User manual

User Manual

TK499

Version: **0.8**

contents

Page 6

Page 7

Page 19

Page 20

Page 22

1. **CRC** calculation unit

1.1 Introduction to **CRC**

The cyclic redundancy check (CRC) calculation unit obtains the CRC calculation result of any 32-bit full word according to a fixed generator polynomial. In other In the application, CRC technology is mainly used to verify the correctness and completeness of data transmission or data storage. Standard EN/IEC 60335-1 namely Provides a method to verify the integrity of flash memory. The CRC calculation unit can calculate the identification of the software while the program is running, and then compare it with the The reference identifier generated during connection is compared and then stored in the designated memory space.

1.2 Main features of **CRC**

- Use CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X_{32} + X_{26} + X_{23} + X_{22} + X_{16} + X_{12} + X_{11} + X_{10} + X_{8} + X_{7} + X_{5} + X_{4} + X_{2} + X +1$
- A 32-bit data register for input/output
- CRC calculation time: 4 AHB clock cycles (HCLK)
- General 8-bit register (can be used to store temporary data)

The figure below is the block diagram of the CRC calculation unit

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

AHB bus

Data register (output)

CRC calculation (polynomial: 0x4C11DB7)

Data register (input)

1.3 **CRC** function introduction

The CRC calculation unit contains a 32-bit data register:

- When writing to this register, it can be used as an input register to input new data for CRC calculation.
- When reading this register, the result of the last CRC calculation is returned.

Each time the data register is written, the calculation result is the combination of the previous CRC calculation result and the new calculation result (for the entire 32-bit word) CRC calculation, not byte by byte).

During the CRC calculation, the CPU write operation is suspended, so you can write back to back to the register CRC_DATA or continuously Write-read operation.

The CRC_DATA register can be reset to 0xFFFF FFFF by setting the RESET bit of the CRC_CTRL register. The operation Does not affect the data in the register CRC_IDATA.

Page 23

1.4 **CRC** register

The CRC calculation unit includes 2 data registers and a control register.

1.4.1 **CRC** data register (**CRC_DR**)

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
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	DR: Data register bits When writing new data of the CRC calculator, it is used as an input register Return CRC calculation result when reading

1.4.2 **CRC** independent data register (**CRC_IDR**)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit 31: 8	Reserve
Bit 7:0	IDR: General-purpose 8-bit data register bits Can be used to temporarily store 1 byte of data. The CRC reset generated by the RESET bit of the CRC_CTRL register has no effect on this register

Note: This register does not participate in *CRC* calculation and can store any data.

1.4.3 **CRC** control register (**CRC_CTRL**)

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															RESET
															w

Bit 31:1	Reserve
Bit 0	**RESET** : Reset the CRC calculation unit (CRC reset) Set the data register to 0xFFFF FFFF. Only write a '1' to this bit, and it will be automatically cleared to '0' by the hardware.

2. Power control (**PWR**)

2.1 Power

The chip's working voltage (V_{DD}) is 2.5V ~ 3.6V. The required 1.2V power supply is provided through the built-in voltage regulator.

When the main power supply V_{DD} is powered down, the V_{BAT} pin provides power for the real-time clock (RTC) and backup registers.

Figure **2.** Power block diagram

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

2.1.1 Independent **A/D** converter power supply and reference voltage

In order to improve the accuracy of the conversion, the ADC uses an independent power supply to filter and shield glitch interference from the printed circuit board.

- The ADC's power supply pin is V_{DDA}
- Independent power ground V_{SSA}

If there is a V_{REF-} pin (depending on the package), it must be connected to V_{SSA} .

2.1.2 Battery backup area

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

V_{BAT} pin provides power for RTC, LSE oscillator and ports PB13 to PB15, which can ensure that RTC can continue to work when the main power supply is cut off. do. The switch to switch to V_{BAT} power supply is controlled by the power-down reset function in the reset module.

warn:

After V_{DD} rises ($t_{RSTTEMPO}$) or PDR (power-down reset) is detected , the power switch between V_{BAT} and V_{DD} will still protect Keep connected to V_{BAT} .

In the V_{DD} rise, if V_{DD} less than $t_{RSTTEMPO}$ reached a steady state (during the time t on $_{RSTTEMPO}$ value of the reference data may be hand Book in relevant part), and $V_{DD} > V_{\text{the BAT}} + 0.6V$ when, through current may V_{DD} and $V_{\text{the BAT}}$ is injected into the interior of the diode V between $_{\text{the BAT}}$.

If the power supply or battery connected to V_{BAT} cannot withstand such injected current, it is strongly recommended to connect an external $V_{\text{BAT to the}}$ power supply Low-dropout diode.

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

When the backup area is powered by the V_{DD} internal analog switch connected to V_{DD} , the following functions are available:

- PB14 and PB15 can be used for GPIO or LSE pins
- PB13 can be used as general I/O port, TAMPER pin, RTC calibration clock, RTC alarm or second output (see backup register (BKP))

Note: Because the analog switch can only pass a small amount of current (*3mA*), it is *possible* to use the *I/O* port function of *PB13* to *PB15* in the output mode Restricted: The speed must be limited below *2MHz* , the maximum load is *30pF* , and these *I/O* ports must not be used as current sources (such as driving *LEDs*).

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

- PB14 and PB15 can be used for LSE pin, and Timer7 function multiplexing
- PB13 can be used as TAMPER pin, RTC alarm or second output (see: RTC clock calibration register (BKP_RTCCR))

2.1.3 Voltage regulator

After reset, the regulator is always enabled and provides 1.2V.

2.1.4 Power-on reset (**POR**) and power-down reset (**PDR**)

There is a complete power-on reset (POR) and power-down reset (PDR) circuit inside TK499. When the power supply voltage reaches 1.45V, the system Both can work normally.

When V_{DD} /$V_{DDA\ is}$ lower than the specified limit voltage V_{POR} /V_{PDR} , the system remains in the reset state without the need for an external reset circuit. About For details of power-on reset and power-down reset, please refer to the electrical characteristics section of the data sheet.

Figure **3.** Waveform diagram of power-on reset and power-down reset

2.2 Low power consumption mode

After the system or power supply is reset, the microcontroller is in operation. When the CPU does not need to continue to run, a variety of low power consumption modes can be used To save power, such as waiting for an external event. Users need based on the lowest power consumption, fastest startup time and available wakeup sources, etc. Conditions, select an optimal low-power mode.

TK499 has two low-power modes:

- Sleep mode (CPU stops, all peripherals including CPU peripherals, such as NVIC, system clock (SysTick), etc. are still running)
- Stop mode (all clocks are stopped)

In addition, in running mode, power consumption can be reduced in one of the following ways:

- Reduce the system clock
- Turn off the unused peripheral clocks on the APB and AHB buses.

Table **1.** List of low-power modes

model	Enter	wake	Impact on 1.2V regional clock	To the V $_{DD}$ area clock Influence	Voltage Regulator
Sleep (SLEEP-NOW or SLEEP-ON-EXIT)	WFI (Wait for Interrupt)	Any interrupt	CPU clock is off, for other No influence on clock and ADC clock	without	open
	WFE (Wait for Event)	Wake up event			
Downtime	PDDS bit +SLEEPDEEP bit +WFI or WFE	Any external interrupt Or external event	All areas that use 1.2V Clocks are off	PLL, HSI and HSE Oscillator is off	open

2.2.1 Reduce the system clock

In running mode, by programming the prescaler register, you can reduce any system clock (SYSCLK, HCLK, PCLK1, PCLK2) speed. Before entering sleep mode, the prescaler can also be used to reduce the peripheral clock.

For details, please refer to: Clock Configuration Register (RCC_CFGR)

2.2.2 Control of external clock

In the run mode, you can reduce power consumption by stopping clocks (HCLK and PCLKx) for peripherals and memory at any time.

In order to reduce power consumption more in sleep mode, the clocks of all peripherals can be turned off before executing WFI or WFE instructions.

By setting AHB1 peripheral clock enable register (RCC_AHB1ENR), AHB2 peripheral clock enable register (RCC_AHB2ENR), APB1 peripheral clock enable register (RCC_APB1ENR) and APB2 peripheral clock enable register (RCC_APB2ENR) to switch the clock of each peripheral module.

2.2.3 Sleep mode

Enter sleep mode

Enter the sleep state by executing the WFI or WFE instruction. According to the value of the SLEEPONEXIT bit in the CPU system control register, There are two options for selecting the sleep mode entry mechanism:

- SLEEP-NOW: If the SLEEPONEXIT bit is cleared, when WFI or WFE is executed, the microcontroller immediately goes to sleep Sleep mode.
- SLEEP-ON-EXIT: If the SLEEPONEXIT bit is set, when the system exits from the lowest priority interrupt handler, The microcontroller immediately enters sleep mode.

In sleep mode, all I/O pins maintain their state in run mode.

For more details on how to enter sleep mode, refer to Table 2.

Exit sleep mode

If the WFI instruction is executed to enter the sleep mode, any peripheral interrupt that is responded to by the nested vector interrupt controller can put the system from sleep Mode wake up.

If the WFE instruction is executed to enter the sleep mode, once a wake-up event occurs, the microprocessor will exit from the sleep mode. Wake up event

It can be generated in the following ways:

- ● Enable an interrupt in the peripheral control register, not in the NVIC (Nested Vectored Interrupt Controller), and in the CPU
  The SEVONPEND bit is enabled in the system control register. When the MCU wakes up from WFE, the interrupt pending bit of the peripheral and the peripheral
  The NVIC interrupt channel suspend bit (in the NVIC interrupt clear suspend register) must be cleared.
- ● Configure an external or internal EXIT line as event mode. When the MCU wakes up from the WFE, because of the hangup corresponding to the event line
  The start bit is not set, and there is no need to clear the interrupt pending bit of the peripheral or the NVIC interrupt channel pending bit of the peripheral.

This mode requires the shortest time to wake up, because there is no time lost in the entry or exit of the interrupt.

For more details on how to exit sleep mode, refer to the table below.

Table **2. SLEEP-NOW** mode

SLEEP-NOW mode	illustrate
Enter	Execute WFI (Wait for Interrupt) or WFE (Wait for Event) instructions under the following conditions: -SLEEPDEEP = 0 and -SLEEPONEXIT = 0 Refer to the CPU system control register.
quit	If executing WFI to enter sleep mode: interrupt wake-up, refer to the interrupt vector table If executing WFE to enter sleep mode: wake up from event, refer to wake up event management
Wake-up delay	without

Table **3. SLEEP-ON-EXIT** mode

SLEEP-ON_EXIT mode	illustrate
Enter	Execute WFI (Wait for Interrupt) or WFE (Wait for Event) instructions under the following conditions: -SLEEPDEEP = 0 and -SLEEPONEXIT = 1 Refer to CPU system control register
quit	If executing WFI to enter sleep mode: interrupt wake-up or clear CPU control register bit 1 If executing WFE to enter sleep mode: wake-up event, refer to wake-up event management
Wake-up delay	without

2.2.4 Stop mode

The stop mode is based on the deep sleep mode of the CPU combined with the peripheral clock control mechanism. In the stop mode, the voltage regulator is still working. do. At this time, all clocks in the 1.2V power supply area are stopped, the functions of PLL, HSI and HSE oscillators are disabled, SRAM and register The contents of the memory are retained.

In stop mode, all I/O pins maintain their state in run mode.

Enter stop mode

See Table 4 for details on how to enter the stop mode.

The following functions can be selected by programming independent control bits:

- Independent watchdog (IWDG): IWDG can be started by writing to the watchdog's key register or hardware selection. Once independent Watchdog, except for system reset, it can no longer be stopped
- Real-time clock (RTC): set by the RTCEN bit of the backup domain control register (RCC_BDCR).
- Internal oscillator (LSI oscillator): set by the LSION bit of the control/status register (RCC_CSR).
- External 32.768KHz oscillator (LSE): set by the LSEON bit of the backup domain control register (RCC_BDCR).

In stop mode, if the ADC and DAC are not turned off before entering this mode, these peripherals still consume current. By setting Set the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register to 0 to turn off these two peripherals.

Exit stop mode

See the table below for details on how to exit the stop mode.

When an interrupt or wake-up event causes the stop mode to exit, the HSE oscillator is selected as the system clock.

Table **4.** Stop modes

Stop mode	illustrate
Enter	Execute WFI (Wait for Interrupt) or WFE (Wait for Event) instructions under the following conditions: -Set the SLEEPDEEP bit in the CPU system control register -Clear the PDDS bit in the power control register (PWR_CTRL) Note: In order to enter the stop mode, all external interrupt request bits (pending register (EXTI_PEND)) and RTC alarm flag The log must be cleared, otherwise the process of entering the stop mode will be skipped and the program will continue to run.
quit	The WFI (Wait for Interrupt) instruction is executed under the following conditions: Any external interrupt pin is set to interrupt mode (corresponding external interrupt vector must be enabled in NVIC), see interrupt vector table. The WFE (Wait for Event) instruction is executed under the following conditions: Any external interrupt pin is set to event mode, see wake-up event management. External reset on the NRST pin (reset the entire system).
Wake-up delay	HSE wake-up time.

2.2.5 Automatic wake-up in low power consumption mode (**AWU**)

RTC can wake up the microcontroller in low-power mode without relying on external interrupts (automatic wake-up mode). RTC provides A programmable time base used to periodically wake up from stop mode. Through the backup area control register (RCC_BDCR) For RTCSEL[1:0] bit programming, two of the three RTC clock sources can be selected to implement this function.

- Low power consumption 32.768KHz external crystal oscillator (LSE)
  - This clock source provides a low-power and accurate time reference. (Consumption is less than 1μA under typical conditions)
- Low-power internal oscillator (LSI oscillator)
  - Using this clock source saves the cost of a 32.768KHz crystal oscillator. But the oscillator will slightly increase power consumption.

In order to use the RTC alarm event to wake the system from stop mode, the following operations must be performed:

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

2.3 Power Control Register

2.3.1 Power Control Register (**PWR_CR**)

Address offset: 0x0

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			PLS[2:0]				DBP	Reserve			PVDE	CSBF	CWUF	PDDS	LPDS
			rw	rw	rw	rw	rw				rw	rc_w1 rw		rw	rw

Bit 31: 9	Reserved, always read as 0

Bit 8	**DBP** : Cancel the write protection of the backup area (domain write protection) After reset, the RTC and backup registers are in a protected state to prevent accidental writes. Setting this bit allows writing to these registers. 1 = Allow writing to RTC and backup registers 0 = Disable writing to RTC and backup registers Note: If the RTC clock is HSE/128, this bit must be kept at '1'.
Bit 7: 3	Reserved, always read as 0
Bit 2	**CWUF** : Clear wakeup flag Always read as 0 1 = Clear the WUF wake-up bit after 2 system clock cycles (write) 0 = no effect
Bit 1: 0	Reserve

2.3.2 Power Control / Status Register (**PWR_CSR**)

Address offset: 0x04

Reset value: 0x0000 0000

Compared with the standard APB read, reading the secondary register requires additional APB cycles

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															WUF
															rw

Bit 31:1	Reserve
Bit 0	**WUF** : Wakeup flag This bit is set by hardware, and can only be set by POR/PDR (power-on/power-down reset) or setting the power control register (PWR_CR) The CWUF bit is cleared. 1 = A wake-up event occurred on the WKUP pin or an RTC alarm event occurred 0 = No wake-up event occurred Note: When the WKUP pin is already high, when the WKUP pin is enabled (by setting the EWUP bit), a Extra events.

3. Backup register (**BKP**)

3.1 Introduction to **BKP**

The backup registers are 20 32-bit registers, which can be used to store 80 bytes of user application data, and 4K bytes of backup SRAM
It can also be used to store user data. They are in the backup domain. When the V $_{DD}$ power supply is cut off, they are still powered by V $_{BAT}$. When the system is restored
They will not be reset when the bit or power is reset.

In addition, the BKP control register is used to manage intrusion detection and RTC calibration functions.

After reset, access to the backup register and RTC is prohibited, and the backup domain is protected to prevent possible accidental write operations. Hold on
Perform the following operations to enable access to the backup register and RTC.

- Turn on the power supply and the clock of the backup interface by setting the PWREN and BKPEN bits in the register RCC_APB1ENR
- The DBP bit of the power control register (PWR_CR) is used to enable access to the backup register and RTC.

3.2 BKP features

- 80 bytes data backup register
- 4K bytes of backup SRAM
- Status/control register used to manage anti-intrusion detection and with medium function
- The check register used to store the RTC check value.
- Output RTC calibration clock, RTC alarm pulse or second pulse on PB13 pin (when this pin is not used for intrusion detection)

3.3 BKP function description

3.3.1 Intrusion detection

When the signal on the TAMPER pin changes from 0 to 1 or from 1 to 0 (depending on the TPAL bit of the backup control register BKP_CR),
An intrusion detection event will be generated. The intrusion detection event will clear the contents of all data backup registers.

However, in order to avoid loss of intrusion events, the intrusion detection signal is the logical AND of the edge detection signal and the intrusion detection permission bit, so that the int
Intrusion events that occur before the access detection pin is allowed can also be detected.

- When TPAL = 0: If the TAMPER pin is already high before intrusion detection (by setting the TPE bit),
  Once the intrusion detection function is activated, an additional intrusion event will be generated (although there is no rising edge after the TPE bit is '1').
- When TPAL = 1: If the pin TAMPER is already at low level (by setting the TPE bit) before the intrusion detection pin TAMPER is activated,
  Once the intrusion detection function is activated, an additional intrusion event will be generated (although there is no falling edge after the TPE bit is '1').

Set the TPIE bit of the BKP_CSR register to '1', and an interrupt will be generated when an intrusion event is detected.

After an intrusion event is detected and cleared, the intrusion detection pin TAMPER should be disabled. Then, write the backup number again
Restart the intrusion detection function with the TPE bit before the data register. In this way, you can prevent the software from
The backup data register is written. This is equivalent to level detection on the intrusion pin TAMPER.

Note: When the *V $_{DD}$* power supply is disconnected, the intrusion detection function is still valid. In order to avoid unnecessary resetting of data backup registers, *TAMPER*
The pins should be connected to the correct level off-chip.

3.3.2 RTC calibration

To facilitate measurement, the RTC clock can be divided by 64 and output to the intrusion detection pin TAMPER. By setting the RTC check register
(BKP_RTCCR) CCO bit to enable this function.

By configuring the CAL[6:0] bits, this clock can be slowed down by up to 121ppm.

3.4 **BKP** register description

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) mode

3.4.1 RTC clock calibration register (**BKP_RTCCR**)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bit 15: 8	Reserved, always read as 0.
Bit 9	ASOS: Alarm or second output selection When the ASOE bit is set, the ASOS bit can be used to select the RTC second pulse or alarm clock output on the TAMPER pin Pulse signal. 0: Output RTC alarm pulse 1: Output second pulse Note: This bit can only be cleared by the reset of the backup area.
Bit 8	ASOE: Allow to output alarm or second pulse (Alarm or second output enable) According to the setting of the ASOS bit, this bit allows the RTC alarm or second pulse to be output to the TAMPER pin. The width of the output pulse is one period of the RTC clock. The TAMPER function cannot be turned on when the ASOE bit is set. Note: This bit can only be cleared by the reset of the backup area.
Bit 7	CCO: Calibration clock output 0: no effect 1: This bit is set to 1 to output the RTC clock divided by 64 at the intrusion detection pin. When the CCO position is 1, it must be closed Intrusion detection function to avoid detecting useless intrusion signals. Note: This bit will be cleared when the VDD power supply is disconnected.
Bit 6:0	CAL[6:0]: Calibration value The calibration value indicates how many clock pulses will be skipped in every 220 clock pulses. This can be used to calibrate the RTC Standard, slow down the clock by the ratio of 1000000/(220)ppm. The RTC clock can be slowed down by 0 ~ 121ppm.

3.4.2 Backup control register (**BKP_CR**)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit 15: 2	Reserved, always read as 0.
Bit 1	TPAL: Intrusion detection TAMPER pin active level (TAMPER pin active level) 0: The high level on the TAMPER pin of intrusion detection will clear all data backup registers (if the TPE bit is 1) 1: Intrusion detection low level on the TAMPER pin will clear all data backup registers (if the TPE bit is 1)
Bit 0	TPE: Start intrusion detection TAMPER pin (TAMPER pin enable) 0: Intrusion detection TAMPER pin is used as general IO port 1: Turn on the intrusion detection pin for intrusion detection

Note: It is always safe to set the *TPAL* and *TPE* bits at the same time . However, removing both at the same time will produce a false intrusion event. Therefore, push It is recommended to change the state of the *TPAL* bit only when *TPE* is *0* .

3.4.3 Backup control / status register (**BKP_CSR**)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit 15: 10	Reserved, always read as 0.
Bit 9	TIF: Tamper interrupt flag When an intrusion event is detected and the TPIE bit is 1, this bit is set to '1' by hardware. Clear this flag bit by writing '1' to the CTI bit (same as The interrupt is also cleared at the time). If the TPIE bit is cleared, this bit will also be cleared. 0: No intrusive interrupt 1: Generate an intrusion interrupt Note: This bit is reset only when the system is reset.
Bit 8	TEF: Tamper event flag This bit is set by hardware when an intrusion event is detected. This flag bit can be cleared by writing '1' to the CTE bit. 0: No intrusion 1: Intrusion event detected Note: An intrusion event will reset all BKP_DRx registers. As long as TEF is 1, all BKP_DRx registers will be kept Hold the reset state. When this bit is set to '1', if BKP_DRx is written, the written value will not be saved.
Bit 7: 3	Reserved, always read as 0.
Bit 2	TPIE: Allow intrusion into TAMPER pin interrupt (TAMPER pin interrupt enable) 0: Disable intrusion detection interrupt 1: Enable intrusion detection interrupt (TPE bit in BKP_CR register must also be set to '1') Note 1: Intrusive interrupts cannot wake up the system core from low-power mode. Note 2: This bit is reset only when the system is reset.
Bit 1	CTI: Clear tamper interrupt This bit can only be written, and the read value is 0. 0: invalid 1: Clear the intrusion detection interrupt and TIF intrusion detection interrupt flag
Bit 0	CTE: Clear tamper event This bit can only be written, and the read value is 0. 0: invalid 1: Clear the TEF intrusion detection event flag (and reset the intrusion detector)

3.4.4 Backup data register **x** (**BKP_DRx**) (**x = 1… 20**)

Address offset: 0x38 ~ 0x84

Reset value: 0x0000 0000

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

Bit 31:0	BKP[31:0]: backup data These bits can be used to write user data. Note: The BKP_DRx register will not be reset by system reset or power reset. They can be reset by the backup domain reset or (If the tamper detection pin TAMPER function is enabled) reset by the intrusion pin event.

4. Reset and clock control (**RCC**)

4.1 Reset

It supports three types of reset, namely system reset, power-on reset and backup area reset.

4.1.1 System reset

The system reset will reset all registers except the reset flag in the clock control register CSR and the registers in the backup area.

When one of the following events occurs, a system reset occurs:

1. Low level on the NRST pin (external reset)

2. Window watchdog counting is terminated (WWDG reset)

3. Independent watchdog count termination (IWDG reset)

4. Software reset (SW reset)

The source of the reset event can be identified by looking at the reset status flag bit in the RCC_CSR control status register.

Software reset

The software reset can be realized by setting the SYSRESETREQ bit in the CPU interrupt application and reset control register to '1'.

4.1.2 Power reset

When the following events occur, a power reset occurs:

1. Power-on **/** power-down reset (**POR/PDR** reset)

Power reset will reset all registers except the backup area.

The reset source in the figure will eventually act on the RESET pin and will remain low during the reset process. The reset entry vector is fixed at the address 0x0000_0004.

Figure **9.** Reset circuit

4.1.3 Backup domain reset

The backup area reset can be generated by setting the BDRST bit in the backup area control register RCC_BDCR, which will register all RTCs
The register and RCC_BDCR register are reset to their respective reset values.

After the power supply VDD and VBAT are both powered off, any one of them is powered on again, and the backup domain will also be reset.

Note: After VBAT is powered on and before BDRST is reset, the backup area is in an unreset state. You need to set the BDRST bit to reset the backup area. Similarly, BKPSRAM also needs to set BKPSRAMRST to reset.

4.2 Clock

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

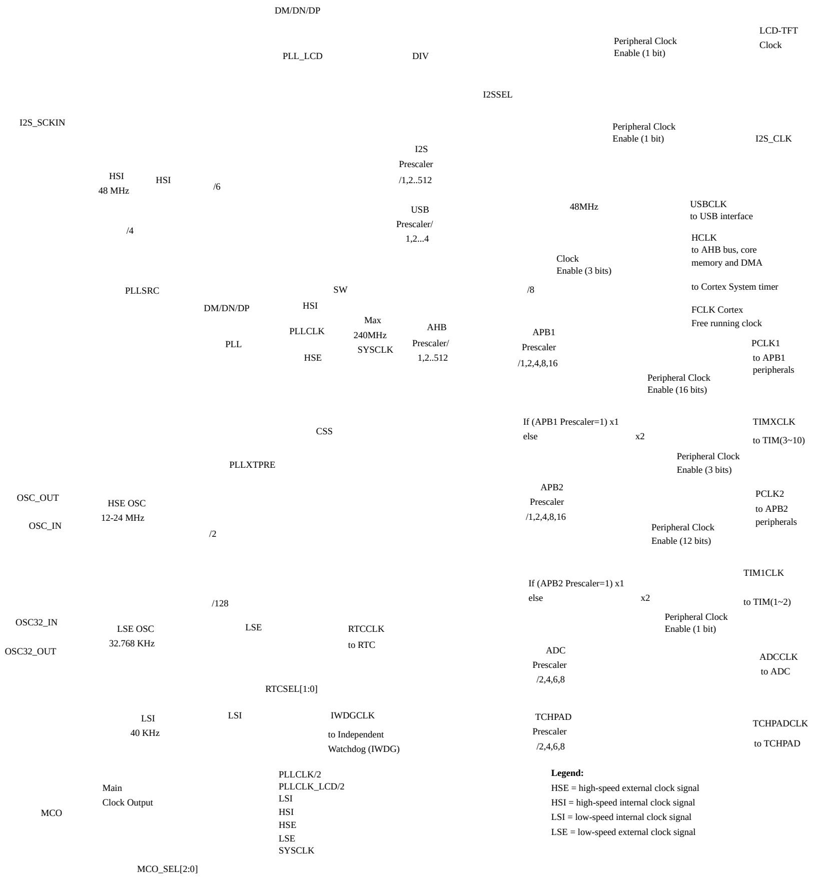
- HSI oscillator clock
- HSE oscillator clock
- PLL clock

These devices have the following two secondary clock sources:

- The 40KHz low-speed internal oscillator can be used to drive independent watchdogs and drive RTC through program selection. RTC is used from downtime Automatically wake up the system.
- 32.768KHz low-speed external crystal can also be used to drive RTC (RTCCLK) by program selection.

When not in use, any clock source can be turned on or off independently, thereby optimizing system power consumption.

Figure **10.** Clock tree


DM/DN/DP
LCD-TFT Clock
Peripheral Clock Enable (1 bit)
PLL_LCD
DIV
I2SSEL
I2S_SCKIN
Peripheral Clock Enable (1 bit)
I2S_CLK
I2S Prescaler /1,2..512
HSI 48 MHz
HSI
/6
USB Prescaler/ 1,2...4
48MHz
USBCLK to USB interface
/4
HCLK to AHB bus, core memory and DMA
Clock Enable (3 bits)
PLLSRC
SW
/8
to Cortex System timer
DM/DN/DP
HSI
FCLK Cortex Free running clock
Max 240MHz SYSCLK
PLLCLK
AHB Prescaler/ 1,2..512
APB1 Prescaler /1,2,4,8,16
PCLK1 to APB1 peripherals
PLL
HSE
Peripheral Clock Enable (16 bits)
If (APB1 Prescaler=1) x1 else x2
TIMXCLK to TIM(3~10)
CSS
Peripheral Clock Enable (3 bits)
PLLXTPRE
APB2 Prescaler /1,2,4,8,16
OSC_OUT
HSE OSC 12-24 MHz
OSC_IN
PCLK2 to APB2 peripherals
Peripheral Clock Enable (12 bits)
/2
TIM1CLK
If (APB2 Prescaler=1) x1 else x2
/128
to TIM(1~2)
Peripheral Clock Enable (1 bit)
OSC32_IN
LSE OSC 32.768 KHz
LSE
RTCCLK to RTC
OSC32_OUT
ADC Prescaler /2,4,6,8
ADCCLK to ADC
RTCSEL[1:0]
LSI 40 KHz
LSI
IWDGCLK to Independent Watchdog (IWDG)
TCHPAD Prescaler /2,4,6,8
TCHPADCLK to TCHPAD
PLLCLK/2
PLLCLK_LCD/2
LSI
HSI
HSE
LSE
SYSCLK
Main Clock Output
MCO
MCO_SEL[2:0]
Legend:
HSE = high-speed external clock signal
HSI = high-speed internal clock signal
LSI = low-speed internal clock signal
LSE = low-speed external clock signal

Users can configure the frequency of the AHB, high-speed APB (APB2) and low-speed APB (APB1) domains through multiple prescalers. AHB domain
The maximum frequency is 240MHz, and the maximum frequency of the APB1 and APB2 domains is 120MHz.

RCC is divided by 8 through the AHB clock and then supplied to the CPU system timer (SysTick) external clock. Through the control of SysTick and

To set the status register, the above clock or AHB clock can be selected as the SysTick clock. The ADC clock is divided by the high-speed APB2 clock through 2, 4, Obtained after dividing by 6 or 8.

The timer clock frequency distribution is automatically set by the hardware according to the following two conditions:

1. If the corresponding APB prescaler coefficient is 1, the clock frequency of the timer is the same as the frequency of the APB bus where it is located.

2. Otherwise, the clock frequency of the timer is set to twice the frequency of the APB bus connected to it.

FCLK is the free running clock of the CPU.

4.2.1 **HSE** clock

The high-speed external clock signal (HSE) is generated by the following two clock sources:

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

In order to reduce the distortion of the clock output and shorten the start-up stabilization time, the crystal/ceramic resonator and load capacitor must be as close as possible to the oscillati 器Pin. The load capacitance value must be adjusted according to the selected oscillator.

Figure **11. HSE/LSE** clock source

External clock source (**HSE** bypass)

In this mode, an external clock must be provided. Its frequency can reach up to 24MHz. The user can set it in the clock control register The HSEBYP and HSEON bits are used to select this mode. The external clock signal (50% duty cycle square wave, sine wave or triangle wave) must be connected To the OSC_IN pin, at the same time ensure that the OSC_OUT pin is floating.

External crystal **/** ceramic resonator (**HSE** crystal)

The 12 ~ 24MHz external oscillator can provide a more accurate main clock for the system. Related hardware configuration can refer to Figure 11 for further information Refer to the electrical characteristics section of the data sheet.

The HSERDY bit in the clock control register RCC_CR is used to indicate whether the high-speed external oscillator is stable. At startup, until this A bit is set to '1' by hardware, and the clock is released. If the interrupt is enabled in the clock interrupt register RCC_CIR, the corresponding Should be interrupted.

The HSE crystal can be turned on and off by setting the HSEON bit in RCC_CR in the clock control register.

4.2.2 **HSI** clock

The HSI clock signal is generated by the internal 48MHz oscillator, which can be divided by 6 as the system clock or 4 as the PLL input. The HSI oscillator can provide the system clock without any external devices. Its startup time is shorter than HSE crystal oscillator.

The HSIRDY bit in the clock control register is used to indicate whether the HSI oscillator is stable. In the clock startup process, until this bit is hard

When the device is set to '1', the HSI oscillator output clock will be released. The HSI oscillator can be turned on and off by the HSION bit in the clock control register.

If the HSE crystal oscillator fails, the HSI clock will be used as a backup clock source. Refer to 7.2.7 Clock Security System (CSS).

4.2.3 PLL

The internal PLL can be used to multiply the output clock of the HSI oscillator or the output clock of the HSE crystal. Refer to Figure 10 and the clock control register.
The PLL setting (selecting the HSI oscillator or HSE oscillator as the input clock of the PLL, and selecting the multiplication factor) must be completed before it is activated
become. Once the PLL is activated, these parameters cannot be changed.

If the PLL interrupt is enabled in the clock interrupt register, an interrupt request can be generated when the PLL is ready. If needed in the application
To use the USB interface, the PLL must be set to output a 48 or 96 MHz clock to provide a 48 MHz USBCLK clock.

4.2.4 **LSE** clock

The LSE crystal is a 32.768KHz low-speed external crystal or ceramic resonator. It provides a real-time clock or other timing functions
A low-power and accurate clock source.

The LSE crystal is turned on and off by the LSEON bit in the backup domain control register (RCC_BDCR). Control register in the backup domain
The LSERDY in (RCC_BDCR) indicates whether the LSE crystal oscillation is stable. In the startup phase, until this bit is set to '1' by the hardware,
The LSE clock signal is released. If it is enabled in the clock interrupt register, an interrupt request can be generated.

External clock source (**LSE** bypass)

In this mode, an external clock source with a frequency of 32.768KHz must be provided. You can control the register by setting in the backup domain
(RCC_BDCR) LSEBYP and LSEON bits to select this mode. External clock signal with 50% duty cycle (square wave,
Sine wave or triangle wave) must be connected to the OSC32_IN pin, while ensuring that the OSC32_OUT pin is floating.

4.2.5 **LSI** clock

The LSI oscillator acts as a low-power clock source. It can keep running in shutdown mode and is an independent watchdog and automatic wake-up unit.
Yuan provides the clock. The LSI clock frequency is about 40KHz (between 26KHz and 52KHz).

The LSI oscillator can be turned on or off by the LSION bit in the control/status register (RCC_CSR). In the control/status register
The LSIRDY bit in (RCC_CSR) indicates whether the low-speed internal oscillator is stable. In the startup phase, until this bit is set to '1' by the hardware
Later, the clock was released. If it is enabled in the clock interrupt register (RCC_CIR), an LSI interrupt request will be generated.

4.2.6 System clock (**SYSCLK**) selection

After the system is reset, the HSI oscillator is selected as the system clock. When the clock source is used as the system clock directly or indirectly through the PLL, it will not
Can be stopped.

Only when the target clock source is ready (after the delay of the start-up stabilization phase or the PLL is stabilized), from one clock source to another clock
The source switch will only happen. When the selected clock source is not ready, the system clock switching will not occur. Not send until the target clock source is ready
Health switch.

The status bit in the clock control register (RCC_CR) indicates which clock is ready and which clock is currently used as the system
Bell.

4.2.7 Clock Security System (**CSS**)

The clock security system can be activated via software. Once it is activated, the clock monitor will be enabled after the HSE oscillator start-up delay,
And turn off after the HSE clock is turned off.

If the HSE clock fails, the HSE oscillator is automatically shut down, and the clock failure event will be sent to the brake output of the advanced timer TIM1.
Into the terminal, and generate a clock safety interrupt CSSI, allowing the software to complete the rescue operation. This CSSI interrupt is connected to the NMI interrupt of the CPU.

Note: Once the CSS is activated and the HSE clock fails, the CSS interrupt will be generated and the NMI will also be generated automatically. NMI will be
Continue to execute until the CSS interrupt pending bit is cleared. Therefore, the clock interrupt register must be set in the NMI processing program
(RCC_CIR) in the CSSC bit to clear the CSS interrupt.

If the HSE oscillator is directly or indirectly used as the system clock, (indirectly means: it is used as the PLL input clock, and the PLL
The clock is used as the system clock). A clock failure will cause the system clock to automatically switch to the HSI oscillator, and the external HSE oscillator will be shut down at the same t
When the clock fails, if the HSE oscillator clock (divided or undivided) is the input clock of the PLL used as the system clock, the PLL will also
is closed.

4.2.8 **RTC** clock

By setting the RTCSEL[1:0] bits in the backup domain control register (RCC_BDCR), the RTCCLK clock source can be set from HSE/128,
LSE or LSI clock is provided. Unless the backup domain is reset, this selection cannot be changed.

The LSE clock is in the backup domain, but the HSE and LSI clocks are not. therefore:

- If LSE is selected as the RTC clock:
  - As long as VBAT maintains power supply, RTC continues to work even though VDD power supply is cut off.
  - If LSI is selected as the automatic wake-up unit (AWU) clock: see 7.2.5 LSI clock for details.
  - If the VDD power supply is cut off, the AWU status cannot be guaranteed
  - If the HSE clock is divided by 128 and used as the RTC clock:
  - If the VDD power supply is cut off or the 1.2V domain power supply is cut off, the RTC status is uncertain.

- The DBP bit (to cancel the write protection of the backup area) of the power control register must be set to '1'.

4.2.9 Watchdog clock

If the independent watchdog has been enabled by hardware options or software, the LSI oscillator will be forced to be on and cannot be turned off. exist After the LSI oscillator stabilizes, the clock is supplied to the IWDG.

4.2.10 Clock output

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

The corresponding GPIO port register must be configured for the corresponding function. The following four clock signals can be selected as the MCO clock:

- SYSCLK
- HSI
- HSE
- LSI
- LSE
- PLL divided by 2 clock
- PLL_LCDCLK divided by 2 clock
  - The clock selection is controlled by the MCO_SEL[2:0] bits in the clock configuration register (RCC_CFGR).

4.3 RCC register description

4.3.1 Clock Control Register (**RCC_CR**)

Address offset: 0x00

Reset value: 0x0003 XX03, X means undefined

Access: No wait state, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve		PLL_ LCD RDY	PLL_ LCD ON	Reserve		PLL RDY	PLL ON	Reserve				CSS ON	HSE BYP	HSE RDY	HSE ON
		r	rw			r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve														HSI RDY	HSI ON
														r	rw

Bit 31: 30	Reserve
Bit 29	**PLL_LCDRDY** : PLL_LCD clock ready flag (PLL_LCD clock ready flag) After the PLL is locked, it is set to '1' by hardware. 0: PLL is not locked 1: PLL locked
Bit 28	**PLL_LCDON:** PLL_LCD enable (PLL_LCD enable) Set or cleared by software. When entering stop mode, this bit is cleared by hardware. 0: PLL_LCD is off 1: PLL_LCD is enabled
Bit 27: 26	Reserve
Bit 25	**PLLRDY** : PLL clock ready flag (PLL clock ready flag) After the PLL is locked, it is set to '1' by hardware. 0: PLL is not locked 1: PLL locked
Bit 24	**PLLON** : PLL enable (PLL enable) Set or cleared by software. When entering stop mode, this bit is cleared by hardware. When the PLL clock is used or selected to be the system clock, this bit cannot It is cleared. 0: PLL is off 1: PLL is enabled
Bit 23: 20	Reserved, always read as 0
Bit 19	**CCSON** : Clock security system enable (Clock security system enable) Set or cleared by software to enable the clock monitor 0: The clock monitor is off 1: If the HSE oscillator is ready, the clock monitor is turned on
Bit 18	**HSEBYP** : External high-speed clock bypass In the debug mode, it is set or cleared by software to bypass the external crystal oscillator. Write only when the external oscillator is turned off This bit 0: The external oscillator is not bypassed 1: The external external crystal oscillator is bypassed
Bit 17	**HSERDY** : External high-speed clock ready flag (External high-speed clock ready flag) Set by hardware to indicate that the HSE clock has stabilized. 0: The external clock is not ready

	1: External clock is ready
Bit 16	**HSEON** : External high-speed clock enable (External high-speed clock enable) Set or cleared by software. When entering the stop mode, this bit is cleared by hardware to turn off the external clock. When the HSE clock is used or selected as the system When clocking, this bit cannot be cleared. 0: HSE oscillator is turned off 1: HSE oscillator is turned on
Bit 15: 2	Reserve
Bit 1	**HSIRDY** : Internal high-speed clock ready flag (Internal high-speed clock ready flag) The hardware is set to '1' to indicate that the HSI clock has stabilized. 0: HSI clock is not ready 1: HSI clock is ready
Bit 0	**HSION** : Internal high-speed clock enable (Internal high-speed clock enable) Set or cleared by software. When returning from stop mode or when the external clock used as the system clock fails, this bit is set to '1' by hardware to start the HSI oscillator. when When the HSI clock is used directly or indirectly or is selected as the system clock, this bit cannot be cleared. 0: HSI oscillator is turned off 1: HSI oscillator is turned on

4.3.2 PLL configuration register (RCC_PLLCFGR)

Offset address: 0x04

Reset value: 0x0000 0000

Access: No waiting period, access by word, half word and byte.

This register is used to configure the PLL clock output according to the formula:

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve									PLL SRC	Reserve					
									rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL BY PASS	PLLIS		PLLDN							PLLDP		PLLDM			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 23	Reserve
Bit 22	**PLLSRC** : PLL entry clock source Set '1' or clear '0' by software to select PLL input clock source. This bit can be written only when the PLL is turned off. 0: HSI clock divided by 4 as PLL input clock 1: HSE clock or divide-by-2 clock as PLL input clock
Bit 21: 13	Reserve
Bit 15	**PLLBYPASS** : PLL bypass control (PLL bypass control) 0: Do not bypass PLL 1: Bypass PLL
Bit 14: 13	**PLLIS** : PLL current control 00: When the DN is 1-20, the configuration is 00 01: When the DN is 21-40, the configuration is 01 10: When the DN is 41-60, the configuration is 10 11: When the DN is above 61, the configuration is 11
Bit 12: 6	**PLLDN** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients.
Bit 5: 4	**PLLDP** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients. 00: P=1 01: P=2 10: P=4 11: P=8
Bit 3: 0	**PLLDM** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients. PLL configuration formula: FCLKO = FREFIN * N / (M*P) FCLKO is the PLL output frequency, FREFIN is the PLL input reference clock frequency N = PLLDN[6:0] + 1 M = PLLDM[3:0] + 1

4.3.3 Clock configuration register (**RCC_CFGR**)

Address offset: 0x08

Reset value: 0x0000 0005

Access: No wait state, word, half word and byte access

Only when the access occurs during a clock switch, will 1 or 2 wait cycles be inserted.

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
I2S_PRE								I2S SEL	MCO_SEL			USBPRE			PLLX TPRE
								rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2			PPRE1			Reserve		HPRE				SWS		SW	
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

Bit 31: 24	**I2SPRE** : I2S prescaler (I2S prescaler) Set to '1' or clear to '0' by software to control the prescaler coefficient of the I2S clock. The prescaler factor is (I2SPRE+1). 00000000: 1 frequency division 00000001: divide by 2 … 11111110: 255 frequency division 11111111: 256 frequency division
Bit 23	**I2SSEL** : I2S clock source selection (I2S clock selection) Set or cleared by software. This bit selects PLLCLK or I2S_SCKIN from the outside as its clock source. 0: PLLCLK is used as I2S clock source 1: External I2S_SCKIN is used as I2S clock source
Bit 22: 20	**MCO_SEL** : Microcontroller clock output select Set or cleared by software. 001: LSI clock output; 010: LSE clock output; 011: System clock (SYSCLK) output; 100: HSI clock output; 101: HSE clock output; 110: PLLCLK clock divided by 2 and output. 111: PLL_LCDCLK clock is divided by 2 and output. Notice: -The clock output may be cut off when starting and switching the MCO clock source. -When the system clock is output to the MCO pin, please ensure that the output clock frequency does not exceed 50MHz (the highest frequency of the IO port)
Bit 19: 17	**USBPRE** : USB prescaler Set '1' or clear '0' by software to generate 48MHz USB clock. Before enabling the USB clock in the RCC_APB1ENR register, It must be ensured that this bit is already valid. 000: PLL clock is directly used as USB clock 001: PLL clock divided by 2 as USB clock … 110: PLL clock divided by 7 as USB clock 111: PLL clock divided by 8 as USB clock
Bit 16	**PLLXTPRE** : HSE divider for PLL entry (HSE divider for PLL entry) Set to '1' or cleared to '0' by software to divide the HSE and use it as the PLL input clock. This bit can be written only when the PLL is turned off. 0: HSE does not divide frequency 1: HSE2 frequency division

Bit 15: 13	**PPRE2** : High-speed APB prescaler (APB2) (APB high-speed prescaler (APB2)) Set to '1' or clear to '0' by software to control the prescaler coefficient of the high-speed APB2 clock (PCLK2). Note: The software must ensure that the APB1 clock frequency does not exceed 120MHz. 0xx: HCLK without frequency division 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
Bit 12: 10	**PPRE1** : Low-speed APB prescaler (APB1) (APB low-speed prescaler (APB1)) Set to '1' or clear to '0' by software to control the prescaler coefficient of the low-speed APB1 clock (PCLK1). Note: The software must ensure that the APB1 clock frequency does not exceed 60MHz. 0xx: HCLK without frequency division 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
Bit 9: 8	Reserve
Bit 7: 4	**HPRE** : AHB Prescaler Set to '1' or clear to '0' by software to control the prescaler coefficient of the AHB clock. 0xxx: SYSCLK is not divided 1000: SYSCLK divided by 2 1100: SYSCLK divided by 64 1001: SYSCLK divided by 4 1101: SYSCLK divided by 128 1010: SYSCLK divided by 8 1110: SYSCLK divided by 256 1011: SYSCLK divided by 16 1111: SYSCLK divided by 512
Bit 3: 2	**SWS** : System clock switch status (System clock switch status) The hardware is set to '1' or cleared to '0' to indicate which clock source is used as the system clock. 00: HSI divided by 6 as the system clock; 01: HSE is used as the system clock; 10: PLL output is used as the system clock; 11: Not available.
Bit 1: 0	**SW** : System clock switch Set '1' or clear '0' by software to select the system clock source. When returning from the stop mode or when the HSE directly or indirectly used as the system clock fails, the selection is forced by hardware HSI as the system clock (if the clock security system has been activated) 00: HSI divided by 6 as the system clock; 01: HSE is used as the system clock; 10: PLL output is used as the system clock; 11: Not available.

4.3.4 Clock interrupt register (**RCC_CIR**)

Offset address: 0x0C

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve								CSSC	PLL LCD RDYC	Reserve	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w	w		w	w	w	w	w
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve	PLL LCD RDYIE	Reserve	PLL RDY IE	HSE RDY IE	HSI RDY IE	LSE RDY IE	LSI RDY IE	CSSF	PLL LCD RDYF	Reserve	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw		rw	rw	rw	rw	rw	r	r		r	r	r	r	r

Bit 31: 24	Reserved, always read as 0
Bit 23	**CSSC** : Clear the clock security system interrupt (Clock security system interrupt clear) Set '1' by software to clear the CSSF safety system interrupt flag bit CSSF. 0: No effect 1: Clear the CSSF safety system interrupt flag bit
Bit 22	**PLL_LCDRDYC** : 0: No effect 1: Clear the PLL_LCD ready interrupt flag bit PLL_LCDRDY
Bit 21	Reserve
Bit 20	**PLLRDYC** : Clear PLL ready interrupt (PLL ready interrupt clear) Set '1' by software to clear the PLL ready interrupt flag bit PLLRDYF. 0: No effect 1: Clear the PLL ready interrupt flag bit PLLRDYF
	HSERDYC : Clear HSE ready interrupt (HSE ready interrupt clear) Set '1' by software to clear the HSE ready interrupt flag bit HSERDYF.

Bit 19	0: No effect 1: Clear the HSE ready interrupt flag bit HSERDYF
Bit 18	**HSIRDYC** : Clear HSI ready interrupt (HSI ready interrupt clear) Set '1' by software to clear the HSI ready interrupt flag HSIRDYF. 0: No effect 1: Clear the HSI ready interrupt flag HSIRDYF
Bit 17	**LSERDYC** : Clear LSE ready interrupt (LSE ready interrupt clear) Set '1' by software to clear the LSE ready interrupt flag bit LSERDYF. 0: No effect 1: Clear the LSE ready interrupt flag bit LSERDYF
Bit 16	**LSIRDYC** : Clear LSI ready interrupt (LSI ready interrupt clear) Set '1' by software to clear the LSI ready interrupt flag bit LSIRDYF. 0: No effect 1: Clear the LSI ready interrupt flag bit LSIRDYF
Bit 15	Reserved, always read as 0
Bit 14	**PLL_LCDRDYIE** : PLL_LCD ready interrupt enable (PLL_LCD ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the PLL_LCD ready interrupt. 0: PLL_LCD ready interrupt is off 1: PLL_LCD ready interrupt enable
Bit 13	Reserve
Bit 12	**PLLRDYIE** : PLL ready interrupt enable (PLL ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the PLL ready interrupt. 0: PLL ready interrupt is turned off 1: PLL ready interrupt enable
Bit 11	**HSERDYIE** : HSE ready interrupt enable (HSE ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the external 8 ~ 24MHz oscillator ready interrupt. 0: HSE ready interrupt is off 1: HSE ready interrupt enable
Bit 10	**HSIRDYIE** : HSI ready interrupt enable (HSI ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the internal 8MHz oscillator ready interrupt. 0: HSI ready interrupt is off 1: HSI ready interrupt enable
Bit 9	**LSERDYIE** : LSE ready interrupt enable (LSE ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the external 32.768KHz oscillator ready interrupt. 0: LSE ready interrupt is closed 1: LSE ready interrupt enable
Bit 8	**LSIRDYIE** : LSI ready interrupt enable (LSI ready interrupt enable) Set to '1' or clear to '0' by software to enable or disable the internal 40KHz oscillator ready interrupt. 0: LSI ready interrupt is off 1: LSI ready interrupt enable

Bit 7	**CSSF** : Clock security system interrupt flag (Clock security system interrupt flag) When the external 8 ~ 24MHz oscillator clock fails, it will be set to '1' by hardware. It is cleared by software by setting the CSSC bit '1'. 0: No safety system interrupt due to HSE clock failure 1: HSE clock failure caused the interruption of the clock security system
Bit 6	**PLL_LCDRDYF** : PLL_LCD ready interrupt flag (PLL_LCD ready interrupt flag) When PLL_LCD is ready and the PLL_LCDRDYIE bit is set to '1', it is set to '1' by hardware. It is cleared by software by setting the PLL_LCDRDYC bit to '1'. 0: No clock ready interrupt generated by PLL_LCD lock 1: PLL_LCD lock causes clock ready interrupt
Bit 5	Reserve
Bit 4	**PLLRDYF** : PLL ready interrupt flag (PLL ready interrupt flag) When the PLL is ready and the PLLRDYIE bit is set to '1', it is set to '1' by hardware. It is cleared by software by setting the PLLRDYC bit '1'. 0: No clock ready interrupt generated by PLL lock 1: PLL lock causes clock ready interrupt
Bit 3	**HSERDYF** : HSE ready interrupt flag (HSE ready interrupt flag) When the external low-speed clock is ready and the HSERDYIE bit is set to '1', it is set to '1' by hardware. It is cleared by software by setting the HSERDYC bit '1'. 0: No clock ready interrupt generated by external 8 ~ 24MHz oscillator 1: External 8 ~ 24MHz oscillator causes clock ready interrupt
Bit 2	**HSIRDYF** : HSI ready interrupt flag (HSI ready interrupt flag) When the internal high-speed clock is ready and the HSIRDYIE bit is set to '1', it is set to '1' by hardware. It is cleared by software by setting the HSIRDYC bit '1'. 0: No clock ready interrupt generated by the internal 8MHz oscillator 1: The internal 8MHz oscillator causes a clock ready interrupt
Bit 1	**LSERDYF** : LSE ready interrupt flag (LSE ready interrupt flag) When the external low-speed clock is ready and the LSERDYIE bit is set to '1', it is set to '1' by hardware. It is cleared by software by setting the '1' LSERDYC bit. 0: No clock ready interrupt generated by external 32.768KHz oscillator; 1: The external 32.768KHz oscillator causes a clock ready interrupt.
	LSIRDYF : LSI ready interrupt flag (LSI ready interrupt flag) When the internal low-speed clock is ready and the LSIRDYIE bit is set to '1', it is set to '1' by hardware.

Bit 0	It is cleared by software by setting '1' LSIRDYC bit. 0: No clock ready interrupt generated by the internal 40KHz oscillator; 1: The internal 40KHz oscillator causes a clock ready interrupt.

