIM_CR.CALCLK_SEL register to select the monitored clock.

3. Set CLKTRIM_CR.CLKEN to enable the monitored and reference clock.

4. Set the CLKTRIM_REFCON.RCNTVAL register to monitor interval time.

5. Set the CLKTRIM_CALCON.CALOVCNT register to the clock counter overflow time to be monitored.

6. Set the CLKTRIM_CR.MON_EN register to enable the monitor function.

7. Set the CLKTRIM_CR.IE register to enable the interrupt.

8. Set the CLKTRIM_CR.TRIM_START register to start monitoring.

9. The reference clock counter and the monitored clock counter start counting. 10.

When the counting of the reference clock counter reaches the monitoring interval time, judge whether the monitored clock counter overflows. If overflow means being

Watchdog clock is working fine. If no overflow indicates that the monitored clock fails, CLKTRIM_IFR.HXT_FAULT or

CLKTRIM_IFR.LXT_FAULT is set to 1 to trigger an interrupt.

11. If RCC_SYSCLKCR.CLKFAILEN is set to 1 (refer to 6.4.8 System clock source configuration register

(RCC_SYSCLKCR)), after the interrupt occurs, it will automatically switch the system clock source to the internal high-speed RC clock (HIRC), processing

Interrupt service subroutine, clear the interrupt flag CLKTRIM_IFR.HXT_FAULT or

CLKTRIM_IFR.LXT_FAULT, clear the CLKTRIM_CR.TRIM_START register to end monitoring.

## 26.4 CLKTRIM register list

Base address: 0x4000 3400

Table 26-1 CLKTRIM register list

| Offset Address Name | Register Description Configuration | Register. | reset value |
|---|---|---|---|
| 0x00 | CLKTRIM_CR | | 0x0000 0000 |
| 0x04 | CLKTRIM_REFCON Reference | counter initial value configuration register. | 0x0000 0000 |
| 0x08 | CLKTRIM_REFCNT Reference | Counter Value Register. | 0x0000 0000 |
| 0x0C | CLKTRIM_CALCNT Calibration | Counter Value Register. Interrupt | 0x0000 0000 |
| 0x10 | CLKTRIM_IFR | flag bit register. | 0x0000 0000 |
| 0x14 | CLKTRIM_CALCON interrupt flag clear | register CLKTRIM_ICLR | 0x0000 0000 |
| 0x18 | calibration counter overflow value | configuration register | 0xFFFF FFFF |

## 26.5 CLKTRIM register description

### 26.5.1 Configuration Register (CLKTRIM_CR)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | CLKEN | IE | MON_EN | CALCLK_SEL[1:0] | | REFCLK_SEL[2:0] | | | TRIM_START |
| | | | | | | | R/W | R/W | R/W | R/W | | R/W | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:9 - | | reserve | 0x0 | - |
| 8 | CLKEN | Reference clock and calibration clock enable<br>0: disabled<br>1: clock enable | 0x0 | R/W |
| 7 | IE | interrupt enable register<br>0: disabled<br>1: Enable | 0x0 | R/W |
| 6 | MON_EN | monitor mode enable register<br>0: disabled<br>1: enable | 0x0 | R/W |
| 5:4 | CALCLK_SEL[1:0] | To-Calibrate/Monitor Clock Select Register<br>00: HIRC<br>01: HXT<br>10: LIRC<br>11: LXT | 0x0 | R/W |
| 3:1 | REFCLK_SEL[2:0] | Reference clock select register:<br>000: HIRC<br>001: HXT<br>010: LIRC<br>011: LXT<br>100: HXT bypass clock | 0x0 | R/W |
| 0 | TRIM_START | Calibration/monitoring start register:<br>0: stop<br>1: start | 0x0 | R/W |

### 26.5.2 Reference Counter Disposition Configuration Register (CLKTRIM_REFCON)

Offset address: 0x04

Reset value: 0xFFFF FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RCNTVAL [31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | RCNTVAL | Reference counter initial value | 0x0 | R/W |

### 26.5.3 Reference Counter Value Register (CLKTRIM_REFCNT)

Offset address: 0x08

Reset value: 0xFFFF FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REFCNT[31:16] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | REFCNT | Reference counter value<br>To read this register, you need to enable the clock first. When TRIM_START is valid, the initial value written will be will be updated to the register | 0x0 | RO |

### 26.5.4 Calibration Counter Value Register (CLKTRIM_CALCNT)

Offset address: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CALCNT[31:16] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

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

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:0 | CALCNT | Calibration counter value | 0x0 | RO |

## 26.5.5 Interrupt Flag Register (CLKTRIM_IFR)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | HXT_ FAULT | LXT_ FAULT | CALCNT_OVF | STOP |
| | | | | | | | | | | | | RO | RO | RO | RO |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:4 - | | reserve | 0x0 | - |
| 3 | HXT_FAULT | HXT failure flag<br>1: HXT invalid<br>0: HXT not invalid | 0 | RO |
| 2 | LXT_FAULT | LXT invalid flag<br>1: LXT failure<br>0: LXT not expired | 0 | RO |
| 1 | CALCNT_OVF | calibration counter overflow flag<br>CLKTRIM_CR.TRIM_START write zero to clear this flag bit | 0 | RO |
| 0 | STOP | reference counter stop flag<br>CLKTRIM_CR.TRIM_START write zero to clear this flag | 0 | RO |

### 26.5.6 Interrupt Flag Clear Register (CLKTRIM_ICLR)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | HXT_ FAUL T_CL R | LXT_ FAUL T_CL R | reserve | |
| | | | | | | | | | | | | WO | WO | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:4 | - | reserve | 0x0 | - |
| 3 | HXT_FAULT_CLR | Clear HXT failure flag, write 1 to clear. | 0x0 | WO |
| 2 | LXT_FAULT_CLR | Clear LXT failure flag, write 1 to clear. | 0x0 | WO |
| 1:0 | - | reserve | 0x0 | - |

### 26.5.7 Trim Counter Overflow Value Configuration Register (CLKTRIM_CALCON)

Offset address: 0x18

Reset value: 0x0000FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | reserve | 0x0 | - |
| 15:0 | CALOVCNT | Calibration counter overflow value comparison value<br>In monitor mode, if the monitored clock counts to the CALOVCNT value during the monitor period, the table<br>Indicates that the monitored clock is working normally; if it is less than the set value, it indicates that the monitored clock stops, and the monitored clock will be set.<br>Fail bit of the monitor clock. | 0x0 | R/W |

# 27 Real Time Clock (RTC)

## 27.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter that provides seconds, minutes, hours (12/24 hour format), week, day, month and year information.

The RTC module has an automatic wake-up function to manage all low-power modes.

Two 32-bit registers store seconds, minutes, hours (12/24 hour clock), week, day, month, and year in BCD format.

The RTC has an automatic month day compensation function, and the number of days in a month and the number of days in a leap year can be automatically adjusted.

Two 32-bit registers are used to store programmable alarm information including seconds, minutes, hours, weeks, days, months, and years.

For any errors caused by the frequency deviation, temperature drift and other reasons of the crystal itself, the digital calibration function of the RTC itself can be used Make corrections.

After a power-on reset, all RTC registers are disabled to prevent accidental writes.

When the device is in run mode, low power mode, or reset state (except power-on reset (POR reset)), as long as the voltage is within the operating range, RTC will keep running normally.

## 27.2 Main features

The main features of the RTC module are as follows:

ÿCalendar function, can display seconds, minutes, hours (12/24 hour format), week, day, month and year.

ÿAutomatic leap year adjustment is possible.

ÿWith alarm interrupt and cycle interrupt function.

ÿDigital calibration circuit (regular correction of the counter): from a calibration window of a few seconds.

ÿClock source optional: external low-speed clock (LXT), internal low-speed clock (LIRC), external high-speed clock (HXT)

ÿ 1Hz square wave output

## 27.3 RTC function description

### 27.3.1 RTC block diagram

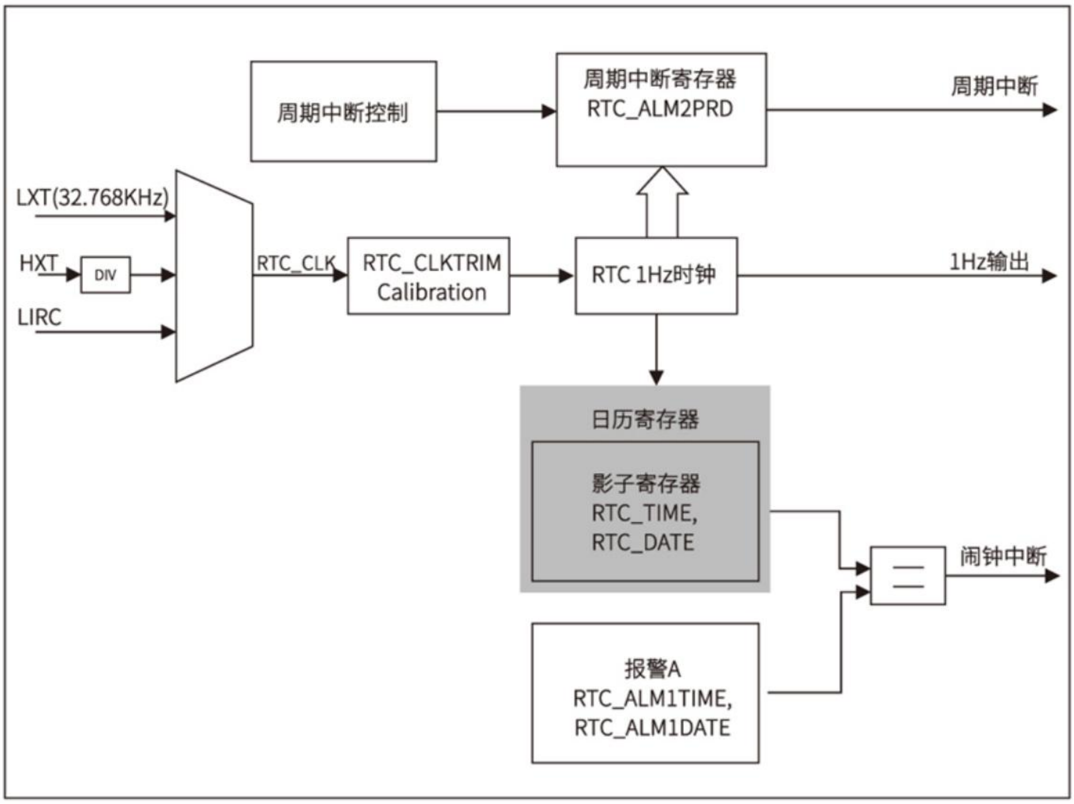


Figure 27-1 RTC block diagram

RTC modules include:

ÿ An alarm clock alarm interruption

ÿ one cycle interrupt

ÿCalibrated 1Hz clock output

ÿ Perpetual Calendar Register

### 27.3.2 RTC clock

The RTC clock source (RTC_CLK) is selected by the clock controller from the following 3 clocks:

ÿ LXT clock as RTC clock;

ÿ LIRC clock as RTC clock;

ÿ HXT clock is used as RTC clock.

For more configuration information about the RTC clock source, please refer to Chapter 6 System Reset and Clock (RCC)

### 27.3.3 Reset Procedure

Any available system reset source will cause the calendar shadow register and the RTC initialization and status register (RTC_ISR) to be reset to the default value.

However, the reset of the following registers has nothing to do with system reset, but only with power-on reset (POR reset): RTC Control Register (RTC_CR), RTC Clock Control Register (RTC_CLKCR), RTC Calibration Register (RTC_CLKTRIM), ALM Register (RTC_ALM1DATE/RTC_ALM1TIME/RTC_ALM2PRD), RTC current calendar register (RTC_TIME/RTC_DATA).

The RTC will remain running on any system reset except power-on reset. After a power-on reset occurs, the RTC stops running, so the There are RTC registers reset to default values.

### 27.3.4 Register Write Protection

After power-on reset, all RTC registers will be write-protected. Write access to the RTC register is enabled by writing the specified keyword to the write protection register RTC_WPR.

Release the write protection of all RTC registers by the following operations.

1. Write '0xCA' to the RTC_WPR register; 2. Write '0x53' to the RTC_WPR register.

Note: After the protection is released, any further write to this register will reactivate the write protection.

### 27.3.5 Calendar initialization and configuration

Initialization of the time and date values, including configuration of the time format and prescaler, is done in the following sequence:

1. Select the RTC timing clock source through RTC_CLKCR.CKSEL. If you choose the HXT clock, you must first set the prescaler.
2. Set RTC_CLKCR.RTCCKEN to enable RTC timing clock.
3. Set the WAIT bit of the RTC_ISR register to "1" to enter initialization mode. 4. Wait for the WAITF bit of the RTC_ISR register to be "1" to ensure that the initialization mode has been formally entered. Due to clock synchronization delays,

   This process takes approximately 2 RTC_CLK clock cycles. In this mode, the calendar counter is suspended, and the time can be updated at this time and the value of the date counter.
5. Set the time format (12-hour/24-hour) by the FMT bit of the RTC_CR register. 6. Load the initial time and date values into the time registers (RTC_TIME and RTC_DATE).
7. When clock error compensation is required, set the clock compensation register RTC_CLKTRIM.
8. Clear the value of the RTC_ISR.WAIT bit to exit initialization mode. The actual value of the calendar counter will be loaded automatically and will be displayed in 4

   Restart after RTCCLK clock cycle.

After the above series of initialization operations are completed, the calendar will start timing.

## 27.3.6 Read Count Register

When the BYPSHAD control bit of the RTC_CR register is cleared:

To ensure normal reading of the RTC calendar registers (RTC_TIME and RTC_DATE) under a secure synchronization mechanism, the APB clock frequency (fPCLK) should be at least 7 times the RTC clock frequency (fRTCCLK) . When the APB clock frequency is lower than 7 times the RTC clock frequency, the software must read the calendar time and date registers twice. If the value read for the second time is the same as the value read for the first time, it means that the return value is correct, otherwise it needs to be read again. In any case, the APB clock frequency must be greater than the RTC clock frequency.

The RSF bit of the RTC_ISR register is set when the contents of the calendar register are copied into the RTC_TIME and RTC_DATE shadow registers. The copy operation is performed every two RTC_CLK cycles. To ensure that the values of the two are consistent, read RTC_TIME At that time, the hardware will lock the value of the RTC_DATE shadow register until the value of RTC_DATE is read.

In order to prevent the software from accessing the calendar multiple times when the time interval is less than 2 RTC_CLK cycles, the RSF bit should be cleared by software every time the calendar is read, and the software must wait for the RSF bit to be set before reading the RTC_TIME and RTC_DATE registers.

After waking up from low-power mode, the RSF bit should be cleared by software, software must wait for the RSF bit to be set again before reading RTC_TIME and RTC_DATE register. The RSF bit should be cleared after wake-up, not before entering low-power mode.

After a system reset, software must wait for the RSF bit to be set before reading the RTC_TIME and RTC_DATE registers. in fact A system reset will cause the shadow registers to reset to their default values.

When the BYPSHAD control bit of the RTC_CR register is set (no need to consider the shadow register):

Read the calendar register to get the value directly from the calendar counter without waiting for the RSF bit to be set. This feature works just after exiting low power mode This is useful because the shadow registers are not automatically updated in low-power modes.

When BYPSHAD is "1", if an RTC_CLK edge occurs between two register reads, results in different registers may be are not related to each other. Also, if an RTC_CLK edge is encountered during a read operation, the value of a register may be incorrect. software must All registers shall be read twice and the results of the two reads shall be compared; or by comparing the results of the two least significant calendar registers, To check whether the data is correct and has a certain relationship.

## 27.3.7 Write Count Register

1. Set RTC_ISR.WAIT=1, stop calendar register counting, and enter write mode; 2. Query until RTC_ISR.WAITF=1;

3. Write the count register value of seconds, minutes, hours, weeks, days, months and years;

4. Set RTC_ISR.WAIT=0, the counter restarts. Note that all write operations must be completed within 1 second; 5. Query until RTC_ISR.WAITF=0.

### 27.3.8 Alarm Clock Setting

Set the ALM1EN bit of the RTC_CR register to start the alarm clock function. When the second, minute, hour, week, day, month, year and report in the calendar If the value set in alarm register RTC_ALM1TIME and RTC_ALM1DATE matches, RTC_ISR.ALM1_F is set to 1 by hardware. Place Any calendar field can be selected as an alarm source by the ALMxEN bit in the RTC_ALM1DATE register. Set RTC_CR send The ALM1_INTEN bit of the register will generate an alarm interrupt.

### 27.3.9 Calibrating the 1Hz Output

The RTC can optionally output a calibrated 1Hz clock. Set the output enable by RTC_CR.RTC1HZOE, by RTC_CLKTRIM to set the calibration value.

### 27.3.10 RTC clock calibration

The RTC module compensates the frequency of the RTC clock by masking the specified number of RTC clock cycles every fixed time period, by RTC_CLKTRIM.MODE[1:0] to select the time interval for adjustment:

0b00: Calibrate every 60 seconds (SEC=00)

0b01: Calibrate every 30 seconds (SEC=00, 30)

0b10: Calibrate every 15 seconds (SEC=00, 15, 30, 45)

0b11: Calibrate every 6 seconds (SEC=00, 06, 12, 18, 24, 30, 36, 42, 48, 54)

The masked RTC clock cycle number is specified by RTC_CLKTRIM.TRIM[7:0].

Note that the value of RTC_CLKTRIM.TRIM[7:0] is a signed integer with a range of -128~+127.

## 27.4 RTC interrupt

The RTC supports two types of interrupts. Alarm clock interruption, regular period interruption. The alarm clock interrupt and the periodic interrupt share the same interrupt signal. The interrupt source is distinguished by the flag register bit.

### 27.4.1 RTC alarm interrupt

1. Set RTC_CR.ALM1EN=0 to disable the alarm clock function;

2. Set the time alarm clock register RTC_ALM1TIME and the date alarm clock register RTC_ALM1DATE;

3. Set RTC_CR.ALM1EN=1, the alarm clock is allowed;

4. Clear the interrupt flag RTC_ISR.ALM1_F;

5. Set RTC_CR.ALM1_INTEN=1, the alarm clock interrupt is allowed, and if the current calendar time is equal to the alarm clock register, the alarm clock will be triggered interruption;

6. Wait for an interrupt to occur;

### 27.4.2 RTC cycle interrupt

When RTC_CR.ALM2_INTEN=1 of the control register RTC_CR, after the selected period occurs, a regular period wake-up interrupt is triggered, which is determined by

Shared interrupts for alarm clock and regular period are distinguished by flag register bits.

1. Set RTC_CR.ALM2_INTEN=0 to disable the periodic interrupt function;

2. Set the periodic alarm clock register RTC_ALM2PRD;

3. Clear the interrupt flag RTC_ISR.ALM2_F;

4. Set RTC_CR.ALM2_INTEN=1, periodic interrupt permission, after the selected cycle occurs, trigger a periodic wake-up interrupt;

5. Wait for an interrupt to occur;

## 27.5 RTC register list

Base address: 0x4000 3000

| offset address | name | describe | Defaults |
|---|---|---|---|
| 0x00 | RTC_CR | RTC Control Register | 0x0000000 |
| 0x04 | RTC_CLKCR | RTC Clock Control Register | 0x0000000 |
| 0x08 | RTC_TIME | RTC time register | 0x0000000 |
| 0x0C | RTC_DATE | RTC date register | 0x0000000 |
| 0x10 | RTC_ALM1TIME | RTC Time Alarm Register | 0x0000000 |
| 0x14 | RTC_ALM1DATE | RTC Date Alarm Register | 0x00000000 |
| 0x18 | RTC_ALM2PRD | RTC Periodic Alarm Register | 0x0000000 |
| 0x1C | RTC_CLKTRIM | RTC clock adjustment register | 0x0000000 |
| 0x20 | RTC_ISR | Initialization and Status Registers | 0x0000000 |
| 0x24 | RTC_INTCLR | RTC Status Clear Register | 0x0000000 |
| 0x28 | RTC_WPR | RTC write protection register | 0x0000000 |

## 27.6 RTC register description

### 27.6.1 RTC Control Register (RTC_CR)

Offset address: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | START | reserve | ALM1EN | ALM2_INTEN | ALM1_INTEN | reserve | FMT | RTC1HZOE. | BYPSHAD |
| | | | | | | | R/W | | R/W | R/W | R/W | | R/W | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:9 | - | reserve | 0x0 | - |
| 8 | START | 0: Stop RTC counter<br>1: Enable RTC counter | 0 | R/W |
| 7 | - | reserve | 0 | - |
| 6 | ALM1EN | ALM1 Alarm function enable.<br>0: disable ALM1 alarm clock function<br>1: Enable ALM1 alarm clock function<br>Note: During calendar counting (RTC_CLKCR.RTCCKEN=1) and<br>Enabled when ALM1 alarm interrupt permission is enabled (ALM1_INTEN=1)<br>When ALM1EN is used, in order to prevent malfunction, please disable the system interrupt. After enabling, please set the<br>ALM1_F flag is cleared. | 0 | R/W |
| 5 | ALM2_INTEN | ALM2 Periodic interrupt enable.<br>0: Disable ALM2 cycle interrupt<br>1: Enable ALM2 period interrupt | 0 | R/W |
| 4 | ALM1_INTEN | ALM1 alarm interrupt enable.<br>0: Disable ALM1 alarm interrupt<br>1: Enable ALM1 alarm interrupt | 0 | R/W |
| 3 | - | reserve | 0 | - |
| 2 | FMT | Time format.<br>0: 12-hour format (AM/PM time format)<br>1: 24-hour format | 0 | R/W |
| 1 | RTC1HZOE | RTC 1Hz output can<br>0: disabled<br>1: enable | 0 | R/W |
| 0 | BYPSHAD | Bypass shadow registers<br>0: read the calendar value from the shadow register, the shadow register every two RTC_CLK weeks<br>updated once<br>1: read the calendar value directly from the calendar counter<br>Note: If the APB clock frequency is less than 7 times the RTCCLK frequency, BYPSHAD must<br>Must be set to "1". | 0 | R/W |

### 27.6.2 RTC Clock Control Register (RTC_CLKCR)

Offset address: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | RTCCKEN | reserve | | RTCCKSEL[1:0] | |
| | | | | | | | | | | | R/W | | | R/W | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | HXTDIV[9:0] | | | | | | | | | |
| | | | | | | R/W | | | | | | | | | |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:21 | - | reserve | 0x0 | - |
| 20 | RTCCKEN | RTC count clock enable:<br>Set or cleared by software<br>Write:<br>0: RTC clock off<br>1: RTC clock is on<br>read:<br>0: RTC clock off<br>1: RTC clock is on | 0 | R/W |
| 19:18 | - | reserve | 0x0 | - |
| 17:16 | RTCCKSEL[1:0] | RTC clock source selection<br>The RTC clock source is selected by software setting. Once the RTC clock source is selected, these bit values do not Can be changed unless RTC is reset. Can be reset by setting the RCC_RTCRST.RTCRST bit bit RTC field.<br>00: LXT oscillator as RTC clock<br>01: LIRC oscillator as RTC clock<br>10: FHXT/(HXTDIV[9:0])<br>11: reserved | 0x0 | R/W |
| 15:10 | - | reserve | 0x0 | - |
| 9:0 | HXTDIV[9:0] | External high-speed crystal oscillator clock frequency division<br>0: stop<br>Other values: F=FHXT/(HXTDIV[9:0]) | 0x0 | R/W |

### 27.6.3 RTC time register (RTC_TIME)

Offset address: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | WEEK[2:0] | | | reserve | | H20_ PA | HOUR19[4:0] | | | | |
| | | | | | R/W | | | | | R/W | R/W | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | MIN[6:0] | | | | | | | reserve | SEC[6:0] | | | | | | |
| | R/W | | | | | | | | R/W | | | | | | |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:27 - | | reserve | 0x0 | - |
| 26:24 | WEEK[2:0] | Week counter. The value of the week counter is binary counting, and the counting range is from 0 to 6 (7 is not used used, unless the week count is not used). The correspondence between the week and the week counter value is defined by the user. (Sunday=0, Monday=1...Saturday=6) | 0x0 | R/W |
| 23:22 - | | reserve | 0x0 | - |
| twenty one | H20_PA | These two bits represent the hour counter. The value of HOUR19 is BCD coded. The time format is from determined by the clock system. In 12-hour format, H20_PA means morning or afternoon. 24-hour clock mode, H20_PA decision Determines whether the tens digit of the counter is 2. 12-hour format, when [H20_PA,HOUR19] counts from [1,11](11PM) to [0, 12](12AM) , the date counter increments by one day. 24-hour mode, when [H20_PA,HOUR19] When counting from [1,3](23H) to [0,0](0H), the date counter increases by one day. | 0 | R/W |
| 20:16 | HOUR19[4:0] | | 0x0 | R/W |
| 15 | - | reserve | 0x0 | - |
| 14:8 | MIN[6:0] | minute counter. The value of the minute counter is BCD coded, and the counting range is from 0 to 59. when the minute When the counter counts from 59 to 0, the hour counter is incremented by 1. When this counter is written When entered, time less than one second will be ignored. | 0x0 | R/W |
| 7 | - | reserve | 0 | - |
| 6:0 | SEC[6:0] | seconds counter. The value of the second counter is BCD coded, and the counting range is from 0 to 59. when seconds count When the counter counts from 59 to 0, the minute counter increments by 1. When this counter is written, less than A time of one second will be ignored. | 0x0 | R/W |

## 27.6.4 RTC date register (RTC_DATE)

Offset address: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | YEAR[7:0] | | | | | | | |
| | | | | | | | | R/W | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CEN | reserve | | MONTH[4:0] | | | | | reserve | | DAY[5:0] | | | | | |
| R/W | | | R/W | | | | | | | R/W | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:24 | - | reserve | 0x0 | - |
| 23:16 | YEAR[7:0] | Year counter. The year counter represents the tens and ones of the decimal year. Year counter in BCD<br>Code, the counting interval is from 00 to 99. The century counter increments when the year counter counts from 99 to 00<br>1.<br>When the century counter is 0, 04, 08, ... 92, 96 are leap years.<br>When the century counter is 1, 00, 04, 08, ... 92, 96 are leap years. | 0x0 | R/W |
| 15 | CEN | Century counter. 0 represents the 20th century, 1 represents the 21st century | 0 | R/W |
| 14:13 | - | reserve | 0x0 | - |
| 12:8 | MONTH[4:0] | Month counter. The month counter is BCD coded, and the counting range is from 01 to 12. Current year counter from<br>12 When counting to 01, the year counter is incremented by 1. | 0x0 | R/W |
| 7:6 | - | reserve | 0x0 | - |
| 5:0 | DAY[5:0] | day counter. The day counter is BCD coded, and the counting range is as follows:<br>01 to 31: January, March, May, July, August, October, December;<br>01 to 30: April, June, September, November;<br>01 to 29: February of leap year<br>01 to 28: February in a non-leap year<br>When the year counter counts from 99 to 00, the century counter increments by 1. | 0x0 | R/W |

### 27.6.5 RTC time alarm clock register (RTC_ALM1TIME)

Offset address: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | ALWEEK[2:0] | | | reserve | | ALH2 0_PA | ALHOUR19[4:0] | | | | |
| | | | | | R/W | | | | | R/W | R/W | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | ALMIN[6:0] | | | | | | | reserve | ALSEC[6:0] | | | | | | |
| | R/W | | | | | | | | R/W | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:27 | - | reserve | 0x0 | - |
| 26:24 | ALWEEK[2:0] | Alarm week setting | 0x0 | R/W |
| 23:22 | - | reserve | 0x0 | - |
| twenty one | ALH20_PA | Alarm hour setting. Refer to Hour Count Register. | 0x0 | R/W |
| 20:16 | ALHOUR19[4:0] | | | |
| 15 | - | reserve | 0 | - |
| 14:8 | ALMIN[6:0] | Alarm minute setting | 0x0 | R/W |
| 7 | - | reserve | 0 | - |
| 6:0 | ALSEC[6:0] | Alarm seconds setting | 0x0 | R/W |

### 27.6.6 RTC date alarm clock register (RTC_ALM1DATE)

Offset address: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | ALMYEAREN | ALMMONEN | ALMDAYEN | ALMWEEKEN | ALMHOUREN | ALMMINEN | ALMSECEN | ALYEAR[7:0] | | | | | | | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALCEN | reserve | | ALMONTH[4:0] | | | | | reserve | | ALDAY[5:0] | | | | | |
| R/W | | | R/W | | | | | | | R/W | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31 | - | reserve | 0 | - |
| 30 | ALM YEAREN | Alarm year setting enabled | 0 | R/W |
| 29 | ALMMONEN | Alarm month setting enable | 0 | R/W |
| 28 | ALM DAYEN | Alarm day setting enabled | 0 | R/W |
| 27 | ALMWEEKEN | Alarm week setting enable | 0 | R/W |
| 26 | ALM HOUREN | Alarm hour setting enable | 0 | R/W |
| 25 | ALMMINEN | Alarm minute setting enable | 0 | R/W |
| twenty four | ALMSECEN | Alarm clock second setting enable | 0 | R/W |
| 23:16 | ALYEAR[7:0] | Alarm year setting | 0x0 | R/W |
| 15 | ALCEN | Alarm clock century setting | 0 | R/W |
| 14:13 | - | reserve | 0x0 | - |
| 12:8 | ALMONTH[4:0] | Alarm month setting | 0x0 | R/W |
| 7:6 | - | reserve | 0x0 | - |
| 5:0 | ALDAY[5:0] | Alarm day setting | 0x0 | R/W |