4.3.5 **AHB1** peripheral reset register (**RCC_AHB1RSTR**)

Offset address: 0x10

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
LCDR ST	Reserve								DMA2 RST	DMA1 RST	Reserve				
rw									rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve			CRCR ST	Reserve							GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
			rw								rw	rw	rw	rw	rw

Page 46

Bit 31	**LCDRST** : LCD-TFT reset (LCD-TFT reset) Set to '1' or clear to '0' by software. 0: No effect 1: LCD-TFT reset
Bit 30:23	Reserve
Bit 22	**DMA2RST** : DMA2 reset (DMA2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: DMA2 reset
Bit 21	**DMA1RST** : DMA1 reset (DMA1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: DMA1 reset
Bit 20: 13	Reserve
Bit 12	**CRCRST** : CRC reset (CRC reset) Set to '1' or clear to '0' by software. 0: No effect 1: CRC reset
Bit 11:5	Reserve
Bit 4	**GPIOERST** : IO port E reset Set to '1' or clear to '0' by software. 0: No effect 1: Port E reset
Bit 3	**GPIODRST** : IO port D reset Set to '1' or clear to '0' by software. 0: No effect 1: Port D reset
Bit 2	**GPIOCRST** : IO port C reset Set to '1' or clear to '0' by software. 0: No effect 1: Port C reset
Bit 1	**GPIOBRST** : IO port B reset Set to '1' or clear to '0' by software. 0: No effect 1: Port B reset
Bit 0	**GPIOARST** : IO port A reset Set to '1' or clear to '0' by software. 0: No effect 1: Port A reset

4.3.6 **AHB2** peripheral reset register (**RCC_AHB2RSTR**)

Offset address: 0x14

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

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

TK80 RST	Reserve														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															

Bit 31	**TK80RST** : TK80 interface reset Set to '1' or clear to '0' by software. 0: No effect 1: TK80 reset
Bit 30:0	Reserve

4.3.7 APB1 peripheral reset register (**RCC_APB1RSTR**)

Offset address: 0x18

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve	I2S1 RST	PWR RST	USB RST	CAN2 RST	CAN1 RST	Reserved		I2C3 RST	I2C2 RST	I2C1 RST	Reserve				
	rw	rw	rw	rw	rw			rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve				WWD GRST	Reserve	BKPR ST	Reserve	TIM10 RST	TIM9 RST	TIM8 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST
				rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31	Reserve
Bit 30	**I2S1RST** : I2S1 interface reset (I2S1 interface reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset I2S1 interface
Bit 29	**PWRRST** : Power interface reset Set to '1' or clear to '0' by software. 0: No effect 1: Reset power interface
Bit 28	**USBRST** : USB reset (USB reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset USB
Bit 27	**CAN2RST** : CAN2 reset (CAN2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset CAN2
Bit 26	**CAN1RST** : CAN1 reset (CAN1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset CAN1
Bit 25:24	Reserve
Bit 23	**I2C3RST** : I2C3 reset (I2C3 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset I2C3
Bit 22	**I2C2RST** : I2C2 reset (I2C2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset I2C2

Bit 21	**I2C1RST** : I2C1 reset (I2C1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset I2C1
Bit 20: 12	Reserve
Bit 11	**WWDGRST** : Window watchdog reset Set to '1' or clear to '0' by software. 0: No effect 1: Reset window watchdog
Bit 10	Reserved, always read as 0.
Bit 9	**BKPRST** : Digital Domain Backup interface reset under the power domain (Digital Domain Backup interface reset) Set to '1' or cleared to '0' by software 0: No effect; 1: Reset the backup interface.
Bit 8	Reserve
Bit 7	**TIM10RST** : Timer 10 reset (Timer10 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM10 timer
Bit 6	**TIM9RST** : Timer 9 reset (Timer9 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM9 timer
Bit 5	**TIM8RST** : Timer 8 reset (Timer8 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM8 timer
Bit 4	**TIM7RST** : Timer7 reset (Timer7 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM7 timer
Bit 3	**TIM6RST** : Timer6 reset (Timer6 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM6 timer
Bit 2	**TIM5RST** : Timer5 reset (Timer5 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM5 timer
Bit 1	**TIM4RST** : Timer 4 reset (Timer4 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM4 timer
Bit 0	**TIM3RST** : Timer3 reset (Timer3 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM3 timer

Page 49

4.3.8 **APB2** peripheral reset register (**RCC_APB2RSTR**)

Offset address: 0x1C

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve						TCH PAD RST	QSPI1 RST	SPI4 RST	SPI3 RST	SPI2 RST	SPI1 RST	Reserve			
						rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserve	SYSC FG RST	Reserved	SDIO2	SDIO1	reserved	reserved	ADC1 RST	Reserve	UART 5RST	UART 4RST	UART 3RST	UART 2RST	UART 1RST	TIM2 RST	TIM1 RST
	rw		rw	rw			rw		rw	rw	rw	rw	rw	rw	rw

Bit 31: 26	Reserve
Bit 25	**TCHPADRST** : TCHPAD reset (TCHPAD reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset TCHPAD
Bit 24	**QSPI1RST** : QSPI reset (QSPI reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset QSPI
Bit 23	**SPI4RST** : SPI4 reset (SPI4 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SPI4
Bit 22	**SPI3RST** : SPI3 reset (SPI3 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SPI3
Bit 21	**SPI2RST** : SPI2 reset (SPI2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SPI2
Bit 20	**SPI1RST** : SPI1 reset (SPI1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SPI1
Bit 19: 15	Reserve
Bit 14	**SYSCFGRST** : System Configuration Controller reset Set to '1' or clear to '0' by software. 0: No effect 1: Reset System Configuration Controller
Bit 13	Reserve
Bit 12	**SDIO2** : SDIO2 reset (SDIO2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SDIO2

Bit 11	**SDIO1** : SDIO1 reset (SDIO1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset SDIO1
Bit 10: 9	Reserve
Bit 8	**ADC1RST** : ADC1 reset (ADC1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset ADC1
Bit 7	Reserve
Bit 6	**UART5RST** : UART5 reset (UART5 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset UART5
Bit 5	**UART4RST** : UART4 reset (UART4 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset UART4
Bit 4	**UART3RST** : UART3 reset (UART3 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset UART3
Bit 3	**UART2RST** : UART2 reset (UART2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset UART2
Bit 2	**UART1RST** : UART1 reset (UART1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset UART1

Bit 1	**TIM2RST** : Timer2 reset (Timer2 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM2 timer
Bit 0	**TIM1RST** : Timer1 reset (Timer1 reset) Set to '1' or clear to '0' by software. 0: No effect 1: Reset the TIM1 timer

4.3.9 AHB1 peripheral clock enable register (**RCC_AHB1ENR**)

Offset address: 0x20

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
LCD EN	Reserve								DMA2 EN	DMA1 EN	Reserve				
rw									rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve		BKP SRAM EN	CRC EN	Reserve							GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw	rw								rw	rw	rw	rw	rw

Bit 31	**LCDEN** : LCD-TFT clock enable (LCD-TFT clock enable) Set to '1' or clear to '0' by software. 0: LCD-TFT clock is off 1: LCD-TFT clock enable
Bit 30:23	Reserve
Bit 22	**DMA2EN** : DMA2 clock enable (DMA2 clock enable) Set to '1' or clear to '0' by software. 0: DMA2 clock is off 1: DMA2 clock enable
Bit 21	**DMA1EN** : DMA1 clock enable (DMA1 clock enable) Set to '1' or clear to '0' by software. 0: DMA1 clock is off 1: DMA1 clock enable
Bit 20: 14	Reserve
Bit 13	**BKPSRAMEN** : BKPSRAM clock enable (BKPSRAM clock enable) Set to '1' or clear to '0' by software. 0: BKPSRAM clock is off 1: BKPSRAM clock is on
Bit 12	**CRCEN** : CRC clock enable Set to '1' or clear to '0' by software. 0: CRC clock is off 1: CRC clock is on
Bit 11:5	Reserve
Bit 4	**GPIOEEN** : Port E clock enable (IO port E clock enable) Set to '1' or clear to '0' by software. 0: Port E clock is off 1: Port E clock is on
Bit 3	**GPIODEN** : Port D clock enable (IO port D clock enable) Set to '1' or clear to '0' by software. 0: Port D clock is off 1: Port D clock is on
Bit 2	**GPIOCEN** : Port C clock enable (IO port C clock enable) Set to '1' or clear to '0' by software. 0: Port C clock is off 1: Port C clock is on
Bit 1	**GPIOBEN** : Port B clock enable (IO port B clock enable) Set to '1' or clear to '0' by software. 0: Port B clock is off 1: Port B clock is off
Bit 0	**GPIOAEN** : Port A clock enable (IO port A clock enable) Set to '1' or clear to '0' by software. 0: Port A clock is off 1: Port A clock is on

Page 52

4.3.10 AHB2 peripheral clock enable register (**RCC_AHB2ENR**)

Offset address: 0x24

Reset value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
TK80 EN	Reserve														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															

Bit 31	**TK80EN** : TK80 clock enable (TK80 clock enable) Set to '1' or clear to '0' by software. 0: TK80 clock is off 1: TK80 clock enable
Bit 30:0	Reserve

4.3.11 APB1 peripheral clock enable register (**RCC_APB1ENR**)

Offset address: 0x28

Clock enable value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve	I2S1 EN	PWR EN	USB EN	CAN2 EN	CAN1 EN	Reserved		I2C3 EN	I2C2 EN	I2C1 EN	Reserve				
	rw	rw	rw	rw	rw			rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve				WWD GEN	Reserve	BKP EN	Reserve	TIM10 EN	TIM9 EN	TIM8 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN
				rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31	Reserve
Bit 30	**I2S1EN** : I2S1 interface clock enable (I2S1 interface clock enable) Set to '1' or clear to '0' by software. 0: I2S1 clock is off 1: I2S1 clock enable
Bit 29	**PWREN** : Power interface clock enable (Power interface clock enable) Set to '1' or clear to '0' by software. 0: PWR clock is off 1: PWR clock enable
Bit 28	**USBEN** : USB clock enable Set to '1' or clear to '0' by software. 0: USB clock is off 1: USB clock enable

Page 53

Bit 27	**CAN2EN** : CAN2 clock enable (CAN2 clock enable) Set to '1' or clear to '0' by software. 0: CAN2 clock is off 1: CAN2 clock enable
Bit 26	**CAN1EN** : CAN1 clock enable (CAN1 clock enable) Set to '1' or clear to '0' by software. 0: CAN1 clock is off 1: CAN1 clock enable
Bit 25: 24	Reserve
Bit 23	**I2C3EN** : I2C3 clock enable (I2C3 clock enable) Set to '1' or clear to '0' by software. 0: I2C3 clock is off 1: I2C3 clock enable
Bit 22	**I2C2EN** : I2C2 clock enable (I2C2 clock enable) Set to '1' or clear to '0' by software. 0: I2C2 clock is off 1: I2C2 clock enable
Bit 21	**I2C1EN** : I2C1 clock enable (I2C1 clock enable) Set to '1' or clear to '0' by software. 0: I2C1 clock is off 1: I2C1 clock enable
Bit 20: 12	Reserve
Bit 11	**WWDGEN** : Window watchdog clock enable Set to '1' or clear to '0' by software. 0: Window watchdog clock is off 1: Window watchdog clock enable
Bit 10	Reserved, always read as 0.
Bit 9	**BKPEN** : Backup interface clock enable (Backup interface clock enable) Set to '1' or cleared to '0' by software 0: The clock of the backup interface is off; 1: The backup interface clock enables the backup interface.
Bit 8	Reserve
Bit 7	**TIM10EN** : Timer 10 clock enable (Timer10 clock enable) Set to '1' or clear to '0' by software. 0: Timer 10 clock is off 1: Timer 10 clock enable
Bit 6	**TIM9EN** : Timer 9 clock enable (Timer9 clock enable) Set to '1' or clear to '0' by software. 0: Timer 9 clock is off 1: Timer 9 clock enable
Bit 5	**TIM8EN** : Timer8 clock enable (Timer8 clock enable) Set to '1' or clear to '0' by software. 0: Timer 8 clock is off 1: Timer 8 clock enable
Bit 4	**TIM7EN** : Timer7 clock enable (Timer7 clock enable) Set to '1' or clear to '0' by software. 0: Timer 7 clock is off 1: Timer 7 clock enable
Bit 3	**TIM6EN** : Timer6 clock enable (Timer6 clock enable) Set to '1' or clear to '0' by software. 0: Timer 6 clock is off 1: Timer 6 clock enable
Bit 2	**TIM5EN** : Timer 5 clock enable (Timer5 clock enable) Set to '1' or clear to '0' by software. 0: Timer 5 clock is off 1: Timer 5 clock enable

Page 54

Bit 1	**TIM4EN** : Timer 4 clock enable (Timer4 clock enable) Set to '1' or clear to '0' by software. 0: Timer 4 clock is off 1: Timer 4 clock enable
Bit 0	**TIM3EN** : Timer 3 clock enable (Timer3 clock enable) Set to '1' or clear to '0' by software. 0: Timer 3 clock is off 1: Timer 3 clock enable

4.3.12 APB2 peripheral clock enable register (**RCC_APB2ENR**)

Offset address: 0x2C

Clock enable value: 0x0000 0000

Access: no wait cycle, word, half word and byte access

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve						TCHP ADEN	QSPI1 EN	SPI4 EN	SPI3 EN	SPI2 EN	SPI1 EN	Reserve			
						rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve	SYSC FGEN	Reserve	SDIO2	SDIO1	Reserve		ADC1 EN	Reserve	UART 5EN	UART 4EN	UART 3EN	UART 2EN	UART 1EN	TIM2 EN	TIM1 EN
	rw		rw	rw			rw		rw	rw	rw	rw	rw	rw	rw

Bit 31: 26	Reserve
Bit 25	**TCHPADEN** : TCHPAD clock enable (TCHPAD clock enable) Set to '1' or clear to '0' by software. 0: TCHPADI clock is off 1: TCHPAD clock enable
Bit 24	**QSPI1EN** : QSPI clock enable (QSPI clock enable) Set to '1' or clear to '0' by software. 0: QSPI clock is off 1: QSPI clock enable
Bit 23	**SPI4EN** : SPI4 clock enable (SPI4 clock enable) Set to '1' or clear to '0' by software. 0: SPI4 clock is off 1: SPI4 clock enable
Bit 22	**SPI3EN** : SPI3 clock enable (SPI3 clock enable) Set to '1' or clear to '0' by software. 0: SPI3 clock is off 1: SPI3 clock enable
Bit 21	**SPI2EN** : SPI2 clock enable (SPI2 clock enable) Set to '1' or clear to '0' by software. 0: SPI2 clock is off 1: SPI2 clock enable
Bit 20	**SPI1EN** : SPI1 clock enable (SPI1 clock enable) Set to '1' or clear to '0' by software. 0: SPI1 clock is off 1: SPI1 clock enable
Bit 19: 15	Reserve
Bit 14	**SYSCFGEN** : System Configuration Controller clock enable (System Configuration Controller clock enable) Set to '1' or clear to '0' by software. 0: The system configuration controller clock is off 1: System configuration controller clock enable

Bit 13	Reserve
Bit 12	**SDIO2** : SDIO2 clock enable (SDIO2 clock enable) Set to '1' or clear to '0' by software. 0: SDIO2 clock is off 1: SDIO2 clock enable
Bit 11	**SDIO1** : SDIO1 clock enable (SDIO1 clock enable) Set to '1' or clear to '0' by software. 0: SDIO1 clock is off 1: SDIO1 clock enable
Bit 10: 9	Reserve
Bit 8	**ADC1EN** : ADC1 clock enable (ADC1 clock enable) Set to '1' or clear to '0' by software. 0: ADC1 clock is off 1: ADC1 clock enable
Bit 7	Reserve
Bit 6	**UART5EN** : UART5 clock enable (UART5 clock enable) Set to '1' or clear to '0' by software. 0: UART5 clock is off 1: UART5 clock enable
Bit 5	**UART4EN** : UART4 clock enable (UART4 clock enable) Set to '1' or clear to '0' by software. 0: UART4 clock is off 1: UART4 clock enable
Bit 4	**UART3EN** : UART3 clock enable (UART3 clock enable) Set to '1' or clear to '0' by software. 0: UART3 clock is off 1: UART3 clock enable

Bit 3	**UART2EN** : UART2 clock enable (UART2 clock enable) Set to '1' or clear to '0' by software. 0: UART2 clock is off 1: UART2 clock enable
Bit 2	**UART1EN** : UART1 clock enable (UART1 clock enable) Set to '1' or clear to '0' by software. 0: UART1 clock is off 1: UART1 clock enable
Bit 1	**TIM2EN** : Timer 2 clock enable (Timer2 clock enable) Set to '1' or clear to '0' by software. 0: TIM2 clock is off 1: TIM2 clock enable
Bit 0	**TIM1EN** : Timer1 clock enable (Timer1 clock enable) Set to '1' or clear to '0' by software. 0: TIM1 clock is off 1: TIM1 clock enable

4.3.13 Backup domain control register (**RCC_BDCR**)

Offset address: 0x30

Reset value: 0x0000 0000, which can only be effectively reset by the backup domain reset

Access: 0 ~ 3 wait cycles, word, half word and byte access

When continuously accessing this register, a wait state will be inserted.

Note: The *LSEON* , *LSEBYP* , *RTCSEL* and *RTCEN* bits in the backup domain control register (*RCC_BDCR*) are in the backup domain. Therefore, these bits are in a write-protected state after reset, and only after the *DBP* bit in the power control register (*PWR_CR*) is *"1"* can they be *checked* . These bits are changed. These bits can only be cleared by resetting the backup domain. Any internal or external reset will not affect these bits.

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve														BKP SRAM RST	BD RST
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Reserve					RTC SEL[1:0]		Reserve					LSE BYP	LSE RDY	LSE ON
rw						rw	rw						rw	rw	rw

Bit 31: 18	Reserved, always read as 0.
Bit 17	**BKPSRAMRST** : Backup SRAM software reset (Backup SRAM software reset) Set to '1' or clear to '0' by software. 0: Reset is not activated 1: Reset the backup SRAM
Bit 16	**BDRST** : Backup domain software reset (Backup domain software reset) Set to '1' or clear to '0' by software. 0: Reset is not activated 1: Reset the entire backup domain
Bit 15	**RTCEN** : RTC clock enable (RTC clock enable) Set to '1' or clear to '0' by software. 0: RTC clock is off 1: RTC clock is on
Bit 14: 10	Reserved, always read as 0.
Bit 9: 8	**RTCSEL[1 : 0]** : RTC clock source selection The RTC clock source is selected by software setting. Once the RTC clock source is selected, it cannot be changed until the next time the backup domain is reset. Change. It can be cleared by setting the BDRST bit. 00: no clock 01: LSE oscillator is used as RTC clock 10: LSI oscillator is used as RTC clock

	11: The HSE oscillator is used as the RTC clock after being divided by 128
Bit 7: 3	Reserved, always read as 0.
Bit 2	**LSEBYP** : External low-speed oscillator bypass (External low-speed oscillator bypass) In the debug mode, it is set to '1' or cleared to '0' by software to bypass the LSE. This can only be written when the external 32.768KHz oscillator is turned off. Bit. 0: LSE clock is not bypassed 1: LSE clock is bypassed
Bit 1	**LSERDY** : External low-speed LSE ready (External low-speed oscillator ready) The hardware is set to '1' or cleared to '0' to indicate whether the external 32.768KHz oscillator is ready. After LSEON is cleared, this bit requires 6 external The period of the low-speed oscillator is cleared. 0: The external 32.768KHz oscillator is not ready 1: External 32.768KHz oscillator is ready

Page 57

Bit 0	**LSEON** : External low-speed oscillator enable (External low-speed oscillator enable) Set to '1' or cleared to '0' by software 0: The external 32.768KHz oscillator is turned off 1: External 32.768KHz oscillator is turned on

4.3.14 Control Status Register (**RCC_CSR**)

Offset address: 0x34

Reset value: 0x0C00 0000, except that the reset flag is cleared by system reset, the reset flag can only be cleared by power reset.

Access: 0 ~ 3 wait cycles, word, half word and byte access

When continuously accessing this register, a wait state will be inserted.

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Keep	RMVF	Reserve							
rw	rw	rw	rw	rw	rw		w								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserve														LSI RDY	LSI ON
														r	rw

Bit 31	**LPWRRSTF** : Low power reset flag Set by hardware when a low-power management reset occurs. Cleared by software by writing the RMVF bit. 0: No low-power management reset occurs 1: A low-power management reset occurs
Bit 30	**WWDGRSTF** : Window watchdog reset flag It is set by hardware when the window watchdog reset occurs. Cleared by software by writing the RMVF bit. 0: No window watchdog reset occurs 1: Window watchdog reset occurs
Bit 29	**IWDGRSTF** : Independent watchdog reset flag It is set by hardware when the independent watchdog reset occurs in the V $_{DD}$ area. Cleared by software by writing the RMVF bit. 0: No independent watchdog reset occurs 1: An independent watchdog reset occurs
Bit 28	**SFTRSTF** : Software reset flag (Software reset flag) Set by hardware when a software reset occurs. Cleared by software by writing the RMVF bit. 0: No software reset occurs 1: A software reset occurred
Bit 27	**PORRSTF** : POR/PDR reset flag Set by hardware when power-on/power-down reset occurs. Cleared by software by writing the RMVF bit. 0: No power-on/power-down reset occurs 1: Power-on/power-down reset occurs
Bit 26	**PINRSTF** : NRST pin reset flag (PIN reset flag) Set by hardware when NRST pin reset occurs. Cleared by software by writing the RMVF bit. 0: No NRST pin reset occurs 1: NRST pin reset occurs
Bit 25	Reserved, read operation returns 0.

Page 58

Bit 24	**RMVF** : Clear reset flag (Remove reset flag) Set '1' by software to clear the reset flag. 0: No effect 1: Clear the reset flag
Bit 23: 2	Reserved, read operation returns 0.
Bit 1	**LSIRDY** : Internal low-speed oscillator ready (Internal low-speed oscillator ready) The hardware is set to '1' or cleared to '0' to indicate whether the internal 40KHz oscillator is ready. After LSION is cleared, LSIRDY is cleared after 3 cycles of the internal 40KHz oscillator. 0: The internal 40KHz oscillator clock is not ready 1: The internal 40KHz oscillator clock is ready
Bit 0	**LSION** : Internal low-speed oscillator enable (Internal low-speed oscillator enable) Set to '1' or clear to '0' by software. 0: The internal 40KHz oscillator is turned off 1: The internal 40KHz oscillator is turned on

4.3.15 **LCDPLL** configuration register (**RCC_LCD_PLLCFGR**)

Offset address: 0x38

Reset value: 0x0000 0000

Access: No waiting period, access by word, half word and byte.

This register is used to configure the PLL clock output according to the formula:

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
PLL LCD BY PASS	PLLLCDIS		LCDPLLDN							LCDPLLDP		LCDPLLDM			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve
Bit 15	**PLLLCDBYPASS** : PLLLCD bypass control (PLLLCD bypass control) 0: Do not bypass PLLLCD 1: Bypass PLLLCD
Bit 14: 13	**PLLLCDIS** : PLL current control (PLLLCD current control) 00: When the DN is 1-20, the configuration is 00 01: When the DN is 21-40, the configuration is 01 10: When the DN is 41-60, the configuration is 10 11: When the DN is above 61, the configuration is 11
Bit 12: 6	**LCDPLLDN** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients.
Bit 5: 4	**LCDPLLDP** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients. 00: P=1 01: P=2 10: P=4 11: P=8

Page 59

Bit 3: 0	**LCDPLLDM** : PLL clock configure factor (PLL clock configure factor) Set and cleared by software, used to control the PLL coefficients. PLL configuration formula: FCLKO = FREFIN * N / (M*P) FCLKO is the PLL output frequency, FREFIN is the PLL input reference clock frequency N = LCDPLLDN[6:0] + 1 M = LCDPLLDM[3:0] + 1

4.3.16 **RCC** dedicated clock configuration register (**RCC_DCKCFGR**)

Offset address: 0x3C

Reset value: 0x0000 0000

Access: No waiting period, access by word, half word and byte.

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve														LCD_PLLDIV	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															

Bit 31: 18	Reserve
Bit 17: 16	**LCD_PLLDIV** : division factor for PLL_LCDCLK (division factor for PLL_LCDCLK) Set and cleared by software to control the frequency division of PLL_LCDCLK. This bit allows writing only when PLL_LCD is disabled. 00: PLL_LCDCLK divided by 2 01: PLL_LCDCLK divided by 4 10: PLL_LCDCLK divided by 6 11: PLL_LCDCLK divided by 8
Bit 15:0	Reserve

5. General function I/O (GPIO)

5.1 GPIO function description

The GPIOA-D port has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), and two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit position/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR), a 32-bit lock register (GPIOx_LCKR) and two alternate function selection registers (GPIOx_AFRH And GPIOx_AFRL).

The GPIOE port has three 32-bit configuration registers (GPIOE_CRL, GPIOE_CRH, GPIOE_CRH_EXT), two 32-bit configuration registers Bit data registers (GPIOE_IDR and GPIOE_ODR), two 32-bit position/reset registers (GPIOE_BSRR and GPIOE_BSRR_EXT), a 24-bit reset register (GPIOE_BRR), a 32-bit lock register (GPIOE_LCKR) And three alternate function selection registers (GPIOE_AFRH, GPIOE_AFRL and GPIOE_AFRH_EXT).

sheet:

	32-bit configuration register	32-bit data register	32 position/reset register	16-bit reset register	32-bit lock register	Reuse function selection register
GPIOA-D	2 pcs	2 pcs	1 piece	1 piece	1 piece	2 pcs
GPIOE	3 pcs	2 pcs	2 pcs	1 piece	1 piece	3 pcs

Each bit of the GPIO port can be configured into multiple modes by software.

- Input floating
- Input pull-up

- Input dropdown
- Analog input
- Push-pull output
- Push-pull multiplexing function
- Open drain multiplexing function (only I2C multiplexing pins have this configuration)

Each I/O port can be freely programmed, but the I/O port registers must be accessed in 32-bit words (half-word or byte access is not allowed). The GPIOx_BSRR and GPIOx_BRR registers allow independent access to read/change any GPIO register; IRQ generated between visits is not dangerous.

Table **5.** Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
Universal output	Push-Pull	0	0	01		0 or 1
Multiplex function output	Push-Pull	1	0	11		Do not use
	Open-Drain		1	See Table 6		Do not use
enter	Analog input	0	0	00		Do not use
	Floating input		1			Do not use
	Drop-down input	1	0			0
	Pull up input					1

Table **6.** Output drive capability

MODE[1:0]	Drive capability
01	8mA
10	12mA
11	16mA

5.1.1 General **I/O** (**GPIO**)

During reset and just after reset, the multiplexing function is not turned on, and the I/O port is configured as a floating input mode (CNFx[1:0]=01b, MODEx[1: 0]=00).

After reset, the SWD pin is placed in input pull-up or pull-down mode:

- PA14: SWC is placed in pull-down mode
- PA15: SWD is placed in pull-up mode

When configured as an output, the value written to the output data register (GPIOx_ODR) is output to the corresponding I/O pin. Can be push-pull Mode (when the output is 0, only N-MOS is turned on) use the output driver.

The input data register (GPIOx_IDR) captures the data on the I/O pin every AHB1 clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down. When configured as inputs, they can be activated or disconnected.

5.1.2 Individual bit setting or bit clearing

When programming individual bits of GPIOx_ODR, the software does not need to disable interrupts: in a single AHB1 write operation, you can change only one or Multiple bits.

This is done by writing "1" to the bit you want to change in the "set/reset register" (GPIOx_BSRR, reset is GPIOx_BRR) Realized, the unselected bits will not be changed.

5.1.3 External interrupt **/** wake-up line

All ports have external interrupt capability. In order to use an external interrupt line, the port must be configured as input mode. More about external interrupts For information, refer to Section 7.2: External Interrupt/Event Controller (EXTI)

5.1.4 Multiplexing function (**AF**)

The port bit configuration register must be programmed before using the default multiplexing function.

- For multiplexed input functions, the port must be configured as an input mode (floating, pull-up or pull-down) and the input pins must be driven externally

Note: It is also possible to simulate multiplex function input pins through software. This simulation can be achieved by programming the *GPIO* controller. at this time, The port should be set to multiplex function output mode. Obviously, at this time, the corresponding pin is no longer driven by the outside, but by the *GPIO* controller. Software to drive.

- For the multiplex output function, the port must be configured as the multiplex function output mode (push-pull)
- For the bidirectional multiplex function, the port bit must be configured with the multiplex function output mode (push-pull). At this time, the input driver is configured to float Null input mode.

If the port is configured as a multiplexed output function, the pin is disconnected from the output register and connected to the output signal of the on-chip peripheral. If soft The device configures a GPIO pin as a multiplexed output function, but the peripheral is not activated, its output will be uncertain.

5.1.5 Software remapping **I/O** multiplexing function

In order to optimize the number of peripheral I/O functions of different device packages, some multiplexed functions can be remapped to other pins. This can be done through software configuration of the corresponding registers (refer to the AFR register description). At this time, the reuse function is no longer mapped to their On the original pin.

5.1.6 GPIO locking mechanism

The locking mechanism allows to freeze the IO configuration. When a LOCK program is executed on a port bit, it will not be possible until the next reset Then change the configuration of the port bit.

5.1.7 Input configuration

When the I/O port is configured as input:

- Output buffer is disabled
- Schmidt trigger input is activated

- Depending on the input configuration (pull-up, pull-down or floating), the weak pull-up and pull-down resistors are connected
- The data appearing on the I/O pin is sampled to the input data register every AHB1 clock
- Read access to the input data register to get the I/O status

The following figure shows the input configuration of the I/O port bits

Figure **4.** Input floating **/** pull-up **/** pull-down configuration

5.1.8 Output configuration

When the I/O port is configured as output:

- The output buffer is activated
  - Push-pull mode: "0" on the output register activates N-MOS, and "1" on the output register activates P-MOS.
- Schmidt input is activated
- Weak pull-up and pull-down resistors are disabled
- The data appearing on the I/O pin is sampled to the input data register every AHB1 clock
- In push-pull mode, the value written last time can be obtained by read access to the output data register.

5.1.9 Multiplexing function configuration

When the I/O port is configured as a multiplex function:

- In the push-pull configuration, the output buffer is turned on
- Built-in peripheral signal drive output buffer (multiplexed function output)
- Schmidt trigger input is activated
- Weak pull-up and pull-down resistors are disabled
- In every AHB1 clock cycle, the data appearing on the I/O pin is sampled to the input data register
- In push-pull mode, the last written value can be obtained when reading the output data register

5.1.10 Analog input configuration

When the I/O port is configured as an analog input configuration:

- Output buffer disabled
- Schmitt trigger input is disabled, achieving zero consumption on each analog I/O pin. Schmitt trigger output value is forced to "0"
- Weak pull-up and pull-down resistors are disabled
- The value is "0" when reading the input data register

The following figure shows the high-impedance input configuration of the I/O port bits:

Page 63

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

5.1.11 GPIO configuration of peripherals

The following table lists the pin configuration of each peripheral.

Table **7.** Advanced timer **TIMx**

TIM1 pin	Configuration	GPIO configuration
TIMx_CHx	Input capture channel x	Floating input
	Output compare channel x	Push-pull multiplexed output
TIMx_CHxN	Complementary output channel x	Push-pull multiplexed output
TIMx_BKIN	Brake input	Floating input
TIMx_ETR	External trigger clock input	Floating input

Table **8.** General-purpose timer **TIMx**

TIMx pin	Configuration	GPIO configuration
TIMx_CHx	Input capture channel x	Floating input
	Output compare channel x	Push-pull multiplexed output
TIMx_ETR	External trigger clock input	Floating input

Table **9. UART**

UART pin	Configuration	GPIO configuration
UART_TX	Full duplex mode	Push-pull multiplexed output
	Half-duplex synchronization module	Push-pull multiplexed output
UART_RX	Full duplex mode	Floating input or with pull-up input
	Half-duplex synchronization mode	Unused, can be used as general I/O
UART_RTS	Hardware flow control	Push-pull multiplexed output
UART_CTS	Hardware flow control	Floating input or with pull-up input

Page 64

Table **10. SPI**

SPI pins	Configuration	GPIO configuration
SPI_SCK	Master mode	Push-pull multiplexed output
	Slave mode	Floating input

SPI_MOSI	Full duplex mode/main mode	Push-pull multiplexed output
	Full duplex mode/slave mode	Floating input or with pull-up input
	Simple two-way data line/master mode	Push-pull multiplexed output
	Simple two-way data line/slave mode	Unused, can be used as general I/O
SPI_MISO	Full duplex mode/main mode	Floating input or with pull-up input
	Full duplex mode/slave mode	Push-pull multiplexed output
	Simple two-way data line/master mode	Unused, can be used as general I/O
	Simple two-way data line/slave mode	Push-pull multiplexed output
SPI_NSS	Hardware master/slave mode	Floating input or pull-up input or pull-down input
	Hardware master mode/NSS output enable	Push-pull multiplexed output
	Software mode	Unused, can be used as general I/O

Table **11. I2C**

I2C pin	Configuration	GPIO configuration
I2C_SCL	I2C clock	Open drain multiplexed output
I2C_SDA	I2C data	Open drain multiplexed output

Table **12. ADC**

ADC pin	GPIO configuration
ADC	Analog input

Table **13.** Other **I/O** pins

Pin	Configuration	GPIO configuration
MCO	Clock output	Push-pull multiplexed output
EXTI input line	External interrupt input	Floating input or with pull-up input or pull-down input

5.2 Multiplexing function **I/O** and debugging configuration

In order to optimize the number of peripherals, some multiplexing functions can be remapped to other pins. Set the multiplex function register (AFR) to achieve Remapping of pins. At this time, the multiplexing functions are no longer mapped to their original allocation.

5.2.1 SWD multiplexing function

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

Table **14.** Debug interface signals

Reuse function	GPIO port
SWD	PA15
SWC	PA14

In order to use more GPIOs during debugging, the above remapping configuration can be changed by setting the multiplex function register.

5.3 GPIO register description

5.3.1 Port configuration low register (**GPIOx_CRL**) (**x=A..E**)

Offset address: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 30 Bit 27: 26	**CNFy[1 : 0]** : Port x configuration bits (0…7) (Port x configuration bits) The software configures the corresponding I/O ports through these bits, please refer to Table 15 Port Bit Configuration Table. In the input mode (MODE[1:0]==00): 00: Analog input mode

Bit 19: 18 Bit 15: 14 Bit 11: 10 Bit 7: 6 Bit 3: 2	01: Floating input mode 10: Pull-up/pull-down input mode 11: reserved In the output mode (MODE[1:0]>00): 00: general push-pull output mode 01: reserved 10: Multiplex function push-pull output mode 11: reserved
Bit 29: 28 Bit 25: 24 Bit 21: 20 Bit 17: 16 Bit 13: 12 Bit 9: 8 Bit 5: 4 Bit 1: 0	**MODEy[1 : 0]** : Port x mode bits (y=0…7) (Port x mode bits) The software configures the corresponding I/O port through these bits, please refer to Table 15 Port Bit Configuration Table 00: Input mode (state after reset) 01: Output drive capacity 8mA 10: Output drive capacity 12mA 11: Output drive capability 16mA

5.3.2 Port configuration high register (**GPIOx_CRH**) (**x=A..E**)

Offset address: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
CNF15[1:0]		MODE15 [1:0]		CNF14[1:0]		MODE14 [1: 0]		CNF13[1:0]		MODE13 [1: 0]		CNF12[1:0]		MODE12 [1: 0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11 [1:0]		CNF10[1:0]		MODE10 [1: 0]		CNF9[1:0]		MODE9[1: 0]		CNF8[1:0]		MODE8[1: 0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Page 66

Bit 31: 30 Bit 27: 26 Bit 23: 22 Bit 19: 18 Bit 15: 14 Bit 11: 10 Bit 7: 6 Bit 3: 2	**CNFy[1 : 0]** : Port x configuration bits (8…15) (Port x configuration bits) The software configures the corresponding I/O ports through these bits, please refer to Table 15 Port Bit Configuration Table. In the input mode (MODE[1:0]==00): 00: Analog input mode 01: Floating input mode 10: Pull-up/pull-down input mode 11: reserved In the output mode (MODE[1:0]>00): 00: general push-pull output mode 01: reserved 10: Multiplex function push-pull output mode 11: reserved
Bit 29: 28 Bit 25: 24 Bit 21: 20 Bit 17: 16 Bit 13: 12 Bit 9: 8 Bit 5: 4 Bit 1: 0	**MODEy[1 : 0]** : Port x mode bits (y=8…15) (Port x mode bits) The software configures the corresponding I/O port through these bits, please refer to Table 15 Port Bit Configuration Table 00: Input mode (state after reset) 01: Output drive capacity 8mA 10: Output drive capacity 12mA 11: Output drive capability 16mA

5.3.3 Port input data register (**GPIOx_IDR**) (**x=A..E**)

Offset address: 0x08

Reset value: 0x0000 xxxx

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve								IDRE[23:16]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31: 24	Reserved, always read as 0
Bit 23: 16	**IDRy[23 : 16]** : Port input data (y=16..23) (Port input data)
Bit 15:0	**IDRy[15 : 0]** : Port input data (y=0..15) (Port input data)

5.3.4 Port output data register (**GPIOx_ODR**) (**x=A..E**)

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								ODR[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserved, always read as 0
Bit 23: 16	**ODRy[23** : **16]** : Port E output data (y=16..23) (Port output data) Note: For GPIOx_BSRR(x=A...D), each ODR bit can be set/cleared independently; For GPIOx_BSRR(x=E) and GPIOE_BSRR_EXT, each ODR bit of GPIOE can be individually performed Set/clear of the stand-alone.
Bit 15:0	**ODRy[15** : **0]** : Port output data (y=0..15) (Port output data) Note: For GPIOx_BSRR(x=A...D), each ODR bit can be set/cleared independently.

Page 67

5.3.5 Port set / clear register (**GPIOx_BSRR**) (**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
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31: 16	**BRy** : Clear the bit y of port x (y=0…15) (Port x Reset bit y) These bits can only be written and can only be manipulated in the form of words (16 bits) 0: No effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit to 0
Bit 15:0	**BSy** : Set bit y of port x (y=0..15) (Port x Set bit y) These bits can only be written and can only be manipulated in the form of words (16 bits). 0: No effect on the corresponding ODRy bit 1: Set the corresponding ODRy bit to 1

5.3.6 Port bit clear register (**GPIOx_BRR**) (**x=A..E**)

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								BR[23:16]							
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31: 24	Reserve
Bit 23: 16	**BRy** : Clear bit y of port E (y=16…23) (Port x Reset bit y) 0: No effect on the corresponding ODREy bit 1: Clear the corresponding ODREY bit to 0
Bit 15:0	**BRy** : Clear the bit y of port x (y=0…15) (Port x Reset bit y) 0: No effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit to 0

Page 68

5.3.7 Port configuration lock register (**GPIOx_LCKR**) (**x=A..E**)

When bit 24 (LCKK) is set by executing the correct write sequence, this register is used to lock the configuration of the port bits. Bit [23:0] is used to lock
The configuration of the GPIO port. During the specified write operation, LCKP[15:0] cannot be changed. When the LOCK sequence is executed on the corresponding port bit
Once listed, the port bit configuration can no longer be changed until the next system reset.

Each lock bit locks the corresponding 4 bits in the control register (CRL, CRH).

Note: Because the pin numbers of *GPIOE* and *GPIOA-D* are different, *8* bits are expanded , and the *LCKK* bit in the lock register is different, so it is divided into
Open description.

GPIOA-D register description:

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

Bit 31: 17	Reserve
Bit 16	**LCKK** : Lock key The given bit can be read at any time, and it can only be modified by the lock key write sequence. 0: Port configuration lock key is activated 1: The port configuration lock key bit is activated, and the GPIOx_LCKR register is locked before the next system reset The write sequence of the lock key: Write 1->write 0->write 1->read 0->read 1 The last reading can be omitted, but it can be used to confirm that the lock key has been activated. Note: When operating the write sequence of the lock key, the value of LCK[15:0] cannot be changed. Any errors in the write sequence of the operation lock key will not be Can activate the lock key
Bit 15:0	**LCKy** : Port x Lock bit y (y=0…15) (Port x Lock bit y) These bits are readable and writable but can only be written when the LCKK bit is 0. 0: Do not lock the configuration of the port 1: Lock the configuration of the port

GPIOE register description:

Note: *GPIOE* is *24* ports.

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				LCKK			LCK[23:16]					
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							LCK[15:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Page 69

Bit 31: 17	Reserve
	LCKK : Lock key The given bit can be read at any time, and it can only be modified by the lock key write sequence.

Bit 24	0: Port configuration lock key is activated 1: The port configuration lock key bit is activated, and the GPIOx_LCKR register is locked before the next system reset The write sequence of the lock key: Write 1->write 0->write 1->read 0->read 1 The last reading can be omitted, but it can be used to confirm that the lock key has been activated. Note: When operating the write sequence of the lock key, the value of LCK[15:0] cannot be changed. Any errors in the write sequence of the operation lock key will not be Can activate the lock key
Bit 23: 16	**LCKy** : Port E Lock bit y (y=16…23) (Port E Lock bit y) These bits are readable and writable but can only be written when the LCKK bit is 0. 0: Do not lock the configuration of the port 1: Lock the configuration of the port
Bit 15:0	**LCKy** : Port x Lock bit y (y=0…15) (Port x Lock bit y) These bits are readable and writable but can only be written when the LCKK bit is 0. 0: Do not lock the configuration of the port 1: Lock the configuration of the port

5.3.8 Port multiplexing function low register (**GPIOx_AFRL**) (**x=A..E**)

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
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**AFRy** : Multiplexing function selection of port x bit y (y=0…7) These bits can be written by software to configure the IO multiplexing function 0000: GPIO_AF_MCO_SW 0001: GPIO_AF_TIM_1_2 0010: GPIO_AF_TIM_34567 0011: GPIO_AF_I2S 0100: GPIO_AF_I2C 0101: GPIO_AF_SPI 0110: GPIO_AF_QSPI 0111: GPIO_AF_UART_2345 1000: GPIO_AF_UART_1 1001: GPIO_AF_CAN 1010: GPIO_AF_USB 1011: AF11 1100: GPIO_AF_TK80_SDIO 1101: GPIO_AF_Touchpad 1110: GPIO_AF_LTDC 1111: AF15

5.3.9 Port multiplexing function high register (**GPIOx_AFRH**) (**x=A..E**)

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
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

	AFRy : the multiplexing function selection of port x's bit y (y=8...15) These bits can be written by software to configure the IO multiplexing function 0000: GPIO_AF_MCO_SW 0001: GPIO_AF_TIM_1_2 0010: GPIO_AF_TIM_34567

Bit 31:0	0011: GPIO_AF_I2S 0100: GPIO_AF_I2C 0101: GPIO_AF_SPI 0110: GPIO_AF_QSPI 0111: GPIO_AF_UART_2345 1000: GPIO_AF_UART_1 1001: GPIO_AF_CAN 1010: GPIO_AF_USB 1011: AF11 1100: GPIO_AF_TK80_SDIO 1101: GPIO_AF_Touchpad 1110: GPIO_AF_LTDC 1111: AF15

5.3.10 Port configuration high register (GPIOE_CRH_EXT)

Offset address: 0x28

Reset value: 0x4444 4444

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
CNF23[1:0]		MODE23 [1:0]		CNF22[1:0]		MODE22 [1: 0]		CNF21[1:0]		MODE21 [1: 0]		CNF20[1:0]		MODE20 [1: 0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF19[1:0]		MODE19 [1:0]		CNF18[1:0]		MODE18 [1: 0]		CNF17[1:0]		MODE17 [1: 0]		CNF16[1:0]		MODE16 [1: 0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 30 Bit 27: 26 Bit 23: 22 Bit 19: 18 Bit 15: 14 Bit 11: 10 Bit 7: 6 Bit 3: 2	**CNFy[1 : 0]** : Port E configuration bits (16…23) (Port x configuration bits) The software configures the corresponding I/O ports through these bits, please refer to Table 15 Port Bit Configuration Table. In the input mode (MODE[1:0]==00): 00: Analog input mode 01: Floating input mode 10: Pull-up/pull-down input mode 11: reserved In the output mode (MODE[1:0]>00): 00: general push-pull output mode 01: reserved 10: Multiplex function push-pull output mode 11: reserved
Bit 29: 28 Bit 25: 24 Bit 21: 20 Bit 17: 16 Bit 13: 12 Bit 9: 8 Bit 5: 4 Bit 1: 0	**MODEy[1 : 0]** : Mode bits of port E (y=16…23) (Port x mode bits) The software configures the corresponding I/O port through these bits, please refer to Table 15 Port Bit Configuration Table 00: Input mode (state after reset) 01: Output drive capacity 8mA 10: Output drive capacity 12mA 11: Output drive capability 16mA

5.3.11 Port setting / clearing register (GPIOE_BSRR_EXT)

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
BR23	BR22	BR21	BR20	BR19	BR18	BR17	BR16	BS23	BS22	BS21	BS20	BS19	BS18	BS17	BS16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 15: 8	**BRy** : Clear bit y of port E (y=16…23) (Port x Reset bit y) These bits can only be written and can only be manipulated in the form of words (16 bits) 0: No effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit to 0
Bit 7:0	**BSy** : Set bit y of port E (y=16..23) (Port x Set bit y) These bits can only be written and can only be manipulated in the form of words (16 bits). 0: No effect on the corresponding ODRy bit 1: Set the corresponding ODRy bit to 1

Page 72

5.3.12 Port multiplexing function high register (**GPIOE_AFRH_EXT**)

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
AFR23[3:0]				AFR22[3:0]				AFR21[3:0]				AFR20[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR19[3:0]				AFR18[3:0]				AFR17[3:0]				AFR16[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**AFRy** : the multiplexing function selection of port E's bit y (y=16...23) These bits can be written by software to configure the IO multiplexing function 0000: GPIO_AF_MCO_SW 0001: GPIO_AF_TIM_1_2 0010: GPIO_AF_TIM_34567 0011: GPIO_AF_I2S 0100: GPIO_AF_I2C 0101: GPIO_AF_SPI 0110: GPIO_AF_QSPI 0111: GPIO_AF_UART_2345 1000: GPIO_AF_UART_1 1001: GPIO_AF_CAN 1010: GPIO_AF_USB 1011: AF11 1100: GPIO_AF_TK80_SDIO 1101: GPIO_AF_Touchpad 1110: GPIO_AF_LTDC 1111: AF15

6. System configuration controller

TK499 has a set of system configuration registers. The main functions of these registers are as follows:

- Remap some peripheral DMA trigger sources to other different DMA channels.
- Manage external interrupts connected to the GPIO port.
- Remap the memory to the code start area.
- External interrupt pin configuration

6.1 SYSCFG register description

6.1.1 SYSCFG configuration register (SYSCFG_CFGR)

This register is specifically used to configure the memory starting area mapping and DMA request remapping. With two configurable memories starting at 0x0000 0000 The control bits of the address storage area type, these two control bits can be configured by software to shield the BOOT selection. After reset, these two control bits are actual BOOT mode configuration.

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
SPI3_RX_DMA1_RMP	SPI3_TX_DMA1_RMP	UART3_TX_DMA1_RMP	I2C1_RX_DMA1_RMP	I2C1_TX_DMA1_RMP	I2C2_RX_DMA1_RMP	TIM3_CH2_DMA1_RMP	TIM3_CH4_DMA1_RMP	TIM3_UP_DMA1_RMP	TIM4_CH3_DMA1_RMP	TIM4_CH4_DMA1_RMP	TIM4_UP_DMA1_RMP	TIM8_UP_DMA1_RMP	TIM9_UP_DMA1_RMP	TIM10_UP_DMA1_RMP	ADC1_DMA2_RMP
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1_RX_DMA2_RMP	SPI1_TX_DMA2_RMP	SPI4_RX_DMA2_RMP	SPI4_TX_DMA2_RMP	UART1_RX_DMA2_RMP	TIM1_CH1_DMA2_RMP	TIM1_CH2_DMA2_RMP	TIM1_TRIG_DMA2_RMP	TIM2_CH2_DMA2_RMP	TIM2_CH3_DMA2_RMP	QSPI_RX_DMA2_RMP	QSPI_TX_DMA2_RMP	SDIO1_DMA2_RMP	SDIO2_DMA2_RMP	MEM_MODE	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bit 31	**SPI3_RX_DMA1_RMP** : SPI3 DMA1 request remapping bit (SPI3 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the SPI3 DMA1 channel. 0: No remapping (SPI3_RX DMA1 request is mapped on DMA1 channel 1) 1: Remapping (SPI3_RX DMA1 request mapping on DMA1 channel 3)
Bit 30	**SPI3_TX_DMA1_RMP** : SPI3 DMA1 request remapping bit (SPI3 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the SPI3 DMA1 channel. 0: No remapping (SPI3_TX DMA1 request mapping on DMA1 channel 6) 1: Remapping (SPI3_TX DMA1 request mapping on DMA1 channel 8)
Bit 29	**UART3_TX_DMA1_RMP** : UART3_TX DMA1 request remapping bit (UART3_TX DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the UART3_TX DMA1 channel. 0: No remapping (UART3_TX DMA1 request is mapped on DMA1 channel 4) 1: Remapping (UART3_ TX DMA1 request is mapped on DMA1 channel 5)
Bit 28	**I2C1_RX_DMA1_RMP** : I2C2_RX DMA1 request remapping bit (I2C2_RX DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the I2C2_RX DMA1 channel. 0: No remapping (I2C2_RX DMA1 request is mapped on DMA1 channel 1) 1: Remapping (I2C2_RX DMA1 request mapping on DMA1 channel 6)
Bit 27	**I2C1_TX_DMA1_RMP** : I2C2_TX DMA1 request remapping bit (I2C2_TX DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the I2C2_TX DMA1 channel. 0: No remapping (I2C2_TX DMA1 request mapping on DMA1 channel 7) 1: Remapping (I2C2_TX DMA1 request mapping on DMA1 channel 8)

I2C2_RX_DMA1_RMP : I2C2_RX DMA1 request remapping bit (I2C2_RX DMA1 request remapping

Bit 26	bit) This bit is set and cleared by software. It controls the remapping requested by the I2C2_RX DMA1 channel. 0: No remapping (I2C2_RX DMA1 request is mapped on DMA1 channel 3) 1: Remapping (I2C2_RX DMA1 request mapping on DMA1 channel 4)
Bit 25	**TIM3_CH2_DMA1_RMP** : TIM3 DMA1 request remapping bit (TIM3 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM3 DMA1 channel. 0: No remapping (TIM3_CH2 DMA1 requests are mapped on DMA1 channel 6 respectively) 1: Remapping (TIM3_CH2 DMA1 requests are mapped on DMA1 channel 7 respectively)
Bit 24	**TIM3_CH4_DMA1_RMP** : TIM3 DMA1 request remapping bit (TIM3 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM3 DMA1 channel. 0: No remapping (TIM3_CH4 DMA1 requests are mapped on DMA1 channel 3 respectively) 1: Remapping (TIM3_CH4 DMA1 requests are mapped on DMA1 channel 7 respectively)
Bit 23	**TIM3_UP_DMA1_RMP** : TIM3 DMA1 request remapping bit (TIM3 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM3 DMA1 channel. 0: No remapping (TIM3_UP DMA1 requests are mapped on DMA1 channel 3 respectively) 1: Remapping (TIM3_UP DMA1 requests are mapped on DMA1 channel 8 respectively)
Bit 22	**TIM4_CH3_DMA1_RMP** : TIM4 DMA1 request remapping bit (TIM4 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM4 DMA1 channel. 0: No remapping (TIM4_CH3 DMA1 requests are mapped on DMA1 channel 1 respectively) 1: Remapping (TIM4_CH3 DMA1 requests are mapped on DMA1 channel 8 respectively)
Bit 21	**TIM4_CH4_DMA1_RMP** : TIM4 DMA1 request remapping bit (TIM4 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM4 DMA1 channel. 0: No remapping (TIM4_CH4 DMA1 requests are mapped on DMA1 channel 2 respectively) 1: Remapping (TIM4_CH4 DMA1 requests are mapped on DMA1 channel 4 respectively)
Bit 20	**TIM4_UP_DMA1_RMP** : TIM4 DMA1 request remapping bit (TIM4 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM4 DMA1 channel. 0: No remapping (TIM4_UP DMA1 requests are mapped on DMA1 channel 1 respectively) 1: Remapping (TIM4_UP DMA1 requests are mapped on DMA1 channel 7 respectively)
Bit 19	**TIM8_UP_DMA1_RMP** : TIM8 DMA1 request remapping bit (TIM8 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM8 DMA1 channel. 0: No remapping (TIM8_UP DMA1 request mapping on DMA1 channel 2) 1: Remapping (TIM8_UP DMA1 request mapping on DMA1 channel 6)
Bit 18	**TIM9_UP_DMA1_RMP** : TIM9 DMA1 request remapping bit (TIM9 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM9 DMA1 channel. 0: No remapping (TIM9_UP DMA1 request mapping on DMA1 channel 3) 1: Remapping (TIM9_UP DMA1 request mapping on DMA1 channel 5)
Bit 17	**TIM10_UP_DMA1_RMP** : TIM10 DMA1 request remapping bit (TIM10 DMA1 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM10 DMA1 channel. 0: No remapping (TIM10_UP DMA1 request mapping on DMA1 channel 1) 1: Remapping (TIM10_UP DMA1 request mapping on DMA1 channel 4)
Bit 16	**ADC1_DMA2_RMP** : ADC1 DMA2 request remapping bit (ADC1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of ADC1 DMA2 channel request 0: No remapping (ADC1 DMA2 request mapping on DMA2 channel 1) 1: Remapping (ADC1 DMA2 request mapping on DMA2 channel 5)
Bit 15	**SPI1_RX_DMA2_RMP** : SPI1 DMA2 request remapping bit (SPI1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the SPI1 DMA2 channel. 0: No remapping (SPI1_RX DMA2 requests are mapped on DMA2 channel 1 respectively) 1: Remapping (SPI1_RX DMA2 requests are mapped on DMA2 channel 3 respectively)
Bit 14	**SPI1_TX_DMA2_RMP** : SPI1 DMA2 request remapping bit (SPI1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the SPI1 DMA2 channel. 0: No remapping (SPI1_TX DMA2 requests are mapped on DMA2 channel 4 respectively) 1: Remapping (SPI1_TX DMA2 requests are mapped on DMA2 channel 6 respectively)
Bit 13	**SPI4_RX_DMA2_RMP** : SPI4 DMA2 request remapping bit (SPI4 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of the SPI4 DMA2 channel request. 0: No remapping (SPI4_RX DMA2 requests are mapped on DMA2 channel 1 respectively) 1: Remapping (SPI4_RX DMA2 requests are mapped on DMA2 channel 4 respectively)

Page 75

Bit 12	**SPI4_TX_DMA2_RMP** : SPI4 DMA2 request remapping bit (SPI4 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of the SPI4 DMA2 channel request. 0: No remapping (SPI4_TX DMA2 requests are mapped on DMA2 channel 2 respectively) 1: Remapping (SPI4_TX DMA2 requests are mapped on DMA2 channel 5 respectively)
Bit 11	**UART1_RX_DMA2_RMP** : UART1_RX DMA2 request remapping bit (UART1_RX DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of UART1_RX DMA2 channel request. 0: No remapping (UART1_ RX DMA2 request is mapped on DMA2 channel 3) 1: Remapping (UART1_ RX DMA2 request is mapped on DMA2 channel 6)
Bit 10	**TIM1_CH1_DMA2_RMP** : TIM1 DMA2 request remapping bit (TIM1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM1 DMA2 channel. 0: No remapping (TIM1_CH1 DMA2 requests are mapped on DMA2 channel 4 respectively) 1: Remapping (TIM1_CH1 DMA2 requests are mapped on DMA2 channel 7 respectively)
Bit 9	**TIM1_CH2_DMA2_RMP** : TIM1 DMA2 request remapping bit (TIM1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM1 DMA2 channel. 0: No remapping (TIM1_CH2 DMA2 requests are mapped on DMA2 channel 3 respectively) 1: Remapping (TIM1_CH2 DMA2 requests are mapped on DMA2 channel 7 respectively)
	TIM1_TRIG_DMA2_RMP : TIM1 DMA2 request remapping bit (TIM1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM1 DMA2 channel.