Machine Translated by Google

### 27.6.7 RTC Periodic Alarm Register (RTC_ALM2PRD)

Offset address: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | ALM2PR_CNT[3:0] | | | | |
| | | | | | | | | | | | R/W | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:4 | - | reserve | 0x0 | - |
| 3:0 | ALM2PR_CNT | Cycle alarm clock 2 count cycle setting.<br>0x0: turn off cycle alarm clock 2<br>0x1:1 second<br>0x2: 1/2 second<br>0x3: 1/4 second<br>0x4: 1/8 second<br>0x5: 1/16 second<br>0x6: 1/32 second<br>0x7: 1/64 second<br>0x8: 1/128 second<br>0x9:10 seconds<br>0xA: 30 seconds<br>0xB: 1 minute<br>0xC: 30 minutes<br>0xD: 60 minutes<br>0xE: 12 hours<br>0xF: 24 hours | 0x0 | R/W |

### 27.6.8 RTC Clock Trim Register (RTC_CLKTRIM)

Offset address: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | TRIM_MODE[1:0] | | TRIM[7:0] | | | | | | | |
| | | | | | | R/W | | R/W | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:10 - | | reserve | 0x0 | - |
| 9:8 | MODE[1:0] | Clock Adjustment Register. Determines the frequency of clock adjustments.<br>0x0: every 60 seconds (SEC=00)<br>0x1: every 30 seconds (SEC=00, 30)<br>0x2: every 15 seconds (SEC=00, 15, 30, 45)<br>0x3: every 6 seconds (SEC=00, 06, 12, 18, 24, 30, 36, 42, 48, 54) | 0x0 | R/W |
| 7:0 | TRIM[7:0] | Clock compensation time register. This register is a signed integer. (-128~+127) | 0x0 | R/W |

### 27.6.9 RTC initialization and status register (RTC_ISR)

Offset address: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | ALM2 _F | ALM1 _F | reserve | RSF | wait f | wait |
| | | | | | | | | | | RO | RO | | R/W | R/W | RW |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:6 - | | reserve | 0x0 | - |
| 5 | ALM2_F | Periodic Alarm 2 Interrupt Raw Status Register.<br>When this register is read, the status value is returned:<br>0: Periodic alarm 2 interrupt is not activated.<br>1: Periodic Alarm 2 interrupt is activated. | 0 | RO |
| 4 | ALM1_F | Alarm interrupt raw status register.<br>When this register is read, the status value is returned:<br>0: Alarm interrupt is not activated.<br>1: Alarm interrupt is activated. | 0 | RO |
| 3 | reserve | | 0 | - |
| 2 | RSF | Register Synchronization Flag<br>Whenever the content of the calendar register is copied to the shadow register (RTC_TIME, RTC_DATE) , this bit is set by hardware. When in ignore shadow register mode (BYPSHAD=1), the Bit is cleared by hardware in initialization mode. This bit can also be cleared by software.<br>In initialization mode, this bit can be cleared by hardware/software.<br>0: The calendar shadow register has not been synchronized;<br>1: The calendar shadow register has been synchronized.<br>Note that the software cannot write 1 | 0 | R/W |
| 1 | WAITF | 0: non-write/configuration state<br>1: write/configuration status<br>Note: WAITF is the valid flag of WAIT bit setting. Please confirm this bit before writing/configuring Is it "1". During the counting process, wait for the writing to be completed after the WAIT bit is cleared to "0". Clear "0". | 0 | R/W |
| 0 | wait | 0: normal counting mode<br>1: Write/configuration mode<br>Note: Please set this bit to "1" when writing/configuring. Since the counter is counting continuously, please Complete the write/configuration operation and clear this bit to "0" within the specified time. | 0 | R/W |

### 27.6.10 RTC Status Clear Register (RTC_INTCLR)

Offset address: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | ALM2<br>_CLR | ALM1<br>_CLR | reserve | | | |
| | | | | | | | | | | WO | WO | | | | |

| bit flag | | Functional description | Reset value read | and write |
|---|---|---|---|---|
| 31:6 - | | reserve | 0x0 | - |
| 5 | ALM2_CLR | Periodic Alarm 2 Interrupt Raw Status Clear Register.<br>When this register is written, the interrupt raw status is required to be cleared:<br>0: no operation.<br>1: Periodic alarm 2 interrupt raw status is cleared. | 0 | WO |
| 4 | ALM1_CLR | Alarm interrupt raw status clear register.<br>When this register is written, the interrupt raw status is required to be cleared:<br>0: no operation.<br>1: Alarm interrupt raw state is cleared. | 0 | WO |
| 3:0 | - | reserve | 0x0 | - |

### 27.6.11 RTC Write Protection Register (RTC_WPR)

Offset address: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | WPR[7:0] | | | | | | | |
| | | | | | | | | WO | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:8 | - | reserve | 0x0 | - |
| 7:0 | WPR | Write the specified keyword to enable the write permission of the RTC register.<br>Write '0xCA' to the RTC_WPR register;<br>Write '0x53' to the RTC_WPR register.<br>Note: After the protection is released, any write to this register will reactivate the write protection protect. | 0x0 | WO |

# 28 Analog/Digital Converter (ADC)

## 28.1 Module Introduction

The chip integrates a 12-bit successive approximation (SAR) analog-to-digital converter (ADC) module with high precision and high conversion rate. have

The following properties:

ÿ 12-bit conversion precision

ÿ 1Msps conversion speed

ÿ 8 conversion channels: 7 pin channels, 1 VCAP calibration channel

ÿReference Voltage is the power supply voltage

ÿ Voltage input range of ADC: 0-VREF

ÿ 3 conversion modes: single conversion, continuous conversion, cumulative conversion

ÿ ADC conversion rate software configurable

ÿ Support on-chip and peripheral interrupts to automatically trigger ADC conversion start, effectively reducing chip power consumption and improving real-time conversion

## 28.2 ADC block diagram

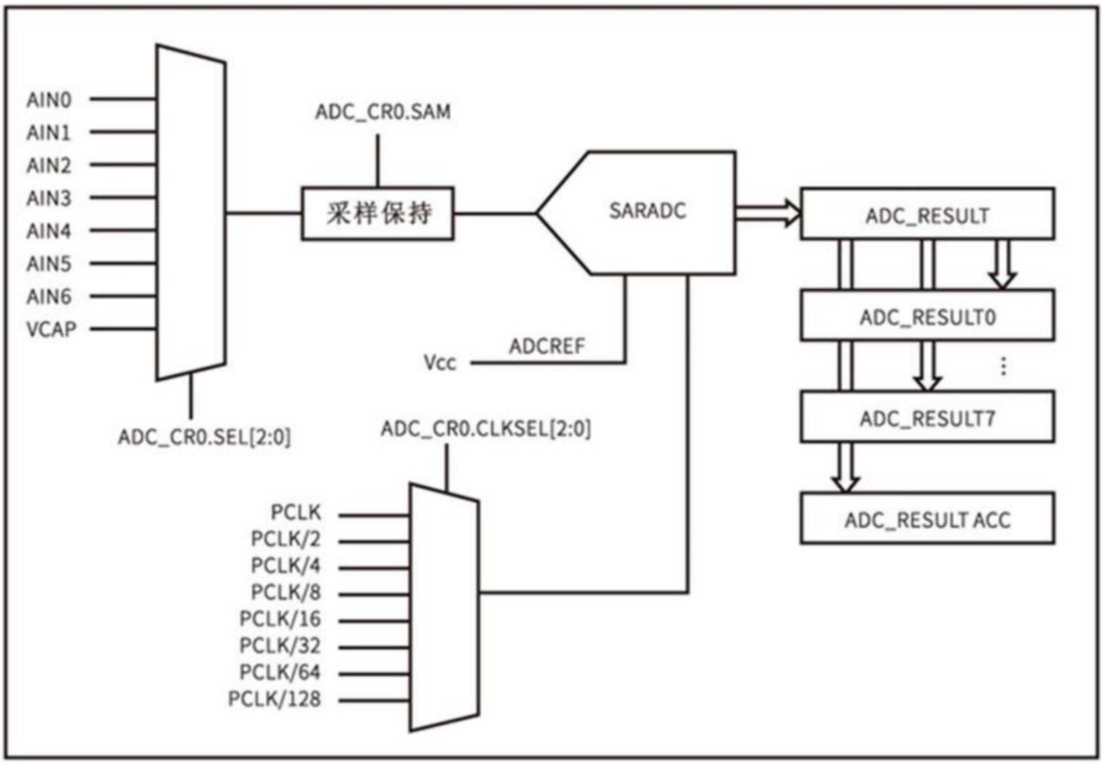


Figure 28-1 Schematic block diagram of ADC

## 28.3 Conversion timing and speed

The ADC conversion timing is shown in the figure below: a complete ADC conversion consists of a sampling process and a successive comparison process. The sampling process 4~8 ADC clocks are required, configured by ADC_CR0.SAM; 12 ADC clocks are required for successive comparisons. so once ADC conversion requires a total of 16~20 ADC clocks.

The unit of ADC conversion speed is sps (samples per second), that is, how many ADC conversions are performed per second. The calculation method of the ADC conversion speed is: the frequency of the ADC clock / the number of ADC clocks required for one ADC conversion.

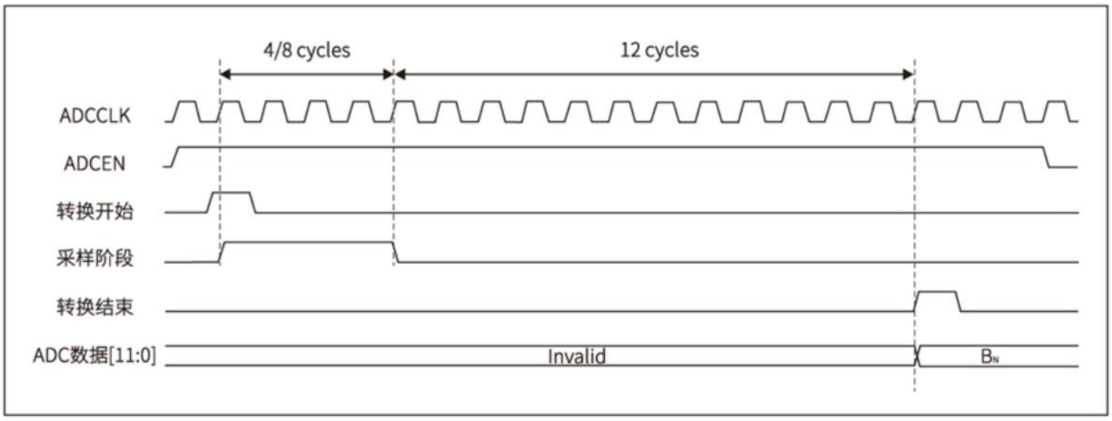


Figure 28-2 ADC conversion timing diagram

## 28.4 Single Conversion Mode

In the single-conversion mode, only one conversion is performed after the ADC is started, and all 8 ADC channels can be converted. This mode can be started by setting the ADC_CR0.START bit or by setting the external trigger of ADC_CR1[9:0]. Once the ADC conversion of the selected channel is completed, the ADC_CR0.START bit is automatically cleared and the conversion result is stored in the ADC_RESULT register.

Start the ADC single conversion operation flow through the START bit:

1. According to the GPIO corresponding to the pin configuration, configure the ADC channel to be converted as an analog port.

2. Set ADC_CR2.CIRCLE_MODE to 0 to select non-circular mode.

3. Set ADC_CR1.CT to 0 to select single conversion mode.

4. Configure ADC_CR0.SAM and ADC_CR0.CLKSEL to set the conversion speed of ADC.

5. Configure ADC_CR0.SEL to select the channel to be converted (note that it must be consistent with step 1).

6. Set ADC_CR0.ADCEN to 1 to enable the ADC module.

7. Set ADC_CR0.START to 1 to start ADC single conversion.

8. Wait for ADC_CR0.START to be 0, read the ADC_RESULT register to get the ADC conversion result.

9. To convert other channels, repeat steps 4~7.

10. Configure ADC_CR0.ADCEN to turn off the ADC module.

Note: The process configuration of ADC conversion triggered by internal signal is similar, and additional trigger selection is required.

## 28.5 Continuous Conversion Mode

In the continuous conversion mode, starting an ADC can perform multiple conversions on multiple channels in sequence; the ADC channels that can be converted are AIN0~AIN7. The total number of ADC conversions is configured by ADC_CR2.ADCCNT[7:0]; the channel to be converted is configured by ADC_CR2.CHEN[7:0] for configuration. This mode can be started either by setting the ADC_CR0.START bit or by setting External trigger start of ADC_CR1[9:0]. After starting the continuous conversion, the ADC module converts the channels to be converted in AIN0~AIN7 in turn until the total number of conversions is complete. The ADC_RAWINTSR.CONT_INTF bit is automatically set to 1 after the ADC module completes the total number of conversions, The conversion result is stored in the ADC_RESULT0~ADC_RESULT7 register corresponding to the conversion channel. If the total conversions are large Depending on the number of ADC channels to be converted, only the last conversion result is saved in the ADC_RESULT0~ADC_RESULT7 registers fruit.

The figure below demonstrates 10 consecutive conversions of AIN0, AIN1, AIN5. After setting START to 1 through the register, The state machine inside the ADC converts AIN0, AIN1, and AIN5 in turn until the count value of ADCCNT becomes 0.