Bit 8	0: No remapping (TIM1_TRIG DMA2 requests are mapped on DMA2 channel 1 respectively) 1: Remapping (TIM1_TRIG DMA2 requests are mapped on DMA2 channel 5 respectively)
Bit 7	**TIM2_CH2_DMA2_RMP** : TIM2 DMA2 request remapping bit (TIM2 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM2 DMA2 channel. 0: No remapping (TIM2_CH2 DMA2 request mapping on DMA2 channel 3) 1: Remapping (TIM2_CH2 DMA2 requests are mapped on DMA2 channel 4 respectively)
Bit 6	**TIM2_CH3_DMA2_RMP** : TIM2 DMA2 request remapping bit (TIM2 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping requested by the TIM2 DMA2 channel. 0: No remapping (TIM2_CH3 DMA2 request mapping on DMA2 channel 3) 1: Remapping (TIM2_CH3 DMA2 requests are mapped on DMA2 channel 5 respectively)
Bit 5	**QSPI_RX_DMA2_RMP** : QSPI DMA2 request remapping bit (QSPI DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of QSPI_TX, QSPI_RX DMA2 channel request 0: No remapping (QSPI_RX DMA2 requests are mapped on DMA2 channel 4 respectively) 1: Remapping (QSPI_RX DMA2 requests are mapped on DMA2 channel 6 respectively)
Bit 4	**QSPI_TX_DMA2_RMP** : QSPI DMA2 request remapping bit (QSPI DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of QSPI_TX, QSPI_RX DMA2 channel request 0: No remapping (QSPI_TX DMA2 requests are mapped on DMA2 channel 5 respectively) 1: Remapping (QSPI_TX DMA2 requests are mapped on DMA2 channel 7 respectively)
Bit 3	**SDIO1_DMA2_RMP** : SDIO1 DMA2 request remapping bit (SDIO1 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of SDIO1 DMA2 channel request 0: No remapping (SDIO1 DMA2 request mapping on DMA2 channel 4) 1: Remapping (SDIO1 DMA2 request mapping on DMA2 channel 7)
Bit 2	**SDIO2_DMA2_RMP** : SDIO2 DMA2 request remapping bit (SDIO2 DMA2 request remapping bit) This bit is set and cleared by software. It controls the remapping of SDIO2 DMA2 channel request 0: No remapping (SDIO2 DMA2 request is mapped on DMA2 channel 2) 1: Remapping (SDIO2 DMA2 request mapping on DMA2 channel 8)
Bit 1: 0	**MEM_MODE** : Memory selection bit These bits are set and cleared by software. It controls the internal mapping of the memory to address 0x0000 0000. 00: ROM is mapped to 0x0000 0000 Other: reserved

6.1.2 External interrupt configuration register **1** (**SYSCFG_EXTICR1**)

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
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve.
Bit 15:0	EXTI3[3:0], EXTI2[3:0], EXTI1[3:0], EXTI0[3:0] EXTIx configuration (x = 0…3) (EXTI x configuration) These bits can be used for software read and write. Used to select the input source of the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

6.1.3 External interrupt configuration register **2** (**SYSCFG_EXTICR2**)

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
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve.
Bit 15:0	EXTI7[3:0], EXTI6[3:0], EXTI5[3:0], EXTI4[3:0] EXTIx configuration (x = 7…4) (EXTI x configuration) These bits can be used for software read and write. Used to select the input source of the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

6.1.4 External interrupt configuration register **3** (**SYSCFG_EXTICR3**)

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
EXTI11[3:0]				EXTI10[3:0]				EXT9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve.
Bit 15:0	EXTI11[3:0], EXTI10[3:0], EXTI9[3:0], EXTI8[3:0] EXTIx configuration (x = 11…8) (EXTI x configuration) These bits can be used for software read and write. Used to select the input source of the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

6.1.5 External interrupt configuration register **4** (**SYSCFG_EXTICR4**)

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
EXTI15[3:0]				EXTI14[3:0]				EXT13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve.
Bit 15:0	EXTI15[3:0], EXTI14[3:0], EXTI13[3:0], EXTI12[3:0] EXTIx configuration (x=15…12) (EXTI x configuration) These bits can be used for software read and write. Used to select the input source of the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin (PE[23:16] does not support external interrupt function)

7. **DMA** controller (**DMA**)

7.1 Introduction to **DMA**

Direct memory access is used to provide high-speed data transfer between peripherals and memory or between memory and memory. No CPU required No intervention, data can be moved quickly through DMA. This saves CPU resources for other operations.

Including DMA1 and DMA2, each DMA controller has 8 channels.

7.2 Main features of **DMA**

- 8 independent configurable channels.
- Each channel is directly connected to a dedicated hardware DMA request, and each channel also supports software triggering. These functions are implemented through software Configuration.
- The priority among the seven requests can be set by software programming (there are four levels: very high, high, medium and low). Priority is determined by hardware (request 0 has priority over request 1, and so on).
- The transmission width (byte, half word, full word) of independent source and target data areas simulates the process of packing and unpacking. Source and destination address Must be aligned according to the data transmission width.
- Support circular buffer management.
- Each channel has 3 event flags (DMA half transfer, DMA transfer completion and DMA transfer error), these 3 event flags The logical OR becomes a separate interrupt request.
- Transfer between storage and storage.
- Peripheral and memory, memory and peripheral transfer.
- SRAM, peripheral SRAM, APB1, APB2 and AHB peripherals can all be used as the source and target of access.
- Programmable number of data transfers: the maximum number of transfers is up to 2^32=4,294,967,296, which can be completed in a single time for high-resolution LCD screens Filled with data.

7.3 Functional description

The DMA controller and CPU core share the system data bus to perform direct memory data transfer. When the CPU and DMA access the same When the target (RAM or peripherals), the DMA request may stop the CPU from accessing the system bus for several cycles, and the bus arbiter performs cyclic scheduling. To ensure that the CPU can get at least half of the system bus (memory or peripheral) bandwidth.

7.3.1 DMA processing

After an event occurs, the peripheral sends a request signal to the DMA controller. The DMA controller processes the request according to the priority of the channel. When the DMA controller starts to access the peripheral, the DMA controller immediately sends an acknowledge signal to the peripheral. When getting from the DMA controller When responding to the signal, the peripheral immediately releases its request. Once the peripheral releases this request, the DMA controller cancels the response signal at the same time. If se When more requests are made, the peripheral can start the next processing.

In summary, each DMA transfer consists of 3 operations:

- The load operation is performed from the peripheral data register or the memory unit at the address specified by the DMA_CMARx register.
- Store data to the peripheral data register or store data to the memory unit at the address specified by the DMA_CMARx register.
- Perform a decrement operation of the DMA_CNDTRx register. This register contains the number of outstanding operations.

7.3.2 Arbiter

The arbiter initiates peripheral/memory access according to the priority of the channel request. Priority management is divided into 2 stages:

- Software: The priority of each channel can be set in the DMA_CCRx register, there are 4 levels:
  - Highest priority
  - high priority
  - Medium priority
  - Low priority
- Hardware: If 2 requests have the same software priority, the channel with the lower number has a higher number than the channel with the higher number Priority. For example, channel 2 has priority over channel 4.

7.3.3 **DMA** channel

Each channel can perform DMA transfers between peripheral registers with fixed addresses and memory addresses. The amount of data transferred by DMA is Programmable, up to 4,294,967,296. The register containing the number of data items to be transferred is decremented after each transfer.

Programmable data volume

The transfer data volume of peripherals and memory can be programmed by the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer increment

By setting the PINC and MINC flags in the DMA_CCRx register, the pointers of peripherals and memory can be selected after each transfer To complete the automatic increment. When set to incremental mode, the next address to be transmitted will be the previous address plus the incremental value. The incremental value depends The selected data width is 1, 2 or 4. The address of the first transfer is stored in the DMA_CPARx/DMA_CMARx register.

When the channel is configured in acyclic mode, after the transfer ends (that is, the transfer count becomes 0), DMA operations will no longer occur.

Channel configuration

The following is the process of configuring DMA channel x (x represents the channel number):

1. Set the address of the peripheral register in the DMA_CPARx register. When a peripheral data transfer request occurs, this address will be the data The source or destination of the transfer.
2. Set the address of the data memory in the DMA_CMARx register. When a peripheral data transfer request occurs, the transferred data will be transferred from here This address is read from or written to.
3. Set the amount of data to be transferred in the DMA_CNDTRx register. After each data transmission, this value is decremented.
4. Set the channel priority in the PL[1:0] bits of the DMA_CCRx register.
5. Set the direction of data transfer, cycle mode, incremental mode of peripherals and memory, peripherals and memory in the DMA_CCRx register. The data width of the memory, half of the transmission generates an interrupt, or the transmission completes an interrupt.
6. Set the ENABLE bit of the DMA_CCRx register to enable the channel. Once the DMA channel is activated, it can respond to DMA requests from peripherals on this channel.

When half of the data is transmitted, the half-transmission flag (HTIF) is set to 1, and when the half-transmission interrupt bit (HTIE) is set, a Interrupt request. After the data transfer is over, the transfer complete flag (TCIF) is set to 1, when the allow transfer complete interrupt bit (TCIE) is set, the An interrupt request is generated.

Cyclic mode

The circular mode is used to process circular buffers and continuous data transmission (such as ADC scan mode). In the DMA_CCRx register The CIRC bit is used to enable this function. When the cyclic mode is activated and the number of data transfers becomes 0, it will automatically be restored to the configured channel DMA operation will continue.

Memory-to-memory mode

The operation of the DMA channel can be carried out without peripheral request, this kind of operation is the memory-to-memory mode. When set up After the MEM2MEM bit in the DMA_CCRx register, the software sets the EN bit in the DMA_CCRx register to start DMA communication. The DMA transfer will start immediately when the channel is running. When the DMA_CNDTRx register becomes 0, the DMA transfer ends. Memory-to-memory mode Cannot be used simultaneously with cyclic mode.

7.3.4 Programmable data transfer width, alignment and data size end

When PSIZE and MSIZE are not the same, the DMA module performs data alignment according to the following table.

Table **15.** Programmable data transfer width and size end operation (when **PINC=MINC=1**)

Source width	Target width	transmission number	Source: address/data	Transfer operation	Target: address/data
8	8	4	0x0 / B0	1: Read B0[7:0] at 0x0, write B0[7:0] at 0x0	0x0 / B0
			0x1 / B1	2: Read B1[7:0] at 0x1 and write B1[7:0] at 0x1	0x1 / B1
			0x2 / B2	3: Read B2[7:0] at 0x2 and write B2[7:0] at 0x2	0x2 / B2
			0x3 / B3	4: Read B3[7:0] at 0x3 and write B3[7:0] at 0x3	0x3 / B3

Source width	Target width	transmission number	Source: address/data	Transfer operation	Target: address/data
8	16	4	0x0 / B0	1: Read B0[7:0] at 0x0, write 00B0[15:0] at 0x0	0x0 / 00B0
			0x1 / B1	2: Read B1[7:0] at 0x1 and write 00B1[15:0] at 0x2	0x2 / 00B1
			0x2 / B2	3: Read B2[7:0] at 0x2 and write 00B2[15:0] at 0x4	0x4 / 00B2
			0x3 / B3	4: Read B3[7:0] at 0x3 and write 00B3[15:0] at 0x6	0x6 / 00B3
8	32	4	0x0 / B0	1: Read B0[7:0] at 0x0, write 000000B0[31:0] at 0x0	0x0 / 000000B0
			0x1 / B1	2: Read B1[7:0] at 0x1, write 000000B1[31:0] at 0x4	0x4 / 000000B1
			0x2 / B2	3: Read B2[7:0] at 0x2 and write 000000B2[31:0] at 0x8	0x8 / 000000B2
			0x3 / B3	4: Read B3[7:0] at 0x3 and write 000000B3[31:0] at 0xC	0xC / 000000B3
16	8	4	0x0 / B1B0	1: Read B1B0[15:0] at 0x0, write B0[7:0] at 0x0	0x0 / B0
			0x2 / B3B2	2: Read B3B2[15:0] at 0x2 and write B2[7:0] at 0x1	0x1 / B2
			0x4 / B5B4	3: Read B5B4[15:0] at 0x4 and write B4[7:0] at 0x2	0x2 / B4
			0x6 / B7B6	4: Read B7B6[15:0] at 0x6 and write B6[7:0] at 0x3	0x3 / B6
16	16	4	0x0 / B1B0	1: Read B1B0[15:0] at 0x0, write B1B0[15:0] at 0x0	0x0 / B1B0
			0x2 / B3B2	2: Read B3B2[15:0] at 0x2 and write B3B2[15:0] at 0x2	0x2 / B3B2

			0x4 / B5B4 0x6 / B7B6	3: Read B5B4[15:0] at 0x4 and write B5B4[15:0] at 0x4 4: Read B7B6[15:0] at 0x6 and write B7B6[15:0] at 0x6	0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: Read B1B0[15:0] at 0x0, write 0000B1B0[31:0] at 0x0 2: Read B3B2[15:0] at 0x2, write 0000B3B2[31:0] at 0x4 3: Read B5B4[15:0] at 0x4, write 0000B5B4[31:0] at 0x8 4: Read B7B6[15:0] at 0x6, write 0000B7B6[31:0] at 0xC	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, write B0[7:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4 and write B4[7:0] at 0x1 3: Read BBBAB9B8[31:0] at 0x8, write B8[7:0] at 0x2 4: Read BFBEBDBC[31:0] at 0xC, write BC[7:0] at 0x3	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, write B1B0[15:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4 and write B5B4[15:0] at 0x2 3: Read BBBAB9B8[31:0] at 0x8, write B9B8[15:0] at 0x4 4: Read BFBEBDBC[31:0] at 0xC, write BDBC[15:0] at 0x6	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	32	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, write B3B2B1B0[31:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4, write B7B6B5B4[31:0] at 0x4 3: Read BBBAB9B8[31:0] at 0x8, write BBBAB9B8[7:0] at 0x8 4: Read BFBEBDBC[31:0] at 0xC, write BFBEBDBC[31:0] at 0xC	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC/ BFBEBDBC

Operate an **AHB** device that does not support byte or halfword writing

When the DMA module starts an AHB byte or halfword write operation, the data will be in the unused part of the HWDATA[31:0] bus
repeat. Therefore, if DMA writes bytes or halfwords to AHB devices that do not support byte or halfword write operations (that is, HSIZE is not suitable for this mode).
Block), no error will occur, DMA will write 32-bit HWDATA data according to the following two examples:

- When HSIZE=half word, write half word '0xABCD', DMA will set HWDATA bus to '0xABCDABCD'.
- When HSIZE=byte, write byte '0xAB', DMA will set HWDATA bus to '0xABABABAB'.

Assuming that the AHB/APB bridge is an AHB 32-bit slave device, it does not process the HSIZE parameter, it will convert any AHB
The byte or half word above is transferred to APB in 32 bits:

- An AHB write byte data '0xB0' operation to address 0x0 (or 0x1, 0x2 or 0x3) will be converted to APB.
  The write data '0xB0B0B0B0' operation of address 0x0.
- An AHB write halfword data '0xB1B0' to address 0x0 (or 0x2) will be converted to APB address 0x0
  The writing data '0xB1B0B1B0' operation.

For example, if you want to write to the APB backup register (16-bit register aligned with a 32-bit address), you need to configure the memory data source width
(MSIZE) is '16 bits', and the peripheral target data width (PSIZE) is '32 bits'.

7.3.5 Error Management

Reading and writing a reserved address area will cause a DMA transfer error. When a DMA transfer error occurs during DMA read and write operations,
The hardware will automatically clear the EN bit of the channel configuration register (DMA_CCRx) corresponding to the channel where the error occurred, and the channel operation will be

At this time, the transmission error interrupt flag (TEIF) corresponding to the channel in the DMA_IFT register will be set, if it is registered in DMA_CCRx
If the transmission error interrupt enable bit is set in the device, an interrupt will be generated.

7.3.6 Interrupt

Each DMA channel can generate interrupts when the DMA transfer is halfway, the transfer is complete, and the transfer is wrong. Considering the flexibility of the application, through
These interrupts can be turned on by setting different bits of the register.

Table **16. DMA** interrupt request

Interrupt event	Event flag	Enable control bit
Halfway through	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transmission error	TEIF	TEIE

7.3.7 DMA request image

DMA controller

The DMA request generated from the peripheral is input to the DMA controller through logic OR, which means that only one request is valid at the same time. Peripherals
The DMA request can be independently turned on or off by setting the control bit in the corresponding peripheral register. See DMA1, DMA2 in the figure below
Mapping list.

Table **17. DMA1** channel mapping list

Peripherals	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8
SPI2				SPI2_RX	SPI2_TX			
SPI3	SPI3_RX (1)		SPI3_RX (2)			SPI3_TX (1)		SPI3_TX (2)
UART2						UART2_RX	UART2_TX	
UART3		UART3_RX		UART3_TX (1)	UART3_TX (2)			
UART4			UART4_RX		UART4_TX			

UART5	UART5_RX							UART5_TX
I2C1	I2C1_RX (1)					I2C1_RX (2)	I2C1_TX (1)	I2C1_TX (2)
I2C2			I2C2_RX (1)	I2C2_ RX (2)				I2C2_TX
I2C3			I2C3_RX		I2C3_TX			
I2C4		I2C4_ TX (1)				I2C4_RX	I2C4_TX (2)	
TIM3			TIM3_CH4 (1) TIM3_UP (1)		TIM3_CH1 TIM3_TRIG	TIM3_CH2 (1)	TIM3_CH2 (2) TIM3_CH4 (2)	TIM3_CH3 TIM3_UP (2)
TIM4	TIM4_CH3 (1) TIM4_UP (1)	TIM4_CH4 (1) TIM4_TRIG	TIM4_CH1	TIM4_CH4 (2) TIM4_TRIG	TIM4_CH2		TIM4_UP (2)	TIM4_CH3 (2)
TIM8		TIM8_UP (1)				TIM8_UP (2)		
TIM9			TIM9_UP (1)		TIM9_UP (2)			
TIM10	TIM10_UP (1)			TIM10_UP (2)				

Table **18. DMA2** channel mapping list

Peripherals	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8
ADC	ADC (1)				ADC (2)			
SPI1	SPI1_RX (1)		SPI1_RX (2)	SPI1_TX (1)		SPI1_TX (2)		
SPI4	SPI4_RX (1)	SPI4_TX (1)		SPI4_RX (2)	SPI4_TX (2)			
UART1			UART1_RX (1)			UART1_RX (2)		UART1_TX
TIM1	TIM1_TRIG (1)		TIM1_CH2 (1)	TIM1_CH1 (1)	TIM1_CH4 TIM1_TRIG (2) TIM1_COM	TIM1_UP	TIM1_CH1 (2) TIM1_CH2 (2) TIM1_CH3	
TIM2		TIM2_UP	TIM2_CH1 TIM2_CH2 (1) TIM2_CH3 (1)	TIM2_CH2 (2)	TIM2_CH3 (2)			TIM2_CH4 TIM2_TRIG TIM2_COM
QSPI				QSPI_RX (1)	QSPI_TX (1)	QSPI_RX (2)	QSPI_TX (2)	
SDIO1				SDIO1 (1)			SDIO1 (2)	
SDIO2		SDIO2 (1)						SDIO2 (2)
USB				USB[0]	USB[1]			

1. If the mapping bit of the SYSCFG_CFGR1 register is cleared, the request with (1) superscript is mapped on this DMA channel
2. If the mapping bit of the SYSCFG_CFGR1 register is set, the request with (2) superscript is mapped on this DMA channel

7.4 **DMA** register description

7.4.1 **DMA** interrupt status register (**DMA_ISR**)

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
TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31, 27, 23, 19, 15, 11, 7, 3	**TEIFx** : Channel x transfer error flag (x = 1… 8) (Channel x transfer error flag) The hardware sets these bits. Write '1' to the corresponding bit of the DMA_IFCR register to clear the corresponding flag bit here. 0: There is no transmission error (TE) on channel x; 1: A transmission error (TE) has occurred on channel x.
Bit 30, 26, 22, 18, 14, 10, 6, 2	**HTIFx** : Half transfer flag of channel x (x = 1… 8) (Channel x half transfer flag) The hardware sets these bits. Write '1' to the corresponding bit of the DMA_IFCR register to clear the corresponding flag bit here. 0: There is no half-transmission event (HT) on channel x; 1: A half transfer event (HT) is generated on channel x.
Bit 29, 25,	**TCIFx** : Channel x transfer complete flag (x = 1… 8) (Channel x transfer complete flag) The hardware sets these bits. Write '1' to the corresponding bit of the DMA_IFCR register to clear the corresponding flag bit here.

21, 17, 13, 9, 5, 1
0: There is no transmission completion event (TC) on channel x;
1: A transmission completion event (TC) is generated on channel x.

Bit 28, 24, 20, 16, 12, 8, 4, 0
GIFx : Channel x global interrupt flag (x = 1… 8) (Channel x global interrupt flag)
The hardware sets these bits. Write '1' to the corresponding bit of the DMA_IFCR register to clear the corresponding flag bit here.
0: There is no TE, HT or TC event on channel x;
1: A TE, HT or TC event occurred on channel x.

Page 83

7.4.2 **DMA** interrupt flag clear register (**DMA_IFCR**)

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
CTE IF8	CHT IF8	CTC IF8	CG IF8	CTE IF7	CHT IF7	CTC IF7	CG IF7	CTE IF6	CHT IF6	CTC IF6	CG IF6	CTE IF5	CHT IF5	CTC IF5	CG IF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31, 27, 23, 19, 15, 11, 7, 3
CTEIFx : Clear the transmission error flag of channel x (x = 1… 8) (Channel x transfer error clear)
These bits are set and cleared by software.
0: does not work
1: Clear the corresponding TEIF flag in the DMA_ISR register.

Bit 30, 26, 22, 18, 14, 10, 6, 2
CHTIFx : Clear the half transfer flag of channel x (x = 1… 8) (Channel x half transfer clear)
These bits are set and cleared by software.
0: does not work
1: Clear the corresponding HTIF flag in the DMA_ISR register.

Bit 29, 25, 21, 17, 13, 9, 5, 1
CTCIFx : Clear the transfer complete flag of channel x (x = 1… 8) (Channel x transfer complete clear)
These bits are set and cleared by software.
0: does not work
1: Clear the corresponding TCIF flag in the DMA_ISR register.

Bit 28, 24, 20, 16, 12, 8, 4, 0
CGIFx : Clear the global interrupt flag of channel x (x = 1… 8) (Channel x global interrupt clear)
These bits are set and cleared by software.
0: does not work
1: Clear the corresponding GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register.

7.4.3 **DMA** channel **x** configuration register (**DMA_CCRx**) (**x = 1 … 8**)

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

Reset value: 0x0000 0000

31	30	29 28 27 26 25 24			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	MEM 2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC PINC CIRC DIR TEIE HTIE TCIE EN							
	rw	rw rw rw rw rw rw			rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 15
Reserved, always read as 0.

Bit 14
MEM2MEM : Memory to memory mode (Memory to memory mode)
This bit is set and cleared by software.
0: non-memory to memory mode
1: Start memory to memory mode

Bit 13: 12
PL[1 : 0] : Channel priority level
These bits are set and cleared by software.
00: low
01: Medium
10: high
11: highest

Page 84

Bit 11: 10	**MSIZE[1 : 0]** : Memory size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: reserved
Bit 9: 8	**PSIZE[1 : 0]** : Peripheral size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: reserved
Bit 7	**MINC** : Memory increment mode (Memory increment mode) This bit is set and cleared by software. 0: Do not perform memory address increment operation 1: Execute memory address increment operation
Bit 6	**PINC** : Peripheral increment mode This bit is set and cleared by software. 0: Do not perform peripheral address increment operation 1: Perform peripheral address increment operation
Bit 5	**CIRC** : Circular mode This bit is set and cleared by software. 0: Do not perform loop operation 1: Perform a loop operation
Bit 4	**DIR** : Data transfer direction This bit is set and cleared by software. 0: read from peripheral 1: Read from memory
Bit 3	**TEIE** : Transfer error interrupt enable This bit is set and cleared by software. 0: Disable TE interrupt 1: Enable TE interrupt
Bit 2	**HTIE** : Half transfer interrupt enable (Half transfer interrupt enable) This bit is set and cleared by software. 0: Disable HT interrupt 1: Allow HT interrupt
Bit 1	**TCIE** : Allow transfer complete interrupt (Transfer complete interrupt enable) This bit is set and cleared by software. 0: Disable TC interrupt 1: Allow TC interrupt
Bit 0	**EN** : Channel enable This bit is set and cleared by software. 0: The channel is not working 1: The channel is open

7.4.4 **DMA** channel **x** transfer quantity register (**DMA_CNDTRx**) (**x = 1 … 8**)

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

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
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Page 85

Bit 31: 16	Reserved, always read as 0.
Bit 15:0	**NDT[15 : 0]** : Number of data to transfer The number of data transfers is 0 to 65535. This register can only be written when the channel is not working (EN=0 of DMA_CCRx). After the channel is turned on, this register becomes read-only, indicating the number of bytes remaining to be transferred. Register content after each DMA transfer Decreasing. After the data transfer is over, the contents of the register will either become 0; or when the channel is configured in auto-reload mode, register The content of the device will be automatically reloaded to the value of the previous configuration. When the content of the register is 0, no data transmission will occur regardless of whether the channel is open or not.

7.4.5 **DMA** channel **x** peripheral address register (**DMA_CPARx**) (**x = 1 … 8**)

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

Reset value: 0x0000 0000

This register cannot be written when the channel is turned on (EN=1 of DMA_CCRx).

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

Bit 31:0	**PA[31 : 0]** : Peripheral address The base address of the peripheral data register is used as the source or destination of data transfer. When PSIZE='01' (16 bits), the PA[0] bit is not used. The operation is automatically aligned with the halfword address. When PSIZE='10' (32 bits), PA[1:0] bits are not used. The operation is automatically aligned with the word address.

7.4.6 **DMA** channel **x** memory address register (**DMA_CMARx**) (**x = 1** … **8**)

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

Reset value: 0x0000 0000

This register cannot be written when the channel is turned on (EN=1 of DMA_CCRx).

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

Bit 31:0	**MA[31 : 0]** : Memory address The memory address serves as the source or destination of data transfer. When MSIZE='01' (16 bits), the MA[0] bit is not used. The operation is automatically aligned with the halfword address. When MSIZE='10' (32 bits), the MA[1:0] bits are not used. The operation is automatically aligned with the word address.

8. Interrupts and events

8.1 Nested Vectored Interrupt Controller

feature

- All interrupts can be masked (except NMI)
- 16 programmable priority levels (using 4-bit interrupt priority levels)
- Low-latency exception and interrupt handling
- Power management control
- Realization of System Control Register

The Nested Vectored Interrupt Controller (NVIC) is closely connected with the interface of the processor core, which can realize low-latency interrupt processing and efficient processing Late interruption.

The nested vectored interrupt controller manages interrupts including core exceptions. For more exceptions and NVIC programming instructions, please refer to CPU technology Reference book.

8.1.1 System tick (**SysTick**) calibration value register

The system tick calibration value is fixed at 9000. When the system tick clock is set to 9MHz (the maximum value of HCLK/8), a 1mS time base is generated allow.

8.1.2 Interrupt and exception vectors

The following table lists the product vector table.

Table **33.** Product vector table

Location	priority	Priority type	name	illustrate	address
	-	-	-	Reserve	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004

	-2	fixed	NMI	Non-maskable interrupt	
RCC clock security System (CSS) Link Receive NMI vector	0x0000_0008				
	-1	fixed	Hardware failure (HardFault)	All types of failures	0x0000_000C
	0	Storage management (MemManage) can be set		Memory management	0x0000_0010
	1	Can be set	Bus fault (BusFault)	Prefetch instruction failed, memory access failed 0x0000_0014	
	2	Can be set	Application Fault (UsageFault)	Undefined instruction or illegal status	0x0000_0018
	-	-	-	Reserve	0x0000_001C
~0x0000_002B					
	3	Can be set	SVCall	Call 0x0000_002C through the system service of the SWI instruction	
	4	Debug monitor (DebugMonitor) can be set		Debug monitor	0x0000_0030
	-	-	-	Reserve	0x0000_0034
	5	Can be set	PendSV	Suspendable system services	0x0000_0038
	6	Can be set	SysTick	System tick timer	0x0000_003C
0	7	Can be set	WWDG	Window timer interrupt	0x0000_0040
1	-	-	-	Reserve	0x0000_0044
2	9	Can be set	TAMPER	Intrusion detection interrupted	0x0000_0048
3	10	Can be set	RTC	Real-time clock (RTC) global interrupt 0x0000_004C	

Location	priority	Priority type	name	illustrate	address
4	-	-	-	Reserve	0x0000_0050
5	12	Can be set	RCC	Reset and clock control (RCC) interrupt 0x0000_0054	
6	13	Can be set	EXTI0	EXTI line 0 interrupt	0x0000_0058
7	14	Can be set	EXTI1	EXTI line 1 interrupted	0x0000_005C
8	15	Can be set	EXTI2	EXTI line 2 interrupted	0x0000_0060
9	16	Can be set	EXTI3	EXTI line 3 interrupted	0x0000_0064
10	17	Can be set	EXTI4	EXTI line 4 interrupted	0x0000_0068
11	18	Can be set	DMA1 channel 1	DMA1 channel 1 global interrupt	0x0000_006C
12	19	Can be set	DMA1 channel 2	DMA1 channel 2 global interrupt	0x0000_0070
13	20	Can be set	DMA1 channel 3	DMA1 channel 3 global interrupt	0x0000_0074
14	twenty one	Can be set	DMA1 channel 4	DMA1 channel 4 global interrupt	0x0000_0078
15	twenty two	Can be set	DMA1 channel 5	DMA1 channel 5 global interrupt	0x0000_007C
16	twenty three	Can be set	DMA1 channel 6	DMA1 channel 6 global interrupt	0x0000_0080
17	twenty four	Can be set	DMA1 channel 7	DMA1 channel 7 global interrupt	0x0000_0084
18	25	Can be set	ADC1	ADC1 interrupt	0x0000_0088
19	26	Can be set	CAN1	CAN1 interrupt	0x0000_008C
20	-	-	-	Reserve	0x0000_0090
twenty one	-	-	-	Reserve	0x0000_0094
twenty two	-	-	-	Reserve	0x0000_0098
twenty three	30	Can be set	EXTI9_5	EXTI line [9:5] is interrupted	0x0000_009C
twenty four	31	Can be set	TIM1_BRK	TIM1 disconnect interrupt	0x0000_00A0
25	32	Can be set	TIM1_UP	TIM1 update interrupt	0x0000_00A4
26	33	Can be set	TIM1_TRG_COM	TIM1 trigger and communication interrupt	0x0000_00A8
27	34	Can be set	TIM1_CC	TIM1 capture compare interrupt	0x0000_00AC
28	35	Can be set	TIM3	TIM3 global interrupt	0x0000_00B0
29	36	Can be set	TIM4	TIM4 global interrupt	0x0000_00B4
30	37	Can be set	TIM5	TIM5 global interrupt	0x0000_00B8
31	38	Can be set	TIM6	TIM6 global interrupt	0x0000_00BC
32	39	Can be set	TIM7	TIM7 global interrupt	0x0000_00C0
33	40	Can be set	I2C1	I2C1 global interrupt	0x0000_00C4
34	41	Can be set	I2C2	I2C2 global interrupt	0x0000_00C8

35	42	Can be set	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	Can be set	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	Can be set	UART1	UART1 global interrupt	0x0000_00D4
38	45	Can be set	UART2	UART2 global interrupt	0x0000_00D8
39	46	Can be set	UART3	UART3 global interrupt	0x0000_00DC
40	47	Can be set	EXTI15_10	EXTI line [15:10] is interrupted	0x0000_00E0
41	48	Can be set	RTC_ALARM	RTC alarm interrupt 0x0000_00E4 connected to EXTI17	

Page 88

Location	priority	Priority type	name	illustrate	address
42	49	Can be set	USB	Wake from USB standby connected to EXTI18 Interrupt	0x0000_00E8
43	50	Can be set	TIM2_BRK	TIM2 disconnect interrupt	0x0000_00EC
44	51	Can be set	TIM2_UP	TIM2 update interrupt	0x0000_00F0
45	52	Can be set	TIM2_TRG_COM	TIM2 trigger and communication interrupt	0x0000_00F4
46	53	Can be set	TIM2_CC	TIM2 capture, compare interrupt	0x0000_00F8
47	54	Can be set	DMA1 channel 8	DMA1 channel 8 global interrupt	0x0000_00FC
48	55	Can be set	TK80	TK80 global interrupt	0x0000_0100
49	56	Can be set	SDIO1	SDIO1 global interrupt	0x0000_0104
50	57	Can be set	SDIO2	SDIO2 global interrupt	0x0000_0108
51	58	Can be set	SPI3	SPI3 global interrupt	0x0000_010C
52	59	Can be set	UART4	UART4 global interrupt	0x0000_0110
53	60	Can be set	UART5	UART5 global interrupt	0x0000_0114
54	-	-	-	Reserve	0x0000_0118
55	62	Can be set	TIM8	TIM8 global interrupt	0x0000_011C
56	63	Can be set	DMA2 channel 1	DMA2 channel 1 global interrupt	0x0000_0120
57	64	Can be set	DMA2 channel 2	DMA2 channel 2 global interrupt	0x0000_0124
58	65	Can be set	DMA2 channel 3	DMA2 channel 3 global interrupt	0x0000_0128
59	66	Can be set	DMA2 channel 4	DMA2 channel 4 global interrupt	0x0000_012C
60	67	Can be set	DMA2 channel 5	DMA2 channel 5 global interrupt	0x0000_0130
61	68	Can be set	TIM9	TIM9 global interrupt	0x0000_0134
62	69	Can be set	TIM10	TIM10 global interrupt	0x0000_0138
63	70	Can be set	CAN2	CAN2 global interrupt	0x0000_013C
64	-	-	-	Reserve	0x0000_0140
65	-	-	-	Reserve	0x0000_0144
66	-	-	-	Reserve	0x0000_0148
67	74	Can be set	USB	USB global interrupt	0x0000_014C
68	75	Can be set	DMA2 channel 6	DMA2 channel 6 global interrupt	0x0000_0150
69	76	Can be set	DMA2 channel 7	DMA2 channel 7 global interrupt	0x0000_0154
70	77	Can be set	DMA2 channel 8	DMA2 channel 8 global interrupt	0x0000_0158
71	-	-	-	Reserve	0x0000_015C
72	79	Can be set	I2C3	I2C3 global interrupt	0x0000_0160
73	80	Can be set	I2C4	I2C4 global interrupt	0x0000_0164
74	-	-	-	Reserve	0x0000_0168
75	-	-	-	Reserve	0x0000_016C
76	-	-	-	Reserve	0x0000_0170
77	-	-	-	Reserve	0x0000_0174
78	-	-	-	Reserve	0x0000_0178
79	-	-	-	Reserve	0x0000_017C

Page 89

Location	priority	Priority type	name	illustrate	address
80	-	-	-	Reserve	0x0000_0180
81	88	Can be set	FPU	FPU global interrupt	0x0000_0184
82	-	-	-	Reserve	0x0000_0188
83	-	-	-	Reserve	0x0000_018C
84	91	Can be set	SPI4	SPI4 global interrupt	0x0000_0190
85	-	-	-	Reserve	0x0000_0194
86	93	Can be set	TCHPAD	TCHPAD global interrupt	0x0000_0198
87	94	Can be set	QSPI	QPI4 global interrupt	0x0000_019C
88	95	Can be set	LCD-TFT	LCD global interrupt	0x0000_01A0
89	-	-	-	Reserve	0x0000_01A4
90	97	Can be set	I2S1	I2S1 global interrupt	0x0000_01A8

8.2 External interrupt **/** event controller (**EXTI**)

The external interrupt and time controller (EXTI) manages external and internal asynchronous events/interrupts, and generates corresponding event requests to the CPU/interrupts The controller and the wake-up request to the power management.

There are 21 edge detectors that can generate event/interrupt requests. Each input line can be independently configured for input type (pulse or suspension) and pair The corresponding trigger event (triggering on rising or falling edges or both edges). Each input line can be shielded independently. Pending register hold The interrupt request of the status line.

8.2.1 Main features

The main features of the EXTI controller are as follows:

- Each interrupt/event has independent triggering and shielding
- Each interrupt line has a dedicated status bit
- Support up to 21 software interrupt/event requests
- Detect external signals whose pulse width is lower than the APB2 clock width. Refer to the relevant parameters in the electrical characteristics section of the data sheet.

Page 90

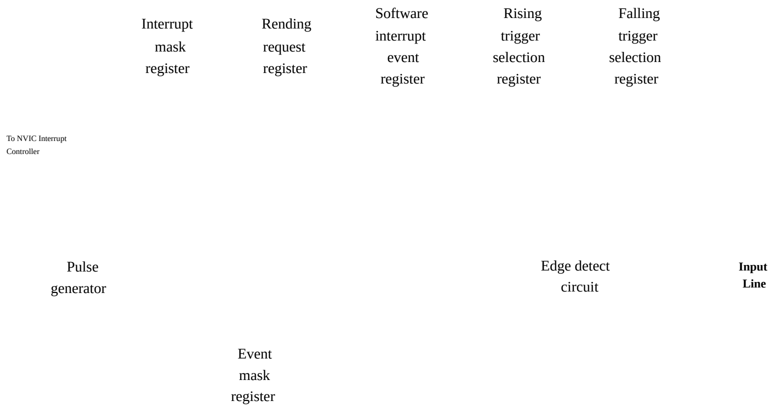
8.2.2 Block Diagram

Figure **17.** Block diagram of external interrupt / event controller

AMBA APB bus

PCLK2

Peripheral interface


Interrupt mask register
Rending request register
Software interrupt event register
Rising trigger selection register
Falling trigger selection register
To NVIC Interrupt Controller
Pulse generator
Edge detect circuit
Input Line
Event mask register

8.2.3 Wake-up event management

TK499 can handle external or internal events to wake up the core (WFE). Wake-up events can be generated by the following configuration:

- Enable an interrupt in the control register of the peripheral, but not in the NVIC, and enable it in the system control register of the CPU SEVONPEND bit. When the CPU recovers from WFE, it is necessary to clear the interrupt pending bit of the corresponding peripheral and the peripheral NVIC interrupt commun Track suspend bit (in the NVIC interrupt clear suspend register).
- Configure an external or internal EXTI line as the event mode. When the CPU recovers from WFE, because the suspend bit of the corresponding event line is not If it is set, it is not necessary to clear the interrupt pending bit of the corresponding peripheral or the NVIC interrupt channel pending bit.

Use the external I/O port as a wake-up event, please refer to the function description in the next section

8.2.4 Function description

To generate an interrupt, you must first configure and enable the interrupt line. Set up 2 trigger registers according to the required edge detection, and at the same time

Write a '1' to the corresponding bit of the interrupt mask register to enable interrupt request. When the expected edge occurs on the external interrupt line, an interrupt will be generated Request, the corresponding suspend bit is also set to '1'. Write a '1' to the corresponding bit of the suspend register to clear the interrupt request.

If you need to generate an event, you must first configure and enable the event line. According to the required edge detection by setting 2 trigger registers, the same Write '1' in the corresponding bit of the event mask register to allow event requests. When the required edge occurs on the event line, an event will be generated. For pulse, the corresponding suspend bit is not set to '1'.

By writing '1' in the software interrupt/event register, an interrupt/event request can also be generated by software.

Hardware interrupt selection

Use the following process to configure 21 lines as interrupt sources:

- Configure the mask bits of 21 interrupt lines (EXTI_IMR)

- Configure the trigger selection bits of the selected interrupt line (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits corresponding to the NVIC interrupt channel of the external interrupt controller (EXTI), so that the Request can be correctly responded

Hardware event selection

Through the following process, 21 lines can be configured as event sources:

- Configure 2 event line shielding bits (EXTI_EMR)
- Configure the trigger selection bit of the event line (EXTI_RTSR and EXTI_FTSR)

Software interrupt / event selection

21 lines can be configured as software interrupt/event lines. The following is the process of generating a software interrupt:

- Configure 21 interrupt/event line shielding bits (EXTI_IMR, EXTI_EMR)
- Set the request bit of the software interrupt register (EXTI_SWIER)

8.2.5 External interrupt / event line image

The general-purpose I/O ports are connected to 16 external interrupt/event lines as shown in the following figure:

Figure **18.** External interrupt general **I/O** image

EXTI0[3:0] bits in SYFCFG _EXTICR1 register

PA 0
PB 0
PD 0

EXTI0

EXTI3[3:0] bits in SYFCFG _EXTICR1 register

PA 3
PB 3
PD 3

EXTI3

EXTI4[3:0] bits in SYFCFG _EXTICR1 register

PA 4
PB 4

EXTI4

EXTI13[3:0] bits in SYFCFG _EXTICR4 register

PA 13
PB 13
PC 13

EXTI13

EXTI15[3:0] bits in SYFCFG _EXTICR4 register

PA 15

PB 15 PC 15	EXTI15

The connections of the other three other external interrupt/event controllers are as follows:

- EXTI line 16, 19, 20 reserved
- EXTI line 17 is connected to the RTC alarm event
- EXTI line 18 connected to USB wake-up event

8.3 **EXTI** register description

8.3.1 Interrupt Mask Register (**EXTI_IMR**)

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											MR20	MR19	MR18	MR17	MR16
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 21	Reserve.
Bit 20:0	**IMRx** : Interrupt Mask on line x 1 = open interrupt request from line x 0 = shield the interrupt request from line x

8.3.2 Event Mask Register (**EXTI_EMR**)

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											MR20	MR19	MR18	MR17	MR16
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 21	Reserve.
Bit 20:0	**EMRx** : Event Mask on line x 1 = Open event requests from line x 0 = shield event requests from line x

Page 94

8.3.3 Rising edge trigger selection register (**EXTI_RTSR**)

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											TR20	TR19	TR18	TR17	TR16
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 21	Reserve.
Bit 20:0	**TRx** : Rising trigger event configuration bit of line x 1 = Allow rising edge triggers on input line x (interrupts and events) 0 = Disable rising edge triggers on input line x (interrupts and events)

Note: The external wake-up lines are edge-triggered, and glitch signals cannot appear on these lines.
When writing the *EXTI_RTSR* register, the rising edge signal on the external interrupt line cannot be recognized, and the suspend bit will not be set.
On the same interrupt line, both rising and falling edge triggers can be set at the same time. That is, any edge can trigger an interrupt.

8.3.4 Falling edge trigger selection register (**EXTI_FTSR**)

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											TR20	TR19	TR18	TR17	TR16
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 21	Reserve.
Bit 20:0	**TRx** : Falling trigger event configuration bit of line x 1 = Allow falling edge triggers on input line x (interrupts and events) 0 = Disable falling edge triggers on input line x (interrupts and events)

Note: The external wake-up lines are edge-triggered, and glitch signals cannot appear on these lines.
When writing the *EXTI_FTSR* register, the falling edge signal on the external interrupt line cannot be recognized, and the suspend bit will not be set.
On the same interrupt line, both rising and falling edge triggers can be set at the same time. That is, any edge can trigger an interrupt.

Page 95

8.3.5 Software interrupt event register (**EXTI_SWIER**)

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											SW IER20	SW IER19	SW IER18	SW IER17	SW IER16

											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW IER15	SW IER14	SW IER13	SW IER12	SW IER11	SW IER10	SW IER9	SW IER8	SW IER7	SW IER6	SW IER5	SW IER4	SW IER3	SW IER2	SW IER1	SW IER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 21	Reserve.
Bit 20:0	**SWIERx** : Software interrupt on line x When this bit is '0', writing '1' will set the corresponding suspend bit in EXTI_PR. If in EXTI_INTMASK and If the interrupt is allowed in EXTI_EVNTMASK, an interrupt will be generated at this time. Note: By clearing the corresponding bit of EXTI_PEND (write '1'), this bit can be cleared to '0'.

8.3.6 Software interrupt event registration (**EXTI_PR**)

Offset address: 0x14

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve											PR20	PR19	PR18	PR17	PR16
											rc w1	rc w1	rc w1	rc w1	rc w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1	rc w1

Bit 31: 21	Reserve.
Bit 20:0	**PRx** : Pending bit 1 = The selected trigger request has occurred 0 = No trigger request occurred When the selected edge event occurs on the external interrupt line, this bit is set to '1'. Write a '1' in this bit to clear it, or through Cleared by changing the polarity of edge detection.

9. Analog / digital conversion (ADC)

9.1 Introduction to ADC

The 12-bit ADC is a successive approximation analog-to-digital converter (SAR A/D converter). It has 10 channels.

The A/D converter supports multiple working modes: single conversion and continuous conversion mode, and you can select the channel to automatically scan. A/D conversion start There are software settings, external pin triggers, and start of various timers.

The window comparator (analog watchdog) allows the application to detect whether the input voltage exceeds the high/low threshold set by the user.

ADC input clock must not exceed 15MHz, it is generated by PCLK2 through frequency division.

9.2 Main features of ADC

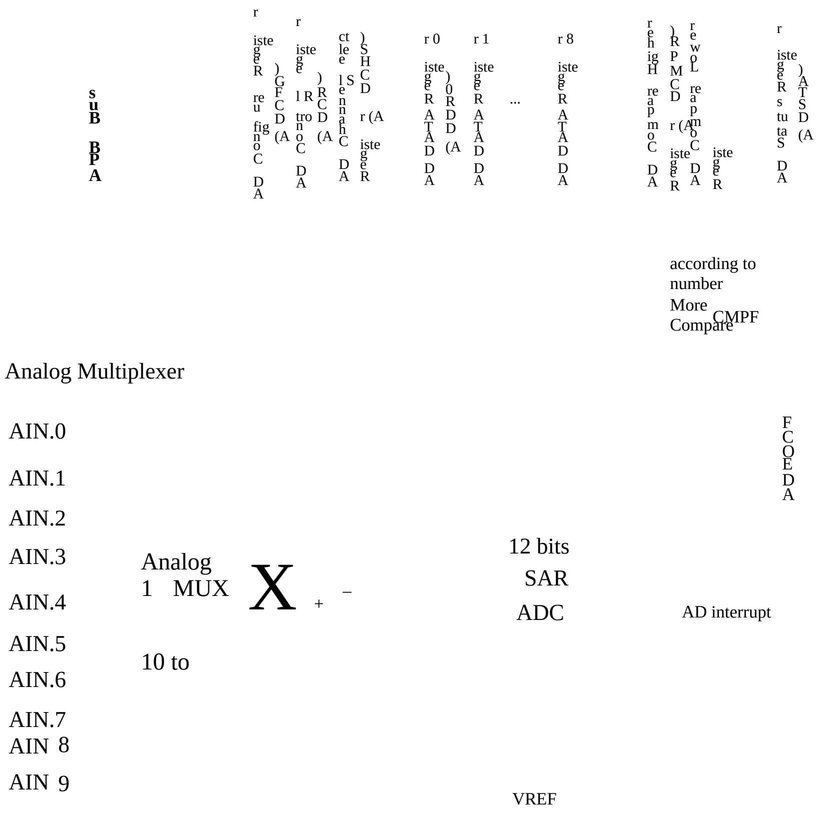
- 12-bit SAR ADC, up to 10 external input channels.
- Up to 1Msps conversion rate;
- Support multiple working modes
  - Single conversion mode: A/D conversion completes one conversion in the specified channel
  - Single cycle scan mode: A/D conversion completes one cycle (from low sequence number channel to high sequence number channel) conversion on all specified channels
  - Continuous scan mode: A/D conversion continuously executes single-cycle scan mode until the software stops A/D conversion
- Support DMA transfer
- A/D conversion start condition
  - Software start
  - External trigger start
  - Timer1/2/3/4 match or TRGO signal

- Analog watchdog, the conversion result can be compared with the specified value, when the conversion value matches the set value, the user can set whether to generate Interrupt request;
- Analog TouchPad, external resistive touch screen device, can be used as the control circuit of the resistive touch screen.

9.3 **ADC** function description

The following figure shows the AD block diagram

Figure **6. AD** block diagram


APB Bus
ADC Config Register (ADCFG)
ADC Control Register (ADCR)
ADC Channel Select Register (ADCHS)
ADC Data Register 0 (ADDR0)
ADC Data Register 1
...
ADC Data Register 8
ADC Compare Higher Register (ADCMPR)
ADC Compare Lower Register
ADC Status Register (ADSTA)
according to number More Compare
CMPF
ADEOF
Analog Multiplexer
AIN.0
AIN.1
AIN.2
AIN.3
AIN.4
AIN.5
AIN.6
AIN.7
AIN 8
AIN 9
Analog 1 MUX
10 to
X
+
–
12 bits SAR ADC
AD interrupt
VREF

9.3.1 **ADC** switch control

The ADC can be powered on by setting the ADEN bit in the ADCFG register. When the ADEN bit is set for the first time, it will

Wake up in the state.

After ADC power-on delay for a period of time (tSTAB), the ADST bit is set to start conversion.

The conversion can be stopped by clearing the ADST bit, and the ADEN bit can be set to power-down mode.

9.3.2 ADC clock

The ADCCLK clock provided by the clock controller is synchronized with PCLK2 (APB2 clock). The RCC controller provides a dedicated
For the programmable prescaler used, see the reset and clock control (RCC) chapter for details.

9.3.3 Channel selection

There are 10 external input channels.

Each external input channel has an independent enable bit, which can be set by setting the corresponding bit in the ADCHS register.

9.4 ADC working mode

9.4.1 Single conversion mode

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

- Set ADST through software, external trigger input and timer overflow, CONT=0, start A/D conversion.
- When the A/D conversion is completed, the data value of the A/D conversion will be stored in the A/D data registers ADDATA and ADDRn.
- The A/D conversion is complete, and the ADIF bit of the status register ADSTA is set to 1. If the ADIE bit of the control register ADCRL is set at this time,
  An AD conversion end interrupt request will be generated.
- During A/D conversion, the ADST bit remains at 1. When the A/D conversion is over, the ADST bit is automatically cleared to 0, and the A/D converter enters idle mode.

Note: In the single conversion mode, if the software enables more than one channel, the channel with the smallest serial number will be converted and the other channels wi

Figure **7.** Timing diagram of single conversion mode

9.4.2 Single cycle scan mode

In the single-cycle scan mode, an A/D conversion will be performed from the enabled channel with the smallest sequence number to the channel with the largest sequence number. The c
Down:

Software or external trigger sets ADST to start the A/D conversion from the channel with the smallest sequence number to the channel with the largest sequence number.

After each channel of A/D conversion is completed, the A/D conversion value will be sequentially loaded into the data register of the corresponding channel, and the ADIF conversion er
If the conversion end interrupt is set, an interrupt request will be generated after all channel conversions are completed.

After the conversion is completed, the ADST bit is automatically cleared to 0, and the A/D converter enters an idle state.

Page 99

Figure **8.** Enable channel conversion timing diagram under single-cycle scan

9.4.3 Continuous Scan Mode

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

Software or external trigger sets ADST to start the A/D conversion from the channel with the smallest sequence number to the channel with the largest sequence number.

When the A/D conversion of all channels is completed once, the A/D conversion values will be loaded into the corresponding data registers in order, and the ADIF conversion will end The flag is set. If the conversion end interrupt is set, an interrupt request will be generated after all channel conversions are completed.

As long as the ADST bit remains at 1, repeat steps 2 to 3. When the ADST bit is cleared to 0, the A/D conversion stops and the A/D converter enters idle state. When ADST is cleared to 0, the A/D conversion will complete the current conversion.

Figure **9.** Continuous scan mode enable channel conversion timing diagram

9.4.4 DMA request

The value of channel conversion during single cycle scan and continuous scan is stored in the data register (ADDRn) of the respective channel, and the last converted value The result is also stored in the ADDATA register. During DMA transfer, you can choose to transfer the data of a specific channel, or transfer all scans The result of the channel.

Page 100

9.4.5 Sampling frequency setting

The ADC clock ADCLK is obtained by dividing the frequency of PCLK2, and the frequency division coefficient can be determined by setting the ADCPRE bit in the ADCFG register. That is, PCLK2 (n+1) is divided into ADC clock. When the ADC is working, it is sampled every 15 ADCLK cycles, that is, the sampling frequency is

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

9.5 Data alignment

The ALIGN bit in the ADCR register selects the alignment of data storage after conversion. Data may be left-justified or right-justified, as shown in FIG 10 Suo Show.

Figure **10.** Data alignment

Data is right aligned

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

Data is aligned to the left

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

9.6 External trigger conversion

ADC conversion can be triggered by external events (eg timer capture, EXTI line). If the TRGEN bit of the ADCFG register is set, You can use an external event to trigger the conversion. The external trigger source can be selected by setting the TRGSEL bit.

For the specific external trigger source selection, please refer to the description of AD control register (ADCR) bit[6:4] TRGSEL bits.

9.7 AD conversion result monitoring in window comparator mode

In the compare mode, two compare registers are provided, the upper limit and the lower limit. The monitoring channel can be selected by setting the CMPCH bit by software.

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

When CPMHDATA<CPMLDATA, the comparison result is greater than or equal to the specified value of CMPHDATA and less than the specified value of CMPLDATA Value, the ADWIF bit of the status register ADSTA is 1.

If the ADWIE of the control register ADCR is set, an ADINT interrupt request will be generated.

9.8 AD conversion result monitoring in touch screen mode

Two data registers of X axis and Y axis are provided in the touch screen mode. The touch screen mode can be turned on by setting the TPEN bit in the software. when After the AD conversion result is less than the specified value of TPCMP in ADCTPCR and the number of times reaches the specified value of TPCNT N+1, the status register ADSTA The ADTPIF position is 1.

If the ADTPIE of the control register ADCR is set, an ADINT interrupt request will be generated.

The ADC will start the continuous scan mode when the touch screen mode is enabled.

9.9 ADC register description

9.9.1 A/D Data Register (ADC_ADDATA)

Address offset: 0x0

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										VALID	OVERRUN	CHANNELSEL			
										r	r	r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r															

Bit 31: 22	Reserve. Must remain at 0.
Bit 21	**VALID** : Valid flag (read only) (Valid flag) 1 = DATA[11:0] bit data is valid. 0 = DATA[11:0] bit data is invalid. After the conversion of the corresponding analog channel is completed, set this bit. After reading the ADDATA register, this bit will be cleared by the hardware.

Bit 20	**OVERRUN** : Data overwrite flag bit (read only) (Overrun flag) 1 = DATA [11:0] The data is overwritten. 0 = DATA [11:0] The last conversion result of data. Before the new conversion result is loaded into the register, if the data of DATA[11:0] has not been read, OVERRUN will be set to 1. read After the ADDATA register, this bit is cleared by hardware.
Bit 19: 16	**CHANNELSEL** : The 4 digits show the channel corresponding to the current data (Channel selection) 0000 = conversion data of channel 0 0001 = conversion data of channel 1 0010 = conversion data of channel 2 0011 = conversion data of channel 3 0100 = conversion data of channel 4 0101 = conversion data of channel 5 0110 = conversion data of channel 6 0111 = conversion data of channel 7 1000 = conversion data of channel 8 1001 = conversion data of channel 9 Other: invalid
Bit 15:0	**DATA** : 12-bit A/D conversion result (Transfer data) Align left or right according to the setting.

9.9.2 **A/D** configuration register (**ADC_ADCFG**)

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									ADCPRE			Reserve		ADWEN	ADEN
									rw				rw	rw	rw

Bit 31: 7	Reserve. Must remain at 0.
Bit 6: 4	**ADCPRE** : ADC prescaler (ADC prescaler) Set "1" or clear "0" by software to determine ADC clock frequency n: PCLK2 2* (n+1) divided as ADC clock
Bit 3: 2	Reserve
Bit 1	**ADWEN** : A/D window comparator enable (ADC window comparison enable) 1 = A/D window comparator is enabled 0 = A/D window comparator is disabled
Bit 0	**ADEN** : A/D conversion enable (ADC enable) 1 = enable 0 = disabled

9.9.3 **AD** control register (**ADC_ADCR**)

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
CMPCH				ALIGN	ADMD		ADST	Reserve	TRGSEL			DMAEN	TRGEN	ADWIE	ADIE
rw				rw	rw		rw		rw			rw	rw	rw	rw

Bit 31: 16	Reserve. Must remain at 0.
Bit 15: 12	**CMPCH** : Window comparison channel selection 0000 = select compare channel 0 conversion result 0001 = Select to compare the conversion result of channel 1 0010 = Select to compare the conversion result of channel 2 0011 = Select to compare the conversion result of channel 3 0100 = Select to compare the conversion result of channel 4 0101 = Select to compare the conversion result of channel 5 0110 = Select to compare the conversion result of channel 6 0111 = Select to compare the conversion result of channel 7 1000 = Select to compare the conversion result of channel 8 1001 = Select to compare the conversion result of channel 9

	1111 = all scan channels Other: invalid
Bit 11	**ALIGN** : Data alignment 0: Right justified 1: Left aligned
Bit 10: 9	**ADMD** : A/D conversion mode (ADC mode) 00: Single conversion 01: Single cycle scan 10: Continuous scanning When changing the conversion mode, the software must first disable the ADST bit.
Bit 8	**ADST** : A/D conversion start (ADC start) 1 = start of conversion. 0 = End of conversion or enter idle state. There are two ways to set ADST: In single mode or single cycle mode, after the conversion is completed, ADST will be automatically cleared by the hardware, In continuous scan mode, A/D conversion will continue until the software writes 0 to this bit or the system is reset.
Bit 7	Reserve

Page 103

Bit 6: 4	**TRGSEL** : External trigger selection 000: TIM1_CC1 001: TIM1_CC2 010: TIM1_CC3 011: TIM2_CC2 100: TIM3_TRGO 101: TIM4_CC4 110: TIM3_CC1 111: EXTI line 11
Bit 3	**DMAEN** : DMA enable (Direct memory access enable) 1 = DMA request is enabled 0 = DMA disabled
Bit 2	**TRGEN** : External hardware trigger source (External trigger enable) 1 = Use external trigger signal to start A/D conversion 0 = Do not use external trigger signal to start A/D conversion
Bit 1	**ADWIE** : A/D window comparator interrupt enable (ADC window comparator interrupt enable) 1 = Enable A/D window comparator interrupt 0 = Disable A/D window comparator interrupt
Bit 0	**ADIE** : A/D interrupt enable (ADC interrupt enable) 1 = Enable A/D interrupt 0 = Disable A/D interrupt If ADINT is set, an interrupt request is generated after the A/D conversion is completed.

9.9.4 **AD** channel selection register (**ADC_ADCHS**)

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						CHEN9	CHEN8	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 9	Reserve. Must remain at 0.
Bit 9	**CHEN9** : Analog input channel 9 enable (Analog input channel 9 enable) 1 = enable 0 = disabled
Bit 8	**CHEN8** : Analog input channel 8 enable (Analog input channel 8 enable) 1 = enable 0 = disabled
Bit 7	**CHEN7** : Analog input channel 7 enable (Analog input channel 7 enable) 1 = enable 0 = disabled
Bit 6	**CHEN6** : Analog input channel 6 enable (Analog input channel 6 enable) 1 = enable 0 = disabled
Bit 5	**CHEN5** : Analog input channel 5 enable (Analog input channel 5 enable) 1 = enable 0 = disabled
Bit 4	**CHEN4** : Analog input channel 4 enable (Analog input channel 4 enable) 1 = enable 0 = disabled

Bit 3	**CHEN3** : Analog input channel 3 enable (Analog input channel 3 enable) 1 = enable 0 = disabled
Bit 2	**CHEN2** : Analog input channel 2 enable (Analog input channel 2 enable) 1 = enable 0 = disabled
Bit 1	**CHEN1** : Analog input channel 1 enable (Analog input channel 1 enable) 1 = enable 0 = disabled
Bit 0	**CHEN0** : Analog input channel 0 enable (Analog input channel 0 enable) 1 = enable 0 = disabled

Note: If the channel enable is all *0* , then channel *0 is* enabled.

9.9.5 A/D window compare register (ADC_ADCMPR)

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

Bit 31: 9	Reserve
Bit 27: 16	**CMPHDATA** : Compare data high limit The 12-bit value will be compared with the conversion result of the specified channel.
Bit 15: 12	Reserve
Bit 11:0	**CMPLDATA** : Compare data low limit The 12-bit value will be compared with the conversion result of the specified channel.

9.9.6 A/D Status Register (ADC_ADSTA)

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			OVERRUN[9:0]									Reserve			
			r												r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID[8:0]								CHANNEL[3:0]				Reserve	BUSY	ADWIF	ADIF
r								r					r	rc_w1	rc_w1

Bit 31: 29	Reserve
Bit 28: 20	**OVERRUN** : Data overrun flag of channel 0〜8 (Overrun flag) Read only.

Bit 19: 17	Reserve
Bit 16: 9	**VALID** : valid flag bit of channel 0〜8 (Valid flag) Read only.

Bit 8: 4	**CHANNEL** : Current conversion channel (Current conversion channel) When BUSY = 1, these 4 bits indicate the channel that is being converted. When BUSY = 0, it indicates the channel that can be converted next.
Bit 3	Reserve
Bit 2	**BUSY** : busy/idle (Busy) 1 = A/D converter is busy 0 = A/D converter is idle
Bit 1	**ADWIF** : compare flag bit (ADC window comparator interrupt flag) The result of the selected A/D conversion channel is greater than or equal to ADCMPHR or less than ADCMPLR, and this bit is set to 1. Write 1 to clear this bit.
Bit 0	**ADIF** : A/D conversion end flag (ADC interrupt flag) This bit is set by the hardware at the end of the channel group conversion, and cleared by the software 1 = A/D conversion completed 0 = A/D conversion is not completed Write 1 to this flag to clear it.

9.9.7 A/D data register (ADC_ADDR0~9)

Address offset: 0x18~0x40

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

Bit 31: 22	Reserve
Bit 21	**VALID** : Valid flag (read only) (Valid flag) 1 = DATA[11:0] bit data is valid. 0 = DATA[11:0] bit data is invalid. After the conversion of the corresponding analog channel is completed, set this bit. After reading the ADDATA register, this bit will be cleared by the hardware.
Bit 20	**OVERRUN** : Data overwrite flag bit (read only) (Overrun flag) 1 = DATA [11:0] The data is overwritten. 0 = DATA [11:0] The last conversion result of data. Before the new conversion result is loaded into the register, if the data of DATA[11:0] has not been read, OVERRUN will be set to 1. read After the ADDATA register, this bit is cleared by hardware.
Bit 19: 16	Reserve
Bit 15:0	**DATA** : 12-bit A/D conversion result of channel 0~9 (Transfer data) Align left or right according to the setting.

9.9.8 A/D touch screen X+ data register (ADC_TPXDR)

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

Bit 31: 22	Reserve
Bit 21	**valid_x** : valid flag bit (read only) (Valid flag) 1 = DATA[11:0] bit data is valid. 0 = DATA[11:0] bit data is invalid.

	After the conversion of the corresponding analog channel is completed, set this bit. After reading the register, this bit will be cleared by the hardware.
Bit 20	**overrun_x** : Data overwrite flag bit (read only) (Overrun flag) 1 = DATA [11:0] The data is overwritten. 0 = DATA [11:0] The last conversion result of data. Before the new conversion result is loaded into the register, if the data of DATA[11:0] has not been read, OVERRUN will be set to 1. Read and send After registering, this bit is cleared by hardware.
Bit 19: 16	Reserve
Bit 15:0	**data_x** : 12-bit A/D conversion result of channel 0~9 (Transfer data) Align left or right according to the setting.

9.9.9 A/D touch screen **Y+** data register (**ADC_YPDR**)

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

Bit 31: 22	Reserve
Bit 21	**valid_y** : valid flag bit (read only) (Valid flag) 1 = DATA[11:0] bit data is valid. 0 = DATA[11:0] bit data is invalid. After the conversion of the corresponding analog channel is completed, set this bit. After reading the register, this bit will be cleared by the hardware.

Bit 20	**overrun_y** : Data overwrite flag bit (read only) (Overrun flag) 1 = DATA [11:0] The data is overwritten. 0 = DATA [11:0] The last conversion result of data. Before the new conversion result is loaded into the register, if the data of DATA[11:0] has not been read, OVERRUN will be set to 1. Read and send After registering, this bit is cleared by hardware.
Bit 19: 16	Reserve
Bit 15:0	**data_y** : 12-bit A/D conversion result of channel 0~9 (Transfer data) Align left or right according to the setting.

9.9.10 A/D touch screen control register (**ADC_TPCR**)

Address offset: 0x50

Reset value: 0x0000 0000

Bit 16	**adtp_if** : compare flag bit (ADC TouchPad interrupt flag) The selected A/D TouchPad conversion channel result is less than ADTPCMP and is continuously valid, this bit is 1. Write 1 to clear this bit.
Bit 1	**adtp_ie** : A/D touchpad interrupt enable (ADC touchpad interrupt enable) 1 = Enable A/D touch screen interrupt 0 = Disable A/D touch screen interrupt
Bit 0	**adtp_en** : TouchPad mode enable control bit (TouchPad mode enable) 1 = touch screen mode enabled 0 = touch screen mode disabled

9.9.11 A/D touch screen filter register (**ADC_TPFR**)

Address offset: 0x54

Reset value: 0x00ff ffff

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
TPCNT[3:0]				TPCMP[11:0]											
r															

Bit 31: 28	Reserve
Bit 27: 24	**adtp_cnt** : n+1 times of continuous use in touch screen mode.
Bit 23: 12	**adtp_cmpy** : effective threshold in Y direction in touch screen mode. When the AD conversion result is less than the threshold, it is a valid sampling value.
Bit 11:0	**adtp_cmpx** : the effective threshold in the X direction in the touch screen mode. When the AD conversion result is less than the threshold, it is a valid sampling value.

9.9.12 A/D touch screen channel selection register (ADC_TPCSR)

Address offset: 0x58

Reset value: 0x0000 0010

Bit 31: 8	Reserve
Bit 7: 4	**adtp_chy** : AD channel used for sampling in Y direction in touch screen mode
Bit 3: 0	**adtp_chx** : AD channel used for sampling in X direction in touch screen mode

9.10 IF

9.10.1 Hardware IF

signal	
AHB bus	...
ADC signals	...
xp_oe	X+ port of control panel 1: Output 1 to the IO port corresponding to X+ 0: Set the IO port corresponding to X+ to analog input mode
xn_oe	X-port of control panel 1: Output 0 to the IO port corresponding to X- 0: Set the IO port corresponding to X- to floating mode
yp_oe	Y+ port of control panel 1: Output 1 to the IO port corresponding to Y+ 0: Set the IO port corresponding to Y+ to analog input mode
yn_oe	Y-port of control panel 1: Output 0 to the IO port corresponding to Y- 0: Set the IO port corresponding to Y- to floating mode
touch_on	Control the connection of IO and gpx_p, gpx_n, gpy_p, gpy_n when IO mux 1: AD is collecting the AD channel data corresponding to the screen 0: AD channel data corresponding to the screen not collected by AD

9.10.2 Software information

It is basically the same as the continuous scan mode of AD, the difference is:

1. The configuration of the mode is not through the admd bit of the configuration register ADC_ADCR, but the adtp_en of the configuration register ADC_TPCR Bit.
2. The filter register ADC_TPFR needs to be configured in advance (you can also keep the default value).
3. The corresponding channel of the screen is configured through ADC_TPCSR, and the corresponding channel needs to be opened in ADC_ADCHS (default Y direction Use channel1 for sampling and channel0 for X-direction sampling).
4. The sampling result is confirmed through the register ADC_TPCR and obtained through ADC_TPXDR and ADC_TPYDR.

10. LCD-TFT Controller

10.1 Introduction

LTDC provides 24-bit parallel digital RGB (red, green, blue), and all transmitted signals can be directly matched with a resolution of up to 1024x600 LCD and TFT panel interface.

10.2 Main features

- Support standard horizontal/vertical synchronization digital video format
- Adjustable output digital video timing
- The output supports RGB888 format and is backward compatible with RGB666, RGB565, etc.
- Two display layers with dedicated FIFO (FIFO depth 480x64)
- VGA output, support refresh rate no less than 20Hz: 640x480, 800x600

10.3 Register description

Table **19.** Controller register description

Offset	Name	Access	Default	Description
00h	DP_ADDR0	RW	00000000h	Bits[31:8]: The 24MSB of the start address of the display image buffer0 in the system memory. The 8LSB will always treated as all zero. Bits[7:0]: Reserved. Always zero.
04h	DP_ADDR1	RW	00000000h	Bits[31:8]: The 24MSB of the start address of the display image buffer1 in the system memory. The 8LSB will always treated as all zero. Bits[7:0]: Reserved. Always zero.
08h	P_HOR	RW	00000697h	Bit[31:16]: Reserved. Bit[15:0]: This value plus 1 define the total width counted in unit of one pixel. For example, 679h mean the output image width is 1688 pixels.
0ch	HSYNC	RW	0030009fh	Bit[31:16]: Horizontal sync start This value define the hsync signal started by horizontal counter Bit[15:0]: Horizontal sync end This value define the hsync signal ended by horizontal counter
10h	A_HOR	RW	01980697h	Bit[31:16]: A_HOR_START This value define the active area ended by horizontal counter Bit[15:0]: A_HOR_END This value define the active area started by horizontal counter. And A_HOR_END must equal to A_HOR_START+A_HOR_LEN
14h	A_HOR_LEN	RW	000004FFh	Bit[31:16]: Reserved Bit[15:0]: This value plus 1 define the total number of horizontal active area in unit of pixel. The LST 2bit are always zero This value plus 1 must multiple of 8pixel
18h	BLK_HOR	RW	00000197h	Bit[15:0] BLK_HOR_START This value define the blanking area started by horizontal counter Bit[31:16]: BLK_HOR_END This value define the blanking area ended by horizontal counter.
1ch	P_VER	RW	00000429h	Bit[15:0]: This value plus 1 define the total height counted in unit of one line. For example, 429h mean the output image height is 1066 lines. Bit[31:16]: Reserved.

Offset	Name	Access	Default	Description
20h	VSYNC	RW	00010003h	Bit[31:16]: Vertical sync start This value define the vsync signal started by vertical counter Bit[15:0]: Vertical sync end This value define the vsync signal ended by vertical counter
24h	A_VER	RW	002a0429h	Bit[31:16]: A_VER_START This value define the vertical active area ended by vertical counter Bit[15:0]: A_VER_END This value define the vertical active area started by vertical counter
				Bit[31:16]: Reserved

28h	A_VER_LEN	RW	000003ffh	Bit[15:0]: This value plus 1 define the total number of vertical active area in unit of line
2ch	BLK_VER	RW	00000029h	Bit[31:16]: BLK_VER_START This value define the vertical blanking area ended by vertical counter Bit[15:0]: BLK_VER_END This value define the vertical blanking area started by vertical counter
30h	BLK_DATA	RW	00000000h	Bit[31:24]: Reserved Bit[23:0]: The blank data that will be inserted into the data stream when blank period. This register is also be used to fill the color of the image background.
34h	POL_CTL	RW	00000000h	Bit[31:4]: Reserved Bit[3]: Polarity of the pixel output clock p_out_clk. 1: negative edge sampling. 0: post edge sampling. Bit[2]: data_en polarity 1: low enable 0: high enable Bit[1]:vsync polarity 1: low active 0: high active Bit[0]: hsync polarity 1: low active 0: high active

Offset	Name	Access	Default	Description
38h	OUT_EN	RW	00000000h	Bit[31:9] Reserved Bit[8]: Global enable. 1: Enable the controller. When enabled, the controller will regenerate the output frame from the start of the image 0: Disable the whole controller, VO can be configured Bit[7:3]: Reserved Bit[2]: data_en output enable. Bit[1]: v_sync output enable. Bit[0]: h_sync output enable. 0: disable. 1: enable.
3ch	INTR_STA	RO	00000000h	Interrupt status bits Bit[31:6] Reserved Bit[5]:Active_cmd_finish interrupt, assert when axi finish all active cmd. This interrupt only generate when global enable bit is disabled Bit[4:2]: Reserved Bit[1]: frame over int Bit[0]: line buffer Error int
40h	INTR_EN	RW	00000000h	Bit[31:6] Reserved Bit[5]:Acitve_cmd_finish interrupt enable Bit[4:2]: Reserved Bit[1]: frame over int enable Bit[0]: line buffer error int enable, high active
				Interrupt clear register Write 1 to each bit will clear correspond interrupt in

44h	INTR_CLR	WO	-	INTR_STA Bit[31:6] Reserved Bit[5]:Active_cmd_finish interrupt clear Bit[4:2]: Reserved Bit[1]: frame buffer over int clear Bit[0]: line buffer error int clear
48h	DP_SWT	RW	00000000h	Bit[31:1]: Reserved Bit[0]: Next display frame buffer 0: Next frame is buffer0 1: Next frame is buffer1 When the display buffer finished display the frame in current buffer, it will check this register's next displaying frame buffer, and get the corresponding data.

Page 113

Offset	Name	Access	Default	Description
4ch	VI_FORMAT	RW	00000000h	video input format bit[31:6] Reserved bit[5:4] Indian mode 3: 64bit big-endian 2: 32bit big-endian 1: 16bit big-endian 0: little-endian bit[3:1] Reserved bit[0] Input video format 0: RGB888 1: RGB565

10.4 Interface definition

10.4.1 Pin list

Table **20.** Controller pin list

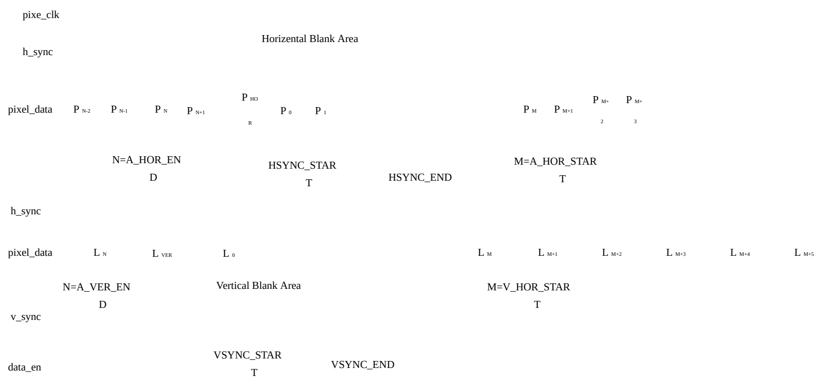
Pin Name	IO	Description
		Video Output Interface
data_r[7:0]	O	Pixel “r” data output
data_g[7:0]	O	Pixel “g” data output
data_b[7:0]	O	Pixel “b” data output
pix_clk_o	O	output pixel clock
hsync	O	Output video horizontal synchronize signal.
vsync	O	Output video vertical synchronize
data_en	O	Output video data_en signal.

10.4.2 Interface signal description

Video output interface

The video output interface has an independent synchronous control signal. The following figure is an example of the waveform:

Figure **11.** Video output waveform



All signals including h_syn, v_syn, pixel_data and data_en are sampled on the valid edge of clk_p.

The row position of the first valid pixel is set by A_HOR_START. The number of effective pixels in a row is set by the register A_HOR_LEN Certainly. A_HOR_END must be equal to A_HOR_START+A_HOR_LEN-1.

The position of the first valid line in a frame is set by A_VER_START. The total number of valid lines is set by the register A_VER_LEN.

Digital pixel value only supports RGB888 format.

10.5 Application Note

10.5.1 Controller initialization

To initialize the controller, write 0 to the register OUT_EN[8] first.

The following is an example of the initialization steps:

Table **21.** Controller initialization steps

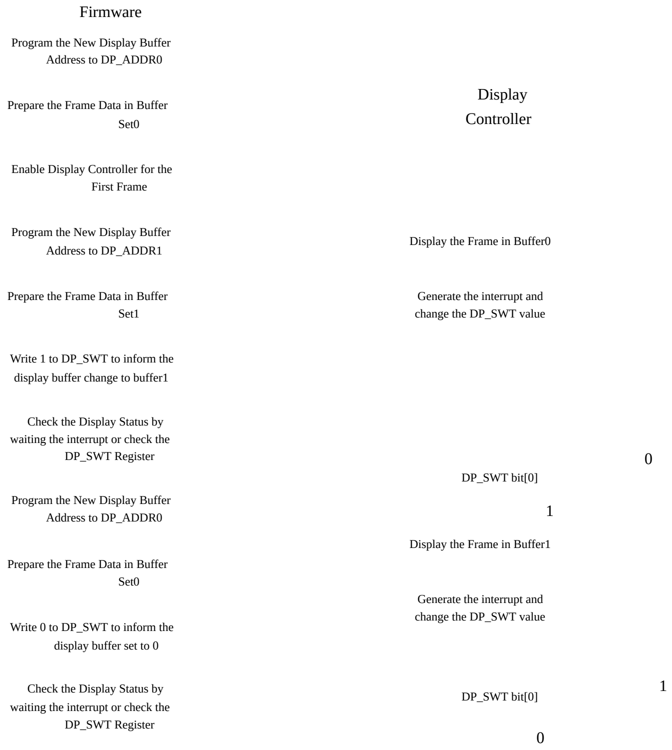
Steps	Description
1	Write OUT_EN with zero.
2	Configure PLL_config register, enable PLL
3	Configure DAC_control register
4	Configure DP_ADDR0, DP_ADDR1
5	Configure register addressed from 08h to 34h in any order
6	Configure INTR_EN register
7	After PLL output stable pixel clock. Video out controller start to work
8	Enable the output by write the corresponding bit with 1 to OUT_EN register.
9	After one frame been displayed, VO will check DP_SWT bit[0] and get the correspond DP_ADDR to display. Before VO get the DP_ADDR, CPU or GPU can change DP_ADDR any time.

10.5.2 Alternate use of display layers

The controller supports alternate use of 2 display layers to enhance the stability of the video output when the refresh rate is low. It also helps to use alternate video frame order The base address of the column.

Using the display layer alternately requires interaction between the controller and the software. The following figure illustrates the interaction process:

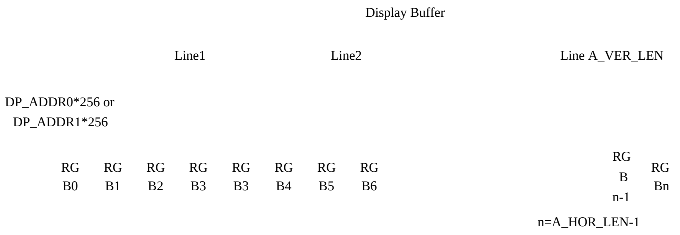
Figure **12.** Switching control of the display layer


Firmware
Program the New Display Buffer Address to DP_ADDR0
Display Controller
Prepare the Frame Data in Buffer Set0
Enable Display Controller for the First Frame
Program the New Display Buffer Address to DP_ADDR1
Display the Frame in Buffer0
Prepare the Frame Data in Buffer Set1
Generate the interrupt and change the DP_SWT value
Write 1 to DP_SWT to inform the display buffer change to buffer1
Check the Display Status by waiting the interrupt or check the DP_SWT Register
0
DP_SWT bit[0]
Program the New Display Buffer Address to DP_ADDR0
1
Display the Frame in Buffer1
Prepare the Frame Data in Buffer Set0
Generate the interrupt and change the DP_SWT value
Write 0 to DP_SWT to inform the display buffer set to 0
Check the Display Status by waiting the interrupt or check the DP_SWT Register
1
DP_SWT bit[0]
0

10.5.3 Reading data from the display cache

The state of the display cache is set by DP_ADD0/1. The display cache storage method is shown in the following figure:

Figure **13.** Shows how the cache is stored



10.5.4 Display Cache **Endian mode**

LTDC display layer supports little-endian and big-endian(3type):

Figure **14. RGB 565 Display Buffer Storage**

addr	3	2	1	0
	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0
RGB565 little-endian	pixel 2		pixel 1	
	7	6	5	4
	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0
	pixel 4		pixel 3	

addr	3	2	1	0
	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3
RGB565 big-endian 16bit mode	pixel 2		pixel 1	
	7	6	5	4
	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3
	pixel 4		pixel 3	

addr	3	2	1	0
	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3
RGB565 big-endian 32bit mode	pixel 1		pixel 2	
	7	6	5	4
	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3	G2 G1 G0 B4 B3 B2 B1 B0	R4 R3 R2 R1 R0 G5 G4 G3

pixel 3 pixel 4

addr 3 2 1 0

G G G B B B B B R R R R R G G G G G G B B B B B R R R R R G G G

2 1 0 4 3 2 1 0 4 3 2 1 0 5 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 5 4 3

RGB565
big-endian
64bit mode

pixel 3 pixel 4

7 6 5 4

G G G B B B B B R R R R R G G G G G G B B B B B R R R R R G G G

2 1 0 4 3 2 1 0 4 3 2 1 0 5 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 5 4 3

pixel 1 pixel 2

Page 118

Figure **15. RGB 888 Display Buffer Storage**

addr 3 2 1 0

X X X X X X X X R R R R R R R R G G G G G G G G B B B B B B B B

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

RGB888
little-
endian

pixel 1

7 6 5 4

X X X X X X X X R R R R R R R R G G G G G G G G B B B B B B B B

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

pixel 2

addr 3 2 1 0

R R R R R R R R X X X X X X X X B B B B B B B B G G G G G G G G

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

RGB888
big-endian
16bit mode

pixel 1

7 6 5 4

R R R R R R R R X X X X X X X X B B B B B B B B G G G G G G G G

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

pixel 2

3 2 1 0

B B B B B B B B G G G G G G G G R R R R R R R R X X X X X X X X

addr

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

pixel 1

RGB888
big-endian
32bit mode

7 6 5 4

B B B B B B B B G G G G G G G G R R R R R R R R X X X X X X X X

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

pixel 2

3 2 1 0

B B B B B B B B G G G G G G G G R R R R R R R R X X X X X X X X

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

addr

pixel 2

RGB888
big-endian
64bit mode

7 6 5 4

B B B B B B B B G G G G G G G G R R R R R R R R X X X X X X X X

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

pixel 1

10.5.5 VESA timing

VESA standard resolution frequency:

Table **22. Resolution horizontal value**

Resolution	p_hor	sync	sync	active	active	active	blk	blk
		start	end	start	end	len	start	end
800x600	41f h	28 h	a7 h	100 h	41f h	31f h	0 h	ff h
640x480	31f h	8 h	67 h	98 h	317 h	27f h	0 h	8f h

Table **23. Resolution vertical configure value**

Resolution	pver	sync	sync	active	active	active	blk	blk
		start	end	start	end	len	start	end
800x600	273 h	1 h	4 h	1c h	273 h	257 h	0 h	1b h
640x480	20c h	2 h	3 h	25 h	204 h	1df h	0 h	1c h

Notice: 1. 800X480 is not the standard VESA STD, so its values are estimated.

11. Advanced control timer (**TIM1/2**)

11.1 Introduction to **TIM1** and **TIM2**

The advanced control timers (TIM1 and TIM2) consist of a 32-bit auto-load counter, which is driven by a programmable prescaler move.

It is suitable for many purposes, including measuring the pulse width of input signals (input capture), or generating output waveforms (output comparison, PWM, Complementary PWM with embedded dead time, etc.).

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

Advanced control timers (TIM1 and TIM2) and general timers (TIMx) are completely independent, they do not share any resources. they It can be operated synchronously, please refer to the chapter of general timer synchronization for details.

11.2 Main features

The functions of TIM1 and TIM2 timers include:

- 32-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, the frequency division factor of the counter clock frequency is any number between 1 and 65536 value
- Up to 4 independent channels:
  - Input capture
  - Output comparison
  - PWM generation (edge or center aligned mode)
  - Single pulse mode output
- Complementary output with programmable dead time

- Synchronization circuit that uses external signal to control timer and timer interconnection
- Allows to update the repeat counter of the timer register after a specified number of counter cycles
- The brake input signal can put the timer output signal in a reset state or a known state
- Interrupt/DMA is generated when the following events occur:
  - Update: Counter overflow/downflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output comparison
  - Brake signal input
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning
- Trigger input as external clock or cycle current management

Figure **16.** Block diagram of advanced control timer

Note: According to the setting of the control bit, the content of the preload register is transferred to the working register in the *U* event

event

Interrupt and *DMA* output

11.3 Functional description

11.3.1 Time base unit

The main part of the programmable advanced control timer is a 32-bit counter and its associated auto-loading register. This counter can Count up, count down, or count up and down in both directions. This counter clock is divided by the prescaler.

The counter, auto-load register and prescaler register can be read and written by software, even if the counter is still running, the read and write are still valid.

The time base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)

- Auto reload register (TIMx_ARR)
- Repeat count register (TIMx_RCR)

The auto-load register is pre-loaded, and writing or reading the auto-reload register will access the pre-load register. According to the sent in TIMx_CR1 The auto-loading pre-loading enable bit (ARPE) setting in the register, the content of the pre-loading register is immediately or in every update event UEV When transferred to the shadow register. When the counter reaches the overflow condition (underflow condition when counting down) and when the UDIS in the TIMx_CR1 register When the bit is equal to 0, an update event is generated. Update events can also be generated by software. The generation of update events under each configuration will be described in detail

The counter is driven by the clock output CK_CNT of the prescaler, only when the counter enable bit in the counter TIMx_CR1 register is set (CEN), CK_CNT is valid. (For more details about enabling the counter, please refer to the description of the slave mode of the controller).

Note: One clock cycle after setting the *CEN* bit of the *TIMx_CR* register , the counter starts counting.

Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a (TIMx_PSC register In the device) a 32-bit counter controlled by a 16-bit register. Because this control register has a buffer, it can be changed at runtime. new The parameters of the prescaler will be used when the next update event arrives.

The following two figures show examples of changing counter parameters when the prescaler is running.

Figure **17.** When the prescaler parameter changes from **1** to **2** , the timing diagram of the counter

Figure **18.** Timing diagram of the counter when the parameter of the prescaler is changed from **1** to **4**

11.3.2 Counting mode

Up counting mode

In the up-counting mode, the counter counts from 0 to the auto-load value (the content of the TIMx_ARR counter), and then restarts from 0 Count and generate a counter overflow event.

If the repeat counter function is used, an update event will be generated when the up count reaches the set repeat count number (TIMx_RCR) (UEV); otherwise, an update event is generated every time the counter overflows.

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

Setting the UDIS bit in the TIMx_CR1 register can disable the update event; this can avoid writing new data to the preload register. The shadow register is updated when the value is set. Until the UDIS bit is cleared to 0, no update event will be generated. But when an update event should be generated, the counter It will still be cleared to 0, and the count of the prescaler will be set to 0 (but the value of the prescaler will not change). In addition, if the TIMx_CR1 register is set URS bit (select update request) in the device, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (that is, no Generate an interrupt or DMA request). This is to avoid generating update and capture interrupts at the same time when clearing the counter in capture mode.

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

- The repeat counter is reloaded with the contents of the TIMx_RCR register.
- The autoload shadow register is reset to the value of the preload register (TIMx_ARR).
- The buffer of the prescaler is put into the value of the preload register (the content of the TIMx_PSC register).

The following figure gives some examples, when TIMx_ARR = 0x36, the action of the counter under different clock frequencies.

Figure **19.** Counter timing diagram, the internal clock division factor is **1**

Figure **20.** Counter timing diagram, the internal clock division factor is **2**

Figure **21.** Counter timing diagram, the internal clock division factor is **4**

Figure **22.** Counter timing diagram, the internal clock division factor is **N**

Figure **23.** Counter timing diagram, update event when **ARPE = 0** (**TIMx_ARR is** not preloaded)

Page 125

Figure **24.** Counter timing diagram, update event when **ARPE = 1** (**TIMx_ARR is preloaded**)

Down counting mode

In the down mode, the counter starts from the automatically loaded value (the value of the TIMx_ARR counter) and counts down to 0, and then starts from the automatically loaded valu
The entered value restarts and a counter underflow event is generated.

If a repeat counter is used, when the count down repeats the number of times set in the repeat count register (TIMx_RCR), it will generate
Update event (UEV), otherwise the update event will be generated every time the counter underflows.

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

The UEV event can be disabled by setting the UDIS bit in the TIMx_CR1 register. This can avoid writing a new value to the preload register
The shadow register is updated at time. Therefore, no update event will be generated before the UDIS bit is cleared to 0. However, the counter will still automatically load the value from the cu
Restart counting, and the counter of the prescaler restarts from 0 (but the rate of the prescaler cannot be modified).

In addition, if the URS bit in the TIMx_CR1 register is set (select update request), setting the UG bit will generate an update event
UEV but does not set the UIF flag (so interrupts and DMA requests are not generated). This is to avoid the occurrence of a capture event and clear the counter.
Both update and capture interrupts are generated at the same time.

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

- The repetition counter is reset to the contents of the TIMx_RCR register.
- The buffer of the prescaler is loaded with the preloaded value (the value of the TIMx_PSC register).
- The current autoload register is updated to the preload value (contents in the TIMx_ARR register).

Note: 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 under different clock frequencies when TIMx_ARR = 0x36.

Page 126

Figure **25.** Counter timing diagram, the internal clock division factor is **1**

Figure **26.** Counter timing diagram, the internal clock division factor is **2**

Figure **27.** Counter timing diagram, the internal clock division factor is **4**

Figure **28.** Counter timing diagram, the internal clock division factor is **N**

Figure **29.** Counter timing diagram, update event when repeated counter is not used

Center alignment mode (count up / down)

In the center-aligned mode, the counter starts counting from 0 to the automatically loaded value (TIMx_ARR register) -1, resulting in a counter overflow Event, then count down to 1 and generate a counter underflow event; then restart counting from 0.

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current counting direction.

The update event can be generated at every count overflow and every count underflow; it can also be set by (software or using slave mode controller) The UG bit in the TIMx_EGR register is generated. At this time, the counter starts counting from 0 again, and the prescaler also starts counting from 0 again.

The UEV event can be disabled by setting the UDIS bit in the TIMx_CR1 register. This can avoid writing a new value to the preload register The shadow register is updated at time. Therefore, no update event will be generated before the UDIS bit is cleared to 0. However, the counter will still be automatically re-added based on the The loaded value continues to count up or down.

In addition, if the URS bit in the TIMx_CR1 register is set (select update request), setting the UG bit will generate an update event UEV but does not set the UIF flag (so interrupts and DMA requests are not generated). This is to avoid the occurrence of a capture event and clear the counter. Both update and capture interrupts are generated at the same time.

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

- The repetition counter is reset to the contents of the TIMx_RCR register.
- The buffer of the prescaler is loaded with the value of the preload (TIMx_PSC register).
- The current autoload register is updated to the preload value (contents in the TIMx_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 operations at different clock frequencies:

Figure **30.** Counter timing diagram, the internal clock division factor is **1** , **TIMx_ARR = 0x6**

Figure **31.** Counter timing diagram, the internal clock division factor is **2**

Page 129

Figure **32.** Counter timing diagram, the internal clock division factor is **4** , **TIMx_ARR = 0x36**

Figure **33.** Counter timing diagram, the internal clock division factor is **N**

Figure **34.** Counter timing diagram, update event when **ARPE = 1** (counter underflow)

Page 130

Figure **35.** Counter timing diagram, update event when **ARPE = 1** (counter overflow)

11.3.3 Repeat Counter

'Time base unit' explains how the update event (UEV) is generated when the counter overflows/underflows, but in fact it can only be counted up to Generated when it reaches 0. This feature is very useful for generating PWM signals.

This means that every N counts overflow or underflow, the data is transferred from the preload register to the shadow register (TIMx_ARR is automatically reloaded) Input register, TIMx_PSC preload register, and capture/compare register TIMx_CCRx in compare mode), N is TIMx_RCR repeats the value in the count register.

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

- Each time the counter overflows in the up-counting mode,
- Each time the counter underflows in the down-counting mode,
- Every time it overflows and every time it underflows in center-aligned mode. Although this limits the maximum PWM cycle period to 128, it can The duty cycle is updated twice in each PWM cycle. In the center-aligned mode, because the waveform is symmetrical, if every PWM cycle Only refresh the compare register once during the period, the maximum resolution is 2xTck.

The repetition counter is automatically loaded, and the repetition rate is defined by the value of the TIMx_RCR register (see Figure 66). When the update event by the soft Generated (by setting the UG bit in TIMx_EGR) or generated by the hardware slave mode controller, regardless of whether the value of the repeat counter is How much, an update event occurs immediately, and the contents of the TIMx_RCR register are reloaded into the repeat counter.

Page 131

Figure **36.** Examples of update rates in different modes and register settings of **TIMx_RCR**

11.3.4 Clock selection

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

- Internal clock (CK_INT).
- External clock mode 1: External input pin (TIx).
- External clock mode 2: External trigger input (ETR).
- Internal trigger input (ITRx): Use a timer as the prescaler of another timer, for example, you can configure a timer

  Timer1 is used as a prescaler for another timer Timer2.

Internal clock source (**CK_INT**)

If the slave mode controller is disabled (SMS=000), the CEN, DIR (TIMx_CR1 register) and UG bit (TIMx_EGR Register) is the de facto control bit and can only be modified by software (the UG bit is still automatically cleared). Once the CEN bit is written as 1, pre-divide The clock of the frequency converter is provided by the internal clock CK_INT.

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

Figure **37.** The control circuit in normal mode, the internal clock division factor is **1**

External clock source mode **1**

When SMS=111 in the TIMx_SMCR register, this mode is selected. The counter can be at each rising edge or down of the selected input Falling edge count.

Figure **38. TI2** external clock connection example

For example, to configure the up counter to count on the rising edge of the T12 input, use the following steps:

1. Configure TIMx_CCMR1 register CC2S=01, configure channel 2 to detect the rising edge of TI2 input.
2. Configure IC2F[3:0] in the TIMx_CCMR1 register, select the input filter bandwidth (if no filter is required, keep IC2F=0000).
3. Configure CC2P=0 in the TIMx_CCER register to select the rising edge polarity.
4. Configure SMS=111 in the TIMx_SMCR register and select timer external clock mode 1.
5. Configure TS=110 in the TIMx_SMCR register and select TI2 as the trigger input source.
6. Set CEN=1 in the TIMx_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 the 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 clock of the counter depends on the resynchronization circuit at the input of TI2.

Figure **39.** Control circuit in external clock mode **1**

External clock source mode **2**

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

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

The following figure is the overall block diagram of the external trigger input:

Figure **40.** External trigger input block diagram

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

1. No filter is needed in this example, set ETF[3:0]=0000 in the TIMx_SMCR register

2. Set the prescaler, set ETPS[1:0]=01 in the TIMx_SMCR register
3. Select the rising edge detection of ETR, set ETP=0 in the TIMx_SMCR register
4. Turn on external clock mode 2, write ECE=1 in the TIMx_SMCR register
5. Start the counter and write CEN=1 in the TIMx_CR1 register

The counter counts once every 2 ETR rising edges.

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

Figure **41.** Control circuit in external clock mode **2**

11.3.5 Capture / Compare Channel

Each capture/compare channel is surrounded by a capture/compare register (including shadow registers), including the captured input part (data Word filtering, r [redacted] rescaler), and output section (comparator and output control).

Figure 42 overview of the capture/compare channels.

The input orresponding TIx input signal and generates a filtered signal TIxF. Then, an edge with polarity selection The monitor generates a signal (TIxFPx), which can be used as an input trigger from the mode controller or as a capture control. The signal passes the pre The frequency is divided into the capture register (ICxPS).

Figure **42.** Capture / compare channel (for example: input part of channel **1**)

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

Figure **43.** Main circuit of capture / compare channel **1**

Figure **44.** The output section of the capture / compare channel (channels **1** to **3**)

Figure **45.** The output section of the capture / compare channel (channel **4**)

The capture/compare module consists of a preload register and a shadow register. The read and write process only manipulates the preload register. In capture mode Under the formula, the capture occurs on the shadow register and then copied to the preload register.

In the 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.

11.3.6 Input Capture Mode

In the input capture mode, when the corresponding edge on the ICx signal is detected, the current value of the counter is latched into the capture/compare register (TIMx_CCRx). When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, if the Interrupt or DMA operation, an interrupt or DMA request will be generated. If the CCxIF flag is already high when the capture event occurs, repeat the capture Get the flag CCxOF (TIMx_SR register) is set. Write CCxIF=0 to clear CCxIF, or read stored in TIMx_CCRx register The captured data in the device can also clear CCxIF. Write CCxOF=0 to clear CCxOF.

The following example shows how to capture the counter value into the TIMx_CCR1 register at the rising edge of TI1 input. The steps are as follows:

- Select a valid input terminal: TIMx_CCR1 must be connected to the TI1 input, so write CC1S=01 in the TIMx_CCR1 register, Once CC1S is not 00, the channel is configured as an input, and the TIMx_CCR1 register becomes read-only.
- 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 ICxF bit in the TIMx_CCMRx register). Assuming that the input signal jitters within a maximum of 5 clock cycles, we must The bandwidth of the configuration filter is longer than 5 clock cycles; therefore, we can sample 8 times continuously (at fDTS frequency) to confirm that the The last real edge transition of TI1, that is, write IC1F=0011 in the TIMx_CCMR1 register.
- Select the valid conversion edge of the TI1 channel, and write CC1P=0 (rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler Disabled (write IC1PS=00 in TIMx_CCMR1 register).
- Set CC1E of the TIMx_CCER register to 1 to allow the value of the counter to be captured into the capture register.
- If necessary, enable related interrupt requests by setting the CC1IE bit in the TIMx_DIER register, and by setting TIMx_DIER The CC1DE bit in the register allows DMA requests.

When an input capture occurs:

- When a valid level transition occurs, the value of the counter is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures have occurred, and CC1IF has not been cleared, CC1OF Also set to 1.
- If the CC1IE bit is set, an interrupt will be generated.
- If the CC1DE bit is set, a DMA request will also 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 loss in reading the capture overflow flag Capture overflow information that may occur after and before the data is read.

Note: By setting the corresponding *CCxG* bit in the *TIMx_EGR* register, the input capture interrupt and */* or *DMA* request can be generated by software .

11.3.7 PWM input mode

This mode is a special case of the input capture mode, except for the following differences, the operation is the same as the input capture mode:

- The two ICx signals are mapped to the same TIx input.
- The two ICx signals are edge-valid, but the polarity is opposite.
- One of the TIxFP signals is used as a trigger input signal, and the slave mode controller is configured to reset mode. For example, you need to test Quantify the length (TIMx_CCR1 register) and duty cycle (TIMx_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)

- Select the valid input of TIMx_CCR1: set CC1S=01 in the TIMx_CCMR1 register (select TI1).
- Select the valid polarity of TI1FP1 (used to capture data to TIMx_CCR1 and clear the counter): set CC1P=0 (rising edge efficient).
- Select the valid input of TIMx_CCR2: set CC2S=10 in the TIMx_CCMR1 register (select TI1).
- Select the valid polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P=1 (falling edge valid).
- Select a valid trigger input signal: set TS=101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller to reset mode: set SMS=100 in TIMx_SMCR.
- Enable capture: set CC1E=1 and CC2E=1 in the TIMx_CCER register.

Figure **46.** Timing of **PWM** input mode

Because only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can only use TIMx_CH1 / TIMx_CH2 signal.

11.3.8 Forced output mode

In the output mode (CCxS=00 in the TIMx_CCMRx register), output the compare signal (OCxREF and corresponding OCx/OCxN) It can be directly forced into a valid or invalid state by software, without relying on the comparison result between the output compare register and the counter.

Set the corresponding OCxM=101 in the TIMx_CCMRx register to force the output comparison signal (OCxREF/OCx) to be in a valid state. In this way, OCxREF is forced to be high (OCxREF is always active high), and at the same time, OCx gets a signal with the opposite polarity of CCxP.

For example: CCxP=0 (OCx high level is valid), then OCx is forced to high level.

Set OCxM=100 in the TIMx_CCMRx register to force the OCxREF signal to be low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress, and the corresponding flags will also be modified. because This will still generate corresponding interrupts and DMA requests. This will be introduced in the output comparison mode section below.

11.3.9 Output Compare Mode

This function is used to control an output waveform or indicate when a given time has elapsed.

When the contents of the counter and the capture/compare register are the same, the output compare function does the following:

- The output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (the TIMx_CCER register in the The value defined by the CCxP bit) is output to the corresponding pin. During a comparison match, the output pin can maintain its level (OCxM=000), It is set to an effective level (OCxM=001), is set to no effective level (OCxM=010) or is reversed (OCxM=011).
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- If the corresponding interrupt mask (CCxIE bit in the TIMx_DIER register) is set, an interrupt is generated.
- If the corresponding enable bit is set (CCxDE bit in TIMx_DIER register, CCDS bit in TIMx_CR2 register is selected Select the DMA request function), a DMA request is generated.

The OCxPE bit in TIMx_CCMRx selects whether the TIMx_CCRx register needs to use the preload register.

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

The accuracy of synchronization can reach one counting cycle of the counter. The output compare mode (in the single pulse mode) can also be used to output a single pulse.

Configuration steps of output comparison mode:

- Select counter clock (internal, external, prescaler)
- Write the corresponding data into the TIMx_ARR and TIMx_CCRx registers
- If you want to generate an interrupt request, set the CCxIE bit
- Select the output mode, for example:
  - It is required that the output pin of OCx is flipped when the counter matches CCRx, and OCxM=011 is set
  - Set OCxPE = 0 to disable the preload register
  - Set CCxP = 0 to select the polarity to be active high
  - Set CCxE = 1 to enable output
- Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_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 TIMx_CCRx can only be updated when the next update event occurs). The figure below gives a example.