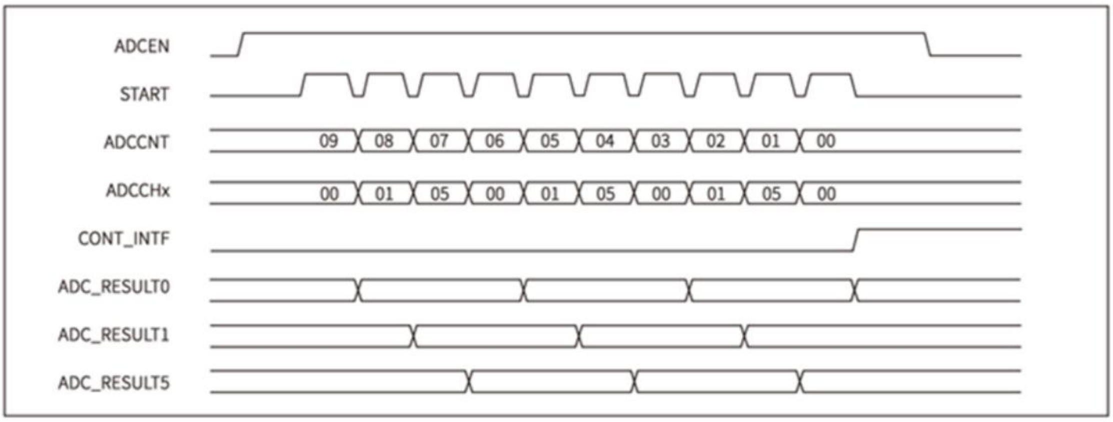


Figure 28-3 Example of ADC continuous conversion process

Configuration steps:

1. According to the GPIO corresponding to the pin configuration, configure the ADC channel to be converted as an analog port. 2. Set ADC_CR2.circle_mode to 0 to select non-circular mode. 3. Set ADC_CR1.ct to 1 to select continuous conversion mode. 4. Configure ADC_CR2.adccnt[7:0] to select the total number of conversions for continuous conversions. 5. Configure ADC_CR0.sam and ADC_CR0.clksel to set the conversion speed of ADC. 6. Configure ADC_CR2.chen[7:0] to enable the channel to be converted. 7. Set ADC_INTCLR.cont_intc to 1 and clear ADC_RAWINTSR.cont_intf flag. 8. Set ADC_INTEN.CONT_IEN to 1 to enable continuous conversion complete interrupt mask.

9. Set ADC_CR0.staterst to 1 to reset the continuous conversion state.

10. Set ADC_CR0.adcen to 1 to enable the ADC module.

11. Set ADC_CR0.start to 1 to start ADC continuous conversion.

12. Wait for ADC_RAWINTSR.cont_intf to become 1, read the ADC_result0~ADC_result7 registers to get the

Convert the result.

13. To convert other channels, repeat steps 6~11.

14. Configure ADC_CR0.adcen and turn off the ADC module.

## 28.6 Continuous Conversion Accumulation Mode

In the continuous conversion and accumulation mode, starting an ADC can perform multiple conversions on multiple channels and accumulate the results of each conversion;

The converted ADC channels are AIN0~AIN7. The total number of ADC conversions is configured by ADC_CR2.ADCCNT[7:0]; the channel to be converted is configured by ADC_CR2.CHEN[7:0].

This mode can be started by setting the ADC_CR0.START bit, or by setting the external trigger of ADC_CR1[9:0]. After starting the continuous conversion, the ADC module converts AIN0~AIN7

sequentially

Converted channels until the total number of conversions completed. After the ADC module completes the total number of conversions,

The ADC_RAWINTSR.CONT_INTF bit is automatically set to 1, and the accumulated value of the conversion result is stored in the ADC_RESULT_ACC register

middle.

The figure below demonstrates the process of accumulating 10 consecutive conversions of AIN0, AIN1, and AIN5. After setting START to 1 through the register, the ADC

The internal state machine converts AIN0, AIN1, and AIN5 in turn until the count value of ADCCNT becomes 0. The ADC_RESULT_ACC register is automatically incremented each time a conversion

is complete. The conversion results of AIN0, AIN1, and AIN5 given in the figure depend on

The times are 0x010, 0x020, 0x040.

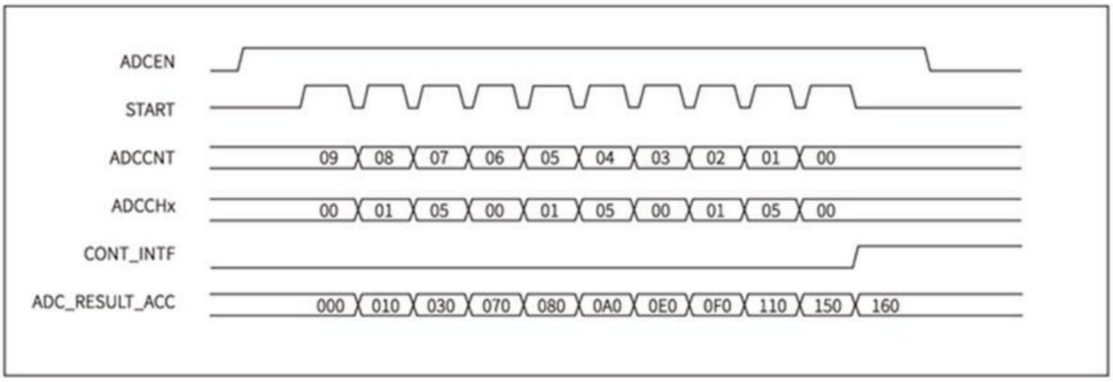


Figure 28-4 Example of ADC continuous conversion accumulation process

Configuration steps:

1. According to the GPIO corresponding to the pin configuration, configure the ADC channel to be converted as an analog port. 2. Set

ADC_CR2.circle_mode to 0 to select non-circular mode. 3. Set ADC_CR1.ct to 1 to select continuous conversion

mode. 4. Set ADC_CR1.racc_en to 1 to select the automatic accumulation function of ADC

conversion. 5. Configure ADC_CR2.adccnt[7:0] to select the total number of conversions for continuous conversions. 6. Configure

ADC_CR0.sam and ADC_CR0.clksel to set the conversion speed of ADC. 7. Configure ADC_CR2.chen[7:0] to select the

channel to be converted. 8. Set ADC_INTCLR.cont_intc to 1 and clear ADC_RAWINTSR.cont_intf flag. 9. Set ADC_INTEN.CONT_IEN to

1 to enable continuous conversion complete interrupt mask. 10. Set ADC_CR1.racc_clr to 1 to clear the

ADC_result_acc register. 11. Set ADC_CR0.staterst to 1 to reset the continuous conversion state. 12. Set ADC_CR0.adcen to 1 to enable the ADC module.

13. Set ADC_CR0.start to 1 to start ADC continuous conversion.

14. Wait for ADC_RAWINTSR.cont_intf to become 1, read the ADC_result_ACC register to get the continuous conversion accumulation result.

15. To convert other channels, repeat steps 6~11.

16. Configure ADC_CR0.adcen and turn off the ADC module.

## 28.7 ADC conversion result comparison

When the ADC conversion is completed, the ADC conversion result can be compared with the threshold set by the user, which supports upper threshold comparison, lower threshold comparison, zone

Comparison between values. This function needs to set the corresponding control bits ADC_CR1.HTCMP, ADC_CR1.LTCMP,

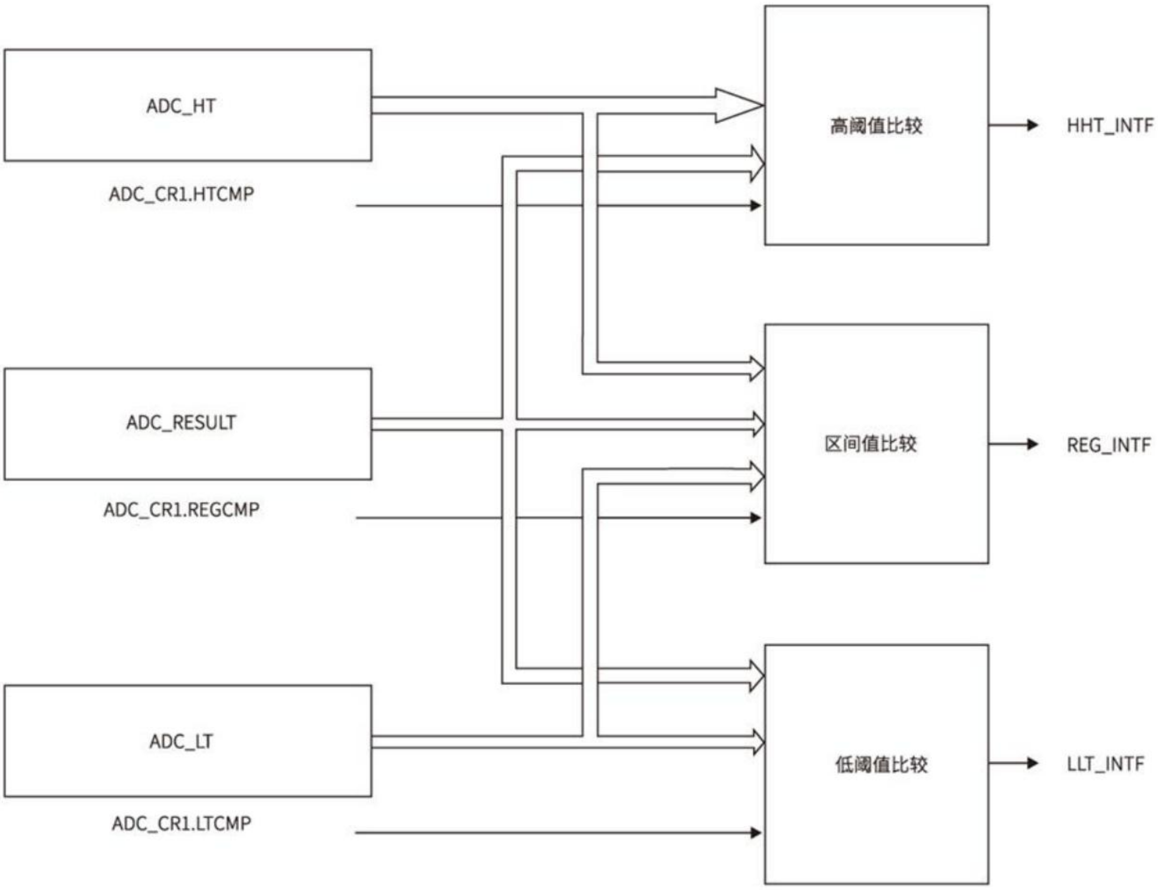
ADC_CR1.REGCMP is set. This function can realize the automatic monitoring of the analog quantity until the ADC conversion result meets the user's expectations.

Generate an interrupt to request user program interface entry.

ÿ Upper threshold comparison: When the ADC conversion result is within the range [ADC_HT, 4095], then Set ADC_RAWINTSR.HHT_INTF to 1; write 1 to ADC_INTCLR.HHT_INTC to clear ADC_RAWINTSR.HHT_INTF.

ÿLower threshold comparison: when the ADC conversion result is in the range [0, ADC_LT), then ADC_RAWINTSR.LLT_INTF Set; writing 1 to ADC_INTCLR.LLT_INTC clears ADC_RAWINTSR.LLT_INTF.

ÿInterval value comparison: when the ADC conversion result is within the interval [ADC_LT, ADC_HT), then Set ADC_RAWINTSR.REG_INTF to 1; write 1 to ADC_INTCLR.REG_INTC to clear ADC_RAWINTSR.REG_INTF.



## 28.8 ADC Interrupt

The ADC interrupt requests are shown in the table below:

| Interrupt | Interrupt flag | interrupt enable mask |
|---|---|---|
| source ADC continuous | ADC_MSKINTSR.CONT_MIF | ADC_INTEN.CONT_IEN |
| conversion completed ADC conversion result is in interval value area | | ADC_INTEN.REG_IEN |
| ADC_MSKINTSR.REG_MIF ADC conversion result is in upper threshold area | | ADC_INTEN.HHT_IEN |
| ADC_MSKINTSR.HHT_MIF ADC conversion result is compared in lower threshold area ADC_MSKINTSR.LLT_MIF | | ADC_INTEN.LLT_IEN |

## 28.9 Register List

Base address: 0x4000 2C00

| offset address name | | describe | reset value |
|---|---|---|---|
| 0x00 | ADC_CR0 | ADC Configuration Register 0 | 0x0000 0000 |
| 0x04 | ADC_CR1 | ADC configuration register 1 | 0x0000 7000 |
| 0x08 | ADC_CR2 | ADC configuration register 2 | 0x0000 0000 |
| 0x0C | ADC_RESULT0 | ADC channel 0 conversion result | 0x0000 0000 |
| 0x10 | ADC_RESULT1 | ADC channel 1 conversion result | 0x0000 0000 |
| 0x14 | ADC_RESULT2 | ADC channel 2 conversion result | 0x0000 0000 |
| 0x18 | ADC_RESULT3 | ADC channel 3 conversion result | 0x0000 0000 |
| 0x1C | ADC_RESULT4 | ADC channel 4 conversion result | 0x0000 0000 |
| 0x20 | ADC_RESULT5 | ADC channel 5 conversion result | 0x0000 0000 |
| 0x24 | ADC_RESULT6 | ADC channel 6 conversion result | 0x0000 0000 |
| 0x28 | ADC_RESULT7 | ADC channel 7 conversion result | 0x0000 0000 |
| 0x2C | ADC_RESULT | ADC conversion result | 0x0000 0000 |
| 0x30 | ADC_RESULT_ACC | Accumulated value of ADC conversion result | 0x0000 0000 |
| 0x34 | ADC_HT | ADC comparison upper threshold | 0x0000 0FFF |
| 0x38 | ADC_LT | ADC comparison lower threshold | 0x0000 0000 |
| 0x44 | ADC_INTEN | ADC Interrupt Enable Register | 0x0000 0000 |
| 0x48 | ADC_INTCLR | ADC Interrupt Clear Register | 0x0000 0000 |
| 0x4C | ADC_RAWINTSR | ADC Pre-Mask Interrupt Status Register | 0x0000 0000 |
| 0x50 | ADC_MSKINTSR | ADC Post Mask Interrupt Status Register | 0x0000 0000 |