Figure **47.** Output compare mode, flip **OC1**

11.3.10 PWM mode

The pulse width modulation mode can generate a frequency determined by the TIMx_ARR register and the duty cycle determined by the TIMx_CCRx register. Signal.

Write '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bit in the TIMx_CCMRx register, which can be independent Set each OCx output channel to generate a PWM. The corresponding preset must be enabled by setting the OCxPE bit in the TIMx_CCMRx register. Load the register, and finally set the ARPE bit of the TIMx_CR1 register to enable the preload register for automatic reloading (in the upward counting or Center symmetry mode).

Because only when an update event occurs, the preload register can be transferred to the shadow register, so the counter starts counting Previously, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The polarity of OCx can be set by software in the CCxP bit in the TIMx_CCER register, it can be set to active high or low power Ping and effective. OCx output enable (in the TIMx_CCER and TIMx_BDTR registers) CCxE, CCxNE, MOE, OSSI Combination control with OSSR bit. See the description of the TIMx_CCER register for details.

In PWM mode (mode 1 or mode 2), TIMx_CNT and TIMx_CCRx are always being compared (according to the counter count Counting direction) to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx.

According to the state of the CMS bit in the TIMx_CR1 register, the timer can generate edge-aligned PWM signals or center-aligned PWM signals Signal.

PWM edge alignment mode

Up counting configuration

When the DIR bit in the TIMx_CR1 register is low, the count-up is executed. See section 11.3.2 .

The following is an example of PWM mode 1. When TIMx_CNT <TIMx_CCRx, the PWM reference signal OCxREF is high, Otherwise low. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains at '1'. If it is better than If the comparison value is 0, OCxREF remains at '0'. Figure 48 shows an example of the edge-aligned PWM waveform when TIMx_ARR = 8.

Figure **48.** Edge-aligned **PWM** waveform (**ARR = 8**)

Countdown configuration

When the DIR bit of the TIMx_CR1 register is high, the down count is executed. See section 11.3.2 .

In PWM mode 1, the reference signal OCxREF is low when TIMx_CNT> TIMx_CCRx, otherwise it is high. If TIMx_CCRx If the comparison value in TIMx_ARR is greater than the auto-reload value in TIMx_ARR, OCxREF remains at '1'. 0% PWM wave cannot be generated in this mode shape.

PWM center alignment mode

When the CMS bit in the TIMx_CR1 register is not '00', it is the center-aligned mode (all other configurations affect the OCxREF/O Cx signal Have the same effect). According to the setting of different CMS bits, the comparison flag can be set when the counter is counting up, and the comparison flag can be set when the counter is d It is set to 1 when counting, or is set to '1' when the counter is counting up and down. The counting direction bit (DIR) in the TIMx_CR1 register is controlled by hardware Update, do not modify it with software. See. 11 .3 Section 2 center-aligned mode.

Figure 49 shows some examples of center-aligned PWM waveforms

- TIMx_ARR=8
- PWM mode 1
- CMS=01 in the TIMx_CR1 register, in center-aligned mode 1, set the compare flag when the counter counts down

Figure **49.** Center-aligned **PWM** waveform (**APR = 8**)

Tips for using center alignment mode:

- When entering center-aligned mode, the current up/down count configuration is used; this means that the counter counts up or down depending on The current value of the DIR bit in the TIMx_CR1 register. In addition, the software cannot modify the DIR and CMS bits at the same time.

- It is not recommended to rewrite the counter when running in center-aligned mode because it will produce unpredictable results. In particular:
  - If the value of the write counter is greater than the value of auto-reload (TIMx_CNT>TIMx_ARR), the direction will not be updated
  - For example, if the counter is counting up, it will continue counting up
  - If 0 or the value of TIMx_ARR is written into the counter, the direction is updated, but no update event UEV is generated
- The safest way to use center-aligned mode is to generate a software update (set the TIMx_EGR bit before starting the counter). UG bit in), do not modify the value of the counter during the counting process.

11.3.11 Complementary output and dead zone insertion

Advanced control timers (TIM1 and TIM2) can output two complementary signals, and can manage the instantaneous turn-off and turn-on of the output. this A period of time is usually called a dead zone, and the user should base it on the connected output devices and their characteristics (delay of level conversion, delay of power switch Etc.) to adjust the dead time.

Configure the CCxP and CCxNP bits in the TIMx_CCER register to independently select the polarity for each output (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 in the TIMx_CCER register, The MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers, see Table 56. With brake function The complementary output channels OCx and OCxN control bits. In particular, the dead zone 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 dead zone. If there is a brake circuit, the MOE bit should also be set. Every channel has A 10-bit dead zone generator. The reference signal OCxREF can generate 2 outputs OCx and OCxN. If OCx and OCxN are high efficient:

- The OCx output signal is the same as the reference signal, except that its rising edge has a delay relative to the rising edge of the reference signal.
- The OCxN output signal is opposite to the reference signal, except that its rising edge has a delay relative to the falling edge of the reference signal. If you extend If it is longer than the current effective output width (OCx or OCxN), the corresponding pulse will not be generated.

The following figures show the relationship between the output signal of the dead zone generator and the current reference signal OCxREF. (Assuming CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1, and CCxNE = 1).

Figure **50.** Complementary output with dead zone insertion

Figure **51.** Dead zone waveform delay is greater than negative pulse

Figure **52.** Dead zone waveform delay is greater than positive pulse

The dead-band delay of each channel is the same, which is programmed and configured by the DTG bit in the TIMx_BDTR register. See section 11.4.18 for details The delay calculation in.

Redirect **OCxREF** to **OCx** or **OCxN**

In the output mode (forced setting, output compare or PWM), by configuring the CCxE and CCxNE bits of the TIMx_CCER register,

OCxREF can be redirected to the output of OCx or OCxN.

This function can send a special waveform (such as PWM or static Effective level). Another function is to make the two outputs at the inactive level at the same time, or at the active level and complementary output with dead zone.

Note: When only *OCxN is enabled* (*CCxE = 0* , *CCxNE = 1*), it will not be inverted, and immediately becomes high when *OCxREF is* valid. example For example, if *CCxNP = 0* , then *OCxN = OCxREF* . On the other hand, when both *OCx* and *OCxN* are enabled (*CCxE = CCxNE = 1*), when *OCxREF* is high *OCx* active; and *OCxN* contrary, when *OCxREF* low *OCxN* becomes active.

11.3.12 Using the brake function

When using the brake function, according to the corresponding control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, The OISx and OISxN bits in the TIMx_CR2 register), the output enable signal and the invalid level will be modified. But whenever, OCx and The OCxN output cannot be at the active level at the same time. For details, please refer to the complementary output channel OCx with brake function in the register table and Control bit of OCxN.

The brake source can be either a brake input pin or a clock failure event. The clock failure event is reset by the clock in the clock controller Security system is produced.

After the system is reset, the brake circuit is disabled and the MOE bit is low. Set the BKE bit in the TIMx_BDTR register to enable the brake function can. 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.

Because the falling edge of MOE can be asynchronous, the actual signal (acting on the output) and the synchronous control bit (in the TIMx_BDTR register A resynchronization circuit is set up between the middle). This resynchronization circuit creates a delay between the asynchronous signal and the synchronous signal. Especially if

When it is low, write MOE=1, then a delay (null instruction) must be inserted before reading it to read the correct value. This is because of the written It is an asynchronous signal and the read is a synchronous signal.

When a brake occurs (the selected level appears at the brake input), the following actions are taken:

- 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 The feature is still valid when the oscillator of the MCU is turned off.
- Once MOE = 0, each output channel outputs the level set by the OISx bit in the TIMx_CR2 register. If OSSI = 0, the timer releases the enable output, otherwise the enable output is always high.
- When using complementary output:
  - The output is first placed in the reset state, that is, the inactive state (depending on the polarity). This is an asynchronous operation, even when the timer does not have a clock, This function is also effective.
  - If the timer’s clock still exists, the dead-band generator will re-validate. The level shown drives the output port. Even in this case, OCx and OCxN cannot be driven to a valid level at the same time. Note that due to the resynchronization of the MOE, the dead time is longer than usual (about 2 CK_TIM clock cycles).
  - If OSSI = 0, the timer releases the enable output, otherwise it keeps the enable output; or once one of CCxE and CCxNE goes high When, the enable output goes high.
- If the BIE bit in the TIMx_DIER register is set, when the brake status flag (BIF bit in the TIMx_SR register) is '1' When, an interrupt is generated. If the BDE bit in the TIMx_DIER register is set, a DMA request is generated.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit will be automatically set in the next update event UEV; For example, this can be used for shaping. Otherwise, MOE remains low until it is set to '1' again; at this time, this feature can be used in For safety, you can connect the brake input to the power-driven alarm output, thermal sensor or other safety devices.

Note: The brake input is level effective. Therefore, when the brake input is valid, *MOE* cannot be set at the same time (automatically or through software) . At the same time, the status flag *BIF* cannot be cleared.

The brake is generated by the BRK input, its effective polarity is programmable, and it is turned on by the BKE bit in the TIMx_BDTR register.

In addition to brake input and output management, write protection is also implemented in the brake circuit to ensure the safety of the application. It allows users to freeze several Configuration parameters (dead zone length, OCx/OCxN polarity and disabled state, OCxM configuration, brake enable and polarity). Users can pass For the LOCK bit in the TIMx_BDTR register, select one of the three levels of protection, see section 11. 4.18 . LOCK bit after MCU reset Can only be modified once.

The following figure shows an example of output in response to brakes:

Figure **53.** Output in response to brakes

11.3.13 Clear **OCxREF** signal on external event

For a given channel, the high voltage at the ETRF input (set the corresponding OCxCE bit in the TIMx_CCMRx register to '1')
Ping can pull 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 comparison and PWM mode, and cannot be used in forced mode.

For example, the OCxREF signal can be connected to an external input. At this time, ETR must be configured as follows:

- The external trigger prescaler must be turned off: ETPS[1:0] = 00 in the TIMx_SMCR register.
- The external clock mode 2: ECE = 0 in the TIMx_SMCR register must be disabled.
- External trigger polarity (ETP) and external trigger filter (ETF) can be configured as required.

The figure below shows the action of the OCxREF signal corresponding to different OCxCE values when the ETRF input goes high. In this example,
The timer TIMx is placed in PWM mode.

Figure **54.** Clear **OCxREF** of **TIMx**

11.3.14 Generate six-step **PWM** output

When complementary output is required on a channel, the preload bits are OCxM, CCxE and CCxNE. When a COM commutation event occurs,
These 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 in the TIMx_EGR register, or by hardware on the rising edge of TRGI.
Pieces are produced.

When a COM event occurs, a flag bit (COMIF bit in the TIMx_SR register) is set. At this time, if it has been set
The COMIE bit of the TIMx_DIER register generates an interrupt; if the COMDE bit of the TIMx_DIER register has been set, then
Generate a DMA request.

The following figure shows the output of OCx and OCxN in three different configurations when a COM event occurs.

Figure **55.** Example of generating six-step **PWM** using **COM** (**OSSR = 1**)

11.3.15 Single pulse mode

Single pulse mode (OPM) is a special case of many of the aforementioned modes. This mode allows the counter to respond to a stimulus and in a program
After the controllable delay, a pulse with programmable pulse width is generated.

The counter can be started by the slave mode controller to generate waveforms in output comparison mode or PWM mode. Set TIMx_CR1 to send
The OPM bit in the register will select the single pulse mode, so that the counter can automatically stop when the next update event UEV is generated.

Only when the comparison value is different from the initial value of the counter can a pulse be generated. Before starting (when the timer is waiting to be triggered), the
Must be configured as follows:

- Up counting method: counter CNT <CCRx ≤ ARR (especially, 0 <CCRx)
- Down counting method: counter CNT> CCRx

Figure **56.** Example of single pulse mode

For example, you need to detect a rising edge on the TI2 input pin, and after a delay of t_{DELAY} , a length of

t_{PULSE} positive pulse.

Assuming TI2FP2 as trigger 1:

- Set CC2S=01 in the TIMx_CCMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable TI2FP2 to detect the rising edge.
- Set TS=110 in the TIMx_SMCR register, and TI2FP2 is used as the trigger (TRGI) of the slave mode controller.
- Set SMS=110 (trigger mode) in the TIMx_SMCR register, TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written in the compare register (the clock frequency and counter prescaler should be considered)

- t_{DELAY} is defined by the value in the TIMx_CCR1 register.
- t_{PULSE} is defined by the difference between the autoload value and the comparison value (TIMx_ARR-TIMx_CCR1).
- Suppose that a waveform from 0 to 1 is generated when a comparison match occurs, and a waveform from 1 to 0 is generated when the counter reaches the preload value.
  Waveform; first set the OC1M of the TIMx_CCMR1 register = 111, enter PWM mode 2; selectively use
  Register can be preloaded: set OC1PE in TIMx_CCMR1 = 1 and ARPE in TIMx_CR1 register; then

  Fill in the comparison value in the TIMx_CCR1 register, fill in the auto-load value in the TIMx_ARR register, and set the UG bit to generate
  An update event, and then wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Because only one pulse is needed, OPM = 1 in the TIMx_CR1 register must be set, and in the next update event (when the counter
It stops counting when it rolls over from the auto-load value to 0).

Special case: **OCx** fast enable:

In the single pulse mode, the edge detection logic at the TIx input pin sets the CEN bit to start the counter. Then the counter and comparison value
The comparison operation produces a conversion of the output. But these operations require a certain clock cycle, so it limits the minimum delay t_{DELAY} that can be obtained .

If you want to output the waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register; at this time, force OCxREF
(And OCx) Directly respond to the stimulus and no longer rely on the comparison result, the output waveform is the same as the waveform when the comparison matches. OCxFE only in the c
It works when configured in PWM1 and PWM2 modes.

11.3.16 Encoder interface mode

The method to select the encoder interface mode is: if the counter only counts on the edge of TI2, set the SMS in the TIMx_SMCR register
= 001; if only counting on the edge of TI1, set SMS = 010; if the counter is counting on both edges of TI1 and TI2, set SMS = 011.

By setting the CC1P and CC2P bits in the TIMx_CCER register, the polarity of TI1 and TI2 can be selected; if necessary, you can also
Program the input filter. The two inputs TI1 and TI2 are used as the interface of the incremental encoder. Refer to Table 54, assuming that the counter has been started
(CEN = 1 in the TIMx_CR1 register), the counter is driven by each valid transition on TI1FP1 or TI2FP2. TI1FP1
And TI2FP2 are the signals of TI1 and TI2 after passing the input filter and polarity control; if there is no filtering and disguised phase, then TI1FP1 = TI1;
If there is no filtering and disguising, then TI2FP2 = TI2. According to the jump sequence of the two input signals, count pulses and direction signals are generated. according to
According to the transition sequence of the two input signals, the counter counts up or down, and the hardware performs a corresponding operation on the DIR bit of the TIMx_CR1 register.
set up. Regardless of whether the counter counts on TI1, counts on TI2, or counts on both TI1 and TI2, at either input (TI1 or
The transition of TI2) will recalculate the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter is only between 0 and
TIMx_ARR register auto-load value between continuous counting (according to the direction, or 0 to ARR count, or ARR to 0 count).
Therefore, TIMx_ARR must be configured before starting to count; similarly, the capturer, comparator, prescaler, repeat counter, trigger output special
Sex etc. still work as usual. Encoder mode and external clock mode 2 are not compatible, so they cannot be operated at the same time.

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 editing
The location of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table lists all possible combinations, assuming that TI1 and T
Time changes.

Table **24.** Relation between counting direction and encoder signal

Effective edge	Relative signal level (TI1FP1 corresponds to TI2, TI2FP2 corresponds to TI1)	TI1FP1 signal		TI2FP2 signal	
		rise	decline	rise	decline

Only count at TI1	high	Count down and count up	Not counted	Not counted
	Low	Count up and count down	Not counted	Not counted
Only count at TI2	high	Not counted	Not counted	Count up and count down
	Low	Not counted	Not counted	Count down and count up
Count on TI1 and TI2	high	Count down, count up, count up, count down		
	Low	Count up, count down, count down, count up		

An external incremental encoder can be directly connected to the MCU without the need for external interface logic. However, a comparator is generally used to convert the encoder
The differential output is converted to a digital signal, which greatly increases the ability to resist noise interference. The third signal output by the encoder represents the mechanical zero poin
To connect it to an external interrupt input and trigger a counter reset.

The following figure is an example of counter operation, showing the generation and direction control of the counting signal. It also shows that when both sides are selected,
How input jitter is suppressed; jitter may occur when the sensor is close to a switching point. In this example, we assume
The configuration is as follows:

- CC1S = '01' (TIMx_CCMR1 register, IC1FP1 is mapped to TI1)
- CC2S = '01' (TIMx_CCMR2 register, IC2FP2 is mapped to TI2)
- CC1P = '0' (TIMx_CCER register, IC1FP1 is not inverted, IC1FP1=TI1)

- CC2P = '0' (TIMx_CCER register, IC2FP2 is not inverted, IC2FP2=TI2)
- SMS = '011' (TIMx_SMCR register, all inputs are valid on rising and falling edges).
- CEN = '1' (TIMx_CR1 register, counter enable)

Figure **57.** Example of counter operation in encoder mode

The following figure shows the operation example of the counter when the polarity of IC1FP1 is reversed (CC1P = '1', other configurations are the same as the previous example)

Figure **58. Example of IC1FP1** inverted encoder interface mode

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. Use the second configuration for timing in capture mode
The device measures the interval between two encoder events and can obtain dynamic information (speed, acceleration, deceleration). Encoder indicating mechanical zero point
The output can be used for this purpose. According to the interval between two events, the counter can be read out at a fixed time. If possible, you can
Latch the counter value to the third input capture register (the capture signal must be periodic and can be generated by another timer). it
It can also be read through a DMA request generated by the real-time clock.

11.3.17 Timer input XOR function

The TI1S bit in the TIMx_CR2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, and 3 XOR gates

The input terminals are TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used for all timer input functions, such as triggering or input capture. The following section 11.3.18 gives this feature for connecting Take the example of Hall sensor.

11.3.18 Interface with Hall sensor

When using the advanced control timer (TIM1 or TIM2) to generate PWM signals to drive the motor, you can use another general-purpose TIMx (TIM2, TIM3, TIM4 or TIM5) timers are used as "interface timers" to connect to Hall sensors, as shown in Figure 59 , 3 timer input pins (CC1, CC2, CC3) are connected to the TI1 input channel through an exclusive OR gate (selected by setting the TI1S bit in the TIMx_CR2 register),'connect The port timer' captures this signal.

The slave mode controller is configured in reset mode, and the slave input is TI1F_ED. Whenever one of the 3 inputs changes, the counter starts from 0 again Begin counting. This produces a time reference 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 42). The captured value reflects two The time delay between input changes gives information about the motor speed.

'Interface timer' can be used to generate a pulse in output mode, this pulse can be used to change (by triggering a COM event) Advanced timer (TIM1 or TIM2) attributes of each channel, and advanced control timer generates PWM signal to drive motor. Therefore, "the interface is fixed "Timer" channel must be programmed to generate a positive pulse after a specified delay (output comparison or PWM mode), and this pulse passes The TRGO output is sent to the advanced control timer (TIM1 or TIM2).

Example: The Hall input is connected to the TIMx timer, and it is required to change at a specified time after every change on any Hall input Advanced control timer TIMx PWM configuration.

- Set the TI1S bit of the TIMx_CR2 register to '1', configure three timer input logic or to TI1 input,
- Time base programming: Set TIMx_ARR to its maximum value (the counter must be cleared by the change of TI1). Set the prescaler to get a The maximum counter period, which is longer than the time interval between two changes on the sensor.
- Set channel 1 to capture mode (select TRC): set CC1S = 01 in the TIMx_CCMR1 register, if necessary, you can also To set the digital filter.
- Set channel 2 to PWM2 mode with the required delay: set OC2M = 111 in the TIMx_CCMR1 register and CC2S = 00.
- Select OC2REF as the trigger output on TRGO: set MMS = 101 in the TIMx_CR2 register.

In the advanced control register TIM1, the correct ITR input must be a trigger input, and the timer is programmed to generate a PWM signal to capture The get/compare control signal is preloaded (CCPC=1 in the TIMx_CR2 register), and the input control COM event (TIMx_CR2 CCUS = 1 in the register). After a COM event, write the next PWM control bits (CCxE, OCxM), which can be It is implemented in the interrupt subroutine that handles the rising edge of OC2REF.

The following figure shows this example:

Figure **59.** Example of Hall sensor interface

11.3.19 Synchronization of TIMx timer and external trigger

The TIMx timer can be synchronized with an external trigger in multiple modes: reset mode, gating mode and trigger mode.

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 IMx_CR1 register The URS bit is low, and an update event UEV is also generated; then all preload registers (TIMx_ARR, TIMx_CCRx) are Has been updated.

In the following example, the rising edge of the TI1 input causes the up counter to be cleared:

- 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 the trigger operation, so no configuration is required. CC1S bit only selects input capture Get the source, that is, CC1S=01 in the TIMx_CCMR1 register. Set CC1P = 0 in the TIMx_CCER register to determine the polarity (only Detect rising edge).

- Set SMS = 100 in the TIMx_SMCR register to configure the timer in reset mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.
- Set CEN = 1 in the TIMx_CR1 register to start the counter.

The counter starts to count according to the internal clock, and then runs normally until TI1 has a rising edge; at this time, the counter is cleared and then reset from 0 Restart counting. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, according to the TIE in the TIMx_DIER register The setting of (interrupt enable) bit and TDE (DMA enable) bit generates an interrupt request or a DMA request.

The following figure shows the action when the auto reload register TIMx_ARR = 0x36. Between the rising edge of TI1 and the actual reset of the counter The delay depends on the resynchronization circuit at the input of TI1.

Figure **60.** Control circuit in reset mode

Slave mode: gated mode

The enable of the counter depends on the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- 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 the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, Set CC1S = 01 in the TIMx_CCMR1 register. Set CC1P = 1 in the TIMx_CCER register to determine the polarity (check only Measure the low level).
- Set SMS = 101 in the TIMx_SMCR register to configure the timer as gated mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.
- Set CEN = 1 in the TIMx_CR1 register to start the counter. In gating mode, if CEN=0, the counter cannot be started, 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 goes high. Set when the counter starts or stops Set the TIF flag in TIMx_SR.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the input of TI1.

Figure **61.** Control circuit in gating mode

Slave mode: trigger mode

The enabling of the counter depends on the event on the selected input.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed, keep IC2F = 0000). The capture prescaler is not used in the trigger operation, and no configuration is required. The CC2S bit is only used to select CC2P in the input trap = 1 To determine the polarity (only detect low level).
- Set SMS = 110 in the TIMx_SMCR register to configure the timer as trigger mode; set TS = 110 in the TIMx_SMCR register, Select TI2 as the input source.

When TI2 has a rising edge, the counter starts to count under the internal clock drive, and the TIF flag is set at the same time. TI2 rising edge and counting The delay between the start of the counter depends on the resynchronization circuit at the TI2 input.

Figure **62.** Control circuit in flip-flop mode

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 letter
The signal is used as the input of the external clock, and another input can be selected as the trigger input in reset mode, gate control mode or trigger mode. not recommend
Use the TS bit of the TIMx_SMCR register to select ETR as TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up once on each rising edge of ETR:

- Configure the external trigger input circuit through the TIMx_SMCR register:
  - ETF = 0000: no filtering
  - ETPS = 00: no prescaler
  - ETP = 0: Detect the rising edge of ETR, set ECE = 1 to enable external clock mode 2.
- Configure channel 1 as follows to detect the rising edge of TI:

  - IC1F = 0000: no filtering
  - The capture prescaler is not used in the trigger operation, no configuration is required
  - Set CC1S = 01 in the TIMx_CCMR1 register to select the input capture source
  - Set CC1P = 0 in the TIMx_CCER register to determine the polarity (only the rising edge is detected)
- Set SMS = 110 in the TIMx_SMCR register to configure the timer as trigger mode. Set TS = 101 in the TIMx_SMCR register,
  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 circuit at the ETRP input.

Figure **63.** Control circuit in external clock mode **2** + trigger mode

11.3.20 Timer synchronization

All TIM timers are connected internally for timer synchronization or linking. See the next chapter TIM2/3/4 for details.

11.3.21 Debug mode

When the microcontroller enters the debug mode (CPU core stops), according to the setting of DBG_TIMx_STOP in the DBG module, TIMx
The counter can either continue normal operation or stop. For details, see the subsequent commissioning chapter.

11.4 Register description

11.4.1 Control Register **1** (**TIMx_CR1**)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 10 Reserve

Bit 9: 8	**CKD[1 : 0]** : Clock division factor (Clock division) These 2 bits are defined in the timer clock (CK_INT) frequency, dead time and the dead time generator and digital filter (ETR, TIx) The frequency division ratio between the sampling clocks used. 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: Reserved, do not use this configuration
Bit 7	**ARPE** : Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is loaded into the buffer
Bit 6: 5	**CMS[1 : 0]** : Select Center-aligned mode selection 00: Edge alignment mode. The counter counts up or down according to the direction bit (DIR) 01: Center alignment mode 1. The counter counts up and down alternately. Channel configured as output (TIMx_CCMRx register CCxS = 00) output compare interrupt flag bit, which is only set when the counter is counting down 10: Center alignment mode 2. The counter counts up and down alternately. The counter counts up and down alternately. Configure to lose The output compare interrupt flag bit of the output channel (CCxS = 00 in the TIMx_CCMRx register), only counts up on the counter Is set 11: Center alignment mode 3. The counter counts up and down alternately. The counter counts up and down alternately. Configure to lose The output compare interrupt flag bit of the output channel (CCxS = 00 in the TIMx_CCMRx register), when the counter is up and down Set when counting Note: When the counter is turned on (CEN = 1), it is not allowed to switch from edge-aligned mode to center-aligned mode.
Bit 4	DIR: Direction 0: The counter counts up 1: The counter counts down Note: When the counter is configured in center-aligned mode or encoder mode, this bit is read-only.
Bit 3	**OPM** : One pulse mode 0: When an update event occurs, the counter does not stop 1: When the next update event occurs (clear the CEN bit), the counter stops
Bit 2	**URS** : Update request source The software selects the source of the UEV event through this bit. 0: If an update interrupt or DMA request is allowed, any of the following events will generate an update interrupt or DMA request: − Counter overflow/underflow − Set the UG bit − Updates generated from the mode controller 1: If the update interrupt or DMA request is allowed, only the counter overflow/underflow will generate an update interrupt or DMA request begging
Bit 1	**UDIS** : Update disable The software allows/disables the generation of UEV events through this bit 0: UEV is allowed. Update (UEV) events are generated by any of the following events: − Counter overflow/underflow − Set the UG bit − The updated buffered registers generated from the mode controller are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) maintain their values. If set If the UG bit or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized

Bit 0	**CEN** : Counter enable 0: disable the counter 1: Enable the counter. Note: After the software sets the CEN bit, the external clock, gating mode and encoder mode can only work. Trigger mode can be automatically Set the CEN bit by hardware.

11.4.2 Control Register **2** (**TIMx_CR2**)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Keep	OIS4	OIS3 N	OIS3	OIS2 N	OIS2	OIS1 N	OIS1	TI1S		MMS		CCDS	CCUS	keep	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31: 15	Reserve
Bit 14	**OIS4** : Output idle state 4 (OC4 output). See OIS1 bit.
Bit 13	**OIS3N** : Output idle state 3 (OC3N output). See OIS1N bit.
Bit 12	**OIS3** : Output idle state 3 (OC3 output). See OIS1 bit.
Bit 11	**OIS2N** : Output idle state 2 (OC2N output). See OIS1N bit.
Bit 10	**OIS2** : Output idle state 2 (OC2 output). See OIS1 bit.
Bit 9	**OIS1N** : Output Idle state 1 (OC1N output) (Output Idle state 1) 0: When MOE = 0, OC1N = 0 after the dead zone 1: When MOE = 0, OC1N = 1 after the dead zone Note: After LOCK (TIMx_BKR register) level 1, 2 or 3 has been set, this bit cannot be modified.
Bit 8	**OIS1** : Output Idle state 1 (OC1 output) (Output Idle state 1) 0: When MOE = 0, if OC1N is realized, OC1 = 0 after the dead zone 1: When MOE = 0, if OC1N is realized, OC1 = 1 after the dead zone. Note: After LOCK (TIMx_BKR register) level 1, 2 or 3 has been set, this bit cannot be modified.
Bit 7	**TI1S** : TI1 selection 0: TIMx_CH1 pin is connected to TI1 input 1: TIMx_CH1, TIMx_CH2 and TIMx_CH3 pins are XORed and then connected to TI1 input

Bit 6: 4	**MMS[1 : 0]** : Master mode selection These two bits are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. The possible combinations are as follows: 000: Reset-The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode If the controller is in reset mode) to generate a reset, the signal on TRGO will have a delay relative to the actual reset. 001: Enable-the counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes need to be at the same time Start multiple timers or control to enable slave timers within a period of time. The counter enable signal is controlled by the CEN control bit and gate The logic or generation of the trigger input signal in the mode. When the counter enable signal is controlled by the trigger input, there will be a A delay, unless the master/slave mode is selected (see the description of the MSM bit in the TIMx_SMCR register). 010: Update-The update event is selected as the trigger input (TRGO). For example, the clock of a master timer can be used as a A prescaler for the slave timer. 011: Comparison pulse-once a capture occurs or a comparison is successful, when the CC1IF flag is to be set (even if it has been

	Is high), the trigger output sends a positive pulse (TRGO). 100: Compare-OC1REF signal is used as trigger output (TRGO) 101: Compare-OC2REF signal is used as trigger output (TRGO) 110: Compare-OC3REF signal is used as trigger output (TRGO) 111: Compare-OC4REF signal is used as trigger output (TRGO)
Bit 3	**CCDS** : Capture/compare DMA selection (Capture/compare DMA selection) 0: When a CCx event occurs, send a CCx DMA request 1: When an update event occurs, send a CCx DMA request
Bit 2	**CCUS** : Capture/compare control update selection 0: If the capture/compare control bits are preloaded (CCPC = 1), they can only be updated by setting the COM bit 1: If the capture/compare control bit is preloaded (CCPC = 1), you can set the COM bit or one of the TRGI Updating them Note: This bit only works on channels with complementary outputs.
Bit 1	Reserved, always read as 0.
Bit 0	**CCPC** : Capture/compare preloaded control 0: CCxE, CCxNE and OCxM bits are not preloaded 1: CCxE, CCxNE and OCxM bits are pre-loaded; after setting this bit, they are only updated after setting the COM bit Note: This bit only works on channels with complementary outputs.

11.4.3 Slave mode control register (**TIMx_SMCR**)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Reserve	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit 31: 16	Reserve.

Page 158

Bit 15	**ETP** : External trigger polarity This bit selects whether to use ETR or the inverse of ETR as the trigger operation. 0: ETR is not inverted, high level or rising edge is valid 1: ETR is inverted, low level or falling edge valid
Bit 14	**ECE** : External clock enable bit (External clock enable) This bit enables external clock mode 2. 0: Disable external clock mode 2 1: Enable external clock mode 2, the counter is driven by any valid rising edge on the ETRF signal Note 1: Setting the ECE bit is related to selecting external clock mode 1 and connecting TRGI to ETRF (SMS = 111 and TS = 111). The same effect. Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gate control mode and trigger mode; however, this When TRGI cannot be connected to ETRF (TS bit cannot be 111). Note 3: When external clock mode 1 and external clock mode 2 are enabled at the same time, the input of the external clock is ETRF.
Bit 13: 12	**ETPS[1** : **0]** : External trigger prescaler (External trigger prescaler) The frequency of the external trigger signal ETRP must be at most 1/4 of the TIMxCLK frequency. When inputting a faster external clock, you can use Use prescaler to reduce the frequency of ETRP. 00: Turn off prescaler 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8
	ETF[3 : **0]** : External trigger filter These bits define the frequency of sampling the ETRP signal and the bandwidth of the ETRP digital filtering. In fact, the digital filter is a Event counter, it will generate an output transition after recording N events. 0000: No filter, sampling with f_{DTS}

Bit 11: 8	0001: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 2 0010: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 4 0011: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 8 0100: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /2, N = 6 0101: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /2, N = 8 0110: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /4, N = 6 0111: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /4, N = 8 1000: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /8, N = 6 1001: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /8, N = 8 1010: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /16, N = 5 1011: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /16, N = 6 1100: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /16, N = 8 1101: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /32, N = 5 1110: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /32, N = 6 1111: Sampling frequency $f_{SAMPLING} = f_{DTS}$ /32, N = 8
Bit 7	**MSM** : Master/slave mode 0: No effect 1: The event on the trigger input (TRGI) is delayed to allow the current timer (via TRGO) and its slave timing Perfect synchronization between devices, which is very useful when it is required to synchronize several timers to a single external event

Bit 6: 4	TS[2:0]: Trigger selection These 3 bits select the trigger input for the synchronization counter. 000: internal trigger 0 (ITR0) 001: Internal trigger 1 (ITR1) 010: Internal trigger 2 (ITR2) 011: Internal trigger 3 (ITR3) 100: TI1 edge detector (TI1F_ED) 101: Filtered timer input 1 (TI1FP1) 110: Filtered timer input 2 (TI2FP2) 111: External trigger input (ETRF) For more details about ITRx, see the table below. Note: These bits can only be changed when they are not used (such as SMS = 000) to avoid false edge detection when changing.
Bit 3	Reserved, always read as 0.
Bit 2: 0	**SMS** : Slave mode selection When the external signal is selected, the effective edge of the trigger signal (TRGI) is related to the selected external input polarity (see input control register). Description of registers and control registers) 000: Disable slave mode-if CEN = 1, the prescaler is directly driven by the internal clock. 001: Encoder mode 1-According to the level of TI1FP1, the counter counts up/down on the edge of TI2FP2. 010: Encoder mode 2-According to the level of TI2FP2, the counter counts up/down on the edge of TI1FP1. 011: Encoder mode 3-According to the level of another input, the counter counts up/down on the edge of TI1FP1 and TI2FP2. 100: Reset mode-the rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register The signal of the memory. 101: Gated mode-When the trigger input (TRGI) is high, the counter clock is turned on. Once the trigger input goes low, then The counter is stopped (but not reset). The start and stop of the counter are controlled. 110: Trigger mode-the counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is affected Controlled. 111: External clock mode 1-The rising edge of the selected trigger input (TRGI) drives the counter. Note: If TI1F_EN is selected as the trigger input (TS = 100), do not use the gated mode. This is because TI1F_ED A pulse is output every time TI1F changes, but the gate control mode is to check the level of the trigger input.

Table **25. TIMx** internal trigger connection

Slave timer	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM4	TIM5	TIM2	TIM3
TIM2	TIM4	TIM1	TIM6	TIM3

11.4.4 DMA/ interrupt enable register (TIMX_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Keep	TDE	COM DE	CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE	BIE	TIE	COM IE	CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 15 Reserve

Bit 14 **TDE** : Allow to trigger DMA request (Trigger DMA request enable)
0: Disable triggering of DMA request
1: Allow to trigger DMA request

Bit 13 **COMDE** : Allow COM DMA request (COM DMA request enable)
0: Disable COM's DMA request
1: Allow COM DMA request

Bit 12 **CC4DE** : Allow capture/compare 4 DMA request (Capture/Compare 4 DMA request enable)
0: Disable capture/compare 4 DMA request
1: Allow capture/compare 4 DMA request

Bit 11 **CC3DE** : Allow capture/compare 3 DMA request (Capture/Compare 3 DMA request enable)
0: Disable capture/compare 3 DMA request
1: Allow capture/compare 3 DMA request

Bit 10 **CC2DE** : Allow capture/compare 2 DMA request (Capture/Compare 2 DMA request enable)
0: Disable capture/compare 2 DMA request
1: Allow capture/compare 2 DMA request

Bit 9 **CC1DE** : Allow capture/compare 1 DMA request (Capture/Compare 1 DMA request enable)
0: Disable capture/compare 1 DMA request
1: Allow capture/compare 1 DMA request

Bit 8 **UDE** : Update DMA request enable (Update DMA request enable)
0: Prohibit updated DMA request
1: Allow updated DMA request

Bit 7 **BIE** : Break interrupt enable (Break interrupt enable)
0: Disable brake interruption
1: Allow brake interruption

Bit 6 **TIE** : Trigger interrupt enable (Trigger interrupt enable)
0: Disable triggering interrupt
1: Enable trigger interrupt

Bit 5 **COMIE** : COM interrupt enable (COM interrupt enable)
0: Disable COM interrupt
1: Allow COM interrupt

Page 161

Bit 4	**CC4IE** : Allow capture/compare 4 interrupt (Capture/Compare 4 interrupt enable) 0: Disable capture/compare 4 interrupt 1: Allow capture/compare 4 interrupts
Bit 3	**CC3IE** : Allow capture/compare 3 interrupt enable (Capture/Compare 3 interrupt enable) 0: Disable capture/compare 3 interrupt 1: Allow capture/compare 3 interrupt
Bit 2	**CC2IE** : Allow capture/compare 2 interrupt (Capture/Compare 2 interrupt enable) 0: Disable capture/compare 2 interrupt 1: Allow capture/compare 2 interrupt
Bit 1	**CC1IE** : Allow capture/compare 1 interrupt (Capture/Compare 1 interrupt enable) 0: Disable capture/compare 1 interrupt 1: Allow capture/compare 1 interrupt
Bit 0	**UIE** : Update interrupt enable (Update interrupt enable) 0: Disable update interrupt 1: Allow update interruption

11.4.5 Status Register (TIMx_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve			CC4 OF	CC3 OF	CC2 OF	CC1 OF	Keep BIF TIF			COM IF	CC4 IF	CC3 IF	CC2 IF	CC1 IF	UIF
			rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 13	Reserve
Bit 12	**CC4OF** : Capture/Compare 4 overcapture flag See CC1OF description.
Bit 11	**CC3OF** : Capture/Compare 3 overcapture flag See CC1OF description.
Bit 10	**CC2OF** : Capture/Compare 2 overcapture flag See CC1OF description.
Bit 9	**CC1OF** : Capture/Compare 1 overcapture flag This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear this bit. 0: No repeated capture is generated; 1: When the value of the counter is captured into the TIMx_CCR1 register, the state of CC1IF is already 1.
Bit 8	Reserved, always read as 0.
Bit 7	**BIF** : Break interrupt flag (Break interrupt flag) Once the brake input is valid, the position is '1' by the hardware. If the brake input is invalid, this bit can be cleared to “0” by software 0: No brake event is generated 1: The effective level is detected on the brake input

Page 162

Bit 6	**TIF** : Trigger interrupt flag (Trigger interrupt flag) When a trigger event occurs (when the slave mode controller is in a mode other than gated mode, detect at the TRGI input To the valid edge, or any edge in the gated mode), this bit is set by the hardware. It is cleared by software. 0: No trigger event is generated 1: Trigger interrupt waiting for response
Bit 5	**COMIF** : COM interrupt flag (COM interrupt flag) Once a COM event is generated (when the capture/compare control bits: CCxE, CCxNE, OCxM have been updated), this bit is hardened Piece set 1. It is cleared by software. 0: No COM event is generated

	1: COM interrupt waiting for response
Bit 4	**CC4IF** : Capture/Compare 4 interrupt flag Refer to CC1IF description.
Bit 3	**CC3IF** : Capture/Compare 3 interrupt flag Refer to CC1IF description.
Bit 2	**CC2IF** : Capture/Compare 2 interrupt flag Refer to CC1IF description.
Bit 1	**CC1IF** : Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: When the counter value matches the comparison value, this bit is set to '1' by hardware, except in the center symmetric mode (refer to TIMx_CR1 CMS bit of the register). It is cleared to '0' by software. 0: No match occurred 1: The value of TIMx_CNT matches the value of TIMx_CCR1 If channel CC1 is configured as input mode: When a capture event occurs, this bit is set to '1' by hardware, and it is cleared to 0 by software or cleared to '0' by reading TIMx_CCR1. 0: No input capture is generated 1: The counter value has been captured (copied) to TIMx_CCR1 (the same edge as the selected polarity is detected on IC1)
Bit 0	**UIF** : Update interrupt flag (Update interrupt flag) This bit is set by hardware when an update event is generated. It is cleared to '0' by software. 0: No update event is generated 1: Update event waiting for response. This bit is set by hardware when the register is updated: − If the UDIS of the TIMx_CR1 register = 0, an update event is generated when REP_CNT = 0 (repeated down counter When overflow or underflow) − If UDIS = 0 and URS = 0 in the TIMx_CR1 register, when UG = 1 in the TIMx_EGR register, a change will occur. New event (software reinitializes the counter CNT) − If UDIS = 0 and URS = 0 in the TIMx_CR1 register, when the counter CNT is reinitialized by a trigger event Health update event. (Refer to the description of the synchronization control register)

11.4.6 Event generation register (**TIMx_EGR**)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

Bit 31: 8	Reserve
Bit 7	**BG** : Generate a brake event (Break generation) This bit is set to '1' by software to generate a brake event, and is automatically cleared to '0' by hardware. 0: No action 1: Generate a brake event. At this time, MOE = 0, BIF = 1, if the corresponding interrupt and DMA are turned on, the corresponding Corresponding interrupts and DMA
Bit 6	**TG** : Generate trigger event (Trigger generation) This bit is set to '1' by software to generate a brake event, and is automatically cleared to '0' by hardware. 0: No action 1: TIF of the TIMx_SR register = 1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated
	COMG : Capture/Compare event, generate control update (Capture/Compare control update generation) This bit is set to '1' by software and automatically cleared to '0' by hardware.

Bit 5	0: No action 1: When CCPC = 1, it is allowed to update the CCxE, CCxNE, OCxM bits Note: This bit is only valid for channels with complementary outputs.
Bit 4	**CC4G** : Generate capture/compare 4 generation events (Capture/Compare 4 generation) Refer to CC1G description.
Bit 3	**CC3G** : Generate Capture/Compare 3 generation events (Capture/Compare 3 generation) Refer to CC1G description.
Bit 2	**CC2G** : Generate Capture/Compare 2 generation events (Capture/Compare 2 generation) Refer to CC1G description.
Bit 1	**CC1G** : Generate capture/compare 1 event (Capture/Compare 1 generation) This bit is set by software to generate a capture/compare event and is automatically cleared by hardware. 0: No action 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as output: Set CC1IF=1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated. If channel CC1 is configured as input: The current counter value is captured to the TIMx_CCR1 register, set CC1IF = 1, if the corresponding interrupt and DMA, the corresponding interrupt and DMA are generated. If CC1IF is already 1, set CC1OF = 1.

Bit 0	**UG** : Generate update event (Update generation) This bit is set to '1' by software and automatically cleared to '0' by hardware. 0: No action 1: Reinitialize the counter and generate an update event. Note that the counter of the prescaler is also cleared to '0' (but the prescaler The frequency division coefficient remains unchanged). If in the center symmetry mode or DIR = 0 (counting up), the counter is cleared to '0'; if DIR = 1 (count down), the counter takes the value of TIMx_ARR.

11.4.7 Capture **/** Compare Mode Register **1** (**TIMx_CCMR1**)

Offset address: 0x18

Reset value: 0x0000

The channel can be used for input (capture mode) or output (comparison mode), and the direction of the channel is defined by the corresponding CCxS. This register other The role of the bit is different from that in the output mode. OCxx describes the function of the channel in output mode, and ICxx describes the function of the channel in output mode. can. Therefore, it must be noted that the function of the same bit in output mode and input mode is different.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C2C E	OC2M			OC2P E	OC2F E	CC2S		0C1C E	0C1M			OC1P E	OC1F E	CC1S	
IC2F				IC2PSC				IC1F				IC1PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output comparison mode:

Bit 15	**OC2CE** : Output compare 2 clear enable
Bit 14: 12	**0C2M[2** : **0]** : Output compare 2 mode
Bit 11	**OC2PE** : Output compare 2 preload enable
Bit 10	**OC2FE** : Output compare 4 fast enable
Bit 9: 8	**CC2S[1** : **0]** : Capture/Compare 2 selection This bit defines the direction of the channel (input/output), and the selection of input pins: 00: CC2 channel is configured as output 01: CC2 channel is configured as input, IC2 is mapped on TI2 10: CC2 channel is configured as input, IC2 is mapped on TI1 11: The CC2 channel is configured as an input, and IC2 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).

Bit 7	**OC1CE** : Output compare 1 clear enable (Output compare 1 clear enable) 0: OC1REF is not affected by ETRF input 1: Once the ETRF input high level is detected, clear OC1REF = 0

Bit 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 values of OC1 and OC1N. OC1REF is effective at high level, while the effective level of OC1 and OC1N depends on the CC1P and CC1NP bits. 000: Freeze. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT does not affect OC1REF effect 001: Set channel 1 as the effective level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, force OC1REF to be high 010: Set channel 1 to an invalid level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, force OC1REF to be low 011: Flip. When TIMx_CCR1=TIMx_CNT, flip the level of OC1REF 100: Forced to an invalid level. Force OC1REF to low 101: Forced to be a valid level. Force OC1REF to be high 110: PWM mode 1-When counting up, once TIMx_CNT <TIMx_CCR1, channel 1 is the active level, Otherwise, it is an invalid level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is an invalid level (OC1REF = 0), otherwise the effective level (OC1REF = 1) 111: PWM mode 2-When counting up, once TIMx_CNT <TIMx_CCR1, channel 1 becomes an invalid level, Otherwise, it is the effective level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is the effective level, Otherwise invalid level Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S = 00 (this pass Channel is configured as output) then this bit cannot be modified. Note 2: In PWM mode 1 or PWM mode 2, only when the comparison result changes or freezes from the output comparison mode The OC1REF level only changes when the mode is switched to PWM mode.
Bit 3	**OC1PE** : Output compare 1 preload enable 0: Disable the preload function of the TIMx_CCR1 register, and can write to the TIMx_CCR1 register at any time, and write a new one The value of has an immediate effect 1: Turn on the preload function of the TIMx_CCR1 register, read and write operations only operate on the preload register, The preload value of TIMx_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 TIMx_BDTR register) and CC1S = 00 (this pass Channel is configured as output) then this bit cannot be modified. Note 2: Only in single pulse mode (OPM = 1 in the TIMx_CR1 register), you can preload the register without confirming In this case, use the PWM mode, otherwise its action is uncertain.
Bit 2	**OC1FE** : Output compare 1 fast enable This bit is used to speed up the response of the CC output to the trigger input event. 0: According to the value of the counter and CCR1, CC1 operates normally, even if the trigger is turned on. When the trigger input has At a valid edge, the minimum delay for activating the CC1 output is 5 clock cycles 1: The effective edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to compare power It has nothing to do with the comparison result. The delay between the valid edge of the sampling flip-flop and the output of CC1 is shortened to 3 clock cycles. OCFE only works when the channel is configured in PWM1 or PWM2 mode

Page 166

Bit 1: 0	**CC1S[1 : 0]** : Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 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 only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

Input capture mode:

Bit 15: 12	**IC2F[3 : 0]** : Input capture 2 filter
Bit 11: 10	**IC2PSC[1 : 0]** : Input/capture 2 prescaler (Input capture 2 prescaler)
Bit 9: 8	**CC2S[1 : 0]** : Capture/Compare 2 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC2 channel is configured as output 01: CC2 channel is configured as input, IC2 is mapped on TI2 10: CC2 channel is configured as input, IC2 is mapped on TI1 11: CC2 channel is configured as input, IC2 is mapped on TRC, this mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).
Bit 7: 4	**IC1F[3 : 0]** : 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 group After it records N events, it will produce an output transition: 0000: No filter, sampling with f_{DTS} 1000: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 6 0001: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 2 1001: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 8 0010: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 4 1010: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 5 0011: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 8 1011: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 6 0100: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 6 1100: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 8 0101: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 8 1101: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 5 0110: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 6 1110: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 6 0111: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 8 1111: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 8

Page 167

Bit 3: 2	**IC1PSC[1 : 0]** : Input/capture 1 prescaler (Input capture 1 prescaler) These 2 bits define the prescaler coefficient of the CC1 input (IC1). Once CC1E = 0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, every edge detected on the capture input port triggers a capture 01: Trigger a capture every 2 events 10: Trigger a capture every 4 events 11: Trigger a capture every 8 events

Bit 1: 0	**CC1S[1 : 0]** : Capture/compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 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 only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

11.4.8 Capture **/** Compare Mode Register **2** (**TIMx_CCMR2**)

Offset address: 0x1C

Reset value: 0x0000

See the description of the CCMR1 register above.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C4CE	OC4M			OC4PE	OC4FE	CC4S		0C3CE	0C3M			OC3PE	OC3FE	CC3S	
IC4F				IC4PSC				IC3F				IC3PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15	**OC4CE** : Output compare 4 clear enable (Output compare 4 clear enable)
Bit 14: 12	**OC4M[2 : 0]** : Output compare 4 mode (Output compare 4 mode)
Bit 11	**OC4PE** : Output compare 4 preload enable
Bit 10	**OC4FE** : Output compare 4 fast enable
Bit 9: 8	**CC4S[1 : 0]** : Capture/Compare 4 selection The 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped to TRC, this mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC4S is only writable when the channel is closed (CC4E = 0 in the TIMx_CCER register).
Bit 7	**OC3CE** : Output compare 3 clear enable (Output compare 3 clear enable)
Bit 6: 4	**OC3M[2 : 0]** : Output compare 3 mode (Output compare 3 mode)

Bit 3	**OC3PE** : Output compare 3 preload enable
Bit 2	**OC3FE** : Output compare 3 fast enable
Bit 1: 0	**CC3S[1 : 0]** : Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC, this mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC3S is only writable when the channel is closed (CC3E = 0 in the TIMx_CCER register).

Input comparison mode

Bit 15: 12	**IC4F[3 : 0]** : Input capture 4 filter
Bit 11: 10	**IC4PSC[1 : 0]** : Input/capture 4 prescaler (Input capture 4 prescaler)
Bit 9: 8	**CC4S[1 : 0]** : Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3

	11: CC4 channel is configured as input, IC4 is mapped to TRC, this mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC4S is only writable when the channel is closed (CC4E = 0 in the TIMx_CCER register).
Bit 7: 4	**IC3F[3 : 0]** : Input capture 3 filter
Bit 3: 2	**IC3PSC[1 : 0]** : Input/capture 3 prescaler (Input capture 3 prescaler)
Bit 1: 0	**CC3S[1 : 0]** : Capture/compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped on TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 11: CC3 channel is configured as input, IC3 is mapped on TRC, this mode only works when the internal trigger input is selected Time (selected by TS bit of TIMx_SMCR register) Note: CC3S is only writable when the channel is closed (CC3E = 0 in the TIMx_CCER register).

11.4.9 Capture / Compare Enable Register (**TIMx_CCER**)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC1E	CC1NP	CC1NE	CC1P	CC1E
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 15: 14	Reserved, always read as 0.
Bit 13	**CC4P** : Input/Capture 4 output polarity (Capture/Compare 4 output polarity) Refer to the description of CC1P.
Bit 12	**CC4E** : Input/Capture 4 output enable (Capture/Compare 4 output enable) Refer to the description of CC1E.
Bit 11	**CC3NP** : Input/Capture 3 complementary output polarity (Capture/Compare 3 complementary output polarity) Refer to the description of CC1NP.
Bit 10	**CC3NE** : Input/Capture 3 complementary output enable (Capture/Compare 3 complementary output enable) Refer to the description of CC1NE.
Bit 9	**CC3P** : Input/Capture 3 output polarity (Capture/Compare 3 output polarity) Refer to the description of CC1P.
Bit 8	**CC3E** : Input/Capture 3 output enable (Capture/Compare 3 output enable) Refer to the description of CC1E.
Bit 7	**CC2NP** : Input/Capture 2 complementary output polarity (Capture/Compare 2 complementary output polarity) Refer to the description of CC1NP.
Bit 6	**CC2NE** : Capture/Compare 2 complementary output enable Refer to the description of CC1NE.
Bit 5	**CC2P** : Input/Capture 2 output polarity (Capture/Compare 2 output polarity) Refer to the description of CC1P.
Bit 4	**CC2E** : Capture/Compare 2 output enable Refer to the description of CC1E.
	CC1NP : Input/Capture 1 complementary output polarity (Capture/Compare 1 complementary output polarity)

Bit 3	0: OC1N is active at high level 1: OC1N is active at low level Note: Once the LOCK level (LCCK bit in the TIMx_BDTR register) is set to 3 or 2 and CC1S = 00 (channel configuration Set to output) then this bit cannot be modified.
Bit 2	**CC1NE** : Input/Capture 1 complementary output enable (Capture/Compare 1 complementary output enable) 0: Off-OC1N prohibits output, so the output level of OC1N depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bit value 1: On-OC1N signal is output to the corresponding output pin, and its output level depends on MOE, OSSI, OSSR, Values of OIS1, OIS1N and CC1E bits

Bit 1	**CC1P** : Input/Capture 1 output polarity (Capture/Compare 1 output polarity) CC1 channel is configured as output: 0: OC1 is active at high level 1: OC1 is active at low level CC1 channel is configured as input: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: No inversion: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 does not invert 1: Reverse: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 reverses Note: Once the LOCK level (LCCK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.
Bit 0	**CC1E** : Input/Compare 1 output enable (Capture/Compare 1 output enable) CC1 channel is configured as output: 0: Off-OC1 is forbidden to output, so the output level of OC1 depends on MOE, OSSI, OSSR, OIS1, Value of OIS1N and CC1NE bits 1: On-OC1 signal is output to the corresponding output pin, and its output level depends on MOE, OSSI, OSSR, Values of OIS1, OIS1N and CC1NE bits CC1 channel is configured as input: This bit determines whether the value of the counter can be captured into the TIMx_CCR1 register. 0: Capture prohibited 1: Capture enable

Table **26.** Control bits of complementary output channels **OCx** and **OCxN** with brake function

Control bit					Output status (1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output status	OCxN output status
1	X	0	0	0	Output prohibited (disconnected from timer) OCx = 0, OCx_EN = 0	Output prohibited (disconnected from timer) OCxN = 0, OCxN_EN = 0
		0	0	1	Output prohibited (disconnected from timer) OCx = 0, OCx_EN = 0	OCxREF + polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	0	OCxREF + polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Output prohibited (disconnected from timer) OCxN = 0, OCxN_EN = 0
		0	1	1	OCxREF + polarity + dead zone, OCx_EN=1	OCxREF inversion + polarity + dead zone, OCxN_EN = 1
		1	0	0	Output prohibited (disconnected from timer) OCx = CCxP, OCx_EN = 0	Output prohibited (disconnected from timer) OCxN = CCxNP, OCxN_EN = 0
		1	0	1	Closed state (the output is enabled and invalid Ping) OCx = CCxP, OCx_EN = 1	OCxREF + polarity, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	0	OCxREF + polarity, OCx = OCxREF xor CCxP, OCx_EN = 1	Closed state (output enabled and invalid level) OCxN = CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + polarity + dead zone, OCx_EN = 1	OCxREF inversion + polarity + dead zone, OCxN_EN = 1
0	0	X	0	0	Output prohibited (disconnected from the timer)	

Control bit			Output status (1)
MOE bit OSSI bit OSSR bit CCxE bit CCxNE bit			OCx output status OCxN output status
0	0	1	Asynchronously: OCx = CCxP, OCx_EN = 0,
0	1	0	OCxN = CCxNP, OCxN_EN = 0;
0	1	1	If the clock exists: OCx = OISx after a dead time, OCxN = OISxN, Assume that OISx and OISxN do not both correspond to the effective levels of OCx and OCxN.
1	0	0	Closed state (output enabled and invalid level)
1	0	1	Asynchronously: OCx = CCxP, OCx_EN = 1,
1	1	0	OCxN = CCxNP, OCxN_EN = 1;
1	1	1	If the clock exists: OCx = OISx after a dead time, OCxN = OISxN, Assume that OISx and OISxN do not both correspond to the effective levels of OCx and OCxN.

1. If both outputs of a channel are not used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP and CCxNP
Both must be cleared.

Note: The status of the external *I/O* pins connected to the complementary *OCx* and *OCxN* channels depends on the status of the *OCx* and *OCxN* channels and *GPIO* and *AFIO* registers.

11.4.10 Counter (**TIMx_CNT**)

Offset address: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CNT[31 : 0]** : Counter value

11.4.11 Prescaler (**TIMx_PSC**)

Offset address: 0x28

Reset value: 0x0000

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

Bit 15:0	**PSC[15 : 0]** : Prescaler value The clock frequency of the counter (CK_CNT) is equal to f $_{CK\_PSC}$ / (PSC[15:0] + 1). The PSC contains the value loaded into the current prescaler register each time an update event occurs. Update event including count The device is cleared to '0' by the UG bit of TIM_EGR or is cleared to '0' by the slave controller working in reset mode.

11.4.12 Auto-load register (**TIMx_ARR**)

Offset address: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**ARR[31 : 0]** : Prescaler value ARR contains the value to be loaded into the actual auto-reload register. For details, refer to section 13.3.1: Updates and actions related to ARR. When the value of auto reload is empty, the counter does not work.

11.4.13 Repeat count register (**TIMx_RCR**)

Offset address: 0x30

Reset value: 0x0000

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

Bit 15: 8	Reserved, always read as 0.
Bit 7:0	**REP[7 : 0]** : Repetition counter value After the preload function is enabled, these bits allow the user to set the update rate of the compare register (that is, periodically from the preload The load register is transferred to the current register); if the update interrupt is allowed, it will also affect the generation of the update interrupt. rate. Every time the down counter REP_CNT reaches 0, an update event is generated and the counter REP_CNT restarts from The REP value starts counting. Since REP_CNT only reloads the REP value when the period update event U_RC occurs, so The new value written to the TIMx_RCR register only takes effect when the next cycle update event occurs. This means that in the PWM mode, (REP+1) corresponds to: -In edge-aligned mode, the number of PWM cycles -In center symmetric mode, the number of PWM half cycles

11.4.14 Capture **/** Compare Register **1** (**TIMx_CCR1**)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR1[31 : 0]** : Capture/Compare 1 value If the CC1 channel is configured as output: CCR1 contains the value loaded into the current capture/compare 1 register (preload value). If the preload function is not selected in the TIMx_CCMR1 register (OC1PE bit), the written value will be transmitted immediately Input to the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 1 In the register. The current capture/compare register participates in the comparison with the counter TIMx_CNT, and is generated on the OC1 port

Produce output signal.

If the CC1 channel is configured as input:

CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).

11.4.15 Capture **/** Compare Register **2** (**TIMx_CCR2**)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR2[31** : **0]** : Capture/Compare 2 value If the CC2 channel is configured as output: CCR2 contains the value loaded into the current capture/compare 2 register (preload value). If the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit), the written value will be transmitted immediately Input to the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 2 In the register. The current capture/compare register participates in the comparison with the counter TIMx_CNT, and is generated on the OC2 port Produce output signal. If the CC2 channel is configured as input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).

11.4.16 Capture **/** Compare Register **3** (**TIMx_CCR3**)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR3[31** : **0]** : Capture/Compare 3 value If the CC3 channel is configured as output: CCR3 contains the value loaded into the current capture/compare 3 register (preload value). If the preload feature is not selected in the TIMx_CCMR3 register (OC3PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 3 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC3 port Signal. If the CC3 channel is configured as input: CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).

11.4.17 Capture **/** Compare Register **4** (**TIMx_CCR4**)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR4[31** : **0]** : Capture/Compare 4 value If the CC4 channel is configured as output: CCR4 contains the value loaded into the current capture/compare 4 register (preload value). If the preload feature is not selected in the TIMx_CCMR4 register (OC4PE bit), the written value will be transmitted immediately Input to the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 4 In the register. The current capture/compare register participates in the comparison with the counter TIMx_CNT and is generated on the OC4 port output signal. If the CC4 channel is configured as input: CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).

11.4.18 Brake and dead zone register (**TIMx_BDTR**)

Offset address: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK		DTG							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: According to the lock setting, *AOE* , *BKP* , *BKE* , *OSSI* , *OSSR* and *DTG[7* : *0]* bits can all be write-protected, it is necessary to They are configured each time the *TIMx_BDTR* register is written .

Bit 15	**MOE** : Main output enable Once the brake input is valid, this bit is asynchronously cleared to '0' by the hardware. According to the setting value of the AOE bit, this bit can be cleared to '0' by software Or it is automatically set to '1'. It is only valid for channels configured as output. 0: Prohibit OC and OCN output or force to idle state 1: If the corresponding enable bit (CCxE and CCxNE bits in the TIMx_CCER register) is set, then OC and OCN output For details about OC/OCN enable, see Section 15.4.9, Capture/Compare Enable Register (TIMx_CCER).
Bit 14	**AOE** : Automatic output enable 0: MOE can only be set to '1' by software 1: MOE can be set to '1' by the software or automatically set to 1 in the next update event (if the brake input is invalid) Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.
Bit 13	**BKP** : Break polarity 0: The brake input is active at low level 1: The brake input is active at high level Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.
Bit 12	**BKE** : Break enable 0: Prohibit brake input (BRK and BRK_ACTH) 1: Turn on the brake input (BRK and BRK_ACTH) Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.
Bit 11	**OSSR** : Off-state selection for Run mode This bit is used when MOE = 1 and the channel is complementary output. There is no OSSR bit in timers without complementary outputs. Refer to the detailed description of OC/OCN enable (Section 12.4.9, Capture/Compare Enable Register (TIMx_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, first turn on OC/OCN and output an invalid level. Then set OC/OCN enable output signal = 1 Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.
Bit 10	**OSSI** : Off-state selection for Idle mode This bit is used when MOE = 0 and the channel is set to output. Refer to the detailed description of OC/OCN enable (Section 15.4.9, Capture/Compare Enable Register (TIMx_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 outputs its idle level first, Then OC/OCN enable output signal = 1 Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.

Bit 9: 8	**LOOK[1 : 0]** : Lock configuration This bit provides write protection to prevent software errors. 00: lock is off, the register is not write-protected 01: Lock level 1, cannot write DTG, BKE, BKP, AOE bits and TIMx_CR2 of the TIMx_BDTR register OISx/OISxN bits of the register 10: Lock level 2, you cannot write each bit in lock level 1, nor can you write the CC polarity bit (once the relevant channel is connected The CCxS bit is set to output, the CC polarity bit is the CCxP/CCNxP bit of the TIMx_CCER register) and OSSR/OSSI bit 11: Lock level 3, you cannot write each bit in the lock level 2, nor can you write the CC control bit (once the relevant channel is connected The CCxS bit is set to output, and the CC control bit is the OCxM/OCxPE bit of the TIMx_CCMRx register) Note: After the system is reset, the LOCK bit can only be written once, once written to the TIMx_BDTR register, its content is frozen Until reset.
Bit 7:0	**UTG[7 : 0]** : Dead-time generator setup These bits define the duration of the dead zone 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 is 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 TIMx_BDTR register) is set to 1, 2 or 3, these cannot be modified Bit.

11.4.19 **DMA** control register (**TIMx_DCR**)

Offset address: 0x48

Reset value: 0x0000

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

Bit 15: 13	Reserved, always read as 0.

DBL[4 : 0] : DMA continuous transfer length (DMA burst length)

These bits define the transfer length of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register,

The timer performs a continuous transmission), that is, the number of transmissions is defined. The transmission can be half-word (double-byte) or word

Bit 12: 8	Festival: 00000: 1 transmission 00001: 2 transmissions 00010: 3 transmissions .... 10001: 18 transmissions Example: We consider such a transmission: DBL = 7, DBA = TIM2_CR1 -If DBL=7, DBA = TIM2_CR1 represents the address of the data to be transmitted, then the transmitted address is given by the following formula: (Address of TIMx_CR1) + DBA + (DMA index), where DMA index = DBL Among them (the address of TIMx_CR1) + DBA plus 7, gives the address of the data to be written or read, so Data transfer will occur in 7 registers starting from address (TIMx_CR1 address) + DBA. According to DMA For the setting of data length, the following situations may occur: -If the data is set to a half word (16 bits), the data will be transferred to all 7 registers. -If the data is set to bytes, the data will still be transferred to all 7 registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. So for the timer, the user must Specify the data width to be transferred by DMA.
Bit 7: 5	Reserved, always read as 0.
Bit 4: 0	**DBA[4 : 0]** : DMA base address These bits define the base address of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register), DBA is defined as the offset from the address where the TIMx_CR1 register is located: 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR ......

11.4.20 DMA address in continuous mode (TIMx_DMAR)

Offset address: 0x4C

Reset value: 0x0000

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

Bit 15:0	**DMAB[15 : 0]** : DMA continuous transfer register (DMA register for burst accesses) Reading or writing to the TIMx_DMAR register will result in the access operation to the register at the following address: TIMx_CR1 address + DBA + DMA index, where:'TIMx_CR1 address' is the control register 1 (TIMx_CR1) Address 'DBA' is the base address defined in the TIMx_DCR register; 'DMA index' is the offset automatically controlled by DMA, which depends on the DBL defined in the TIMx_DCR register.

12. General timer (TIM3/4)

12.1 Introduction to TIMx

The general-purpose timer is a 32-bit auto-loading counter driven by a programmable prescaler. It is suitable for many occasions, including testing Measure the pulse length of the input signal (input capture) or generate the output waveform (output comparison and PWM).

Using timer prescaler and RCC clock controller prescaler, the pulse length and waveform period can be between several microseconds to several milliseconds Adjustment.

TIMx timers are completely independent and do not share any resources with each other. They can operate simultaneously.

12.2 TIMx main functions

Common TIMx (TIM3, TIM4) timer functions include:

- 32-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, the frequency division factor of the counter clock frequency is any number between 1 and 65536 value

- 4 independent channels
  - Input capture
  - Output comparison
  - PWM generation (edge or center aligned mode)
  - Single pulse mode output
- Use external signal to control timer and timer interconnection synchronization circuit
- Interrupt/DMA is generated when the following events occur:
  - Update: Counter overflow/downflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output comparison
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning
- Trigger input as external clock or cycle current management

Figure **64.** Block diagram of a general-purpose timer

Note: According to the setting of the control bit, the content of the preload register is transferred to the working register in the *U* event

event

Interrupt and *DMA* output

12.3 **TIMX** function description

12.3.1 Time base unit

The main part of the programmable general-purpose timer is a 32-bit counter and its associated auto-loading register. This counter can go up
Count, count down, or count up and down in both directions. This counter clock is divided by the prescaler.

The counter, auto-load 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:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto reload register (TIMx_ARR)

The auto-load register is pre-loaded, and writing or reading the auto-reload register will access the pre-load register. According to the post in TIMX_CR1
The auto-loading pre-loading enable bit (ARPE) setting in the register, the content of the pre-loading register is immediately or in every update event UEV

When transferred to the shadow register. When the counter reaches the overflow condition (underflow condition when counting down) and when the UDIS in the TIMX_CR1 register
When the bit is equal to 0, an update event is generated. Update events can also be generated by software. The generation of update events under each configuration will be described in detail

The counter is driven by the clock output CK_CNT of the prescaler, only when the counter enable in the counter TIMX_CR1 register is set
CK_CNT is valid only when bit (CEN). (For details of counter enable, please refer to the description of the slave mode of the controller).

Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a (sent in TIMx_PSC
32-bit counter controlled by 16-bit register. Because this control register has a buffer, it can be changed during operation.
The parameters of the new prescaler will be adopted when the next update event arrives.

The following two figures show examples of changing counter parameters when the prescaler is running.

Figure **65.** When the prescaler parameter changes from **1** to **2** , the timing diagram of the counter

Figure **66.** When the prescaler parameter changes from **1** to **4** , the timing diagram of the counter

12.3.2 Counting mode

Up counting mode

In the up-counting mode, the counter counts from 0 to the auto-load value (the content of the TIMx_ARR counter), and then restarts from 0 Count and generate a counter overflow event.

An update event can be generated every time the counter overflows. Set the UG bit in the TIMx_EGR register (by software or by using slave Mode controller) can also generate an update event.

Setting the UDIS bit in the TIMx_CR1 register can disable the update event; this can avoid writing new data to the preload register. The shadow register is updated when the value is set. Until the UDIS bit is cleared to 0, no update event will be generated. But when an update event should be generated, the counter It will still be cleared to 0, and the count of the prescaler will be set to 0 (but the value of the prescaler will not change). In addition, if the TIMx_CR1 register is set URS bit (select update request) in the device, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (that is, no Generate an interrupt or DMA request). This is to avoid generating update and capture interrupts at the same time when clearing the counter in capture mode.

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

- The buffer of the prescaler is put into the value of the preload register (the content of the TIMx_PSC register)
- The autoload shadow register is reset to the value of the preload register (TIMx_ARR)

The following figure gives some examples, when TIMx_ARR = 0x36, the action of the counter at different clock frequencies:

Figure **67.** Counter timing diagram, the internal clock division factor is **1**

Figure **68.** Counter timing diagram, the internal clock division factor is **2**

Figure **69.** Counter timing diagram, the internal clock division factor is **4**

Figure **70.** Counter timing diagram, the internal clock division factor is **N**

Figure **71.** Counter timing diagram, update event when **ARPE = 0** (**TIMx_ARR is** not preloaded)

Figure **72.** Counter timing diagram, update event when **ARPE = 1** (**TIMx_ARR is preloaded**)

Down counting mode

In the down mode, the counter starts from the automatically loaded value (the value of the TIMx_ARR counter) and counts down to 0, and then starts from the automatically loaded valu
The entered value restarts and a counter underflow event is generated.

An update event can be generated every time the counter overflows. Set the UG bit in the TIMx_EGR register (by software or by using slave
Mode controller) can also generate an update event.

The UEV event can be disabled by setting the UDIS bit in the TIMx_CR1 register. This can avoid changing when writing a new value to the preload register.
New shadow register. Therefore, no update event will be generated before the UDIS bit is cleared to 0. However, the counter will still be reloaded from the current autoload value.
Start counting, and the counter of the prescaler restarts from 0 (but the rate of the prescaler cannot be modified).

In addition, if the URS bit in the TIMx_CR1 register is set (select update request), setting the UG bit will generate an update event
UEV but does not set the UIF flag (so interrupts and DMA requests are not generated). This is to avoid the occurrence of a capture event and clear the counter.
Both update and capture interrupts are generated at the same time.

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

- The buffer of the prescaler is set to the value of the preload register (the value of the TIMx_PSC register).
- The current autoload register is updated to the preload value (contents in the TIMx_ARR register). Note: Automatic loading in the count
  The device is updated before reloading, so the next cycle will be the expected value.

The following are some examples of counter operations at different clock frequencies when TIMx_ARR = 0x36:

Figure **73.** Counter timing diagram, the internal clock division factor is **1**

Figure **74.** Counter timing diagram, the internal clock division factor is **2**

Figure **75.** Counter timing diagram, the internal clock division factor is **4**

Figure **76.** Counter timing diagram, the internal clock division factor is **N**

Figure **77.** Counter timing diagram, update event when repeated counter is not used

Center alignment mode (count up **/** down)

In the center-aligned mode, the counter starts counting from 0 to the automatically loaded value (TIMx_ARR register) -1, resulting in a counter overflow Event, then count down to 1 and generate a counter underflow event; then restart counting from 0.

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current counting direction. Update event It can be generated at every count overflow and every count underflow; it can also be set by (software or using a slave mode controller) to set the TIMx_EGR register. The UG bit in the register is generated. At this time, the counter starts counting from 0 again, and the prescaler also starts counting from 0 again.

The UEV event can be disabled by setting the UDIS bit in the TIMx_CR1 register. This can avoid writing a new value to the preload register The shadow register is updated at time. Therefore, no update event will be generated before the UDIS bit is cleared to 0. However, the counter will still be automatically re-added based on the The loaded value continues to count up or down.

In addition, if the URS bit in the TIMx_CR1 register is set (select update request), setting the UG bit will generate an update event

UEV but does not set the UIF flag (so interrupts and DMA requests are not generated). This is to avoid the occurrence of a capture event and clear the counter.
Both update and capture interrupts are generated at the same time.

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

The UIF bit in the memory) is also set.

- The buffer of the prescaler is loaded with the value of the preload (TIMx_PSC register).
- The current autoload register is updated to the preload value (contents in the TIMx_ARR register). Note: If because of the counter
  If an update occurs due to overflow, the auto-reload will be updated before the counter is reloaded, so the next cycle will be the expected value (counter
  The counter is loaded with the new value).

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

Figure **78.** Counter timing diagram, the internal clock division factor is **1** , **TIMx_ARR = 0x6**

Figure **79.** Counter timing diagram, the internal clock division factor is **2**

Figure **80.** Counter timing diagram, the internal clock division factor is **4** , **TIMx_ARR = 0x36**

Figure **81.** Counter timing diagram, the internal clock division factor is **N**

Figure **82.** Counter timing diagram, update event when **ARPE = 1** (counter underflow)

Figure **83.** Counter timing diagram, update event when **ARPE = 1** (counter overflow)

12.3.3 Clock selection

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

- Internal clock (CK_INT)
- External clock mode 1: External input pin (TIx)
- External clock mode 2: External trigger input (ETR)
- Internal trigger input (ITRx): Use a timer as the prescaler of another timer, for example, you can configure a timer Timer1 is used as a prescaler for another timer Timer2.

Internal clock source (**CK_INT**)

If the slave mode controller is disabled (SMS = 000), the CEN, DIR (TIMx_CR1 register) and UG bit (TIMx_EGR Register) is the de facto control bit and can only be modified by software (the UG bit is still automatically cleared). Once the CEN bit is written as 1, pre-divide The clock of the frequency converter is provided by the internal clock CK_INT.

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

Figure **84.** The control circuit in normal mode, the internal clock division factor is **1**

External clock source mode **1**

When SMS = 111 in the TIMx_SMCR register, this mode is selected. The counter can be at each rising edge or down of the selected input Falling edge count.

Figure **85. TI2** external clock connection example

For example, to configure the up counter to count on the rising edge of the T12 input, use the following steps:

1. Configure TIMx_CCMR1 register CC2S = 01, configure channel 2 to detect the rising edge of TI2 input
2. Configure IC2F[3:0] in the TIMx_CCMR1 register, select the input filter bandwidth (if no filter is required, keep IC2F = 0000)

Note: The capture prescaler is not used as a trigger, so there is no need to configure it

3. Configure CC2P of the TIMx_CCER register = 0, select the rising edge polarity
4. Configure SMS = 111 in the TIMx_SMCR register, select timer external clock mode 1
5. Configure TS = 110 in the TIMx_SMCR register and select TI2 as the trigger input source

6. Set CEN = 1 in the TIMx_CR1 register to start the counter

When the 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 clock of the counter depends on the resynchronization circuit at the input of TI2.

Figure **86.** Control circuit in external clock mode **1**

External clock source mode **2**

The method to select this mode is: make ECE = 1 in the TIMx_SMCR register, the counter can trigger externally on each ETR Counting on rising or falling edges.

The following figure is the overall block diagram of the external trigger input:

Figure **87.** External trigger input block diagram

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

1. No filter is needed in this example, set ETF[3:0] in the TIMx_SMCR register = 0000
2. Set the prescaler, set ETPS[1:0] = 01 in the TIMx_SMCR register
3. Set the rising edge detection of ETR, set ETP = 0 in the TIMx_SMCR register
4. Turn on external clock mode 2, set ECE = 1 in the TIMx_SMCR register
5. Start the counter and set CEN = 1 in the TIMx_CR1 register

The counter counts once every 2 ETR rising edges.

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

Figure **88.** Control circuit in external clock mode **2**

12.3.4 Capture / Compare Channel

Each capture/compare channel is surrounded by a capture/compare register (including shadow registers), including the captured input part (data Word filtering, multiplexing and prescaler), and output section (comparator and output control).

The following pictures are an overview of the capture/compare channel. The input part samples the corresponding TIx input signal and generates a filtered signal No. TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx), which can be used as the input of the slave mode controller Trigger or as capture control. This signal enters the capture register (ICxPS) by prescaler.

Figure **89.** Capture / compare channel (eg: input part of channel **1**)

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

Figure **90.** Main circuit of capture / compare channel **1**

Figure **91.** The output section of the capture **/** compare channel (channel **1**)

The capture/compare module consists of a preload register and a shadow register. The read and write process only manipulates the preload register. In capture mode Under the formula, the capture occurs on the shadow register and then copied to the preload register.

In the 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.

12.3.5 Input Capture Mode

In the input capture mode, when the corresponding edge on the ICx signal is detected, the current value of the counter is latched into the capture/compare register (TIMx_CCRx). When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, if the Interrupt or DMA operation, an interrupt or DMA operation will be generated. If the CCxIF flag is already high when the capture event occurs, repeat the capture Get the flag CCxOF (TIMx_SR register) is set. Write CCxIF = 0 to clear CCxIF, or read stored in the TIMx_CCRx register The captured data in the memory can also clear CCxIF. Write CCxOF = 0 to clear CCxOF.

The following example shows how to capture the counter value into the TIMx_CCR1 register at the rising edge of TI1 input. The steps are as follows:

- Select a valid input terminal: TIMx_CCR1 must be connected to the TI1 input, so write CC1S in the TIMx_CCR1 register = 01, once CC1S is not 00, the channel is configured as an input, and the TM1_CCR1 register becomes read-only.
- 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 ICxF bit in the TIMx_CCMRx register). Assuming that the input signal jitters within a maximum of 5 clock cycles, we must The bandwidth of the configuration filter is longer than 5 clock cycles. So we can sample 8 times continuously (at fDTS frequency) to confirm that The last real edge transition of TI1, that is, write IC1F = 0011 in the TIMx_CCMR1 register.
- Select the valid conversion edge of the TI1 channel, and write CC1P = 0 (rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler Disabled (write IC1PS = 00 in the TIMx_CCMR1 register).
- Set CC1E = 1 in the TIMx_CCER register to allow the value of the counter to be captured into the capture register.
- If necessary, enable related interrupt requests by setting the CC1IE bit in the TIMx_DIER register, and by setting TIMx_DIER The CC1DE bit in the register allows DMA requests.

Occurs when an input is captured:

- When a valid level transition occurs, the value of the counter is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures have occurred, CC1IF has not been cleared.
- CC1OF is also set.

- If the CC1IE bit is set, an interrupt will be generated.
- If the CC1DE bit is set, a DMA request will also 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 loss in reading the capture overflow flag Capture overflow information that may occur after and before the data is read.

Note: By setting the corresponding *CCxG* bit in the *TIMx_EGR* register, the input capture interrupt and */* or *DMA* request can be generated by software .

12.3.6 PWM input mode

This mode is a special case of the input capture mode, except for the following differences, the operation is the same as the input capture mode:

- The two ICx signals are mapped to the same TIx input.
-

The two ICx signals are edge-valid, but the polarity is opposite.

- One of the TIxFP signals is used as a trigger input signal, and the slave mode controller is configured to reset mode.

For example, you need to measure the length (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of the PWM signal input to TI1. Register), the specific steps are as follows (depending on the frequency of CK_INT and the value of the prescaler)

- Select the valid input of TIMx_CCR1: set CC1S = 01 in the TIMx_CCMR1 register (select TI1).
- Select the valid polarity of TI1FP1 (used to capture data to TIMx_CCR1 and clear the counter): set CC1P = 0 (rising Valid along).
- Select the valid input of TIMx_CCR2: set CC2S in the TIMx_CCMR1 register = 10 (select TI1).
- Select the valid polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P = 1 (falling edge valid).
- Select a valid trigger input signal: set TS = 101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller to reset mode: set SMS = 100 in TIMx_SMCR.
- Enable capture: set CC1E = 1 and CC2E = 1 in the TIMx_CCER register.

Figure **92.** Timing of **PWM** input mode

Because only TI1FP1 and TI2FP2 are connected to the slave mode controller. So PWM input mode can only use TIMx_CH1 / TIMx_CH2 signal.

12.3.7 Forced output mode

In the output mode (CCxS = 00 in the TIMx_CCMRx register), the output compare signal (OCxREF and corresponding OCx) can be It can be directly forced into the valid or invalid state by software, without relying on the comparison result between the output compare register and the counter.

Set the corresponding OCxM = 101 in the TIMx_CCMRx register to force the output comparison signal (OCxREF/OCx) to be valid state. In this way, OCxREF is forced to be high (OCxREF is always active high), and at the same time, OCx gets the opposite polarity of CCxP value.

For example: CCxP = 0 (OCx is active at high level), then OCx is forced to be high.

Set OCxM = 100 in the TIMx_CCMRx register to force the OCxREF signal to be low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress, and the corresponding flags will also be modified. because This will still generate corresponding interrupts and DMA requests. This will be introduced in the output comparison mode section below.

12.3.8 Output Compare Mode

This function is used to control an output waveform or indicate when a given time has elapsed.

When the contents of the counter and the capture/compare register are the same, the output compare function does the following:

- The output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (the TIMx_CCER register in the The value defined by the CCxP bit) is output to the corresponding pin. During a comparison match, the output pin can maintain its level (OCxM = 000), set to effective level (OCxM = 001), set to no effective level (OCxM = 010), or reverse (OCxM = 011).
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- If the corresponding interrupt mask (CCXIE bit in the TIMx_DIER register) is set, an interrupt is generated.
- If the corresponding enable bit is set (CCxDE bit in TIMx_DIER register, CCDS bit in TIMx_CR2 register is selected

Select the DMA request function), a DMA request is generated.

The OCxPE bit in TIMx_CCMRx selects whether the TIMx_CCRx register needs to use the preload register.

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

The accuracy of synchronization can reach one counting cycle of the counter. The output compare mode (in the single pulse mode) can also be used to output a single pulse.

Configuration steps of output comparison mode:

1. Select the counter clock (internal, external, prescaler)
2. Write the corresponding data into the TIMx_ARR and TIMx_CCRx registers
3. To generate an interrupt request and/or a DMA request, set the CCxIE bit and/or CCxDE bit.
4. Select the output mode, for example: OCxM = '011', OCxPE = '0', CCxP = '0' and CCxE = '1' must be set, when counting
   When the CNT and CCRx match, the output pin of OCx is flipped, CCRx is preloaded unused, and the OCx output is turned on and the high level is active.
5. Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_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 TIMx_CCRx shadow register can only be updated when the next update event occurs). The following figure shows an example son.

Figure **93.** Output compare mode, flip **OC1**

12.3.9 PWM mode

The pulse width modulation mode can generate a frequency determined by the TIMx_ARR register and the duty cycle determined by the TIMx_CCRx register. Signal.

Write '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bit in the TIMx_CCMRx register, which can be independent Set each OCx output channel to generate a PWM. The OCxPE bit in the TIMx_CCMRx register must be set to enable the corresponding preload Register, and finally set the ARPE bit of the TIMx_CR1 register to enable automatic reloading of the preload register (in the upward counting or center In symmetric mode).

Because only when an update event occurs, the preload register can be transferred to the shadow register, so the counter starts counting Previously, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The polarity of OCx can be set by software in the CCxP bit in the TIMx_CCER register, and it can be set to active high or low The level is valid. The CCxE bit in the TIMx_CCER register controls the OCx output enable. See the description of the TIMx_CCERx register for details.

In the PWM mode (mode 1 or mode 2), TIMx_CNT and TIM1_CCRx are always being compared (according to the counter count Number direction) to determine whether it meets TIM1_CCRx ≤ TIM1_CNT or TIM1_CNT ≤ TIM1_CCRx. However in order to The function of OCREF_CLR (before the next PWM cycle, an external event on the ETR signal can clear OCxREF) is the same, The OCxREF signal can only be generated under the following conditions:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bit in the TIMx_CCMRx register) is switched from'freeze' (no comparison, OCxM = '000')

To a certain PWM mode (OCxM = '110' or '111').

In this way, the PWM output can be forced by software during operation. According to the state of the CMS bit in the TIMx_CR1 register, the timer can Generate edge-aligned PWM signals or center-aligned PWM signals.

PWM edge alignment mode

Up counting configuration

When the DIR bit in the TIMx_CR1 register is low, the count-up is executed.

The following is an example of PWM mode 1. When TIMx_CNT <TIMx_CCRx, the PWM signal reference OCxREF is high, no It is low. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains at '1'. If it is better than If the comparison value is 0, OCxREF remains at '0'. The following figure shows an example of an edge-aligned PWM waveform when TIMx_ARR = 8.

Figure **94.** Edge-aligned **PWM** waveform (**ARR = 8**)

Countdown configuration

When the DIR bit of the TIMx_CR1 register is high, the down count is executed.

In PWM mode 1, the reference signal OCxREF is low when TIMx_CNT> TIMx_CCRx, otherwise it is high. if The comparison value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF remains at '1'. 0% cannot be generated in this mode The PWM waveform.

PWM center alignment mode

When the CMS bit in the TIMx_CR1 register is not '00', it is the center-aligned mode (all other configurations affect the OCxREF/OCx signal Have the same effect). According to the setting of different CMS bits, the comparison flag can be set when the counter is counting up, and the comparison flag can be set when the counter is d It is set to 1 when counting, or when the counter is counting up and down. The counting direction bit (DIR) in the TIMx_CR1 register is updated by hardware New, don't modify it with software. See the chapter on center alignment mode.

The following figure shows some examples of center-aligned PWM waveforms

- TIMx_ARR = 8
- PWM mode 1
- CMS = 01 in the TIMx_CR1 register. In the center-aligned mode 1, the compare flag is set when the counter counts down.

Page 197

Figure **95.** Center-aligned **PWM** waveform (**APR = 8**)

Tips for using center alignment mode:

- When entering center-aligned mode, the current up/down count configuration is used; this means that the counter counts up or down depending on The current value of the DIR bit in the TIMx_CR1 register. In addition, the software cannot modify the DIR and CMS bits at the same time.
- It is not recommended to rewrite the counter when running in center-aligned mode because it will produce unpredictable results. In particular:
  - If the value of the written counter is greater than the value of auto-reload (TIMx_CNT> TIMx_ARR), the direction will not be updated. example For example, if the counter is counting up, it will continue counting up.
  - If 0 or the value of TIMx_ARR is written into the counter, the direction is updated, but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate a software update (set the TIMx_EGR bit before starting the counter). UG bit in), do not modify the value of the counter during the counting process.

12.3.10 Single pulse mode

Single pulse mode (OPM) is a special case of many of the aforementioned modes. This mode allows the counter to respond to a stimulus and in a program After the controllable delay, a pulse with programmable pulse width is generated.

The counter can be started by the slave mode controller to generate waveforms in output comparison mode or PWM mode. Set TIMx_CR1 The OPM bit in the register will select the single pulse mode, so that the counter can automatically stop when the next update event UEV is generated.

Only when the comparison value is different from the initial value of the counter can a pulse be generated. Before starting (when the timer is waiting to be triggered), the Must be configured as follows:

Page 198

- Up counting method: CNT <CCRx ≤ ARR (especially, 0<CCRx)
- Counting down method: CNT> CCRx

Figure **96.** Example of single pulse mode

For example, you need to detect a rising edge on the TI2 input pin, and after a delay of t DELAY , a length of t PULSE positive pulse.

Assuming TI2FP2 as trigger 1:

- Set CC2S = 01 in the TIMx_CCMR1 register to map TI2FP2 to TI2.
- Set CC2P = 0 in the TIMx_CCER register to enable TI2FP2 to detect the rising edge.
- Set TS = 110 in the TIMx_SMCR register, and TI2FP2 is used as the trigger (TRGI) of the slave mode controller.
- Set SMS in the TIMx_SMCR register = 110 (trigger mode), TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written in the compare register (the clock frequency and counter prescaler should be considered).

- t DELAY is defined by the value written to the TIMx_CCR1 register.
- t PULSE is defined by the difference between the autoload value and the comparison value (TIMx_ARR-TIMx_CCR1).
- Suppose that when a comparison match occurs, a waveform from 0 to 1 is to be generated. When the counter reaches the preload value, a waveform from 1 to 0 is generated.
  Waveform; first set the OC1M of the TIMx_CCMR1 register = 111, enter PWM mode 2; selectively use
  Register can be preloaded: set OC1PE in TIMx_CCMR1 = 1 and ARPE in TIMx_CR1 register; then
  Fill in the comparison value in the TIMx_CCR1 register, fill in the auto-load value in the TIMx_ARR register, and modify the UG bit to generate
  An update event, and then wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Because only one pulse is required, OPM = 1 in the TIMx_CR1 register must be set, and in the next update event (when the counter is from Stop counting when the auto-load value rolls over to 0).

Special case: **OCx** fast enable:

In the single pulse mode, the edge detection logic at the TIx input pin sets the CEN bit to start the counter. Then the counter and comparison value The comparison operation produces a conversion of the output. But these operations require a certain clock cycle, so it limits the minimum delay t DELAY that can be obtained .

If you want to output the waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register; at this time, force OCxREF (and OCx) is forced to respond to the stimulus instead of relying on the result of the comparison, and the output waveform is the same as the waveform when the comparison matches. OCxFE is on It works when set to PWM1 and PWM2 mode.

12.3.11 Clear **OCxREF** signal on external event

For a given channel, set the corresponding OCxCE bit in the TIMx_CCMRx register to a high level of '1' at the ETRF input The OCxREF signal can 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 comparison and PWM mode, and cannot be used in forced mode.

For example, the OCxREF signal can be connected to an external input. At this time, ETR must be configured as follows:

- The external trigger prescaler must be turned off: ETPS[1:0] = 00 in the TIMx_SMCR register.
- The external clock mode 2: ECE = 0 in the TIMx_SMCR register must be disabled.
- External trigger polarity (ETP) and external trigger filter (ETF) can be configured as required.

The figure below shows the action of the OCxREF signal corresponding to different OCxCE values when the ETRF input goes high. In this example, The timer TIMx is placed in PWM mode.

Figure **97.** Clear **OCxREF** of **TIMx**

12.3.12 Encoder interface mode

The method to select the encoder interface mode is: if the counter only counts on the edge of TI2, set the SMS in the TIMx_SMCR register = 001; if only counting on the edge of TI1, set SMS = 010; if the counter is counting on both edges of TI1 and TI2, set SMS = 011.

By setting the CC1P and CC2P bits in the TIMx_CCER register, the polarity of TI1 and TI2 can be selected; if necessary, you can also Program the input filter.

The two inputs TI1 and TI2 are used as the interface of the incremental encoder. The following table assumes that the counter has been started (TIMx_CR1 register CEN = 1), the counter is driven by each valid transition on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are TI1 and TI2 is the signal after passing the input filter and polarity control; if there is no filtering and disguising, then TI1FP1 = TI1; if there is no filtering and disguising, Then TI2FP2 = TI2. According to the jump sequence of the two input signals, count pulses and direction signals are generated. According to the transition of the two input signals Sequence, the counter counts up or down, and the hardware sets the DIR bit of the TIMx_CR1 register accordingly. Regardless of whether the counter is based on Count on TI1, count on TI2, or count on both TI1 and TI2. A transition on either input terminal (TI1 or TI2) will re-Calculate the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter is only between 0 and TIMx_ARR register auto-load value between continuous counting (according to the direction, or 0 to ARR count, or ARR to 0 count). Therefore, TIMx_ARR must be configured before starting to count; similarly, the capturer, comparator, prescaler, trigger output characteristics, etc. still work as often.

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 editing The location of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table lists all possible combinations, assuming that TI1 and Time changes.

Table **27.** Relation between counting direction and encoder signal

Effective edge	Relative signal level (TI1FP1 corresponds to TI2, TI2FP2 corresponds to TI1)	TI1FP1 signal		TI2FP2 signal	
		rise	decline	rise	decline
Only count at TI1	high	Count down	Count up	Not counted	Not counted
	Low	Count up	Count down	Not counted	Not counted
Only count at TI2	high	Not counted	Not counted	Count up	Count down
	Low	Not counted	Not counted	Count down	Count up
Count on TI1 and TI2	high	Count down	Count up	Count up	Count down
	Low	Count up	Count down	Count down	Count up

An external incremental encoder can be directly connected to the MCU without the need for external interface logic. However, a comparator is generally used to convert the encoder The differential output is converted to a digital signal, which greatly increases the ability to resist noise interference. The third signal output by the encoder represents the mechanical zero poin To connect it to an external interrupt input and trigger a counter reset.

The following figure is an example of counter operation, showing the generation and direction control of the counting signal. It also shows that when both sides are selected, How input jitter is suppressed; jitter may occur when the sensor is close to a switching point. In this example, we assume The configuration is as follows:

- CC1S = '01' (TIMx_CCMR1 register, IC1FP1 is mapped to TI1)
- CC2S = '01' (TIMx_CCMR2 register, IC2FP2 is mapped to TI2)
- CC1P = '0' (TIMx_CCER register, IC1FP1 is not inverted, IC1FP1 = TI1)
- CC2P = '0' (TIMx_CCER register, IC2FP2 is not inverted, IC2FP2 = TI2)
- SMS = '011' (TIMx_SMCR register, all inputs are valid on rising and falling edges).
- CEN = '1' (TIMx_CR1 register, counter enable)

Page 201

Figure **98.** Example of counter operation in encoder mode

The following figure shows the operation example of the counter when the polarity of IC1FP1 is reversed (CC1P = '1', other configurations are the same as the previous example)

Figure **99. IC1FP1** inverted encoder interface mode example

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. Use the second configuration timer in capture mode
By measuring the interval between two encoder events, dynamic information (speed, acceleration, deceleration) can be obtained. Encoder output indicating mechanical zero
Out can be used for this purpose. According to the interval between two events, the counter can be read out at a fixed time. If possible, you can put
The value of the counter is latched into the third input capture register (the capture signal must be periodic and can be generated by another timer). It also
Its value can be read through a DMA request generated by the real-time clock.

12.3.13 Timer input XOR function

The TI1S bit in the TIMx_CR2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, and 3 XOR gates
The input terminals are TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used for all timer input functions, such as triggering or input capture. Section 15.3.18 of the previous chapter gave this feature for
Example of connecting a Hall sensor.

Page 202

12.3.14 Timer and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in multiple modes: reset mode, gating mode and trigger mode.

Slave mode: reset mode

When a trigger input event occurs, the counter and its prescaler can be initialized again; at the same time, if TIMx_CR1 is registered The URS bit of the device is low, and an update event UEV is also generated; then all preload registers (TIMx_ARR, TIMx_CCRx) Have been updated.

- In the following example, the rising edge of the TI1 input causes the up counter to be cleared:
- 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 the trigger operation, so no configuration is required. CC1S bit only selects input capture Get the source, that is, CC1S = 01 in the TIMx_CCMR1 register. Set CC1P = 0 in the TIMx_CCER register to determine the polarity (Only the rising edge is detected).
- Set SMS = 100 in the TIMx_SMCR register to configure the timer in reset mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.
- Set CEN = 1 in the TIMx_CR1 register to start the counter.

The counter starts to count according to the internal clock, and then runs normally until TI1 has a rising edge; at this time, the counter is cleared and then reset from 0 Restart counting. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, according to the TIE in the TIMx_DIER register The setting of (interrupt enable) bit and TDE (DMA enable) bit generates an interrupt request or a DMA request.

The following figure shows the action when the auto reload register TIMx_ARR = 0x36. Between the rising edge of TI1 and the actual reset of the counter The delay depends on the resynchronization circuit at the input of TI1.

Figure **100.** Control circuit in reset mode

Slave mode: gated mode

The enable of the counter depends on the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- 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 the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, Set CC1S = 01 in the TIMx_CCMR1 register. Set CC1P = 1 in the TIMx_CCER register to determine the polarity (check only Measure the low level).
- Set SMS = 101 in the TIMx_SMCR register to configure the timer as gated mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.

Page 203

- Set CEN = 1 in the TIMx_CR1 register to start the counter. In gating mode, if CEN = 0, the counter cannot be started 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. Set when the counter starts or stops TIF marking in TIMx_SR.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the input of TI1.

Figure **101.** Control circuit in gating mode

Slave mode: trigger mode

The enabling of the counter depends on the event on the selected input.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed, keep IC2F = 0000). The capture prescaler is not used in the trigger operation, and no configuration is required. The CC2S bit is only used to select the input capture source, set CC2S = 01 in the TIMx_CCMR1 register. Set CC1P = 1 in the TIMx_CCER register to determine the polarity (only detect low Level).
- Set SMS = 110 in the TIMx_SMCR register to configure the timer as trigger mode; set TS = 110 in the TIMx_SMCR register, Select TI2 as the input source.

When TI2 has a rising edge, the counter starts to count under the internal clock drive, and the TIF flag is set 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 input of TI2.

Figure **102.** Control circuit in flip-flop mode

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 letter The signal is used as the input of the external clock, and another input can be selected as the trigger input in reset mode, gating mode or trigger mode. Not build It is recommended to use the TS bit of the TIMx_SMCR register to select ETR as TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up once on each rising edge of ETR:

- Configure the external trigger input circuit through the TIMx_SMCR register:
  - ETF = 0000: no filtering
  - ETPS = 00: no prescaler
  - ETP = 0: detect the rising edge of ETR, set ECE = 1 to enable external clock mode 2
- Configure channel 1 as follows to detect the rising edge of TI:
  - IC1F=0000: no filtering
  - The capture prescaler is not used in the trigger operation, no configuration is required
  - Set CC1S = 01 in the TIMx_CCMR1 register to select the input capture source
  - Set CC1P = 0 in the TIMx_CCER register to determine the polarity (only the rising edge is detected)
- Set SMS = 110 in the TIMx_SMCR register to configure the timer as trigger mode. Set TS = 101 in the TIMx_SMCR register, 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 circuit at the ETRP input.

Figure **103.** Control circuit in external clock mode **2 +** trigger mode

12.3.15 Timer synchronization

All TIMx timers are connected internally for timer synchronization or linking. When a timer is in the master mode, it can
The counter of the timer in the slave mode performs operations such as resetting, starting, stopping, or providing a clock.

The figure below shows an overview of the trigger selection and main mode selection modules.

Use one timer as a prescaler for another timer

Figure **104.** Examples of master **/** slave timers

For example: Timer 1 can be configured as the prescaler of Timer **3** . Refer to the figure above and perform the following operations:

- Configure Timer 1 as the main mode, which can output a periodic trigger signal at each update event UEV. exist
  When the MMS of the TIM1_CR2 register = '010', a rising edge signal is output on TRGO1 whenever an update event is generated.
- Connect TRGO1 of timer 1 to timer **3** , set TS of TIM3_SMCR register = '000', configure timer **3**
  To use ITR1 as the internally triggered slave mode.
- Then put the slave mode controller in external clock mode 1 (SMS = 111 in the TIM3_SMCR register); in this way, timer **3**
  It can be driven by the periodic rising edge of Timer 1 (that is, the counter overflow of Timer 1).
- Finally, the CEN bit of the corresponding (TIMx_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 counting of Timer ***3***
Device.

Use one timer to enable another timer

In this example, the operation of Timer **3** is controlled by the output comparison of Timer 1. Refer to Figure 42 for connections. Only for timer 1
Timer **3** counts the divided internal clock only when OC1REF is high . The clock frequency of the two timers is determined by the prescaler to CK_INT
Divide by 3 (fCK_CNT = fCK_INT/3) to get.

- Configure timer 1 as the main mode, and send out its output comparison reference signal (OC1REF) as the trigger output (TIM1_CR2 register MMS = 100)
- Configure the OC1REF waveform of timer 1 (TIM1_CCMR1 register)
- Configure Timer **3 to** get the input trigger from Timer 1 (TS = 001 in the TIM3_SMCR register)
- Configure Timer **3** in gating mode (SMS in TIM3_SMCR register = 101)

- Set CEN = 1 in TIM3_CR1 register to enable timer **3**
- Set CEN = 1 in TIM1_CR1 register to start timer 1

Note: The clock of Timer ***3*** is not synchronized with the clock of Timer *1*. This mode only affects the enable signal of the Timer ***3*** counter.

FIG. **105.** Timer **1** is **OC1REF** control timer **3**

In the example above, before Timer **3** starts, their counter and prescaler are not initialized, so they start from the current count The value starts counting. You can reset the 2 timers before starting Timer 1, so that they start from a given value, that is, in the timer counter Write any value you want. Write the UG bit in the TIMx_EGR register to reset the timer.

In the next example, timer 1 and timer **3** need to be synchronized . Timer 1 is in master mode and starts from 0, timer **3** is in slave mode And start from 0xE7; the prescaler coefficients of the two timers are the same. Writing 0 to the CEN bit of TIM1_CR1 will disable Timer 1. Timer **3** stopped immediately.

- Configure timer 1 as the main mode, and send the output comparison 1 reference signal (OC1REF) as the trigger output (TIM1_CR2 register The MMS of the device = 100).
- Configure the OC1REF waveform of Timer 1 (TIM1_CCMR1 register).
- Configure Timer **3 to** get the input trigger from Timer 1 (TS = 000 in the TIM3_SMCR register)
- Configure Timer **3** in gating mode (SMS in TIM3_SMCR register = 101)
- Set UG = 1 in the TIM1_EGR register to reset timer 1.
- Set UG = 1 in the TIM3_EGR register to reset timer **3** .
- Write 0xE7 to the timer **3** counter (TIM3_CNTL), and initialize it to 0xE7.
- Set CEN = 1 in the TIM3_CR1 register to enable Timer **3** .
- Set CEN = 1 in the TIM1_CR1 register to start timer 1.
- Set CEN = 0 in the TIM1_CR1 register to stop Timer 1.

Figure **106.** Timer **3** can be controlled by enabling Timer **1**

Use one timer to start another timer

In this example, the update event of timer 1 is used to enable timer **3** . Refer to Figure 43 for connections. Once timer 1 generates an update event Timer **3** starts counting from its current value (which can be non-zero) according to the divided internal clock. When the trigger signal is received, the timer The CEN bit of **3** is automatically set to 1, and the counter starts counting until 0 is written to the CEN bit of the TIM3_CR1 register. Two timer The clock frequency is divided by 3 by the prescaler pair CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Configure timer 1 as the main mode, and send its update event (UEV) as a trigger output (MMS in the TIM1_CR2 register = 010).
- Configure the period of timer 1 (TIM1_ARR register).
- Configure Timer **3 to** get the input trigger from Timer 1 (TS = 000 in the TIM3_SMCR register)
- Configure timer **3** as trigger mode (SMS in TIM3_SMCR register = 110)
- Set CEN = 1 in the TIM1_CR1 register to start timer 1.

Figure **107.** Use the update of Timer **1** to trigger Timer **3**

In the previous example, two counters can be initialized before starting counting. The figure below shows that in the same configuration as 0, using touch Sending mode instead of gating mode (SMS = 110 in the TIM3_SMCR register).

Figure **108.** Use the enable of Timer **1** to trigger Timer **3**

Use one timer as a prescaler for the other

This example uses Timer 1 as the prescaler for Timer **3** . The configuration is as follows:

- Configure Timer 1 as the main mode and send its update event UEV as the trigger output (MMS in the TIM1_CR2 register = '010'). Then output a periodic signal every time the counter overflows.
- Configure the period of timer 1 (TIM1_ARR register).
- Configure Timer **3 to** get the input trigger from Timer 1 (TS = 000 in the TIM3_SMCR register)
- Configure Timer **3 to** use external clock mode (SMS in TIM3_SMCR register = 111)
- Set CEN = 1 in TIM1_CR2 register to start timer **3**
- Set CEN = 1 in TIM1_CR1 register to start timer 1

Use an external trigger to start **2** timers synchronously

In this example, when the TI1 input of timer 1 rises, timer 1 is enabled, and timer 1 is enabled to enable timer **3** . To guarantee the counter Alignment, timer 1 must be configured as master/slave mode (corresponding to TI1 as slave, corresponding to timer **3** as master):

- Configure timer 1 as the main mode, and send its enable as a trigger output (MMS='001' in the TIM1_CR2 register)
- Configure timer 1 as slave mode, get input trigger from TI1 (TS = '100' in TIM1_SMCR register)
- Configure timer 1 as trigger mode (SMS='110' in TIM1_SMCR register)
- Configure Timer **3 to** get the input trigger from Timer 1 (TS = 000 in the TIM3_SMCR register)
- Configure timer **3** as trigger mode (SMS in TIM3_SMCR register = 110)

When a rising edge occurs on TI1 of timer 1, the two timers start counting synchronously according to the internal clock, and the two TIF flags also It is set at the same time.

NOTE: In this example, before starting the two timers are initialized (set corresponding *UG* bits), two counters are from *0* to open Start, but you can insert an offset between the timers by writing to any counter register (*TIMx_CNT*). The master */* slave can be seen in the picture below In the timer mode *1* of *CNT_EN* and *CK_PSC* between have a delay.

FIG. **109. The** use of the timer **1** is **TI1** input trigger Timer **1** and Timer **3**

12.3.16 Debug mode

When the microcontroller enters the debug mode (CPU core is stopped), according to the setting of DBG_TIMx_STOP in the DBG module, the TIMx counts The counter will either continue normal operation or stop. For details, see the chapter Debugging Module.

12.4 **TIMx** register description

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) mode.