## 28.10 Register Description

### 28.10.1 ADC Configuration Register 0 (ADC_CR0)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STATERST | reserve | | | SAM | SEL[2:0] | | | reserve | CLKSEL[2:0] | | | reserve | | START | ADCEN |
| R/W | | | | R/W | R/W | | | | R/W | | | | | R/W | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | - | Reserved, always reads as 0. | 0 | - |
| 15 | STATERST | ADC Continuous Conversion State Control<br>0: invalid<br>1: Reset ADC continuous conversion status | 0x0 | R/W |
| 14:12 | - | Reserved, always reads as 0. | 0 | - |
| 11 | SAM | ADC sampling period selection<br>0: 4 sampling periods<br>1: 8 sampling periods | 0x0 | R/W |
| 10:8 | SEL[2:0] | ADC conversion channel selection (single conversion mode) 000: select channel 0<br>001: select channel 1<br>010: Select channel 2<br>011: Select channel 3<br>100: select channel 4<br>101: Select channel 5<br>110: select channel 6<br>111: Select channel 7 (VCAP) | 0x0 | R/W |
| 7 | - | Reserved, always reads as 0. | 0 | - |
| 6:4 | CLKSEL[2:0] | ADC clock selection<br>000: PCLK clock<br>001: PCLK clock divided by 2<br>010: PCLK clock divided by 4<br>011: PCLK clock divided by 8<br>100: PCLK clock divided by 16<br>101: PCLK clock divided by 32<br>110: PCLK clock divided by 64<br>111: PCLK clock divided by 128 | 0x0 | R/W |
| 3:2 | - | Reserved, always reads as 0. | 0 | - |
| 1 | START | ADC conversion control<br>0: Stop ADC conversion<br>1: start ADC conversion<br>Note: This bit field is written to 1 by software and cleared to 0 by hardware. | 0x0 | R/W |
| 0 | ADCEN | ADC enable control<br>0: Disable ADC<br>1: enable ADC | 0x0 | R/W |

## 28.10.2 ADC Configuration Register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 7000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RAC<br>C_CL<br>R | REG<br>CMP | HTC<br>MP | LTCM<br>P | RAC<br>C_EN | CT | TRIGS1[4:0] | | | | | TRIGS0[4:0] | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | | | | | R/W | | | | |

| Bitmark 31:16 | | The function | Reset value read | and write |
|---|---|---|---|---|
| | - | description is reserved and always reads as 0. | 0 | - |
| 15 | RACC_CLR | ADC conversion result accumulation register is cleared<br>0: no effect;<br>1: Clear the ADC conversion result accumulation register (ADC_RESULT_ACC).<br>Note: This bit is read as 0, so special attention should be paid to the value of this bit when operating this register<br>In case of misoperation. | 0x0 | R/W |
| 14 | REGCMP | ADC Interval Comparison Control<br>0: Interval comparison is prohibited<br>1: Enable range comparison | 0x1 | R/W |
| 13 | HTCMP | ADC High Threshold Compare Control<br>0: Disable high threshold comparison<br>1: Enable High Threshold Compare | 0x1 | R/W |
| 12 | LTCMP | ADC Low Threshold Compare Control<br>0: Disable low threshold comparison<br>1: Enable Low Threshold Compare | 0x1 | R/W |
| 11 | RACC_EN | ADC conversion result automatic accumulation control<br>0: Disable the automatic accumulation function of ADC conversion results<br>1: Enable the automatic accumulation function of ADC conversion results | 0x0 | R/W |
| 10 | CT | ADC conversion mode selection<br>0: single conversion mode<br>1: Continuous conversion mode | 0x0 | R/W |
| 9:5 | TRIGS1[4:0] | ADC conversion automatic trigger selection 1:<br>00000: Disable automatic triggering of ADC conversions<br>00001: Timer10 interrupt, trigger ADC conversion automatically<br>00010: Timer11 interrupt, automatically trigger ADC conversion<br>00011: TIM1 interrupt, automatically trigger ADC conversion<br>00100: LPTIM interrupt, trigger ADC conversion automatically<br>00101: TIM1 TRGO, automatically trigger ADC conversion<br>00110: TIM2 TRGO, automatically trigger ADC conversion<br>00111: TIM2 interrupt, automatically trigger ADC conversion<br>01000: UART0 interrupt, trigger ADC conversion automatically<br>01001: UART1 interrupt, trigger ADC conversion automatically<br>01010: LPUART interrupt, trigger ADC conversion automatically<br>01011: VC0 interrupt, trigger ADC conversion automatically<br>01100: N.C.<br>01101: RTC interrupt, automatically trigger ADC conversion<br>01110: PCA interrupt, automatically trigger ADC conversion<br>01111: SPI interrupt, automatically trigger ADC conversion<br>10000: PA1 interrupt, automatically trigger ADC conversion<br>10001: PA2 interrupt, automatically trigger ADC conversion<br>10010: PA3 interrupt, automatically trigger ADC conversion<br>10011: PB4 interrupt, automatically trigger ADC conversion<br>10100: PB5 interrupt, automatically trigger ADC conversion | 0x0 | R/W |

| | | | | |
|---|---|---|---|---|
| | | 10101: PC3 interrupt, trigger ADC conversion automatically<br>10110: PC4 interrupt, automatically trigger ADC conversion<br>10111: PC5 interrupt, automatically trigger ADC conversion<br>11000: PC6 interrupt, automatically trigger ADC conversion<br>11001: PC7 interrupt, automatically trigger ADC conversion<br>11010: PD1 interrupt, automatically trigger ADC conversion<br>11011: PD2 interrupt, automatically trigger ADC conversion<br>11100: PD3 interrupt, automatically trigger ADC conversion<br>11101: PD4 interrupt, automatically trigger ADC conversion<br>11110: PD5 interrupt, automatically trigger ADC conversion<br>11111: PD6 interrupt, trigger ADC conversion automatically<br>Note:<br>The ADC is triggered using the rising edge of each interrupt flag bit. If repeated triggering is required, the interrupt flag needs to be cleared. If you don't need to enter the interrupt service routine, please don't enable the interrupt enable of NVIC. | | |
| 4:0 | TRIGS0[4:0] | ADC conversion automatic trigger<br>selection 1: 00000: Disable automatic triggering<br>of ADC conversion 00001: Timer10 interrupt, automatically trigger<br>ADC conversion 00010: Timer11 interrupt, automatically trigger<br>ADC conversion 00011: TIM1 interrupt, automatically trigger<br>ADC conversion 00100: LPTIM interrupt, automatically trigger<br>ADC conversion 00101: TIM1 TRGO, automatically trigger ADC<br>conversion 00110: TIM2 TRGO, automatically trigger ADC<br>conversion 00111: TIM2 interrupt, automatically trigger ADC<br>conversion 01000: UART0 interrupt, automatically trigger ADC<br>conversion 01001: UART1 interrupt, automatically trigger ADC<br>conversion 01010: LPUART interrupt, Automatically trigger ADC<br>conversion 01011: VC0 interrupt, automatically trigger ADC<br>conversion<br>01100: NC interrupt, automatically trigger ADC conversion<br>01110: PCA interrupt, automatically trigger ADC conversion<br>01111: SPI interrupt, automatically trigger ADC conversion<br>10000: PA1 interrupt, automatically trigger ADC conversion<br>10001: PA2 interrupt, automatically trigger ADC conversion<br>10010: PA3 interrupt, automatically trigger ADC conversion<br>10011: PB4 interrupt, automatically trigger ADC conversion<br>10100: PB5 interrupt, automatically trigger ADC conversion<br>10101: PC3 interrupt, automatically trigger ADC conversion<br>10110: PC4 Interrupt, automatically trigger ADC conversion<br>10111: PC5 interrupt, automatically trigger ADC conversion<br>11000: PC6 interrupt, automatically trigger ADC conversion<br>11001: PC7 interrupt, automatically trigger ADC conversion<br>11010: PD1 interrupt, automatically trigger ADC conversion<br>11011: PD2 interrupt, automatically trigger ADC Conversion<br>11100: PD3 interrupt, automatically triggers ADC conversion<br>11101: PD4 interrupt, automatically triggers ADC conversion<br>11110: PD5 interrupt, automatically triggers ADC conversion<br>11111: PD6 interrupt, automatically triggers ADC conversion<br>Note:<br>triggering ADC uses the rise of each interrupt flag bit along. If repeated triggering is required, the interrupt flag needs to be cleared. If you don't need to enter the interrupt service routine, please don't enable the interrupt enable of NVIC. | 0x0 | R/W |

### 28.10.3 ADC Configuration Register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | CIRCLE_MODEs |
| | | | | | | | | | | | | | | | R/W |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCCNT[7:0] | | | | | | | | CHEN[7:0] | | | | | | | |
| R/W | | | | | | | | R/W | | | | | | | |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:17 | - | Reserved, always reads as 0. | 0 | - |
| 16 | CIRCLE_MODE | ADC conversion cycle mode selection<br>0: Acyclic mode<br>1: cycle mode | 0x0 | R/W |
| 15:8 | ADCCNT[7:0] | ADC Continuous Conversion Count Configuration<br>0: Continuous conversion once<br>1: Continuously convert 2 times<br>...<br>255: 256 consecutive conversions | 0x0 | R/W |
| 7:0 | CHEN[7:0] | ADC continuous conversion channel 7~0 enable<br>0: disabled<br>1: enable | 0x0 | R/W |

### 28.10.4 ADC channel 0 conversion result (ADC_RESULT0)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT0[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT0[11:0] | ADC channel 0 conversion result | 0x0 | RO |

### 28.10.5 ADC Channel 1 Conversion Result (ADC_RESULT1)

Address offset: 0x10

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT1[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | reset value | read and write |
|---|---|---|---|---|
| 31:12 - | | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT1[11:0] | ADC channel 1 conversion result | 0x0 RO | |

### 28.10.6 ADC channel 2 conversion result (ADC_RESULT2)

Address offset: 0x14

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | RESULT2[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT2[11:0] | ADC channel 2 conversion result | 0x0 | RO |

### 28.10.7 ADC channel 3 conversion result (ADC_RESULT3)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | RESULT3[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT3[11:0] | ADC channel 3 conversion result | 0x0 | RO |

### 28.10.8 ADC channel 4 conversion result (ADC_RESULT4)

Address offset: 0x1C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT4[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT4[11:0] | ADC channel 4 conversion result | 0x0 | RO |

### 28.10.9 ADC channel 5 conversion result (ADC_RESULT5)

Address offset: 0x20

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT5[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT5[11:0] | ADC channel 5 conversion result | 0x0 | RO |

Machine Translated by Google

### 28.10.10 ADC channel 6 conversion result (ADC_RESULT6)

Address offset: 0x24

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT6[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 - | | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT6[11:0] | ADC channel 6 conversion result | 0x0 | RO |

### 28.10.11 ADC channel 7 conversion result (ADC_RESULT7)

Address offset: 0x28

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | RESULT7[11:0] | | | | | | | | | | | |
| | | | | RO | | | | | | | | | | | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT7[11:0] | ADC channel 7 conversion result | 0x0 | RO |

### 28.10.12 ADC conversion result (ADC_RESULT)

Address offset: 0x2C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

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

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | RESULT[11:0] | ADC conversion result | 0x0 | RO |

### 28.10.13 ADC Conversion Result Accumulated Value (ADC_RESULT_ACC)

Address offset: 0x30

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | RESULT_ACC[19:16] | | | |
| | | | | | | | | | | | | RO | | | |

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

| bit flag | | Functional description | Reset value read and write | |
|---|---|---|---|---|
| 31:20 | - | Reserved, always reads as 0. | 0 | - |
| 19:0 | RESULT_ACC[19:0] | ADC conversion result accumulation value | 0x0 | RO |

### 28.10.14 ADC Compare Upper Threshold (ADC_HT)

Address offset: 0x34

Reset value: 0x0000 0FFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | HT[11:0] | | | | | | | | | | | |
| | | | | R/W | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | HT[11:0] | ADC conversion result comparison upper threshold | 0xFFF | R/W |

### 28.10.15 ADC Compare Lower Threshold (ADC_LT)

Address offset: 0x38

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | LT[11:0] | | | | | | | | | | | |
| | | | | R/W | | | | | | | | | | | |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11:0 | LT[11:0] | ADC conversion result comparison lower threshold | 0x0 | R/W |

### 28.10.16 ADC Interrupt Enable Register (ADC_INTEN)

Address offset: 0x44

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | CONT_IEN | REG_IEN | HHT_IEN | LLT_IEN | ADCXIEN[7:0] | | | | | | | |
| | | | | R/W | R/W | R/W | R/W | R/W | | | | | | | |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11 | CONT_IEN | Continuous Conversion Complete Interrupt Mask Configuration<br>0: disable interrupt<br>1: enable interrupt | 0x0 | R/W |
| 10 | REG_IEN | ADC conversion result comparison interval interrupt mask configuration<br>0: disabled<br>1: enable | 0x0 | R/W |
| 9 | HHT_IEN | ADC Conversion Result Comparison Upper Threshold Interrupt Mask Configuration<br>0: disabled<br>1: enable | 0x0 | R/W |
| 8 | LLT_IEN | ADC Conversion Result Compare Lower Threshold Interrupt Mask Configuration<br>0: disabled<br>1: enable | 0x0 | R/W |
| 7:0 | ADCXIEN[7:0] | ADC channel 7~0 interrupt mask configuration<br>0: disabled<br>1: enable | 0x0 | R/W |

## 28.10.17 ADC Interrupt Clear Register (ADC_INTCLR)

Address offset: 0x48

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | CONT_INTC | REG_INTC | HHT_INTC | LLT_INTC | ADCICLR[7:0] | | | | | | | |
| | | | | WO WO WO WO WO | | | | WO | | | | | | | |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11 | CONT_INTC | Write 1 to clear the continuous conversion complete flag<br>Write 0 has no | 0x0 | WO |
| 10 | REG_INTC | effect Write 1 to clear the ADC conversion result comparison interval flag<br>Write 0 has no effect | 0x0 | WO |
| 9 | HHT_INTC | Write 1 to clear ADC conversion result comparison upper threshold<br>Writing 0 has no effect | 0x0 | WO |
| 8 | LLT_INTC | Write 1 to clear ADC conversion result comparison lower threshold flag<br>Write 0 has no | 0x0 | WO |
| 7:0 | ADCICLR[7:0] | effect Write 1 to clear ADC channel 7~0 interrupt status<br>Writing 0 has no effect | 0x0 | WO |

### 28.10.18 ADC Before Mask Interrupt Status Register (ADC_RAWINTSR)

Address offset: 0x4C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | CONT_INTF | REG_INTF | HHT_INTF | LLT_INTF | ADCRIS[7:0] | | | | | | | |
| | | | | RO | RO | RO | RO | RO | | | | | | | |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11 | CONT_INTF | Continuous Conversion Complete Flag<br>0: ADC continuous conversion not completed<br>1: ADC continuous conversion completed | 0x0 | RO |
| 10 | REG_INTF | ADC conversion result comparison interval flag<br>0: ADC conversion result is outside [ADC_LT, ADC_HT) range<br>1: ADC conversion result is within [ADC_LT, ADC_HT) range | 0x0 | RO |
| 9 | HHT_INTF | ADC conversion result comparison upper threshold flag<br>0: ADC conversion result is outside [ADC_HT, 4095] range<br>1: The ADC conversion result is within the range [ADC_HT, 4095] | 0x0 | RO |
| 8 | LLT_INTF | ADC conversion result comparison lower threshold flag<br>0: The ADC conversion result is outside the interval [0, ADC_LT)<br>1: The ADC conversion result is in the range [0, ADC_LT) | 0x0 | RO |
| 7:0 | ADCRIS[7:0] | ADC channel 7~0 conversion complete interrupt status (before mask) | 0x0 | RO |

### 28.10.19 ADC Post Mask Interrupt Status Register (ADC_MSKINTSR)

Address offset: 0x50

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | CON T_MI f | REG _MIF | HHT_ MIF | LLT_ MIF | ADCMIS[7:0] | | | | | | | |
| | | | | RO | RO | RO | RO | RO | | | | | | | |

| bit | mark | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:12 | - | Reserved, always reads as 0. | 0 | - |
| 11 | CONT_MIF | Interrupt after continuous conversion completes mask<br>0: ADC continuous conversion not completed<br>1: ADC continuous conversion completed | 0x0 | RO |
| 10 | REG_MIF | Interrupt after ADC conversion result compare interval mask<br>0: ADC conversion result is outside [ADC_LT, ADC_HT) range<br>1: ADC conversion result is within [ADC_LT, ADC_HT) range | 0x0 | RO |
| 9 | HHT_MIF | The ADC conversion result is compared to the upper threshold mask<br>0: ADC conversion result is outside [ADC_HT, 4095] range<br>1: The ADC conversion result is within the range [ADC_HT, 4095] | 0x0 | RO |
| 8 | LLT_MIF | Interrupt after ADC conversion result compares lower threshold mask<br>0: The ADC conversion result is outside the interval [0, ADC_LT)<br>1: The ADC conversion result is in the range [0, ADC_LT) | 0x0 | RO |
| 7:0 | ADCMIS[7:0] | ADC channel 7~0 conversion completed interrupt status (after mask) | 0x0 | RO |

Machine Translated by Google

# 29 Low Voltage Detector (LVD)

## 29.1 Introduction to LVD

LVD can be used to monitor the operating voltage. When the comparison result of the monitored voltage and the LVD threshold meets the trigger condition, the LVD will generate an interrupt or reset signal. An interrupt or reset signal can only be cleared by an interrupt or reset clear signal. Only when the interrupt or reset signal is cleared will the On the trigger condition, an interrupt or reset signal is generated again.

The sampling filter clock is configurable and can be configured as APB clock or LIRC. The filter count value is configurable. Sampling reaches filter count value times When counting, the results are consistent and output.

3 trigger conditions: combination of high level, rising edge, and falling edge.

2 types of trigger results: interrupt, reset signal (it is forbidden to choose to generate a reset signal when the filter clock selects PCLK). interrupt and reset numbers cannot be generated at the same time.

## 29.2 LVD Block Diagram

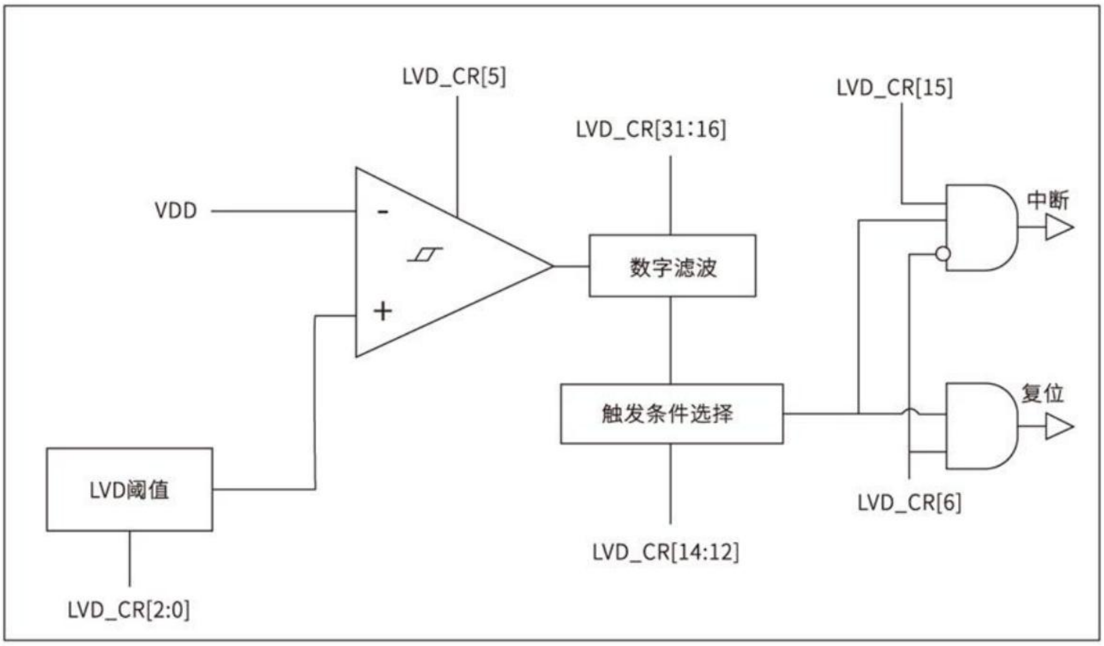


Figure 29-1 LVD structure block diagram

## 29.3 Digital Filtering

If the working environment of the chip is bad, the output of the hysteresis comparator will appear noise signal. enable the digital filter block, the hysteresis comparator Noise signals whose pulse width is less than the setting time of LVD_CR.FLT_NUM[15:0] in the output waveform can be filtered out. Disable digital filtering module, the input and output signals of the digital filter module are the same. Enable the digital filtering module, and the filtering diagram is as follows:

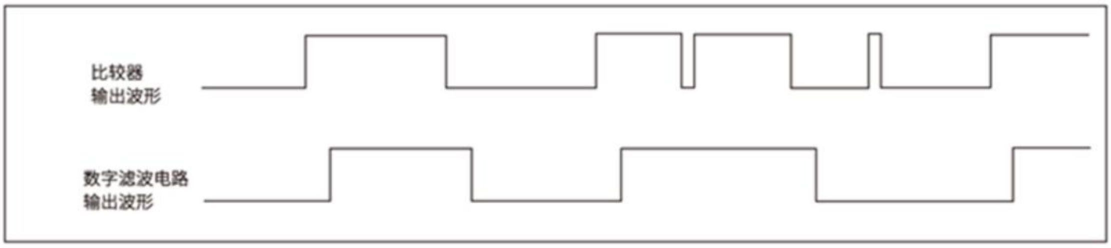


Figure 29-2 LVD filter output

## 29.4 Configuration Example

### 29.4.1 LVD configured as low voltage reset

In this mode, the MCU is reset when the monitored voltage is lower than the threshold voltage. The configuration method is as follows:

Step1: Configure LVD_CR.DIV_SEL to select the voltage division to be monitored.

Step2: Configure LVD_CR.FLT_NUM to select LVD filter time.

Step3: Configure LVD_CR.FLTCLK_SEL to select the filter clock.

Step4: Configure LVD_CR.FLTEN to enable LVD filtering.

Step5: Set LVD_CR.HIGHINTEN to 1, select high level to trigger LVD action.

Step6: Set LVD_CR.ACT to 1, select LVD action as reset.

Step7: Set LVD_CR.LVDEN to 1 to enable LVD.

### 29.4.2 LVD configured as voltage change interrupt

In this mode, an interrupt is generated when the monitored voltage goes above or below the threshold voltage. The configuration method is as follows:

Step1: Configure LVD_CR.DIV_SEL to select the voltage source to be monitored.

Step2: Configure LVD_CR.FLT_NUM to select LVD filter time.

Step3: Configure LVD_CR.FLTCLK_SEL to select the filter clock.

Step4: Configure LVD_CR.FLTEN to enable LVD filtering.

Step5: Set LVD_CR.RISEINTEN to 1, or LVD_CR.FALLINTEN to 1, or both to 1, select the level change trigger

Send LVD action.

Step6: Set LVD_CR.ACT to 0, select LVD action as interrupt.

Step7: Set LVD_CR.INT_EN to 1 to enable LVD interrupt.

Step8: Set LVD_CR.LVDEN to 1 to enable LVD.

Step9: Write 0 to LVD_SR.INTF in the interrupt service routine to clear the interrupt flag.

## 29.5 Register List

Base address: 0x4000 4000

| offset address | name | describe | Defaults |
|---|---|---|---|
| 0x00 | LVD_CR | LVD Control Register | 0x000 0007 |
| 0x04 | LVD_SR | LVD status register | 0x000 0000 |

## 29.6 Register Description

### 29.6.1 LVD Control Register (LVD_CR)

Address offset: 0x00

Reset value: 0x0000 0007

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT_NUM[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT_EN | HIGHINTEN | RISEINTEN | FALLINTEN | reserve | | FLTCLK_SEL[1:0] | | FLTEN | ACT | LVDEN | reserve | | DIV_SEL[2:0] | | |
| R/W | R/W | R/W | R/W | | | R/W | | R/W | R/W | R/W | | | R/W | | |

| bit flag | | Function | Reset value read and write | |
|---|---|---|---|---|
| 31:16 | FLT_NUM[15:0] | description LVD sampling filter count value<br>The sampling clock is ABP clock or LIRC. The filter count value is configurable. Sample times<br>When the count reaches the filter count value, the results are consistent and output.<br>Sample count period = FLT_NUM[15:0] | 0x0 | R/W |
| 15 | INT_EN | LVD interrupt enable<br>0: disabled<br>1: enable | 0 | R/W |
| 14 | HIGHINTEN | High-level trigger enable (VDD is lower than the threshold voltage)<br>0: disabled<br>1: enable | 0 | R/W |
| 13 | RISEINTEN | Rising edge trigger enable (VDD changes from above threshold voltage to below threshold voltage)<br>0: disabled<br>1: enable | 0 | R/W |
| 12 | FALLINTEN | Falling edge trigger enable (VDD changes from below threshold voltage to above threshold voltage)<br>0: disabled<br>1: enable | 0 | R/W |
| 11:10 | - | | 0x0 | – |
| 9:8 | FLTCLK_SEL[1:0] | Preserve filter clock selection<br>00: filter clock invalid<br>01: The filter clock is selected as PCLK (can only be configured as interrupt mode)<br>10: Filter clock selection is LIRC<br>11: reserved | 0x0 | R/W |
| 7 | FLTEN | Digital filter function configuration<br>0: disable digital filtering<br>1: Enable digital filtering | 0 | R/W |
| 6 | ACT | LVD interrupt reset selection bit<br>0: generate an interrupt<br>1: generate a reset | 0 | R/W |
| 5 | LVDEN | LVD enable<br>0: disable LVD<br>1: enable LVD | 0 | R/W |
| 4:3 | - | reserve | 0x0 | – |

| | | | | |
|---|---|---|---|---|
| 2:0 | DIV_SEL[2:0] | LVD divider configuration<br>000: 4.4V<br>001: 4.0V<br>010: 3.6V<br>011: 3.3V<br>100: 3.1V<br>101: 2.9V<br>110: 2.7V<br>111: 2.5V | 0x7 | R/W |

### 29.6.2 LVD Status Register (LVD_SR)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | INTF |
| | | | | | | | | | | | | | | | W0C |

| Bit | Flag | Functional Description | Reset value | read and write |
|---|---|---|---|---|
| 31:1 | – | reserve | 0x0 | – |
| 0 | INTF | LVD interrupt flags:<br>0: No LVD interrupt occurred<br>1: LVD interrupt occurred<br>Writing 0 clears the interrupt flag, writing 1 has no effect. | 0 | W0C |

# 30 Voltage comparator (VC)

## 30.1 Introduction to VC

The analog voltage comparator VC is used to compare the magnitude of two input analog voltages, and output high/low level according to the comparison result. When "+"

When the input terminal voltage is higher than the "-" input terminal voltage, the voltage comparator outputs a high level; when the "+" input terminal voltage is lower than the "-" input terminal

When the terminal voltage is low, the voltage comparator outputs a low level.

Both "+" input and "-" input support 4-way power input selection.

3 trigger conditions: combination of high level, rising edge, and falling edge.

2 kinds of trigger results: interrupt, reset signal. Interrupt and reset signals cannot be generated at the same time.

## 30.2 VC block diagram

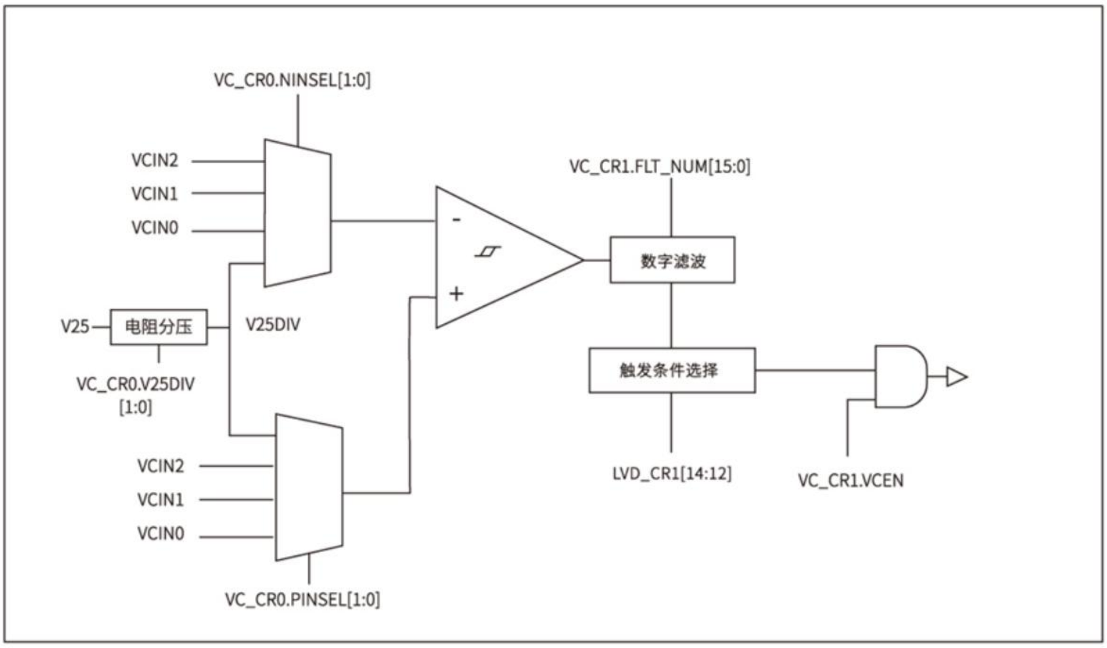


Figure 30-1 VC structure block diagram

## 30.3 Digital Filtering

If the working environment of the chip is bad, the output of the hysteresis comparator will appear noise signal. enable the digital filter block, the hysteresis comparator

Noise signals whose pulse width is less than the time set by VC_CR1.FLT_NUM[15:0] in the output waveform can be filtered out. Disable digital filtering

module, the input and output signals of the digital filter module are the same. Enable the digital filtering module, and the filtering diagram is as follows:

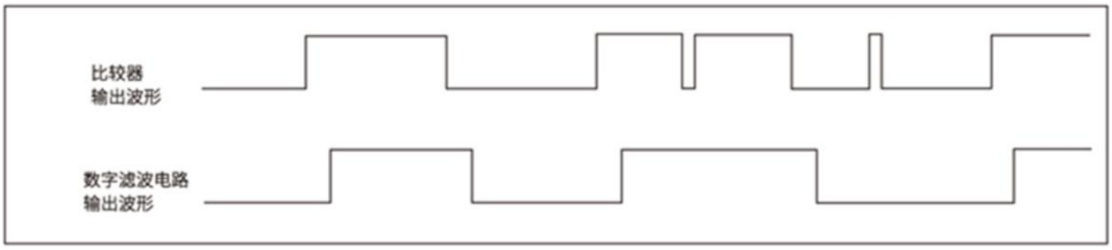


Figure 30-2 VC filter output

## 30.4 Configuration example

In this mode, an interrupt is generated when the monitored voltage goes above or below the threshold voltage. The configuration method is as follows:

Step1: Configure VC_CR0.V25DIV_EN to enable voltage division.