12.4.1 Control Register **1** (**TIMx_CR1**)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 10	Reserve
Bit 9: 8	**CKD[1 : 0]** : Clock division factor (Clock division) These 2 bits are defined in the timer clock (CK_INT) frequency, dead time and the dead time generator and digital filter (ETR, TIx) The frequency division ratio between the sampling clocks used. 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: Reserved, do not use this configuration
Bit 7	**ARPE** : Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is loaded into the buffer

Bit 6: 5	**CMS[1 : 0]** : Select Center-aligned mode selection 00: Edge alignment mode. The counter counts up or down according to the direction bit (DIR). 01: Center alignment mode 1. The counter counts up and down alternately. Channel configured as output (TIMx_CCMRx register The output compare interrupt flag bit of CCxS = 00) is only set when the counter is counting down. 10: Center alignment mode 2. The counter counts up and down alternately. The counter counts up and down alternately. Configure to lose The output compare interrupt flag bit of the output channel (CCxS = 00 in the TIMx_CCMRx register), only counts up on the counter When is set. 11: Center alignment mode 3. The counter counts up and down alternately. The counter counts up and down alternately. Configure to lose The output compare interrupt flag bit of the output channel (CCxS = 00 in the TIMx_CCMRx register), when the counter is up and down It is set when counting. Note: When the counter is turned on (CEN = 1), it is not allowed to switch from edge-aligned mode to center-aligned mode.
Bit 4	**DIR** : Direction 0: The counter counts up 1: The counter counts down Note: When the counter is configured in center-aligned mode or encoder mode, this bit is read-only.
Bit 3	**OPM** : One pulse mode 0: When an update event occurs, the counter does not stop 1: When the next update event occurs (clear the CEN bit), the counter stops
Bit 2	**URS** : Update request source The software selects the source of the UEV event through this bit 0: If an update interrupt or DMA request is allowed, any of the following events will generate an update interrupt or DMA request: − Counter overflow/underflow − Set the UG bit − Updates generated from the mode controller 1: If the update interrupt or DMA request is allowed, only the counter overflow/underflow will generate an update interrupt or DMA request begging
Bit 1	**UDIS** : Update disable The software allows/disables the generation of UEV events through this bit 0: UEV is allowed. Update (UEV) events are generated by any of the following events: − Counter overflow/underflow − Set the UG bit

	− The updated buffered registers generated from the mode controller are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) maintain their values. If set If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.
Bit 0	**CEN** : Counter enable 0: disable the counter 1: Enable the counter Note: After the software sets the CEN bit, the external clock, gating mode and encoder mode can only work. Trigger mode can be automatic The CEN bit is set by hardware. In single pulse mode, when an update event occurs, CEN is automatically cleared.

12.4.2 Control Register 2 (TIMx_CR2)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								TI1S	MMS			CCDS	Reserve		
								rw	rw	rw	rw	rw			

Bit 31: 8	Reserve
Bit 7	**TI1S** : TI1 selection 0: TIMx_CH1 pin is connected to TI1 input; 1: TIMx_CH1, TIMx_CH2 and TIMx_CH3 pins are XORed and then connected to TI1 input.
Bit 6: 4	**MMS[1 : 0]** : Master mode selection These two bits are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. The possible combinations are as follows: 000: Reset-The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode If the controller is in reset mode) to generate a reset, the signal on TRGO will have a delay relative to the actual reset. 001: Enable-the counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes need to be at the same time Start multiple timers or control to enable slave timers within a period of time. The counter enable signal is controlled by the CEN control bit and gate The logic or generation of the trigger input signal in the mode. When the counter enable signal is controlled by the trigger input, there will be a A delay, unless the master/slave mode is selected (see the description of the MSM bit in the TIMx_SMCR register). 010: Update-The update event is selected as the trigger input (TRGO). For example, the clock of a master timer can be used as a A prescaler for the slave timer. 011: Comparison pulse-once a capture occurs or a comparison is successful, when the CC1IF flag is to be set (even if it has been Is high), the trigger output sends a positive pulse (TRGO). 100: Compare-OC1REF signal is used as trigger output (TRGO). 101: Compare-OC2REF signal is used as trigger output (TRGO). 110: Compare-OC3REF signal is used as trigger output (TRGO). 111: Compare-OC4REF signal is used as trigger output (TRGO).
Bit 3	**CCDS** : Capture/Compare DMA selection (Capture/Compare DMA selection) 0: When a CCx event occurs, send a CCx DMA request 1: When an update event occurs, send a CCx DMA request
Bit 2: 0	Reserved, always read as 0.

12.4.3 Slave mode control register (TIMx_SMCR)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF				MSM	TS			Reserve	SMS		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Page 212

Bit 31: 16	Reserve
Bit 15	**ETP** : External trigger polarity This bit selects whether to use ETR or the inversion of ETR as the trigger operation 0: ETR is not inverted, high level or rising edge is valid 1: ETR is inverted, low level or falling edge valid
Bit 14	**ECE** : External clock enable bit (External clock enable) This bit enables external clock mode 2 0: Disable external clock mode 2 1: Enable external clock mode 2. The counter is driven by any valid rising edge on the ETRF signal. Note 1: Setting the ECE bit is related to selecting external clock mode 1 and connecting TRGI to ETRF (SMS = 111 and TS = 111). The same effect. Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gate control mode and trigger mode; however, this When TRGI cannot be connected to ETRF (TS bit cannot be 111). Note 3: When external clock mode 1 and external clock mode 2 are enabled at the same time, the input of the external clock is ETRF.
Bit 13: 12	**ETPS[1 : 0]** : External trigger prescaler (External trigger prescaler) The frequency of the external trigger signal ETRP must be at most 1/4 of the TIMxCLK frequency. When inputting a faster external clock, you can use Use prescaler to reduce the frequency of ETRP. 00: Turn off prescaler 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8
Bit 11: 8	ETF[3:0]: External trigger filter These bits define the frequency of sampling the ETRP signal and the bandwidth of the ETRP digital filtering. In fact, the digital filter is a Event counter, it will generate an output transition after recording N events. 0000: No filter, sampling with f_{DTS} 0001: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$, N = 2 0010: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$, N = 4 0011: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$, N = 8 0100: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N = 6 0101: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, N = 8 0110: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N = 6 0111: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, N = 8 1000: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N = 6 1001: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, N = 8 1010: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N = 5 1011: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N = 6 1100: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, N = 8 1101: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N = 5 1110: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N = 6 1111: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, N = 8
Bit 7	**MSM** : Master/slave mode 0: No effect 1: The event on the trigger input (TRGI) is delayed to allow the current timer (via TRGO) and its slave timing Perfect synchronization between devices. This is very useful when it is required to synchronize several timers to a single external event

Page 213

	TS[2 : 0] : Trigger selection These 3 bits select the trigger input for the synchronization counter. 000: internal trigger 0 (ITR0)

Bit 6: 4	001: Internal trigger 1 (ITR1) 010: Internal trigger 2 (ITR2) 011: Internal trigger 3 (ITR3) 100: TI1 edge detector (TI1F_ED) 101: Filtered timer input 1 (TI1FP1) 110: Filtered timer input 2 (TI2FP2) 111: External trigger input (ETRF) For more details about ITRx, see the table below. Note: These bits can only be changed when they are not used (such as SMS = 000) to avoid false edge detection when changing.
Bit 3	Reserved, always read as 0.
Bit 2: 0	**SMS** : Slave mode selection When the external signal is selected, the effective edge of the trigger signal (TRGI) is related to the selected external input polarity (see input control register). Description of registers and control registers) 000: Disable slave mode-if CEN = 1, the prescaler is directly driven by the internal clock. 001: Encoder mode 1-According to the level of TI1FP1, the counter counts up/down on the edge of TI2FP2. 010: Encoder mode 2-According to the level of TI2FP2, the counter counts up/down on the edge of TI1FP1. 011: Encoder mode 3-According to the level of another input, the counter counts up/down on the edge of TI1FP1 and TI2FP2. 100: Reset mode-the rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register The signal of the memory. 101: Gated mode-When the trigger input (TRGI) is high, the counter clock is turned on. Once the trigger input goes low, then The counter is stopped (but not reset). The start and stop of the counter are controlled. 110: Trigger mode-the counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is affected Controlled. 111: External clock mode 1-The rising edge of the selected trigger input (TRGI) drives the counter. Note: If TI1F_EN is selected as the trigger input (TS = 100), do not use the gated mode. This is because TI1F_ED A pulse is output every time TI1F changes, but the gate control mode is to check the level of the trigger input.

Table **28. TIMx** internal trigger connection

Slave timer	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM3	TIM4	TIM1	TIM2	TIM7
TIM4	TIM5	TIM1	TIM2	TIM3

12.4.4 DMA/ Interrupt Enable Register (**TIMX_DIER**)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved TDE reserved			CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE reserved TIE reserved				CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31: 15	Reserve
Bit 14	**TDE** : Allow to trigger DMA request (Trigger DMA request enable) 0: Disable triggering of DMA request 1: Allow to trigger DMA request
Bit 13	Reserved, always read as 0.
Bit 12	**CC4DE** : Allow capture/compare 4 DMA request (Capture/Compare 4 DMA request enable) 0: Disable capture/compare 4 DMA request 1: Allow capture/compare 4 DMA request
Bit 11	**CC3DE** : Allow capture/compare 3 DMA request (Capture/Compare 3 DMA request enable) 0: Disable capture/compare 3 DMA request 1: Allow capture/compare 3 DMA request
Bit 10	**CC2DE** : Allow capture/compare 2 DMA request (Capture/Compare 2 DMA request enable) 0: Disable capture/compare 2 DMA request 1: Allow capture/compare 2 DMA request
	CC1DE : Allow capture/compare 1 DMA request (Capture/Compare 1 DMA request enable)

Bit 9	0: Disable capture/compare 1 DMA request 1: Allow capture/compare 1 DMA request
Bit 8	**UDE** : Update DMA request enable (Update DMA request enable) 0: Prohibit updated DMA request 1: Allow updated DMA request
Bit 7	Reserved, always read as 0.
Bit 6	**TIE** : Trigger interrupt enable (Trigger interrupt enable) 0: Disable triggering interrupt 1: Enable trigger interrupt
Bit 5	Reserved, always read as 0
Bit 4	**CC4IE** : Allow capture/compare 4 interrupt (Capture/Compare 4 interrupt enable) 0: Disable capture/compare 4 interrupt 1: Allow capture/compare 4 interrupts
Bit 3	**CC3IE** : Allow capture/compare 3 interrupt enable (Capture/Compare 3 interrupt enable) 0: Disable capture/compare 3 interrupt 1: Allow capture/compare 3 interrupt
Bit 2	**CC2IE** : Allow capture/compare 2 interrupt (Capture/Compare 2 interrupt enable) 0: Disable capture/compare 2 interrupt 1: Allow capture/compare 2 interrupt
Bit 1	**CC1IE** : Allow capture/compare 1 interrupt (Capture/Compare 1 interrupt enable) 0: Disable capture/compare 1 interrupt 1: Allow capture/compare 1 interrupt
Bit 0	**UIE** : Update interrupt enable (Update interrupt enable) 0: Disable update interrupt 1: Allow update interruption

12.4.5 Status Register (**TIMx_SR**)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve			CC4 OF	CC3 OF	CC2 OF	CC1 OF	Reserve		TIF reserved		CC4 IF	CC3 IF	CC2 IF	CC1 IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31: 13	Reserve
Bit 12	**CC4OF** : Capture/Compare 4 overcapture flag See CC1OF description.
Bit 11	**CC3OF** : Capture/Compare 3 overcapture flag See CC1OF description.
Bit 10	**CC2OF** : Capture/Compare 2 overcapture flag See CC1OF description.
Bit 9	**CC1OF** : Capture/Compare 1 overcapture flag This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear this bit. 0: no repeated capture 1: When the value of the counter is captured into the TIMx_CCR1 register, the status of CC1IF is already 1
Bit 8: 7	Reserved, always read as 0.
Bit 6	**TIF** : Trigger interrupt flag (Trigger interrupt flag) When a trigger event occurs (when the slave mode controller is in a mode other than gated mode, detect at the TRGI input To the valid edge, or any edge in the gated mode), this bit is set by the hardware. It is cleared by software. 0: No trigger event is generated 1: Trigger interrupt waiting for response
Bit 5	Reserved, always read as 0.
	CC4IF : Capture/Compare 4 interrupt flag

Bit 4	Refer to CC1IF description.
Bit 3	**CC3IF** : Capture/Compare 3 interrupt flag Refer to CC1IF description.
Bit 2	**CC2IF** : Capture/Compare 2 interrupt flag Refer to CC1IF description.

Bit 1	**CC1IF** : Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This bit is set by hardware when the counter value matches the comparison value, except in the center symmetric mode (refer to TIMx_CR1 CMS bit of the register). It is cleared by software. 0: No match occurred 1: The value of TIMx_CNT matches the value of TIMx_CCR1 If channel CC1 is configured as input mode: This bit is set by hardware when a capture event occurs, and it is cleared by software or cleared by reading TIMx_CCR1. 0: No input capture is generated 1: The counter value has been captured (copied) to TIMx_CCR1 (the same edge as the selected polarity is detected on IC1)
Bit 0	**UIF** : Update interrupt flag (Update interrupt flag) This bit is set by hardware when an update event is generated. It is cleared by software. 0: No update event is generated 1: Update event waiting for response. This bit is set by hardware when the register is updated − If the UDIS of the TIMx_CR1 register = 0, an update event is generated when REP_CNT = 0 (repeated down counter When overflow or underflow); − If UDIS = 0 and URS = 0 in the TIMx_CR1 register, when UG = 1 in the TIMx_EGR register, a change will occur. New event (The software reinitializes the counter CNT); − If UDIS = 0 and URS = 0 in the TIMx_CR1 register, when the counter CNT is reinitialized by a trigger event Health update event. (Refer to the description of the synchronization control register)

12.4.6 Event generation register (**TIMx_EGR**)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve									TG	reserved	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bit 31: 7	Reserved, always read as 0.
Bit 6	**TG** : Generate trigger event (Trigger generation) This bit is set by software to generate a trigger event and is automatically cleared by hardware. 0: No action 1: TIF of the TIMx_SR register = 1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated
Bit 5	Reserved, always read as 0.
Bit 4	**CC4G** : Generate capture/compare 4 generation events (Capture/compare 4 generation) Refer to CC1G description.
Bit 3	**CC3G** : Generate capture/compare 3 generation events (Capture/compare 3 generation) Refer to CC1G description.

Bit 2	**CC2G** : Generate capture/compare 2 generation events (Capture/compare 2 generation) Refer to CC1G description.

Bit 1	**CC1G** : Generate capture/compare 1 generation event (Capture/compare 1 generation) This bit is set by software to generate a capture/compare event and is automatically cleared by hardware. 0: No action 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as output: Set CC1IF = 1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated. If channel CC1 is configured as input: The current counter value is captured to the TIMx_CCR1 register, set CC1IF = 1, if the corresponding interrupt and DMA, the corresponding interrupt and DMA are generated. If CC1IF is already 1, set CC1OF = 1.
Bit 0	**UG** : Generate update event (Update generation) This bit is set by software and cleared by hardware automatically. 0: No action 1: Reinitialize the counter and generate an update event. Note that the counter of the prescaler is also cleared to 0 (but the prescaler The frequency division coefficient remains unchanged). If in center symmetric mode or DIR = 0 (counting up), the counter is cleared to 0; if DIR = 1 (count down), the counter takes the value of TIMx_ARR.

12.4.7 Capture **/** Compare Mode Register **1** (**TIMx_CCMR1**)

Offset address: 0x18

Reset value: 0x0000

The channel can be used for input (capture mode) or output (comparison mode), and the direction of the channel is defined by the corresponding CCxS. This register other
The role of the bit is different from that in the output mode. OCxx describes the function of the channel in output mode, and ICxx describes the function of the channel in output mode. can. Therefore, it must be noted that the function of the same bit in output mode and input mode is different.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C2CE	OC2M			OC2PE	OC2FE	CC2S		0C1CE	0C1M			OC1PE	OC1FE	CC1S	
IC2F				IC2PSC				IC1F				IC1PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output comparison mode:

Bit 15	**OC2CE** : Output compare 2 clear enable
Bit 14: 12	**0C2M[2 : 0]** : Output compare 2 mode
Bit 11	**OC2PE** : Output compare 2 preload enable
Bit 10	**OC2FE** : Output compare 2 fast enable

CC2S[1 : 0] : Capture/Compare 2 selection

This bit defines the direction of the channel (input/output), and the selection of input pins:

Bit 9: 8	00: CC2 channel is configured as output; 01: The CC2 channel is configured as an input, and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input, and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input, and IC2 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).
Bit 7	**OC1CE** : Output compare 1 clear enable (Output compare 1 clear enable) 0: OC1REF is not affected by ETRF input; 1: Once the ETRF input high level is detected, clear OC1REF = 0.
Bit 6: 4	**0C1M[2 : 0]** : Output compare 1 mode (Output compare 1 enable) The 3 bits define the action of the output reference signal OC1REF, and OC1REF determines the values of OC1 and OC1N. OC1REF is effective at high level, while the effective level of OC1 and OC1N depends on the CC1P and CC1NP bits. 000: Freeze. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT does not affect OC1REF effect; 001: Set channel 1 as the effective level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, OC1REF is forced to be high. 010: Set channel 1 to an invalid level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, force OC1REF to be low. 011: Flip. When TIMx_CCR1 = TIMx_CNT, the level of OC1REF is inverted. 100: Forced to an invalid level. Force OC1REF to be low. 101: Forced to be a valid level. Force OC1REF to be high. 110: PWM mode 1-when counting up, once TIMx_CNT <TIMx_CCR1, channel 1 is the active level, Otherwise, it is an invalid level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is an invalid level (OC1REF = 0), otherwise it is an effective level (OC1REF = 1). 111: PWM mode 2-When counting up, once TIMx_CNT <TIMx_CCR1, channel 1 becomes an invalid level, Otherwise, it is the effective level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is the effective level, Otherwise, it is an invalid level. Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S = 00 (this pass Channel is configured as output) then this bit cannot be modified. Note 2: In PWM mode 1 or PWM mode 2, only when the comparison result changes or freezes from the output comparison mode The OC1REF level only changes when the mode is switched to PWM mode.
Bit 3	**OC1PE** : Output compare 1 preload enable 0: Disable the preload function of the TIMx_CCR1 register, and can write to the TIMx_CCR1 register at any time, and write a new one The value of will take effect immediately. 1: Turn on the preload function of the TIMx_CCR1 register, read and write operations only operate on the preload register, TIMx_CCR1 The preloaded value of 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 TIMx_BDTR register) and CC1S = 00 (the channel is configured as an output), then this bit cannot modified. Note 2: Only in single pulse mode (OPM = 1 in the TIMx_CR1 register), you can preload the register without confirming In this case, use the PWM mode, otherwise its action is uncertain.

Bit 2	**OC1FE** : Output compare 1 fast enable This bit is used to speed up the response of the CC output to the trigger input event. 0: According to the value of the counter and CCR1, CC1 operates normally, even if the trigger is turned on. When the trigger input has At a valid edge, the minimum delay for activating the CC1 output is 5 clock cycles. 1: The effective edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to compare power It has nothing to do with the comparison result. The delay between the valid edge of the sampling flip-flop and the output of CC1 is shortened to 3 clock cycles. OCFE only works when the channel is configured in PWM1 or PWM2 mode.
Bit 1: 0	**CC1S[1 : 0]** : Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC1 channel is configured as output; 01: The CC1 channel is configured as an input, and IC1 is mapped on TI1; 10: The CC1 channel is configured as an input, and IC1 is mapped on TI2; 11: The CC1 channel is configured as an input, and IC1 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

Input capture mode:

Bit 15: 12	**IC2F[3 : 0]** : Input capture 2 filter
Bit 11: 10	**IC2PSC[1 : 0]** : input/capture 2 prescaler (input capture 2 prescaler)
Bit 9: 8	**CC2S[1 : 0]** : Capture/compare 2 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC2 channel is configured as output; 01: The CC2 channel is configured as an input, and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input, and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input, and IC2 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).

Bit 7: 4	**IC1F[3 : 0]** : 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 group After it records N events, it will produce an output transition: 0000: No filter, sampling with f_{DTS} 1000: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 6 0001: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 2 1001: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 8 0010: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 4 1010: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 5 0011: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 8 1011: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 6 0100: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 6 1100: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 8 0101: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 8 1101: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 5 0110: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 6 1110: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 6 0111: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 8 1111: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 8
Bit 3: 2	**IC1PSC[1 : 0]** : Input/capture 1 prescaler (Input capture 1 prescaler) These 2 bits define the prescaler coefficient of the CC1 input (IC1). Once CC1E = 0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, every edge detected on the capture input port triggers a capture; 01: Trigger a capture every 2 events; 10: Trigger a capture every 4 events; 11: Trigger a capture every 8 events.
	CC1S[1 : 0] : Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins:

Bit 1: 0	00: CC1 channel is configured as output; 01: The CC1 channel is configured as an input, and IC1 is mapped on TI1; 10: The CC1 channel is configured as an input, and IC1 is mapped on TI2; 11: The CC1 channel is configured as an input, and IC1 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

12.4.8 Capture / Compare Mode Register 2 (TIMx_CCMR2)

Offset address: 0x1C

Reset value: 0x0000

See the description of the CCMR1 register above

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C4CE	OC4M			OC4PE	OC4FE	CC4S		0C3CE	0C3M			OC3PE	OC3FE	CC3S	
IC4F				IC4PSC				IC3F				IC3PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15	**OC4CE** : Output compare 4 clear enable (Output compare 4 clear enable)
Bit 14: 12	**0C4M[2 : 0]** : Output compare 4 mode (Output compare 4 mode)
Bit 11	**OC4PE** : Output compare 4 preload enable
Bit 10	**OC4FE** : Output compare 4 fast enable
Bit 9: 8	**CC4S[1 : 0]** : Capture/Compare 4 selection The 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC4 channel is configured as output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: The CC4 channel is configured as an input, and IC4 is mapped on TI3; 11: The CC4 channel is configured as an input, and IC4 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC4S is only writable when the channel is closed (CC4E = 0 in the TIMx_CCER register).
Bit 7	**OC3CE** : Output compare 3 clear enable (Output compare 3 clear enable)
Bit 6: 4	**OC3M[2 : 0]** : Output compare 3 mode (Output compare 3 mode)
Bit 3	**OC3PE** : Output compare 3 preload enable
Bit 2	**OC3FE** : Output compare 3 fast enable
Bit 1: 0	**CC3S[1 : 0]** : Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: The CC3 channel is configured as an input, and IC3 is mapped on TI4; 11: The CC3 channel is configured as an input, and IC3 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC3S is only writable when the channel is closed (CC3E = 0 in the TIMx_CCER register).

Input comparison mode

Bit 15: 12	**IC4F[3 : 0]** : Input capture 4 filter
Bit 11: 10	**IC4PSC[1 : 0]** : input/capture 4 prescaler (input capture 4 prescaler)
Bit 9: 8	**CC4S[1 : 0]** : Capture/compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC4 channel is configured as output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: The CC4 channel is configured as an input, and IC4 is mapped on TI3; 11: The CC4 channel is configured as an input, and IC4 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC4S is only writable when the channel is closed (CC4E = 0 in the TIMx_CCER register).
Bit 7: 4	**IC3F[3 : 0]** : Input capture 3 filter
Bit 3: 2	**IC3PSC[1 : 0]** : Input/capture 3 prescaler (Input capture 3 prescaler)
Bit 1: 0	**CC3S[1 : 0]** : Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: The CC3 channel is configured as an input, and IC3 is mapped on TI4; 11: The CC3 channel is configured as an input, and IC3 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by the TS bit in the TIMx_SMCR register). Note: CC3S is only writable when the channel is closed (CC3E = 0 in the TIMx_CCER register).

12.4.9 Capture **/** Compare Enable Register (**TIMx_CCER**)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve		CC4P	CC4E	Reserve		CC3P	CC3E	Reserve		CC2P	CC1E	Reserve		CC1P	CC1E
		w	w			w	w			w	w			w	w

Bit 15: 14	Reserved, always read as 0.
Bit 13	**CC4P** : Input/Capture 4 output polarity (Capture/Compare 4 output polarity) Refer to the description of CC1P.
Bit 12	**CC4E** : Input/Capture 4 output enable (Capture/Compare 4 output enable) Refer to the description of CC1E.
Bit 11: 10	Reserved, always read as 0.
Bit 9	**CC3P** : Input/Capture 3 output polarity (Capture/Compare 3 output polarity) Refer to the description of CC1P.
Bit 8	**CC3E** : Input/Capture 3 output enable (Capture/Compare 3 output enable) Refer to the description of CC1E.
Bit 7: 6	Reserved, always read as 0.

Bit 5	**CC2P** : Input/Capture 2 output polarity (Capture/Compare 2 output polarity) Refer to the description of CC1P.
Bit 4	**CC2E** : Capture/Compare 2 output enable Refer to the description of CC1E.
Bit 3: 2	Reserved, always read as 0.
Bit 1	**CC1P** : Input/Capture 1 output polarity (Capture/Compare 1 output polarity) CC1 channel is configured as output: 0: OC1 is valid at high level; 1: OC1 is active at low level. CC1 channel is configured as input: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: No inversion: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 does not invert. 1: Inversion: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted. Note: Once the LOCK level (LCCK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.
Bit 0	**CC1E** : Input/Compare 1 output enable (Capture/Compare 1 output enable) CC1 channel is configured as output: 0: Off-OC1 prohibits output, so the output level of OC1 depends on MOE, OSSI, OSSR, OIS1, OIS1N And the value of the CC1NE bit. 1: On-OC1 signal is output to the corresponding output pin, and its output level depends on MOE, OSSI, OSSR, The value of the OIS1, OIS1N, and CC1NE bits. CC1 channel is configured as input: This bit determines whether the value of the counter can be captured into the TIMx_CCR1 register. 0: Capture prohibited; 1: Capture enable.

Table **29.** 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 pin is connected to a standard *OCx* channel external *I / O* pin states, depending *OCx* channel status and *GPIO* and *AFIO* Send Memory.

12.4.10 Counter (**TIMx_CNT**)

Offset address: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CNT[31 : 0]** : Counter value

12.4.11 Prescaler (**TIMx_PSC**)

Offset address: 0x28

Reset value: 0x0000

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

Bit 15:0	**PSC[15 : 0]** : Prescaler value The clock frequency of the counter (CK_CNT) is equal to f $_{CK\_PSC}$ / (PSC[15:0] + 1). The PSC contains the value loaded into the current prescaler register each time an update event occurs. Update event including count The device is cleared to '0' by the UG bit of TIM_EGR or is cleared to '0' by the slave controller working in reset mode.

12.4.12 Auto-load register (**TIMx_ARR**)

Offset address: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**ARR[31 : 0]** : Auto reload value ARR contains the value to be loaded into the actual auto-reload register. For details, refer to section 13.3.1: Updates and actions related to ARR. When the value of auto reload is empty, the counter does not work.

12.4.13 Capture **/** Compare Register **1** (**TIMx_CCR1**)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR1[31 : 0]** : Capture/Compare 1 value If the CC1 channel is configured as output: CCR1 contains the value loaded into the current capture/compare 1 register (preload value). If the preload function is not selected in the TIMx_CCMR1 register (OC1PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 1 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC1 port Signal. If the CC1 channel is configured as input:

CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).

12.4.14 Capture / Compare Register **2** (**TIMx_CCR2**)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR2[31 : 0]** : Capture/Compare 2 value If the CC2 channel is configured as output: CCR2 contains the value loaded into the current capture/compare 2 register (preload value). If the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value is transferred to the current capture/compare 2 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC2 port Signal. If the CC2 channel is configured as input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).

12.4.15 Capture / Compare Register **3** (**TIMx_CCR3**)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR3[31 : 0]** : Capture/Compare 3 value If the CC3 channel is configured as output: CCR3 contains the value loaded into the current capture/compare 3 register (preload value). If the preload feature is not selected in the TIMx_CCMR3 register (OC3PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 3 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC3 port Signal. If the CC3 channel is configured as input: CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).

12.4.16 Capture / Compare Register **4** (**TIMx_CCR4**)

Offset address: 0x40

Reset value: 0x0000

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

CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR4[31 : 0]** : Capture/Compare 4 value If the CC4 channel is configured as output: CCR4 contains the value loaded into the current capture/compare 4 register (preload value). If the preload feature is not selected in the TIMx_CCMR4 register (OC4PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value is transferred to the current capture/compare 4 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC4 port Signal. If the CC4 channel is configured as input: CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).

12.4.17 **DMA** control register (**TIMx_DCR**)

Offset address: 0x48

Reset value: 0x0000

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

Bit 15: 13	Reserved, always read as 0.
Bit 12: 8	**DBL[4 : 0]** : DMA continuous transfer length (DMA burst length) These bits define the transfer length of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register, The timer performs a continuous transmission), that is, the number of transmissions is defined. The transmission can be half-word (double-byte) or word Festival: 00000: 1 transmission 00001: 2 transmissions 00010: 3 transmissions .... 10001: 18 transmissions
Bit 7: 5	Reserved, always read as 0.
Bit 4: 0	**DBA[4 : 0]** : DMA base address These bits define the base address of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register), DBA is defined as the offset from the address where the TIMx_CR1 register is located: 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR ......

12.4.18 DMA address in continuous mode (TIMx_DMAR)

Offset address: 0x48

Reset value: 0x0000

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

Bit 15:0	**DMAB[15 : 0]** : DMA continuous transfer register (DMA register for burst accesses) Reading or writing to the TIMx_DMAR register will result in the access operation to the register at the following address: TIMx_CR1 address + DBA + DMA index, where: 'TIMx_CR1 address' is the address where the control register 1 (TIMx_CR1) is located; 'DBA' is the base address defined in the TIMx_DCR register; 'DMA index' is the offset automatically controlled by DMA, which depends on the DBL defined in the TIMx_DCR register.

13. General timer (TIM5/6/7)

13.1 Introduction to TIMx

The general-purpose timer is a 32-bit auto-loading counter driven by a programmable prescaler. It is suitable for many occasions, including testing Measure the pulse length of the input signal (input capture) or generate the output waveform (output comparison and PWM).

Using timer prescaler and RCC clock controller prescaler, the pulse length and waveform period can be between several microseconds to several milliseconds Adjustment.

TIM5/TIM6/TIM7 timers are completely independent and do not share any resources with each other. They can operate simultaneously.

13.2 Main functions of TIM5/TIM6/TIM7

The general TIM5/TIM6/TIM7 timer functions include:

- 32-bit up autoload counter
- 16-bit programmable (can be modified in real time) prescaler, the frequency division factor of the counter clock frequency is any number between 1 and 65536 value
- 2 independent channels
  - Input capture
  - Output comparison
  - PWM generation (edge-aligned mode)
  - Single pulse mode output
- Use external signal to control timer and timer interconnection synchronization circuit
- An interrupt is generated when the following events occur:
  - Update: counter overflows, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output comparison

Figure **110.** Block diagram of a general-purpose timer

13.3 TIM5/TIM6/TIM7 function description

13.3.1 Time base unit

The main part of the programmable general-purpose timer is a 32-bit counter and its associated auto-loading register. This counter can go up
Count, count down, or count up and down in both directions. This counter clock is divided by the prescaler.

The counter, auto-load 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:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto reload register (TIMx_ARR)

The auto-load register is pre-loaded, and writing or reading the auto-reload register will access the pre-load register. According to the post in TIMX_CR1
The auto-loading pre-loading enable bit (ARPE) setting in the register, the content of the pre-loading register is immediately or in every update event UEV
When transferred to the shadow register. When the counter reaches the overflow condition (underflow condition when counting down) and when the UDIS in the TIMX_CR1 register
When the bit is equal to 0, an update event is generated. Update events can also be generated by software. The generation of update events under each configuration will be described in detail

The counter is driven by the clock output CK_CNT of the prescaler, only when the counter enable in the counter TIMX_CR1 register is set
CK_CNT is valid only when bit (CEN). (For details of counter enable, please refer to the description of the slave mode of the controller).

Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a (sent in TIMx_PSC
32-bit counter controlled by 16-bit register. Because this control register has a buffer, it can be changed during operation.
The parameters of the new prescaler will be adopted when the next update event arrives.

The following two figures show examples of changing counter parameters when the prescaler is running.

Figure **111.** When the prescaler parameter is changed from **1** to **2** , the timing diagram of the counter

Figure **112.** When the prescaler parameter changes from **1** to **4** , the timing diagram of the counter

13.3.2 Counting mode

Up counting mode

In the up-counting mode, the counter counts from 0 to the auto-load value (the content of the TIMx_ARR counter), and then restarts from 0 Count and generate a counter overflow event.

An update event can be generated every time the counter overflows. Set the UG bit in the TIMx_EGR register (by software or by using slave Mode controller) can also generate an update event.

Setting the UDIS bit in the TIMx_CR1 register can disable the update event; this can avoid writing new data to the preload register. The shadow register is updated when the value is set. Until the UDIS bit is cleared to 0, no update event will be generated. But when an update event should be generated, the counter It will still be cleared to 0, and the count of the prescaler will be set to 0 (but the value of the prescaler will not change). In addition, if the TIMx_CR1 register is set URS bit (select update request) in the device, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (that is, no Generate an interrupt). This is to avoid generating update and capture interrupts at the same time when clearing the counter in capture mode.

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

- The buffer of the prescaler is put into the value of the preload register (the content of the TIMx_PSC register)
- The autoload shadow register is reset to the value of the preload register (TIMx_ARR)

The following figure gives some examples, when TIMx_ARR = 0x36, the action of the counter at different clock frequencies:

Figure **113.** Counter timing diagram, the internal clock division factor is **1**

Figure **114.** Counter timing diagram, the internal clock division factor is **2**

Page 233

Figure **115.** Counter timing diagram, the internal clock division factor is **4**

Figure **116.** Counter timing diagram, the internal clock division factor is **N**

Figure **117.** Counter timing diagram, update event when **ARPE = 0** (**TIMx_ARR is** not preloaded)

Page 234

Figure **118.** Counter timing diagram, update event when **ARPE = 1** (**TIMx_ARR is preloaded**)

13.3.3 Clock selection

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

- Internal clock (CK_INT)
- External clock mode 1: External input pin (TIx)
- Internal trigger input (ITRx): Use a timer as the prescaler of another timer, for example, you can configure a timer

  Timer1 is used as a prescaler for another timer Timer2.

Internal clock source (**CK_INT**)

If the slave mode controller is disabled (SMS = 000), the CEN, DIR (TIMx_CR1 register) and UG bit (TIMx_EGR Register) is the de facto control bit and can only be modified by software (the UG bit is still automatically cleared). Once the CEN bit is written as 1, pre-divide The clock of the frequency converter is provided by the internal clock CK_INT.

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

Figure **119.** The control circuit in normal mode, the internal clock division factor is **1**

External clock source mode **1**

When SMS = 111 in the TIMx_SMCR register, this mode is selected. The counter can be at each rising edge or down of the selected input Falling edge count.

Figure **120. TI2** external clock connection example

For example, to configure the up counter to count on the rising edge of the T12 input, use the following steps:

7. Configure TIMx_CCMR1 register CC2S = 01, configure channel 2 to detect the rising edge of TI2 input
8. Configure IC2F[3:0] in the TIMx_CCMR1 register, select the input filter bandwidth (if no filter is required, keep IC2F = 0000)

Note: The capture prescaler is not used as a trigger, so there is no need to configure it

9. Configure CC2P of the TIMx_CCER register = 0, select the rising edge polarity
10. Configure SMS = 111 in the TIMx_SMCR register, select timer external clock mode 1
11. Configure TS = 110 in the TIMx_SMCR register and select TI2 as the trigger input source
12. Set CEN = 1 in the TIMx_CR1 register to start the counter

When the 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 clock of the counter depends on the resynchronization circuit at the input of TI2.

Figure **121.** Control circuit in external clock mode **1**

13.3.4 Capture **/** Compare Channel

Each capture/compare channel is surrounded by a capture/compare register (including shadow registers), including the captured input part (data Word filtering, multiplexing and prescaler), and output section (comparator and output control).

The following pictures are an overview of the capture/compare channel. The input part samples the corresponding TIx input signal and generates a filtered signal No. TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx), which can be used as the input of the slave mode controller Trigger or as capture control. This signal enters the capture register (ICxPS) by prescaler.

Figure **122.** Capture **/** compare channel (eg: input part of channel **1**)

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

Figure **123.** Main circuit of capture **/** compare channel **1**

Figure **124.** The output section of the capture **/** compare channel (channel **1**)

The capture/compare module consists of a preload register and a shadow register. The read and write process only manipulates the preload register. In capture mode Under the formula, the capture occurs on the shadow register and then copied to the preload register.

In the 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 the input capture mode, when the corresponding edge on the ICx signal is detected, the current value of the counter is latched into the capture/compare register (TIMx_CCRx). When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, if the
Interrupt operation, an interrupt operation will be generated. If the CCxIF flag is already high when the capture event occurs, then repeat the capture flag CCxOF (TIMx_SR register) is set. Write CCxIF = 0 to clear CCxIF, or read the captured number stored in the TIMx_CCRx register
According to data, CCxIF can also be cleared. Write CCxOF = 0 to clear CCxOF.

The following example shows how to capture the counter value into the TIMx_CCR1 register at the rising edge of TI1 input. The steps are as follows:

- Select a valid input terminal: TIMx_CCR1 must be connected to the TI1 input, so write CC1S in the TIMx_CCR1 register = 01, once CC1S is not 00, the channel is configured as an input, and the TM1_CCR1 register becomes read-only.
- 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 ICxF bit in the TIMx_CCMRx register). Assuming that the input signal jitters within a maximum of 5 clock cycles, we must
The bandwidth of the configuration filter is longer than 5 clock cycles. So we can sample 8 times continuously (at fDTS frequency) to confirm that The last real edge transition of TI1, that is, write IC1F = 0011 in the TIMx_CCMR1 register.
- Select the valid conversion edge of the TI1 channel, and write CC1P = 0 (rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler Disabled (write IC1PS = 00 in the TIMx_CCMR1 register).
- Set CC1E = 1 in the TIMx_CCER register to allow the value of the counter to be captured into the capture register.
- If necessary, enable related interrupt requests by setting the CC1IE bit in the TIMx_DIER register.

Occurs when an input is captured:

- When a valid level transition occurs, the value of the counter is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures have occurred, CC1IF has not been cleared.
- CC1OF is also set.
- If the CC1IE bit is set, an interrupt will be generated.

Page 238

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 loss in reading the capture overflow flag Capture overflow information that may occur after and before the data is read.

Note: By setting the corresponding *CCxG* bit in the *TIMx_EGR* register, an input capture interrupt request can be generated by software.

13.3.6 PWM input mode

This mode is a special case of the input capture mode, except for the following differences, the operation is the same as the input capture mode:

- The two ICx signals are mapped to the same TIx input.
- The two ICx signals are edge-valid, but the polarity is opposite.
- One of the TIxFP signals is used as a trigger input signal, and the slave mode controller is configured to reset mode.

For example, you need to measure the length (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of the PWM signal input to TI1. Register), the specific steps are as follows (depending on the frequency of CK_INT and the value of the prescaler)

- Select the valid input of TIMx_CCR1: set CC1S = 01 in the TIMx_CCMR1 register (select TI1).
- Select the valid polarity of TI1FP1 (used to capture data to TIMx_CCR1 and clear the counter): set CC1P = 0 (rising Valid along).
- Select the valid input of TIMx_CCR2: set CC2S in the TIMx_CCMR1 register = 10 (select TI1).
- Select the valid polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P = 1 (falling edge valid).
- Select a valid trigger input signal: set TS = 101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller to reset mode: set SMS = 100 in TIMx_SMCR.
- Enable capture: set CC1E = 1 and CC2E = 1 in the TIMx_CCER register.

Figure **125.** Timing of **PWM** input mode

Because only TI1FP1 and TI2FP2 are connected to the slave mode controller. So PWM input mode can only use TIMx_CH1 / TIMx_CH2 signal.

13.3.7 Forced output mode

In the output mode (CCxS = 00 in the TIMx_CCMRx register), the output compare signal (OCxREF and corresponding OCx) can be It can be forced to the valid or invalid state directly by the software, without relying on the comparison result between the output compare register and the counter.

Page 239

Set the corresponding OCxM = 101 in the TIMx_CCMRx register to force the output comparison signal (OCxREF/OCx) to be valid state. In this way, OCxREF is forced to be high (OCxREF is always active high), and at the same time, OCx gets the opposite polarity of CCxP value.

For example: CCxP = 0 (OCx is active at high level), then OCx is forced to be high.

Set OCxM = 100 in the TIMx_CCMRx register to force the OCxREF signal to be low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress, and the corresponding flags will also be modified. because

This will still generate the corresponding interrupt. This will be introduced in the output comparison mode section below.

13.3.8 Output Compare Mode

This function is used to control an output waveform or indicate when a given time has elapsed.

When the contents of the counter and the capture/compare register are the same, the output compare function does the following:

- The output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (the TIMx_CCER register in the
  The value defined by the CCxP bit) is output to the corresponding pin. During a comparison match, the output pin can maintain its level (OCxM = 000), set to effective level (OCxM = 001), set to no effective level (OCxM = 010), or reverse
  (OCxM = 011).
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- If the corresponding interrupt mask (CCXIE bit in the TIMx_DIER register) is set, an interrupt is generated.

The OCxPE bit in TIMx_CCMRx selects whether the TIMx_CCRx register needs to use the preload register.

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

The accuracy of synchronization can reach one counting cycle of the counter. The output compare mode (in the single pulse mode) can also be used to output a single pulse.

Configuration steps of output comparison mode:

6. Select the counter clock (internal, external, prescaler)
7. Write the corresponding data into the TIMx_ARR and TIMx_CCRx registers
8. Select the output mode, for example: OCxM = '011', OCxPE = '0', CCxP = '0' and CCxE = '1' must be set, when counting
   When the CNT and CCRx match, the output pin of OCx is flipped, CCRx is preloaded unused, and the OCx output is turned on and the high level is active.
9. Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_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 TIMx_CCRx shadow register can only be updated when the next update event occurs). The following figure shows an example son.

Figure **126.** Output compare mode, flip **OC1**

13.3.9 PWM mode

The pulse width modulation mode can generate a frequency determined by the TIMx_ARR register and the duty cycle determined by the TIMx_CCRx register. Signal.

Write '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bit in the TIMx_CCMRx register, which can be independent Set each OCx output channel to generate a PWM. The OCxPE bit in the TIMx_CCMRx register must be set to enable the corresponding preload Register, and finally set the ARPE bit of the TIMx_CR1 register to enable automatic reloading of the preload register (in the upward counting or center In symmetric mode).

Because only when an update event occurs, the preload register can be transferred to the shadow register, so the counter starts counting Previously, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The polarity of OCx can be set by software in the CCxP bit in the TIMx_CCER register, and it can be set to active high or low The level is valid. The CCxE bit in the TIMx_CCER register controls the OCx output enable. See the description of the TIMx_CCERx register for details.

In the PWM mode (mode 1 or mode 2), TIMx_CNT and TIM1_CCRx are always being compared (according to the counter count Number direction) to determine whether it meets TIM1_CCRx ≤ TIM1_CNT or TIM1_CNT ≤ TIM1_CCRx. However in order to The function of OCREF_CLR (before the next PWM cycle, an external event on the ETR signal can clear OCxREF) is the same, The OCxREF signal can only be generated under the following conditions:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bit in the TIMx_CCMRx register) is switched from'freeze' (no comparison, OCxM = '000') To a certain PWM mode (OCxM = '110' or '111').

In this way, the PWM output can be forced by software during operation. According to the state of the CMS bit in the TIMx_CR1 register, the timer can Generate edge-aligned PWM signals or center-aligned PWM signals.

PWM edge alignment mode

The following is an example of PWM mode 1. When TIMx_CNT <TIMx_CCRx, the PWM signal reference OCxREF is high, no It is low. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains at '1'. If it is better than If the comparison value is 0, OCxREF remains at '0'. The following figure shows an example of an edge-aligned PWM waveform when TIMx_ARR = 8.

Figure **127.** Edge-aligned **PWM** waveform (**ARR = 8**)

13.3.10 Single pulse mode

Single pulse mode (OPM) is a special case of many of the aforementioned modes. This mode allows the counter to respond to a stimulus and in a program After the controllable delay, a pulse with programmable pulse width is generated.

The counter can be started by the slave mode controller to generate waveforms in output comparison mode or PWM mode. Set TIMx_CR1 The OPM bit in the register will select the single pulse mode, so that the counter can automatically stop when the next update event UEV is generated.

Only when the comparison value is different from the initial value of the counter can a pulse be generated. Before starting (when the timer is waiting to be triggered), the Must be configured as follows:

- CNT <CCRx ≤ ARR (in particular, 0<CCRx)

Figure **128.** Example of single pulse mode

For example, you need to detect a rising edge on the TI2 input pin, and after a delay of t_{DELAY}, a length of t_{PULSE} positive pulse.

Assuming TI2FP2 as trigger 1:

- Set CC2S = 01 in the TIMx_CCMR1 register to map TI2FP2 to TI2.
- Set CC2P = 0 in the TIMx_CCER register to enable TI2FP2 to detect the rising edge.
- Set TS = 110 in the TIMx_SMCR register, and TI2FP2 is used as the trigger (TRGI) of the slave mode controller.
- Set SMS in the TIMx_SMCR register = 110 (trigger mode), TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written in the compare register (the clock frequency and counter prescaler should be considered).

- t_{DELAY} is defined by the value written to the TIMx_CCR1 register.
- t_{PULSE} is defined by the difference between the autoload value and the comparison value (TIMx_ARR-TIMx_CCR1).
- Suppose that when a comparison match occurs, a waveform from 0 to 1 is to be generated. When the counter reaches the preload value, a waveform from 1 to 0 is generated. Waveform; first set the OC1M of the TIMx_CCMR1 register = 111, enter PWM mode 2; selectively use Register can be preloaded: set OC1PE in TIMx_CCMR1 = 1 and ARPE in TIMx_CR1 register; then Fill in the comparison value in the TIMx_CCR1 register, fill in the auto-load value in the TIMx_ARR register, and modify the UG bit to generate An update event, and then wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Because only one pulse is required, OPM = 1 in the TIMx_CR1 register must be set, and in the next update event (when the counter is from Stop counting when the auto-load value rolls over to 0).

Special case: **OCx** fast enable:

In the single pulse mode, the edge detection logic at the TIx input pin sets the CEN bit to start the counter. Then the counter and comparison value The comparison operation produces a conversion of the output. But these operations require a certain clock cycle, so it limits the minimum delay t_{DELAY} that can be obtained .

If you want to output the waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register; at this time, force OCxREF (and OCx) is forced to respond to the stimulus instead of relying on the result of the comparison, and the output waveform is the same as the waveform when the comparison matches. OCxFE is on It works when set to PWM1 and PWM2 mode.

Page 243

13.3.11 Synchronization of timer external trigger

The TIMx timer can be synchronized with an external trigger in multiple modes: reset mode, gating mode and trigger mode.

Slave mode: reset mode

When a trigger input event occurs, the counter and its prescaler can be initialized again; at the same time, if TIMx_CR1 is registered
The URS bit of the device is low, and an update event UEV is also generated; then all preload registers (TIMx_ARR, TIMx_CCRx)
Have been updated.

- In the following example, the rising edge of the TI1 input causes the up counter to be cleared:
- 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 the trigger operation, so no configuration is required. CC1S bit only selects input capture Get the source, that is, CC1S = 01 in the TIMx_CCMR1 register. Set CC1P = 0 in the TIMx_CCER register to determine the polarity (Only the rising edge is detected).
- Set SMS = 100 in the TIMx_SMCR register to configure the timer in reset mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.
- Set CEN = 1 in the TIMx_CR1 register to start the counter.

The counter starts to count according to the internal clock, and then runs normally until TI1 has a rising edge; at this time, the counter is cleared and then reset from 0
Restart counting. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, according to the TIE in the TIMx_DIER register
(Interrupt enable) bit is set to generate an interrupt request.

The following figure shows the action when the auto reload register TIMx_ARR = 0x36. Between the rising edge of TI1 and the actual reset of the counter
The delay depends on the resynchronization circuit at the input of TI1.

Figure **129.** Control circuit in reset mode

Slave mode: gated mode

The enable of the counter depends on the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- 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 the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, Set CC1S = 01 in the TIMx_CCMR1 register. Set CC1P = 1 in the TIMx_CCER register to determine the polarity (check only Measure the low level).
- Set SMS = 101 in the TIMx_SMCR register to configure the timer as gated mode; set TS = 101 in the TIMx_SMCR register, Select TI1 as the input source.

Page 244

- Set CEN = 1 in the TIMx_CR1 register to start the counter. In gating mode, if CEN = 0, the counter cannot be started 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. Set when the counter starts or stops
TIF marking in TIMx_SR.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the input of TI1.

Figure **130.** Control circuit in gating mode

Slave mode: trigger mode

The enabling of the counter depends on the event on the selected input.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed, keep IC2F = 0000). The capture prescaler is not used in the trigger operation, and no configuration is required. The CC2S bit is only used to select the input capture source, set CC2S = 01 in the TIMx_CCMR1 register. Set CC1P = 1 in the TIMx_CCER register to determine the polarity (only detect low Level).
- Set SMS = 110 in the TIMx_SMCR register to configure the timer as trigger mode; set TS = 110 in the TIMx_SMCR register, Select TI2 as the input source.

When TI2 has a rising edge, the counter starts to count under the internal clock drive, and the TIF flag is set 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 input of TI2.

Figure **131.** Control circuit in flip-flop mode

13.3.12 Timer synchronization

TIM timers are linked together internally to achieve timer synchronization or cascading. For details, see ----------.

13.3.13 Debug mode

When the microcontroller enters the debug mode (CPU core is stopped), according to the setting of DBG_TIMx_STOP in the DBG module, the TIMx counts The counter will either continue normal operation or stop. For details, see the chapter Debugging Module.

13.4 **TIMx** register description

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) mode.

13.4.1 Control Register **1** (**TIMx_CR1**)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve						CKD		ARPE	Reserve			OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 10 Reserve

CKD[1 : **0]** : Clock division factor (Clock division)

Bit 9: 8	These 2 bits are defined in the timer clock (CK_INT) frequency, dead time and the dead time generator and digital filter (ETR, TIx) The frequency division ratio between the sampling clocks used. 00: t $_{DTS}$ = t $_{CK\_INT}$ 01: t $_{DTS}$ = 2 x t $_{CK\_INT}$ 10: t $_{DTS}$ = 4 x t $_{CK\_INT}$ 11: Reserved, do not use this configuration
Bit 7	**ARPE** : Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is loaded into the buffer
Bit 6: 4	Reserve
Bit 3	**OPM** : One pulse mode 0: When an update event occurs, the counter does not stop 1: When the next update event occurs (clear the CEN bit), the counter stops
Bit 2	**URS** : Update request source The software selects the source of the UEV event through this bit 0: If the update interrupt is allowed, one of the following events will generate an update interrupt: − Counter overflow/underflow − Set the UG bit − Updates generated from the mode controller 1: If the update interrupt is allowed, only the counter overflow/underflow will generate an update interrupt

Page 246

Bit 1	**UDIS** : Update disable The software allows/disables the generation of UEV events through this bit 0: UEV is allowed. Update (UEV) events are generated by any of the following events: − Counter overflow/underflow − Set the UG bit − The updated buffered registers generated from the mode controller are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) maintain their values. If set If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.
Bit 0	**CEN** : Counter enable 0: disable the counter 1: Enable the counter Note: After the software sets the CEN bit, the external clock, gating mode and encoder mode can only work. Trigger mode can be automatic The CEN bit is set by hardware. In single pulse mode, when an update event occurs, CEN is automatically cleared.

13.4.2 Control Register **2** (**TIMx_CR2**)

Offset address: 0x04

Reset value: 0x0000

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

Bit 31: 7	Reserved, the reset value must be maintained.
	MMS[1 : 0] : Master mode selection These two bits are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. The possible combinations are as follows: 000: Reset-The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode If the controller is in reset mode) to generate a reset, the signal on TRGO will have a delay relative to the actual reset. 001: Enable-the counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes need to be at the same time Start multiple timers or control to enable slave timers within a period of time. The counter enable signal is controlled by the CEN control bit and gate

Bit 6: 4	The logic or generation of the trigger input signal in the mode. When the counter enable signal is controlled by the trigger input, there will be a A delay, unless the master/slave mode is selected (see the description of the MSM bit in the TIMx_SMCR register). 010: Update-The update event is selected as the trigger input (TRGO). For example, the clock of a master timer can be used as a A prescaler for the slave timer. 011: Comparison pulse-once a capture occurs or a comparison is successful, when the CC1IF flag is to be set (even if it has been Is high), the trigger output sends a positive pulse (TRGO). 100: Compare-OC1REF signal is used as trigger output (TRGO). 101: Compare-OC2REF signal is used as trigger output (TRGO). 110: reserved. 111: reserved.
Bit 3: 0	Reserved, the reset value must be maintained.