Step2: Configure VC_CR0.V25DIV and set the voltage division coefficient.

Step3: Configure VC_CR0.NINSEL, and select the voltage source to be monitored at the "-" terminal.

Step4: Configure VC_CR0.PINSEL, and select the voltage source to be monitored at the "+" terminal.

Step5: Configure VC_CR1.FLT_NUM[15:0] to select VC filter time.

Step6: Configure VC_CR1.FLTCLK_SEL to select the filter clock.

Step7: Configure VC_CR1.FLTEN to enable VC filtering.

Step8: Set the HIGHINTEN, RISEINTEN, and FALLINTEN of VC_CR1, and select the trigger mode.

Step9: Set VC_CR1.INT_EN to 1 to enable VC interrupt.

Step10: Set VC_CR1.VCEN to 1 to enable VC.

Step11: Write 0 to VC_SR.INTF in the interrupt service routine to clear the interrupt flag.

## 30.5 VC register list

Base address: 0x4000 4000

| Offset | name | describe | Defaults |
|---|---|---|---|
| address 0x080 | VC_CR0 | VC Control Register 0 | 0x0000000 |
| 0x084 | VC_CR1 | VC Control Register 1 | 0x0000000 |
| 0x088 | VC_OUTCFG | VC output configuration register | 0x0000000 |
| 0x08C | VC_SR | VC status register | 0x0000000 |

## 30.6 VC register description

### 30.6.1 VC Voltage Control Register (VC_CR0)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| reserve | | | | | | | | | V25DIV EN | V25DIV[1:0] | | NINSEL[1:0] | | PINSEL[1:0] | |
| | | | | | | | | | R/W | R/W | | R/W | | R/W | |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:7 | – | reserve | 0x0 | – |
| 6 | V25DIV_EN | V25 frequency division enable control of VC<br>0: disabled<br>1: enable | 0x0 | R/W |
| 5:4 | V25DIV[1:0] | VC voltage divider control.<br>00: 1/4 V25<br>01: 2/4 V25<br>10: 3/4 V25<br>11: V25 | 0x0 | R/W |
| 3:2 | NINSEL[1:0] | "-" input terminal voltage voltage selection<br>select. 00: VCIN[0];<br>01: VCIN[1]<br>10: VCIN[2]<br>11: V25DIV | 0x0 | R/W |
| 1:0 | PINSEL[1:0] | "+" input terminal voltage voltage<br>Select 00:VCIN[0];<br>01: VCIN[1]<br>10: VCIN[2]<br>11: V25DIV | 0x0 | R/W |

### 30.6.2 VC Control Register (VC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT_NUM[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT_<br>EN | HIGH<br>INTE<br>N | RISEI<br>NTE<br>N | FALLI<br>NTE<br>N | reserve | | | FLTE<br>N | reserve | | | | VC_FLTCLK_S<br>EL[1:0] | | reserve | VCE<br>N |
| R/W | R/W | R/W | R/W | | | | R/W | | | | | R/W | | | R/W |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:16 | FLT_NUM[15:0] | VC sampling filter count value<br>The sampling clock is APB clock or LIRC. The filter count value is configurable. sampling<br>When the number of times reaches the filter count value, the results are consistent and output.<br>Sample count period = FLT_NUM[15:0] | 0x0 | R/W |
| 15 | INT_EN | VC interrupt enable<br>0: disabled<br>1: enable | 0 | R/W |
| 14 | HIGHINTEN | VC output signal high level trigger enable<br>0: disabled<br>1: enable | 0 | R/W |
| 13 | RISEINTEN | VC output signal rising edge trigger enable<br>0: disabled<br>1: enable | 0 | R/W |
| 12 | FALLINTEN | VC output signal falling edge trigger enable<br>0: disabled<br>1: enable | 0 | R/W |
| 11:9 | - | reserve | 0x0 | – |
| 8 | FLTEN | Digital filter function configuration<br>0: disable digital filtering<br>1: Enable digital filtering | 0 | R/W |
| 7:4 | - | reserve | 0x0 | – |
| 3:2 | VC_FLTCLK_SEL[1:0] | VC filter clock selection<br>00: filter clock invalid<br>01: The filter clock is selected as PCLK<br>10: Filter clock selection is LIRC<br>11: reserved | 0x0 | R/W |
| 1 | - | reserve | 0 | – |
| 0 | VCEN | VC Enable<br>0: Voltage comparison function disabled<br>1: Enable the voltage comparison function | 0 | R/W |

### 30.6.3 VC Output Configuration Register (VC_OUTCFG)

Address offset: 0x08

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | INV_ PAD | TM1B KE | TM1 CH3_ EN |
| | | | | | | | | | | | | | R/W | R/W | R/W |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INV_ TM1 CH4 | TM1 CH3_ EN | INV_ TM1 CH3 | TM1 CH2_ EN | INV_ TM1 CH2 | TM1 CH1_ EN | INV_ TM1 CH1 | PCA ECI_ EN | PCA CAP0 EN | INV_ PCA | LPTI MEX T_EN | LPTI M_E N | reserve | TIM1 _EN | TIM0 _EN | INV_ TIMX |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |

| Bitmark 31:19 18 | | Functional description | Reset value read and | |
|---|---|---|---|---|
| | – | reserved | write | – |
| | INV_PAD | VC filter result output to PAD reverse enable<br>0: disabled; 1: enabled | 0x0 0 | R/W |
| 17 | TM1BKE | VC interrupt as TIM1 brake control<br>0: disabled; 1: enabled | 0 | R/W |
| 16 | TM1CH4_EN | VC filter result output to TIM1 CH4 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 15 | INV_TM1CH4 | VC filter result output to TIM1 CH4 gate reverse enable<br>0: disabled; 1: enabled | 0 | R/W |
| 14 | TM1CH3_EN | VC filter result output to TIM1 CH3 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 13 | INV_TM1CH3 | VC filter result output to TIM1 CH3 gate reverse enable<br>0: disabled; 1: enabled | 0 | R/W |
| 12 | TM1CH2_EN | VC filter result output to TIM1 CH2 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 11 | INV_TM1CH2 | VC filter result output to TIM1 CH2 gate reverse enable<br>0: disabled; 1: enabled | 0 | R/W |
| 10 | TM1CH1_EN | VC filter result output to TIM1 CH1 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 9 | INV_TM1CH1 | VC filter result output to TIM1 CH1 gate reverse enable<br>0: disabled; 1: enabled | 0 | R/W |
| 8 | PCAECI_EN | VC filter output to PCA external clock enable<br>0: disabled; 1: enabled | 0 | R/W |
| 7 | PCACAP0_EN | VC filter output to PCA capture 0 enable<br>0: disabled; 1: enabled | 0 | R/W |
| 6 | INV_PCA | VC filter result output to PCA reverse enable<br>0: disabled; 1: enabled | 0 | R/W |
| 5 | LPTIMEXT_EN | VC filter output to LPTIM external clock enable control<br>0: disabled; 1: enabled | 0 | R/W |
| 4 | LPTIM_EN | VC filter result output to LPTIM gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 3 | – | reserve | 0 | – |
| 2 | TIM1_EN | VC filter result output to TIM11 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 1 | TIM0_EN | VC filter result output to TIM10 gate enable<br>0: disabled; 1: enabled | 0 | R/W |
| 0 | INV_TIMX | VC filter result output to TIM10, TIM11, LPTIM gate reverse enable<br>0: disabled; 1: enabled | 0 | R/W |

### 30.6.4 VC Status Register (VC_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | VC_FLOUT | INTF |
| | | | | | | | | | | | | | | RO | W0C |

| bit flag | | Functional description | Reset value | read and write |
|---|---|---|---|---|
| 31:2 | – | reserve | 0x0 | – |
| 1 | VC_FLOUT | VC's post-filter state<br>0: VC filtering result is 0<br>1: VC filtering result is 1 | 0 | RO |
| 0 | INTF | VC interrupt flag<br>0: No VC interrupt occurred<br>1: VC interrupt occurs<br>Write 0 to clear the interrupt flag, write 1 to have no effect | 0 | W0C |

# 31 Option Bytes

The option byte area is located at address 0x0800 0000- 0x0800 01FF.

Each configuration field in the option byte area is used for the user to realize the configuration of some system functions.

| address | name | illustrate |
|---|---|---|
| 0x0800 0000 | USERCFG0 | ISP_CON configuration word |
| 0x0800 0004 | USERCFG1 | SWD Protection Bit Configuration Word |
| 0x0800 0008 | USERCFG2 | IWDGCNT[19:0], IWDGMODE, IWDGINTMASK, IWDGON configuration words |

## 31.1 User configuration register 0 (USERCFG0)

Address: 0x0800 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | ISP_<br>CON<br>R/W |

| bit flag | | Functional description | default | read-write |
|---|---|---|---|---|
| 31:1 | - | reserve. | 1 | - |
| 0 | ISP_CON | ISP function enable control bit.<br>0: enable ISP function<br>1: Disable ISP function<br>Note: After the user changes this bit, a reset other than CPURST must be generated for the function of this bit to work.<br>will work. | 1 | R/W |

Machine Translated by Google

## 31.2 User configuration register 1 (USERCFG1)

Address: 0x0800 0004

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | | | | | | | | | | | | SWDP |
| | | | | | | | | | | | | | | | R/W |

| Bit | Flag | Functional Description | Defaults | read and write |
|---|---|---|---|---|
| 31:1 | - | reserve. | 0x7FFFFFFF | - |
| 0 | SWDP | SWDP: SWD protection bit.<br>0: The SWD interface is configured as protected;<br>1: The SWD interface is configured as unprotected.<br>Note: After the user changes this bit, a reset other than CPURST must be generated. The function of this bit will be effective. | 1 | R/W |

## 31.3 User Configuration Register 2 (USERCFG2)

Address: 0x0800 0008

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDT ON | reserve | WDTI NTM ASK | WDT MOD E. | reserve | | | | | | | | WDTCNT[19:16] | | | |
| R/W | | R/W | R/W | | | | | | | | | R/W | | | |

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

| bit flag | | Functional description | default | read-write |
|---|---|---|---|---|
| 31 | IWDGON | IWDGON configuration bit.<br>0: hardware does not automatically start IWDG<br>1: The hardware automatically starts IWDG | 1 | R/W |
| 30 | - | reserve. | 1 | - |
| 29 | IWDGINTMASK | IWDG Interrupt mask bit.<br>0: IWDG does not mask interrupt<br>1: IWDG mask interrupt | 1 | R/W |
| 28 | IWDGMODE | IWDG mode configuration bit.<br>0: reset mode<br>1: interrupt mode | 1 | R/W |
| 27:20 | - | reserve. | 0xFF | - |
| 19:0 | IWDGCNT[19:0] | IWDG count value register | 0xFFFFF | R/W |

## 31.4 Attention

Since the rewriting of the option byte area needs to comply with the erasing sequence of the Flash, when the user rewrites the parameters of the option byte area, the entire option

Read the backup of the data in the byte area, erase the entire option byte area, rewrite the fields that need to be configured, and then write all the options back to the option byte area.

# 32 Debug support (DBG)

The JTAG/SWD of the general-purpose MCU is a non-encrypted traditional JTAG/SWD debugging interface, which protects the confidentiality of the client program and the system

There are very large hidden dangers in terms of security. In order to protect user programs and improve system security, this chip is integrated on the SWD port

A security authorization circuit is designed. When the chip leaves the factory, the port is configured with the SWD debugging interface by default, and cooperates with the host computer software or

In the user program, when writing 0 value in the SWDP position of the upper computer software, the SWD debugging port will be automatically disconnected after reset or next power-on.

The debug interface SWD cannot be opened by writing 0xFF through SWDP. If it is necessary to open the SWD interface, the chip must be fully erased.

Note: SWDP is in the user configuration register 1 (USERCFG1) of the option byte area (Option Bytes), please refer to 31.2 User Configuration

Register 1 (USERCFG1).

Notice:

ÿWhen the user does not set SWDP, PC7 and PD1 are automatically configured as SWD debugging port (PC7 pull up, PD1 down pull); users can also configure the debug interface as GPIO by configuring the RCC_SWDIOCR.SWDPORT register.

ÿWhen the user sets the SWDP value to 0, the PC7 and PD1 ports are automatically disconnected from the SWD debugging port, that is, the SWD cannot be used Debug function. RCC_SWDIOCR.SWDPORT cannot be written to 1.

## 32.1 SWD debugging interface description

### 32.1.1 Pin Assignments for SWD Debug Interface

The 2 GPIOs of this chip can be used as SWD interface pins. These pins are present on all packages.

| SWD interface pin name SWD interface type | | SWD interface function | pin assignment |
|---|---|---|---|
| SWDIO | input Output | Serial data input/output PC7 | |
| SWDCLK | enter | serial clock | PD1 |

### 32.1.2 Internal pull-up and pull-down of SWD pin

It is necessary to ensure that the SWD pins are not floating, since they are directly connected to the D flip-flops that control the debug mode. must

Pay special attention to the SWDCLK pin because it is directly connected to the clock terminal of some D flip-flops.

To avoid any uncontrolled I/O levels, this chip embeds internal pull-up and pull-down resistors on the SWD pin.

ÿ SWDIO: internal pull-up

ÿ SWCLK: Input with pull-down

Software can also configure these I/O ports as GPIO.

## 32.2 How the SWD protection bit works

1. The chip received by the customer is a blank chip, and the value of SWDP is 1, so SWD is enabled by default.

2. The customer uses Keil/IAR for software development. After the development is completed, it can be downloaded directly through Keil/IAR or through the burner download.

3. After the customer downloads the program, configure the value of SWDP to 0 through the host computer and enable the protection bit.

4. After the MCU is reset again, the SWD protection bit takes effect immediately.

5. If the customer needs to perform SWD debugging again, the chip needs to be completely erased to re-open the SWD channel.

## 32.3 Using SWD in Low Power Mode

### 32.3.1 Using SWD in Sleep Mode

While in sleep mode, the system clock keeps running and the SWD connection is not interrupted.

### 32.3.2 Using SWD in Deep Sleep Mode

1. When SYSCON_CFGR0.DBGDLSP_DIS=0 (default value), when entering deep sleep mode system in Debug mode The clock will stop, and the SWD connection will be interrupted; when SYSCON_CFGR0.DBGDLSP_DIS=1, enter in Debug mode In deep sleep mode the system clock will not be stopped and the SWD connection will not be interrupted.

2. After entering Deep Sleep Mode, reset the chip to wake up the chip through the SWD port.

## 32.4 DBG register list

DBG base address: 0x4000 4C00

Table 32-1 Debug register map and reset value

| offset address | name | describe | Defaults |
|---|---|---|---|
| 0x0 | DBG_APBFZ | Debug mode control register | 0x0000 0000 |

## 32.5 DBG register description

### 32.5.1 Debug mode control register (DBG_APBFZ)

Address offset: 0x00

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | twenty four | twenty three | twenty two | twenty one | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY | | | | | | | | | | | | | | | |
| WO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserve | | | | TIM2 DBG STO P | WWD GDB GST OP | IWD G DBG STO P | BEEP DBG STO P | reserve | RTC DBG STO P | TIM1 DBG STO P | PCA DBG STO P | reserve | LPTI MDB GST OP | TIM1 1DB GST OP | TIM1 0DB GST OP |
| | | | | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | | R/W | R/W | R/W |

| bit flag | | Function Description Reset value read and write is valid only when the high bit is written to 0x5A69 to configure this register, and it is invalid | | |
|---|---|---|---|---|
| 31:15 Key 14:12 - | | when other values are written. 0x0 WO | | |
| | | Leave | 0x0 | - |
| 11 | TIM2DBGSTOP | TIM2 debug mode stop working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 10 | WWDGDBGSTOP | WWDG debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 9 | IWDGDBGSTOP | IWDG debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 8 | BEEPDBGSTOP | BEEP debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 7 | - | Keep | 0 | - |
| 6 | RTCDBGSTOP | RTC debug mode stop working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 5 | TIM1DBGSTOP | TIM1 debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 4 | PCADBGSTOP | PCA debug mode stops working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 3 | - | reserve | 0 | - |
| 2 | LPTIMDBGSTOP | Low Power Timer debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 1 | TIM11DBGSTOP | TIM11 debug mode stopped working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |
| 0 | TIM10DBGSTOP | TIM10 debug mode stops working<br>0: Debug mode counters still work<br>1: Debug mode counter stops working | 0 | R/W |

# 33 In-Circuit Programming (ICP)

Program Flash through In-Circuit Programming (ICP). If the product is under development, or in the end customer's product requires a firmware upgrade,

It is very difficult and inconvenient to adopt the hardware programming mode. The ICP approach will be simple and does not require removing the microcontroller from the board

Come. The ICP method also allows customers to program devices on mass-produced circuit boards and program them after the devices are assembled, which allows devices to be programmed

program the latest firmware or customized firmware.

Execute ICP function, only need 5 pins ICP_CLK, ICP_DIN, ICP_CS, ICP_DOUT, RSTN for entering or exiting

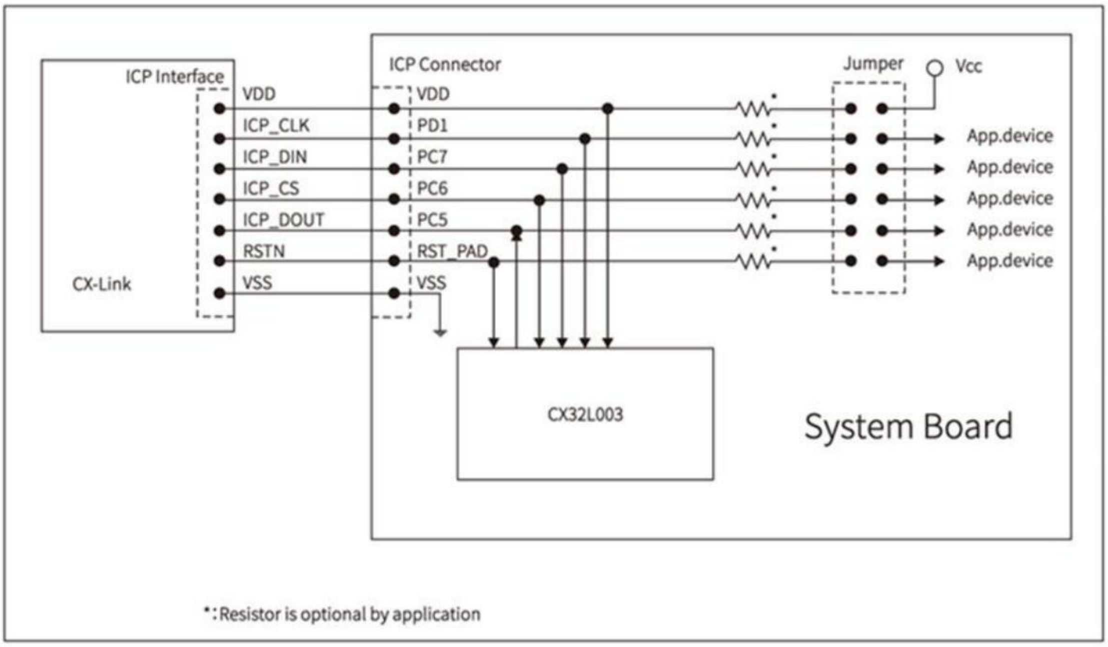
Out of ICP mode, ICP_DIN, ICP_DOUT are data input and output pins, ICP_CLK is programming clock input pin, ICP_CS and RSTN

It is the control function pin during programming. Users need to reserve VDD, GND and these five pins on the system board.

The ICP programmer is specially designed according to the electrical characteristics of the MCU, and it is a very efficient and stable programming method. For details, please refer to

Application Notes.

Schematic diagram of ICP interface connection mode:



Note:

1. When using ICP to update the code, RST_PAD, PD1, PC7, PC6, PC5 must be disconnected from the system load.

2. After the ICP programming is finished, it is recommended to turn off the system power and remove the CX_Link, and then connect the power again.

# 34 Appendix 1: SysTick Timer (SYST)

## 34.1 Introduction to SysTick Timers

If the OS wants to support multitasking, it needs to perform context switching periodically, so hardware resources such as timers are needed to interrupt the program. implement. When the timer interrupt is generated, the processor will perform OS task scheduling in exception handling, and will also perform OS maintenance Work. There is a simple timer called SysTick in the Cortex-M0+ processor, which is used to generate periodic interrupt requests.

SysTick is a 24-bit timer and counts down. After the timer counts down to 0, it will reload a programmable number value, and generate a SysTick interrupt (interrupt number is 15) at the same time, this abnormal event will cause the execution of the SysTick interrupt processing, this Processes are part of the OS.

For systems that do not require an OS, the SysTick timer can also be used for other purposes, such as timing, timing, or for periodic execution Rows of tasks provide interrupt sources. The generation of SysTick interrupt is controllable. If the interrupt is disabled, it can still be used by polling. Use the SysTick timer, for example to check the current count value or poll the overflow flag.

## 34.2 Setting the SysTick timer

Since the reload value and current value of the SysTick timer are undefined at reset, in order to prevent abnormal results, the SysTick The configuration needs to follow a certain process:

1. Set SYST_CSR.ENABLE to 0 to disable SysTick.
2. Configure SYST_CSR.CLKSOURCE to select the clock source of SysTick.
3. Configure SYST_RVR to select the overflow period of SysTick.
4. Write any value to SYST_CVR, clear SYST_CVR and SYST_CSR.COUNTFLAG.
5. Set SYST_CSR.TICKINT to 1 to enable SysTick interrupt.
6. Set SYST_CSR.ENABLE to 1, enable SysTick, and start counting.
7. Read SYST_CSR in the Interrupt Service Routine to clear the overflow flag.

Note: SysTick timer overflow time is SYST_RVR+1 SysTick clock cycle, configuration example:

| SysTick clock source | SysTick clock period | SYST_RVR | SysTick timer overflow time |
|---|---|---|---|
| Core clock HCLK (set to 4MHz) | 0.25us | 3999 | 1ms = 0.25us * (3999+1) |

## 34.3 SysTick Timer Register List

| address | name | describe | reset value | CMSIS symbol |
|---|---|---|---|---|
| 0xE000 E010 | SYST_CSR | SysTick timer control and status register | 0x0000 0000 | SysTick->CTRL |
| 0xE000 E014 | SYST_RVR | SysTick timer reload value register<br>device | undefined | SysTick->LOAD |
| 0xE000 E018 | SYST_CVR | SysTick timer current value register<br>device | undefined | SysTick->VAL |
| 0xE000 E01C | SYST_CALIB | SysTick timer calibration value register<br>device | 0x4000 9C3F | SysTick->CALIB |

Note: SYST_CALIB register has a corresponding relationship with RCC_STICKCR, please refer to Section 6.4.15 SysTickTimer Control Register (RCC_STICKCR).

## 34.4 SysTick Timer Register Description

### 34.4.1 SysTick Timer Control and Status Register (SYST_CSR)

| bit symbol | | Functional description | type reset | value |
|---|---|---|---|---|
| 31:17 reserved | | - | - | - |
| 16 | COUNTFLAG | SysTick timer overflow flag<br>1: SysTick timer underflow occurred.<br>0: The SysTick timer has not overflowed.<br>Reading this register clears the COUNTFLAG flag | RO | 0 |
| 15:3 reserved | | - | - | - |
| 2 | CLK SOURCE | SysTick clock source selection<br>1: Use the core clock (HCLK)<br>0: HCLK/4 | RW 0 | |
| 1 | TICKINT | SysTick interrupt enable<br>1: enable interrupt<br>0: disable interrupt | RW 0 | |
| 0 | ENABLE | SysTick timer enable<br>1: Enable SysTick<br>0: Disable SysTick | RW 0 | |

### 34.4.2 SysTick Timer Reload Value Register (SYST_RVR)

| bit symbol | | Functional description | type reset | value |
|---|---|---|---|---|
| 31:24 | reserve | - | - | - |
| 23:0 | RELOAD | SysTick timer reload value | RW | undefined |

### 34.4.3 SysTick timer current value register (SYST_CVR)

| bit symbol | | Functional description | type reset | value |
|---|---|---|---|---|
| 31:24 | reserve | - | - | - |
| 23:0 | CURRENT | Read this register to get the current count value of the SysTick timer;<br>Write any value to this register, clear this register and COUNTFLAG. | RW undefined | |

### 34.4.4 SysTick Timer Calibration Value Register (SYST_CALIB)

| bit | symbol | Functional description | type | reset value |
|---|---|---|---|---|
| 31 | NOREF | Whether the SysTick timer uses an external reference clock<br>0: HCLK/4<br>1: Use the core clock (HCLK) | RO | 0 |
| 30 | SKEW | Is the 10ms TENMS value accurate?<br>0: accurate<br>1: Inaccurate | RO | 1 |
| 29:24 | reserve | | - | - |
| 23:0 | TENMS[23:0] | SysTick 10ms calibration value, this value uses an external reference clock 10ms calibration value of HCLK/4(4MHZ). | RO | 0x009C3F |

# 35 revision records

| Version | Revision Date | Revision Summary First Edition |
|---|---|---|
| 0.1 | 2019/3/28 | |
| 0.2 | 2019/9/6 | Unify the inconsistency of the context of the module name and register name:<br>Chapter 7 System Control (SYSCON): SYSCFG is unified as SYSCON<br>Chapter 14 Programmable Count Array (PCA): PCA_CCON is unified as PCA_CR, PCA_CMOD Unified as PCA_MOD<br>Chapter 26 Clock Calibration/Monitoring Module (CLKTRIM): CLKTRIM_CALOVCNT Unified as CLKTRIM_CALCON, RTC_ALM2PR unified into RTC_ALM2PRD |
| 0.3 | 2019/9/12 | Added chapter: Appendix 1: SysTick timer (SYST) |
| 0.3.1 | 2019/9/16 | Table 13-2 TIM2 register list and reset value: offset address 0x30 is reserved |
| 0.3.2 | 2019/9/25 | fix some bugs<br>18.4.1 Beeper Control/Status Register (BEEP_CSR): CLKSEL context changed to consistent<br>26.5.1 Configuration register (CLKTRIM_CR): REFCLK_SEL[2:0], 100 select HXT bypass bell |
| 0.3.3 | 2019/10/12 | content update<br>6.4.10 Internal High Speed RC Oscillator Control Register (RCC_HIRCCR): Frequency Calibration Value Address Update<br>New, the calibration value address of the package chip and the KGD chip are different<br>21.7.1 UART control register (UARTx_SCON): bit 8 is FEEN (receive frame error interrupt enable able)<br>21.7.5 UART flag bit register (UARTx_INTSR): bit 3 is reserved<br>22.12.5 LPUART flag register (LPUART_INTSR): bit 3 is reserved<br>error<br>26.5.5 Interrupt Flag Bit Register (CLKTRIM_IFR): HXT_FAULT and LXT_FAULT Survey<br>error |
| 0.3.4 | 2019/10/22 | Errata<br>22.12.2 LPUART control register (LPUART_SCON): PRSC[2:0] reset value is 0x7<br>Table 14-1 PCA comparison/capture function module settings: the MAT value in the last row is 0 |
| 1.0 | 2019/11/19 | Delete the multiplexing function TIM2_CH3 of PD2<br>Official release 1.0 |