13.4.3 Slave mode control register (**TIMx_SMCR**)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								MSM		TS[2:0]		Reserve		SMS[2:0]	
								rw	rw	rw	rw		rw	rw	rw

Bit 31: 8	Reserved, the reset value must be maintained.
Bit 7	**MSM** : Master/slave mode 0: No effect 1: The event on the trigger input (TRGI) is delayed to allow the current timer (via TRGO) and its slave timing Perfect synchronization between devices. This is very useful when it is required to synchronize several timers to a single external event
Bit 6: 4	**TS[2 : 0]** : Trigger selection These 3 bits select the trigger input for the synchronization counter. 000: internal trigger 0 (ITR0) 001: Internal trigger 1 (ITR1) 010: Internal trigger 2 (ITR2) 011: Internal trigger 3 (ITR3) 100: TI1 edge detector (TI1F_ED) 101: Filtered timer input 1 (TI1FP1) 110: Filtered timer input 2 (TI2FP2) 111: reserved For more details about ITRx, see the table below. Note: These bits can only be changed when they are not used (such as SMS = 000) to avoid false edge detection when changing.
Bit 3	Reserved, the reset value must be maintained.
Bit 2: 0	**SMS** : Slave mode selection When the external signal is selected, the effective edge of the trigger signal (TRGI) is related to the selected external input polarity (see input control register). Description of registers and control registers) 000: Slave mode disabled-If CEN = 1, the prescaler is directly driven by the internal clock. 001: reserved. 010: reserved 011: reserved 100: Reset mode-the rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register The signal of the memory. 101: Gated mode-When the trigger input (TRGI) is high, the counter clock is turned on. Once the trigger input goes low, then The counter is stopped (but not reset). The start and stop of the counter are controlled. 110: Trigger mode-the counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is affected Controlled. 111: External clock mode 1-The rising edge of the selected trigger input (TRGI) drives the counter. Note: If TI1F_EN is selected as the trigger input (TS = 100), do not use the gated mode. This is because TI1F_ED A pulse is output every time TI1F changes, but the gate control mode is to check the level of the trigger input.

Table **30. TIMx** internal trigger connection

Slave timer	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM3	TIM4	TIM1	TIM2	TIM7
TIM4	TIM5	TIM1	TIM2	TIM3
TIM5	TIM3	TIM4	TIM1	TIM6
TIM6	TIM3	TIM4	TIM7	TIM2
TIM7	TIM3	TIM4	TIM1	TIM5

13.4.4 DMA/ interrupt enable register (**TIMX_DIER**)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									TIE				CC2 IE	CC1 IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31: 7	Reserved, the reset value must be maintained.
Bit 6	**TIE** : Trigger interrupt enable (Trigger interrupt enable) 0: Disable triggering interrupt 1: Enable trigger interrupt
Bit 5: 3	Reserved, the reset value must be maintained.
Bit 2	**CC2IE** : Allow capture/compare 2 interrupt (Capture/Compare 2 interrupt enable) 0: Disable capture/compare 2 interrupt 1: Allow capture/compare 2 interrupt
Bit 1	**CC1IE** : Allow capture/compare 1 interrupt (Capture/Compare 1 interrupt enable) 0: Disable capture/compare 1 interrupt 1: Allow capture/compare 1 interrupt
Bit 0	**UIE** : Update interrupt enable (Update interrupt enable) 0: Disable update interrupt 1: Allow update interruption

13.4.5 Status Register (**TIMx_SR**)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Reserve			CC2 OF	CC1 OF	Reserve		TIF		Reserve		CC2 IF	CC1 IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bit 31: 11	Reserve
Bit 10	**CC2OF** : Capture/Compare 2 overcapture flag

	See CC1OF description.
Bit 9	**CC1OF** : Capture/Compare 1 overcapture flag This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear this bit. 0: no repeated capture 1: When the value of the counter is captured into the TIMx_CCR1 register, the status of CC1IF is already 1
Bit 8: 7	Reserved, the reset value must be maintained.
Bit 6	**TIF** : Trigger interrupt flag (Trigger interrupt flag) When a trigger event occurs (when the slave mode controller is in a mode other than gated mode, detect at the TRGI input To the valid edge, or any edge in the gated mode), this bit is set by the hardware. It is cleared by software. 0: No trigger event is generated 1: Trigger interrupt waiting for response
Bit 5: 3	Reserved, the reset value must be maintained.
Bit 2	**CC2IF** : Capture/Compare 2 interrupt flag Refer to CC1IF description.
Bit 1	**CC1IF** : Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This bit is set by hardware when the counter value matches the comparison value, except in the center symmetric mode (refer to TIMx_CR1 CMS bit of the register). It is cleared by software. 0: No match occurred 1: The value of TIMx_CNT matches the value of TIMx_CCR1 If channel CC1 is configured as input mode: This bit is set by hardware when a capture event occurs, and it is cleared by software or cleared by reading TIMx_CCR1. 0: No input capture is generated 1: The counter value has been captured (copied) to TIMx_CCR1 (the same edge as the selected polarity is detected on IC1)
Bit 0	**UIF** : Update interrupt flag (Update interrupt flag) This bit is set by hardware when an update event is generated. It is cleared by software. 0: No update event is generated 1: Update event waiting for response. This bit is set by hardware when the register is updated - Overflow and when UDIS = "0" in the TIMx_CR1 register. - When used by software because URS = "0" and UDIS = "0" in the TIMx_CR1 register When the UG bit in the TIMx_EGR register reinitializes the CNT. - When the trigger event is passed due to URS = "0" and UDIS = "0" in the TIMx_CR1 register (please refer to See the description of the synchronization control register) when reinitializing CNT.

13.4.6 Event generation register (**TIMx_EGR**)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve									TG	Reserve			CC2G	CC1G	UG
									w		w	w	w	w	w

Bit 31: 7	Reserved, the reset value must be maintained.
Bit 6	**TG** : Generate trigger event (Trigger generation) This bit is set by software to generate a trigger event and is automatically cleared by hardware. 0: No action 1: TIF = 1 in the TIMx_SR register. If the corresponding interrupt is turned on, the corresponding interrupt will be generated.
Bit 5:3	Reserved, the reset value must be maintained.
Bit 2	**CC2G** : Generate capture/compare 2 generation events (Capture/compare 2 generation)

	Refer to CC1G description.
Bit 1	**CC1G** : Generate capture/compare 1 generation event (Capture/compare 1 generation) This bit is set by software to generate a capture/compare event and is automatically cleared by hardware. 0: No action 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as output: Set CC1IF = 1, if the corresponding interrupt is turned on, the corresponding interrupt will be generated. If channel CC1 is configured as input: The current counter value is captured to the TIMx_CCR1 register, set CC1IF = 1, if the corresponding interrupt is turned on, it will be A corresponding interruption occurs. If CC1IF is already 1, set CC1OF = 1.
Bit 0	**UG** : Generate update event (Update generation) This bit is set by software and cleared by hardware automatically. 0: No action 1: Reinitialize the counter and generate an update event. Note that the counter of the prescaler is also cleared to 0 (but the prescaler The frequency division coefficient remains unchanged).

13.4.7 Capture / Compare Mode Register 1 (TIMx_CCMR1)

Offset address: 0x18

Reset value: 0x0000

The channel can be used for input (capture mode) or output (comparison mode), and the direction of the channel is defined by the corresponding CCxS. This register other The role of the bit is different from that in the output mode. OCxx describes the function of the channel in output mode, and ICxx describes the function of the channel in output mode. can. Therefore, it must be noted that the function of the same bit in output mode and input mode is different.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve	OC2M			OC2PE	OC2FE	CC2S		Reserve	0C1M			OC1PE	OC1FE	CC1S	
IC2F				IC2PSC				IC1F				IC1PSC			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output comparison mode:

Bit 15	Reserved, the reset value must be maintained.
Bit 14: 12	**0C2M[2 : 0]** : Output compare 2 mode
Bit 11	**OC2PE** : Output compare 2 preload enable
Bit 10	**OC2FE** : Output compare 2 fast enable
Bit 9: 8	**CC2S[1 : 0]** : Capture/Compare 2 selection This bit defines the direction of the channel (input/output), and the selection of input pins: 00: CC2 channel is configured as output; 01: The CC2 channel is configured as an input, and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input, and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input, and IC2 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).
Bit 7	Reserved, the reset value must be maintained.

Bit 6: 4	**0C1M[2 : 0]** : Output compare 1 mode (Output compare 1 enable) The 3 bits define the action of the output reference signal OC1REF, and OC1REF determines the values of OC1 and OC1N. OC1REF is effective at high level, while the effective level of OC1 and OC1N depends on the CC1P and CC1NP bits. 000: Freeze. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT does not affect OC1REF effect; 001: Set channel 1 as the effective level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, OC1REF is forced to be high. 010: Set channel 1 to an invalid level when matching. When the value of the counter TIMx_CNT and the capture/compare register 1 (TIMx_CCR1) When the same, force OC1REF to be low. 011: Flip. When TIMx_CCR1 = TIMx_CNT, the level of OC1REF is inverted. 100: Forced to an invalid level. Force OC1REF to be low. 101: Forced to be a valid level. Force OC1REF to be high. 110: PWM mode 1-when counting up, once TIMx_CNT <TIMx_CCR1, channel 1 is the active level, Otherwise, it is an invalid level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is an invalid level (OC1REF = 0), otherwise it is an effective level (OC1REF = 1). 111: PWM mode 2-When counting up, once TIMx_CNT <TIMx_CCR1, channel 1 becomes an invalid level, Otherwise, it is the effective level; when counting down, once TIMx_CNT> TIMx_CCR1, channel 1 is the effective level, Otherwise, it is an invalid level. Note: In PWM mode 1 or PWM mode 2, only when the comparison result changes or freezes from the output comparison mode The OC1REF level only changes when the mode is switched to PWM mode.
Bit 3	**OC1PE** : Output compare 1 preload enable 0: Disable the preload function of the TIMx_CCR1 register, and can write to the TIMx_CCR1 register at any time, and write a new one The value of will take effect immediately. 1: Turn on the preload function of the TIMx_CCR1 register, read and write operations only operate on the preload register, TIMx_CCR1 The preloaded value of 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 TIMx_BDTR register) and CC1S = 00 (the channel is configured as an output), then this bit cannot modified. Note 2: Only in single pulse mode (OPM = 1 in the TIMx_CR1 register), you can preload the register without confirming In this case, use the PWM mode, otherwise its action is uncertain.
Bit 2	**OC1FE** : Output compare 1 fast enable This bit is used to speed up the response of the CC output to the trigger input event. 0: According to the value of the counter and CCR1, CC1 operates normally, even if the trigger is turned on. When the trigger input has At a valid edge, the minimum delay for activating the CC1 output is 5 clock cycles. 1: The effective edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to compare power It has nothing to do with the comparison result. The delay between the valid edge of the sampling flip-flop and the output of CC1 is shortened to 3 clock cycles. OCFE only works when the channel is configured in PWM1 or PWM2 mode.
Bit 1: 0	**CC1S[1 : 0]** : Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC1 channel is configured as output; 01: The CC1 channel is configured as an input, and IC1 is mapped on TI1; 10: The CC1 channel is configured as an input, and IC1 is mapped on TI2; 11: The CC1 channel is configured as an input, and IC1 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register).

Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

Input capture mode:

Bit 15: 12	**IC2F[3 : 0]** : Input capture 2 filter
Bit 11: 10	**IC2PSC[1 : 0]** : input/capture 2 prescaler (input capture 2 prescaler)
Bit 9: 8	**CC2S[1 : 0]** : Capture/compare 2 selection These 2 bits define the direction of the channel (input/output), and the selection of input pins: 00: CC2 channel is configured as output; 01: The CC2 channel is configured as an input, and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input, and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input, and IC2 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC2S is only writable when the channel is closed (CC2E = 0 in the TIMx_CCER register).
Bit 7: 4	**IC1F[3 : 0]** : 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 group After it records N events, it will produce an output transition: 0000: No filter, sampling with f_{DTS} 1000: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 6 0001: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 2 1001: Sampling frequency $f_{SAMPLING} = f_{DTS}/8$, N = 8 0010: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 4 1010: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 5 0011: Sampling frequency $f_{SAMPLING} = f_{CK\_INT}$, N = 8 1011: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 6 0100: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 6 1100: Sampling frequency $f_{SAMPLING} = f_{DTS}/16$, N = 8 0101: Sampling frequency $f_{SAMPLING} = f_{DTS}/2$, N = 8 1101: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 5 0110: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 6 1110: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 6 0111: Sampling frequency $f_{SAMPLING} = f_{DTS}/4$, N = 8 1111: Sampling frequency $f_{SAMPLING} = f_{DTS}/32$, N = 8
Bit 3: 2	**IC1PSC[1 : 0]** : Input/capture 1 prescaler (Input capture 1 prescaler) These 2 bits define the prescaler coefficient of the CC1 input (IC1). Once CC1E = 0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, every edge detected on the capture input port triggers a capture; 01: Trigger a capture every 2 events; 10: Trigger a capture every 4 events; 11: Trigger a capture every 8 events.

CC1S[1 : 0] : Capture/Compare 1 selection

These 2 bits define the direction of the channel (input/output), and the selection of input pins:

Bit 1: 0	00: CC1 channel is configured as output; 01: The CC1 channel is configured as an input, and IC1 is mapped on TI1; 10: The CC1 channel is configured as an input, and IC1 is mapped on TI2; 11: The CC1 channel is configured as an input, and IC1 is mapped on the TRC. This mode only works when the internal trigger input is selected Time (selected by TS bit in TIMx_SMCR register). Note: CC1S is only writable when the channel is closed (CC1E = 0 in the TIMx_CCER register).

13.4.8 Capture **/** Compare Enable Register (**TIMx_CCER**)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserve					CC2P	CC1E	Reserve		CC1P	CC1E
		w	w			w	w			w	w			w	w

Bit 15: 6	Reserved, always read as 0.
Bit 5	**CC2P** : Input/Capture 2 output polarity (Capture/Compare 2 output polarity) Refer to the description of CC1P.
Bit 4	**CC2E** : Capture/Compare 2 output enable Refer to the description of CC1E.
Bit 3: 2	Reserved, always read as 0.
Bit 1	**CC1P** : Input/Capture 1 output polarity (Capture/Compare 1 output polarity) CC1 channel is configured as output: 0: OC1 is valid at high level; 1: OC1 is active at low level. CC1 channel is configured as input: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: No inversion: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 does not invert. 1: Inversion: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted. Note: Once the LOCK level (LCCK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.

Bit 0	**CC1E** : Input/Compare 1 output enable (Capture/Compare 1 output enable) CC1 channel is configured as output: 0: Off-OC1 prohibits output, so the output level of OC1 depends on MOE, OSSI, OSSR, OIS1, OIS1N And the value of the CC1NE bit. 1: On-OC1 signal is output to the corresponding output pin, and its output level depends on MOE, OSSI, OSSR, The value of the OIS1, OIS1N, and CC1NE bits. CC1 channel is configured as input: This bit determines whether the value of the counter can be captured into the TIMx_CCR1 register. 0: Capture prohibited; 1: Capture enable.

Table **31.** 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 pin is connected to a standard *OCx* channel external *I / O* pin states, depending *OCx* channel status and *GPIO* and *AFIO* Send Memory.

13.4.9 Counter (**TIMx_CNT**)

Offset address: 0x24

Reset value: 0x0000

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

Bit 31:0	**CNT[31 : 0]** : Counter value

13.4.10 Prescaler (**TIMx_PSC**)

Offset address: 0x28

Reset value: 0x0000

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

Bit 15:0	**PSC[15 : 0]** : Prescaler value The clock frequency of the counter (CK_CNT) is equal to f $_{CK\_PSC}$ / (PSC[15:0] + 1). The PSC contains the value loaded into the current prescaler register each time an update event occurs. Update event including count The device is cleared to '0' by the UG bit of TIM_EGR or is cleared to '0' by the slave controller working in reset mode.

13.4.11 Auto-load register (**TIMx_ARR**)

Offset address: 0x2C

Reset value: 0x0000

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

Bit 31:0	**ARR[31 : 0]** : Auto reload value ARR contains the value to be loaded into the actual auto-reload register. When the value of auto reload is empty, the counter does not work.

13.4.12 Capture **/** Compare Register **1** (**TIMx_CCR1**)

Offset address: 0x34

Reset value: 0x0000

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

CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR1[31 : 0]** : Capture/Compare 1 value If the CC1 channel is configured as output: CCR1 contains the value loaded into the current capture/compare 1 register (preload value). If the preload function is not selected in the TIMx_CCMR1 register (OC1PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare 1 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC1 port Signal. If the CC1 channel is configured as input: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).

13.4.13 Capture / Compare Register 2 (TIMx_CCR2)

Offset address: 0x38

Reset value: 0x0000

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	Rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**CCR2[31 : 0]** : Capture/Compare 2 value If the CC2 channel is configured as output: CCR2 contains the value loaded into the current capture/compare 2 register (preload value). If the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit), the written value will be transferred immediately To the current register. Otherwise, only when an update event occurs, the preload value is transferred to the current capture/compare 2 register 器中。The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates output on the OC2 port Signal. If the CC2 channel is configured as input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).

Page 258

14. Basic timer (**TIM8/9/10**)

14.1 TIM8 / 9/10 Description

The general-purpose timer is a 32-bit auto-loading counter driven by a programmable prescaler.

TIM8/9/10 timers are completely independent and do not share any resources with each other. They can operate simultaneously.

14.2 Main functions of **TIM8/9/10**

The general TIM8/9/10 timer functions include:

- 32-bit auto-load increment counter.
- 16-bit programmable (can be modified in real time) prescaler, the frequency division factor of the counter clock frequency is any number between 1 and 65536 value
- Interrupt/DMA is generated when the following events occur: the counter overflows

Figure **132.** Basic timer block diagram

14.3 TIM8/9/10 function description

14.3.1 Time base unit

The main part of the programmable general-purpose timer is a 32-bit up counter and its associated auto-loading register. The counter clock is pre-
The frequency is divided by the frequency divider.

The counter, auto-load 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:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)

Page 259

- Auto reload register (TIMx_ARR)

The auto-load register is pre-loaded, and writing or reading the auto-reload register will access the pre-load register. According to the post in TIMX_CR1
The auto-loading pre-loading enable bit (ARPE) setting in the register, the content of the pre-loading register is immediately or in every update event UEV
When transferred to the shadow register. When the counter reaches the overflow condition (underflow condition when counting down) and when the UDIS in the TIMX_CR1 register
When the bit is equal to 0, an update event is generated. Update events can also be generated by software. The generation of update events under each configuration will be described in detail

The counter is driven by the clock output CK_CNT of the prescaler, only when the counter enable in the counter TIMX_CR1 register is set
CK_CNT is valid only when bit (CEN).

Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a (sent in TIMx_PSC
32-bit counter controlled by 16-bit register. Because this control register has a buffer, it can be changed during operation.
The parameters of the new prescaler will be adopted when the next update event arrives.

The following two figures show examples of changing counter parameters when the prescaler is running.

Figure **133.** When the parameter of the prescaler changes from **1** to **2** , the timing diagram of the counter

Figure **134.** When the prescaler parameter changes from **1** to **4** , the timing diagram of the counter

14.3.2 Counting Mode

The counter counts from 0 to the auto-load value (the contents of the TIMx_ARR counter), and then restarts counting from 0 and generates a counter Counter overflow event.

An update event can be generated every time the counter overflows. Set the UG bit in the TIMx_EGR register (by software or by using slave Mode controller) can also generate an update event.

Setting the UDIS bit in the TIMx_CR1 register can disable the update event; this can avoid writing new data to the preload register. The shadow register is updated when the value is set. Until the UDIS bit is cleared to 0, no update event will be generated. But when an update event should be generated, the counter It will still be cleared to 0, and the count of the prescaler will be set to 0 (but the value of the prescaler will not change). In addition, if the TIMx_CR1 register is set URS bit (select update request) in the device, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (that is, no Generate an interrupt or DMA request). This is to avoid generating update and capture interrupts at the same time when clearing the counter in capture mode.

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

- The buffer of the prescaler is put into the value of the preload register (the content of the TIMx_PSC register)
- The autoload shadow register is reset to the value of the preload register (TIMx_ARR)

The following figure gives some examples, when TIMx_ARR = 0x36, the action of the counter at different clock frequencies:

Figure **135.** Counter timing diagram, the internal clock division factor is **1**

Figure **136.** Counter timing diagram, the internal clock division factor is **2**

Figure **137.** Counter timing diagram, the internal clock division factor is **4**

Figure **138.** Counter timing diagram, the internal clock division factor is **N**

Figure **139.** Counter timing diagram, update event when **ARPE = 0** (**TIMx_ARR is** not preloaded)

Figure **140.** Counter timing diagram, update event when **ARPE = 1** (**TIMx_ARR is preloaded**)

14.3.3 Clock Source

The counter clock can be provided by the internal clock (CK_INT) clock source.

The CEN and UG bits (TIMx_EGR register) are de facto control bits and can only be modified by software (the UG bit is still automatically cleared). Once the CEN bit is written as 1, the prescaler clock is provided by the internal clock CK_INT.

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

Figure **141.** The control circuit in normal mode, the internal clock division factor is **1**

14.3.4 Debug Mode

When the microcontroller enters the debug mode (CPU core is stopped), according to the setting of DBG_TIMx_STOP in the DBG module, the TIMx counts The counter will either continue normal operation or stop. For details, see the chapter Debugging Module.

14.4 **TIM8/9/10** register description

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) mode.

14.4.1 Control Register **1** (**TIMx_CR1**)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Reserve				ARPE		Reserve			OPM URS UDIS CEN		
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserved, the reset value must be maintained.
Bit 7	**ARPE** : Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is loaded into the buffer
Bit 6: 4	Reserved, the reset value must be maintained.
Bit 3	**OPM** : One pulse mode 0: When an update event occurs, the counter does not stop 1: When the next update event occurs (clear the CEN bit), the counter stops
Bit 2	**URS** : Update request source The software selects the source of the UEV event through this bit 0: If an update interrupt or DMA request is allowed, any of the following events will generate an update interrupt or DMA request: − Counter overflow − Set the UG bit − Updates generated from the mode controller 1: If an update interrupt or DMA request is allowed, an update interrupt or DMA request will only be generated when the counter overflows
Bit 1	**UDIS** : Update disable The software allows/disables the generation of UEV events through this bit 0: UEV is allowed. Update (UEV) events are generated by any of the following events: − Counter overflow − Set the UG bit − The updated buffered registers generated from the mode controller are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) maintain their values. If set If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.
Bit 0	**CEN** : Counter enable 0: disable the counter 1: Enable the counter Note: After the software sets the CEN bit, the external clock, gating mode and encoder mode can only work. Trigger mode can be automatic The CEN bit is set by hardware. In single pulse mode, when an update event occurs, CEN is automatically cleared.

14.4.2 Control Register **2** (**TIMx_CR2**)

Offset address: 0x04

Reset value: 0x0000

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

Bit 31: 7	Reserved, the reset value must be maintained.
	MMS[1 : **0]** : Master mode selection

Bit 6: 4	These two bits are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. The possible combinations are as follows: 000: Reset-The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode If the controller is in reset mode) to generate a reset, the signal on TRGO will have a delay relative to the actual reset. 001: Enable-the counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes need to be at the same time Start multiple timers or control to enable slave timers within a period of time. The counter enable signal is controlled by the CEN control bit and gate The logic or generation of the trigger input signal in the mode. When the counter enable signal is controlled by the trigger input, there will be a A delay, unless the master/slave mode is selected (see the description of the MSM bit in the TIMx_SMCR register). 010: Update-The update event is selected as the trigger input (TRGO). For example, the clock of a master timer can be used as a A prescaler for the slave timer.
Bit 3: 0	Reserved, the reset value must be maintained.

14.4.3 DMA/ Interrupt Enable Register (**TIMX_DIER**)

Offset address: 0x0C

Reset value: 0x0000

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

Bit 31: 9	Reserved, the reset value must be maintained.
Bit 8	Bit **8 UDE** : Update DMA request enable (Update DMA request enable) 0: Prohibit updating DMA request. 1: Enable update DMA request.
Bit 7:1	Reserved, the reset value must be maintained.
Bit 0	**UIE** : Update interrupt enable (Update interrupt enable) 0: Disable update interrupt 1: Allow update interruption

14.4.4 Status Register (**TIMx_SR**)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															UIF
															rc_w0

Bit 31:1	Reserved, the reset value must be maintained.
Bit 0	**UIF** : Update interrupt flag (Update interrupt flag) This bit is set by hardware when an update event occurs. But it needs to be cleared by software. 0: No update has occurred. 1: Update interruption pending. This bit is set by hardware when updating the register in the following situations: — Overflow or underflow and when UDIS = 0 in the TIMx_CR1 register. — When TIMx_EGR is used by software because URS = 0 and UDIS = 0 in the TIMx_CR1 register When the UG bit in the register reinitializes the CNT.

14.4.5 Event generation register (**TIMx_EGR**)

Offset address: 0x14

Reset value: 0x0000

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

Reserve	UG
	w

Bit 31: 7	Reserved, the reset value must be maintained.
Bit 0	**UG** : Generate update event (Update generation) This bit is set by software and cleared by hardware automatically. 0: No action 1: Reinitialize the counter and generate an update event. Note that the counter of the prescaler is also cleared to 0 (but the prescaler The frequency division coefficient remains unchanged).

14.4.6 Counter (**TIMx_CNT**)

Offset address: 0x24

Reset value: 0x0000

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

Bit 31:0	**CNT[31** : **0]** : Counter value

14.4.7 prescaler (**TIMx_PSC**)

Offset address: 0x28

Reset value: 0x0000

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

Bit 15:0	**PSC[15** : **0]** : Prescaler value The clock frequency of the counter (CK_CNT) is equal to f $_{CK\_PSC}$ / (PSC[15:0] + 1). The PSC contains the value loaded into the current prescaler register each time an update event occurs. Update event including count The device is cleared to '0' by the UG bit of TIM_EGR or is cleared to '0' by the slave controller working in reset mode.

14.4.8 Auto-load register (**TIMx_ARR**)

Offset address: 0x2C

Reset value: 0x0000

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 31:0	**ARR[31 : 0]** : Auto reload value ARR contains the value to be loaded into the actual auto-reload register. For details, refer to section 13.3.1: Updates and actions related to ARR. When the value of auto reload is empty, the counter does not work.

15. Independent watchdog (IWDG)

15.1 Introduction to IWDG

Built-in two watchdogs provide higher security, time accuracy and flexibility of use. Two watchdog devices (independent watchdog Dogs and window watchdogs) can be used to detect and solve faults caused by software errors; when the counter reaches a given timeout value, a medium Interrupt (only applicable to window watchdog) or generate a system reset.

The independent watchdog (IWDG) is driven by a dedicated low-speed clock (LSI), and it is still valid even if the main clock fails. Window gate The dog is driven by the clock obtained by dividing the APB1 clock, and detects abnormally late or early application programs through a configurable time window operate.

IWDG is most suitable for those who need the watchdog as an outside of the main program, can work completely independently, and require time accuracy. Lowest occasions. WWDG is most suitable for applications that require the watchdog to function in a precise timing window.

15.2 Main performance of IWDG

- Free running down counter
- The clock is provided by an independent RC oscillator (can work in stop mode)
- After the watchdog is activated, it will reset when the counter reaches 0x0000.

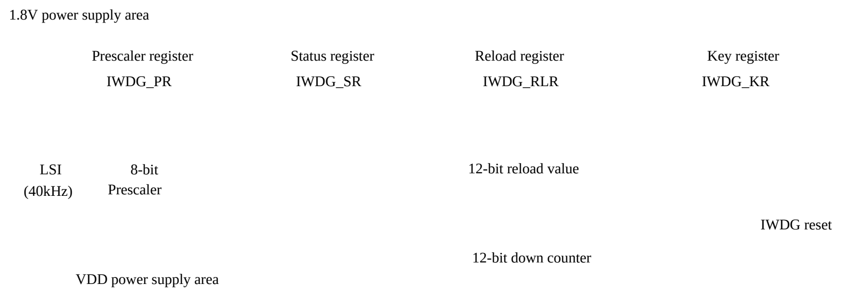
15.3 IWDG function description

The following figure shows the functional block diagram of the independent watchdog module.

Write 0xCCCC in the key register (IWDG_KR). Start to start the independent watchdog; at this time the counter starts from its reset value 0xFFF Count down. When the counter counts to the end 0x000, a reset signal (IWGD_RESET) will be generated.

Whenever you write 0xAAAA in the key register IWDG_KR, the value in IWDG_RLR will be reloaded to the counter. In order to avoid a watchdog reset.

Figure **142.** Independent watchdog block diagram



Note: The watchdog function is in the *VDD* power supply area, that is, it can still work normally in shutdown mode.

Table **32.** Watchdog timeout time (**40kHz** input clock (**LSI**))

Prescaler coefficient	PR[2:0] bit	shortest time	longest time
		RL[11:0]=0x000	RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4

/32	3	0.8	3,276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 or 7)	6.4	26214.4

Note: These times are given in accordance with the *40kHz* clock. In fact, the *RC* frequency inside the *MCU* will vary from *30kHz* to *60kHz* .

In addition, even if the frequency of the RC oscillator is accurate, the exact timing still depends on the APB interface clock and the RC oscillator clock. The phase difference, so there will always be a complete RC cycle is uncertain.

A relatively accurate watchdog timeout period can be obtained by calibrating the LSI.

15.3.1 Hardware Watchdog

If the user activates the "hardware watchdog" function in the selection byte, the watchdog will automatically start to run after the system is powered on and reset; for example, If the software does not write the corresponding value to the key register before the counter ends, the system will reset.

15.3.2 Register access protection

The IWDG_PR and IWDG_RLR registers are write-protected. To modify the values of these two registers, you must first send to IWDG_KR Write 0x5555 in the register. Writing to this register with a different value will disrupt the operation sequence and the register will be protected again. Reload operation (That is, write 0xAAAA) will also activate the write protection function.

The status register indicates whether the prescaler value and the down counter are being updated.

15.3.3 Debug mode

When the microcontroller enters the debug mode (CPU core stops), according to the status of the DBG_IWDG_STOP configuration bit in the debug module, The counter of IWDG can continue to work or stop. See the chapter on debugging module for details.

15.4 **IWDG** register description

15.4.1 Key Register (**IWDG_KR**)

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
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31: 15	Reserved, always read as 0.
Bit 15:0	KEY[15:0]: Key value (write register only, read value is 0x0000) (Key value) The software must write 0xAAAA at a certain interval, otherwise, when the counter is 0, the watchdog will reset. Writing 0x5555 indicates that access to the IWDG_PR and IWDG_RLR registers is allowed. Write 0xCCCC to start watchdog work

15.4.2 Prescaler Register (**IWDG_PR**)

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													PR[2:0]		
													rw	rw	rw

Bit 31: 3	Reserved, always read as 0.
Bit 2: 0	PR[2:0]: Prescaler divider These bits have write protection settings. Select the prescaler factor of the counter clock by setting these bits. To change the pre-score Frequency factor, the PVU bit of the IWDG_SR register must be 0. 000: Prescaler factor=4 100: Prescaler factor=64 001: Prescaler factor=8 100: Prescaler factor=128 010: Prescaler factor=16 100: Prescaler factor=256 011: Prescaler factor=32 100: Prescaler factor=256 Note: Reading this register will return the prescale value from the VDD voltage domain. If the write operation is in progress, The value read back may be invalid. Therefore, only when the PUV bit of that IWDG_SR register is 0, the read Value is valid

15.4.3 Reload register (**IWDG_RLR**)

Offset address: 0x08

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				RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 12	Reserved, always read as 0.
Bit 11:0	RL[11:0]: Watchdog counter reload value These bits are write-protected. Used to define the reload value of the watchdog counter, whenever writing to the IWDG_KR register At 0xAAAA, the reload value will be transferred to the counter. Then the counter starts counting down from this value. The watchdog time-out period can be calculated by the reload value and the clock prescaler value. This register can be modified only when the RVU bit in the IWDG_SR register is 0. Note: Reading this register will return the prescaler value from the VDD voltage domain. If the write operation is in progress, then The value read back may be invalid. Therefore, only when the RUV bit of the IWDG_SR register is 0, the read value has effect.

15.4.4 Status Register (**IWDG_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														RVU	PVU
														r	r

Bit 31: 2	Reserved, always read as 0.

Bit 1	RVU: Watchdog counter reload value update This bit is set to "1" by hardware to indicate that the update of the reload value is in progress. When the reload update in the VDD domain ends After the bundle, this bit is cleared to “0” by hardware (up to 5 40kHz RC cycles are required). The reload value can only be cleared when the RVU bit is cleared. It can be updated after "0".
Bit 0	PVU: Watchdog prescaler value update This bit is set to "1" by hardware to indicate that the update of the prescaler value is in progress. When the prescale value in the VDD domain is updated After the end, this bit is cleared to “0” by hardware (up to 5 40kHz RC cycles are required). The prescaler value can only be set in the RVU bit. It can be updated after clearing "0".

Page 273

Note: If multiple reload values or prescaler values are used in the application, the preload can be changed only after the *RVU* bit is cleared
The prescaler value can only be changed after the *PVU* bit is cleared. However, after the prescaler and */* or reload value is updated, there is no need to wait for *RVU* or *PVU*
Reset, you can continue to execute the following code. (Even in the low-power mode, the write operation will continue to be executed to complete)

16. Window watchdog (**WWDG**)

16.1 Introduction to **WWDG**

The window watchdog is usually used to monitor the deviation of the application from the normal operating sequence caused by external interference or unforeseen logical conditions. The resulting software failure. Unless the value of the down counter is refreshed before the T6 bit becomes 0, the watchdog circuit will Generate an MCU reset. Before the down counter reaches the window register value, if the 7-bit down counter value (in the control register In) is refreshed, then an MCU reset will also be generated. This indicates that the down counter needs to be refreshed in a limited time window.

16.2 Main features of **WWDG**

- Programmable free running down counter
- Condition reset
  - When the value of the down counter is less than 0x40, (if the watchdog is started), a reset is generated.
  - When the down counter is reloaded outside the window, (if the watchdog is started) a reset is generated
- If the watchdog is enabled and interrupts are enabled, an early wake-up interrupt (EWI) is generated when the down counter is equal to 0x40, it can be Used to reload the counter to avoid WWDG reset.

16.3 **WWDG** functional description

If the watchdog is activated (the WDGA bit in the WWDG_CR register is set to '1'), and when the 7 bits (T[6:0]) decrement the counter from When 0x40 rolls over to 0x3F (T6 bit is cleared), a reset is generated. If the software restarts when the counter value is greater than the value in the window register Loading the counter will generate a reset.

Figure **143.** Watchdog block diagram

The application program must periodically write the WWDG_CR register during normal operation to prevent the MCU from resetting. Only when the counter When the value is less than the value of the window register, the write operation can be performed. The value stored in the WWDG_CR register must be between 0xFF and 0xC0 between:

- Start watchdog

After the system is reset, the watchdog is always off. Setting the WDGA bit of the WWDG_CR register can enable the watchdog. After that, it can no longer be turned off unless a reset occurs.

- Control down counter

The down counter is in a free-running state. Even if the watchdog is disabled, the down counter continues to count down. When the watchdog is enabled, The T6 bit must be set to prevent an immediate reset.

The T[5:0] bit contains the number of timings before the watchdog resets; the delay time before reset is between a minimum value and a maximum value

This is because the prescaler value is unknown when writing to the WWDG_CR register.

The configuration register (WWDG_CFR) contains the upper limit of the window: to avoid resetting, the down counter must be smaller than the window
When the value of the register is greater than 0x3F, it is reloaded. The following figure describes the working process of the window register.

Another way to reload the counter is to use the early wake-up interrupt (EWI). Set the WEI bit in the WWDG_CFR register to enable this
Interrupted. When the down counter reaches 0x40, this interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to load the counter to prevent WWDG is reset. Write '0' in the WWDG_SR register to clear the interrupt.

Note: The *T6* bit can be used to generate a software reset (the *WDGA* bit is set and the *T6* bit is cleared)

16.4 How to write a watchdog timeout program

The following figure shows the linear relationship between the 6-bit count value loaded into the watchdog counter (CNT) and the delay time of the watchdog (in ms unit). This graph can be used as a reference for quick calculation without taking the time deviation into consideration. If you need higher accuracy, you can use the following The calculation formula provided in the figure.

Warning: When writing to the WWDG_CR register, always set the T6 bit to '1' to avoid an immediate reset

Figure **144.** Window watchdog timing diagram

16.5 Debug mode

When the microcontroller enters the debug mode (CPU core stops), according to the state of the DBG_WWDG_STOP configuration bit in the debug module,
The counter of WWDG can continue to work or stop. See the chapter about debugging modules for details.

16.6 WWDG register description

16.6.1 Control Register (**WWDG_CR**)

Offset address: 0x00

Reset value: 0x7F

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								WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8 Reserved, always read as 0.

WDGA: Activation bit
This bit is set to "1" by software, but it can only be set to "0" by hardware after reset. When WDGA=1, the watchdog can generate

Bit 7	Reset. 0: Disable watchdog 1: Start watchdog
Bit 6:0	T[6:0]: 7-bit counter (MSB to LSB) (7-bit counter) These bits are used to store the counter value of the watchdog. Subtract 1 every (4096x2WDGTB) cycles of PCLK1. When the counter value When changing from 40h to 3Fh (T6 becomes 0), a watchdog reset is generated.

16.6.2 Configuration Register (**WWDG_CFR**)

Offset address: 0x04

Reset value: 0x7F

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				EWI	WDGTB					W[6:0]			
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserved, always read as 0.
Bit 9	EWI: Early wakeup interrupt If this bit is set to 1, an interrupt will be generated when the counter value reaches 40h. This interrupt can only be cleared by hardware after reset.

Bit 8: 7	WDGTB[1:0]: Time base (Timer base) The time base of the prescaler can be modified as follows: 00: CK timer clock (PCLK1 divided by 4096) divided by 1 01: CK timer clock (PCLK1 divided by 4096) divided by 2 10: CK timer clock (PCLK1 divided by 4096) divided by 4 11: CK timer clock (PCLK1 divided by 4096) divided by 8
Bit 6:0	W[6:0]: 7-bit window value These bits contain the window value used for comparison with the down counter.

16.6.3 Status Register (**WWDG_SR**)

Offset address: 0x08

Reset value: 0x00

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								EWIF
															rc w0

Bit 31:1	Reserved, always read as 0.
Bit 0	EWIF: Early wakeup interrupt flag When the counter value reaches 40h, this bit is set by hardware. It must be cleared by software writing '0'. Writing '1' to this bit is invalid. If the interrupt is not enabled, this bit will also be set to '1'.

17. Real Time Clock (**RTC**)

17.1 Introduction to **RTC**

The real-time clock is an independent timer. The RTC module has a set of continuous counting counters. Under the corresponding software configuration, it can provide time Clock calendar function. Modifying the counter value can reset the current time and date of the system.

The RTC module and the clock configuration system (RCC_BDCR register) are in the backup area, that is, after the system is reset, the RTC setting and time The time remains the same.

After the system is reset, the access to the backup register and RTC is prohibited. This is to prevent accidental write operations to the backup area (BKP). Hold on The following operations will enable access to the backup register and RTC.

- Set the PWREN and BKPEN bits in the RCC_APB1ENR register to enable the power supply and backup interface clock
- Set the DBP bit in the register PWR_CR to enable access to the backup register and RTC.

17.2 Main features

- Programmable pre-scaling factor: the frequency division factor is up to 220
- 32-bit programmable counter for long time period measurement
- 2 separate clocks: PCLK1 and RTC clock used for the APB1 interface (the frequency of the RTC clock must be less than that of PCLK1 More than a quarter of the clock frequency).
- You can choose the clock source of three kinds of RTC
  - Divide the HSE clock by 128;
  - LSE oscillator clock;
  - LSI oscillator clock
- 2 independent reset types:
  - The APB1 interface is reset by the system;
  - The RTC core (prescaler, alarm clock, counter and divider) can only be reset by the backup domain.
- 3 special shielded interrupts:
  - Alarm interrupt, used to generate a software programmable alarm interrupt
  - Second interrupt, used to generate a programmable periodic interrupt signal (up to 1 second)
  - Overflow interrupt, indicating that the internal programmable counter overflows and turns to 0

17.3 Functional description

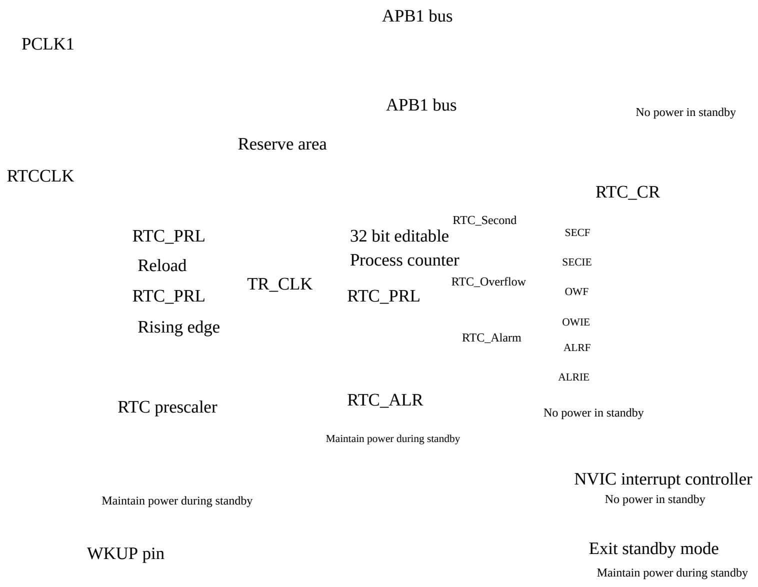
17.3.1 Overview

RTC consists of two main parts. See the picture above. The first part (APB1 interface) is used to connect to the APB1 bus. This unit also includes Contains a set of 16-bit registers, which can be read and written via the APB bus. The APB1 interface is driven by the APB1 bus clock to communicate with APB1 bus interface.

The other part (RTC core) consists of a set of programmable counters, divided into two main modules. The first module is the prescaler module of RTC Block, it can be programmed to generate the RTC time reference TR_CLK up to 1 second. The prescaler module of RTC contains a 20-bit programmable prescaler Frequency converter (RTC prescaler). If the corresponding enable bit is set in the RTC_CR register, then the RTC in each TR_CLK cycle Generate an interrupt (second interrupt). The second module is a 32-bit programmable counter that can be initialized to the current system time. system The time is accumulated according to the TR_CLK cycle and compared with the programmable time stored in the RTC_ALR register. If the RTC_CR control register is If the corresponding enable bit is set, an alarm interrupt will be generated when the comparison matches.

Simplified RTC block diagram below

Figure **145.** Real-time clock block diagram



17.3.2 Reset process

Except for the RTC_PRL, RTC_ALR, RTC_CNT and RTC_DIV registers, all system registers are reset by the system or
The power reset performs an asynchronous reset.

The RTC_PRL, RTC_ALR, RTC_CNT and RTC_DIV registers can only be reset by the backup domain reset signal.

17.3.3 Read **RTC** Register

The RTC core is completely independent of the RTC APB1 interface.

The software accesses the prescaler value, counter value and alarm value of the RTC through the APB1 interface. However, the relevant readable registers are only related to RTC
The rising edge of the RTC clock where the APB1 clock is resynchronized is updated. The same is true for the RTC logo.

This means that if the APB1 interface has been closed, and the read operation is just after the APB1 is reopened, the first internal
Before the register is updated, the value of the RTC register read from APB1 may be corrupted (usually read as 0). Can be issued in the following situations
This situation occurs:

- System reset or power reset occurred
- The system just woke up from halt mode

In all of the above cases, when the APB1 interface is disabled (reset, no clock, or breakpoint), the RTC core remains running.

Therefore, if the APB1 interface of RTC has been in the disabled state when reading the RTC register, the software must first wait for RTC_CRL
The RSF bit (register synchronization flag) in the register is set to "1" by hardware.

Note: The *APB1* interface of *RTC* is not affected by low power consumption modes such as *WFI* and *WFE* .

17.3.4 Configure **RTC** registers

The CNF bit in the RTC_CRL register must be set so that RTC enters the configuration mode before writing RTC_PRL, RTC_CNT,
RTC_ALR register.

In addition, the write operation to any RTC register must be performed after the previous write operation is completed. You can send it by querying RTC_CR
The RTOFF status bit in the register determines whether the RTC register is being updated. It can be written only when the RTOFF status bit is "1"

RTC register.

Configuration process:

- Check the RTOFF bit, know that the value of RTOFF becomes "1"
- Set CNF value to 1, enter configuration mode
- Write one or more RTC registers
- Clear the CNF flag and exit the configuration mode
- Check RTOFF until the RTOFF bit becomes "1" to confirm that the write operation has been completed.
- The write operation can only be performed when the CNF flag bit is cleared. This process requires at least 3 RTCCLK cycles.

17.3.5 Setting of **RTC** flag

In each clock cycle of the RTC core, set the RTC second flag (SECF) before changing the RTC counter.

In the last RTC clock cycle before the counter reaches 0x0000, the RTC overflow flag (OWF) is set.

In the RTC clock cycle before the value of the counter reaches the value of the alarm register plus 1 (RTC_ALR+1), set RTC_Alarm and

RTC alarm flag (ALRF). The write operation to the RTC alarm clock must use one of the following processes to synchronize with the RTC second mark:

- Clock RTC alarm interrupt, and modify the RTC alarm and/or RTC counter in the interrupt handler
- Wait for the SECF bit in the RTC control register to be set, and then change the RTC alarm and/or RTC counter.

Figure **146. RTC** second and alarm waveform example, **PR=0003** , **ALARM=00004**

Figure **147.** Example of **RTC** overflow waveform, **PR=0003**

17.4 **RTC** register description

17.4.1 RTC control register high bit (**RTC_CRH**)

Address offset: 0x00

Reset value: 0x0000

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

Reserve	OWIE	ALRIE	SECIE
	rw	rw	rw

Bit 31: 3	Reserved, forced to 0 by hardware
Bit 2	OWIE: Overflow interrupt enable 0: Mask (not allowed) overflow interrupt 1: Allow overflow interrupt
Bit 1	ALRIE: Allow alarm interrupt (Alarm interrupt enable) 0: Mask (not allowed) overflow interrupt 1: Allow overflow interrupt
Bit 0	SECIE: Second interrupt enable 0: Mask (not allowed) overflow interrupt 1: Allow overflow interrupt

These bits are used to mask interrupt requests.

Note: All interrupts are masked after the system reset, so you can write to the *RTC* register to ensure that there are no suspended interrupts after initialization. begging. When the peripheral is completing the previous write operation (the flag bit *RTOFF=0*), the *RTC_CRH* register cannot be written.

The RTC function is controlled by this control register. The write operation of some bits must go through a special configuration process to complete. (See 17. 3.4 sect. A)

Page 282

17.4.2 RTC control register low bit (RTC_CRL)

Offset address: 0x04

Reset value: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc w0	rc w0	rc w0	rc w0

Bit 0	SECIE: Second interrupt enable (Second interrupt enable)) 0: Mask (not allowed) overflow interrupt 1: Allow overflow interrupt
Bit 15: 6	Reserved, forced to 0 by hardware
Bit 5	RTOFF: RTC operation OFF (RTC operation OFF) The RTC module uses this bit to indicate the status of the last operation performed on its register, indicating whether the operation is complete. If this bit "0", it means that no RTC register can be read or written. This bit is a read-only bit. 0: The last write operation to the RTC register is still in progress; 1: The last write operation to the RTC register has been completed
Bit 4	CNF: Configuration flag This bit must be set to "1" by software to enable static configuration mode, thereby allowing RTC_CNTL/H, RTC_ALRL/H or RTC_PRLL/H Register write data. Only when this bit is set to "1" and cleared to "0" by software again, the write operation will be performed. 0: Exit configuration mode (start to update RTC register) 1: Enter configuration mode
Bit 3	RSF: Registers synchronized flag (Registers synchronized flag) Whenever the RTC_CNT register and RTC_DIV register are updated or cleared to “0” by software, this bit is set to “1” by hardware. At APB1 After reset, or after the APB1 clock stops, this bit must be cleared to “0” by software. Before any read operation, the user program You must wait for this bit to be set to "1" by hardware to ensure that RTC_CNT, RTCALR or RTC_PRL has been synchronized. 0: The register has not been synchronized 1: The register has been synchronized
	OWF: Overflow flag When the 32-bit programmable counter overflows, this bit is set to "1" by hardware. If OWIE=1 in the RTC_CRH register, then

Bit 2	Health interruption. This bit can only be cleared to "0" by software. Writing "1" to this is invalid 0: no overflow 1: 32-bit programmable counter overflow
Bit 1	ALRF: Alarm flag When the 32-bit programmable counter reaches the predetermined value set by the RTC_ALR register, this bit is set to "1" by hardware. if ALRIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared to "0" by software. Write "1" to this bit is none Effective. 0: no alarm clock 1: There is an alarm clock

Bit 0	SECF: Second flag When the 32-bit programmable prescaler overflows, this bit is set to "1" by hardware and the RTC counter is incremented by 1. Therefore, this mark is distinguished The rate programmable RTC counter provides a periodic signal (usually 1 second). If the RTC_CRH register SECIE=1, An interrupt is generated. This bit can only be cleared by software. Writing "1" to this bit is invalid. 0: The second flag condition is not established 1: The second sign condition is established

The function of RTC is controlled by this control register. When the current write operation has not been completed (RTOFF=0), RTC_CR cannot be written register.

Note: *1.* Any flag bit will remain suspended until the appropriate *RTC_CR* request bit is reset by software, indicating the requested Interrupt has been accepted

2. All interrupts are forbidden during reset, and there is no pending interrupt request, and the *RTC* register can be written.

3. When the *APB1* clock is not running, the *OWF* , *ALRF* , *SECF* and *RSF* bits are not updated

4. The OWF , *ALRF* , *SECF* and *RSF* bits can only be set by hardware and cleared by software

5. If *ALRF=1* and *ALRIE=1* , *RTC* global interrupt is allowed to be generated . If *EXTI* register generates a control word order *EXTI* When line *17 is* broken, the *RTC* global interrupt and *RTC* alarm interrupt are allowed to be generated .

6. If *ALRF=1* , if the interrupt mode of *EXTI* line *17* is set in the *EXTI* controller , the *RTC* alarm interrupt is allowed to be generated ;
If the time mode of *EXTI* line *17* is set in the *EXTI* controller, a pulse will be generated on this line (no *RTC* alarm will be generated)
The clock breaks).

17.4.3 RTC prescaler load register (**RTC_PRLH/RTC_PRLL**)

The prescaler load register is used to protect the cycle count value of the RTC prescaler. They are protected by the RTOFF bit of the RTC_CR register, Write operation is allowed only when the RTOFF value is "1"

RTC prescaler load register high bit (**RTC_PRLH**)

Offset address: 0x08

Write only (see section 17.3.4)

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve												PRL[19:16]			
												w	w	w	w

Bit 15: 4	Reserved, forced to 0 by hardware
Bit 3: 0	PRL[19:16]: RTC prescaler reload value high (RTC prescaler reload value high) According to the following formula, these bits are used to define the clock frequency of the counter: $f_{TR\_CLK} = f_{RTCCLK} /(PRL[19:0]+1)$ Note: It is not recommended to use a value of 0, otherwise the RTC interrupt and flag will not be generated correctly

RTC prescaler load register low bit (**RTC_PRLL**)

Offset address: 0x0C

Write only (see section 17.3.4)

Reset value: 0x8000

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

Bit 15:0	PRL[15:0]: RTC prescaler reload value low (RTC prescaler reload value low) According to the following formula, these bits are used to define the clock frequency of the counter: $f_{TR\_CLK} = f_{RTCCLK} /(PRL[19:0]+1)$

Note: If the input clock frequency is *32.768kHz (f $_{RTCCLK}$*), write *7FFFFh in* this register to obtain a signal with a period of *1* second

No

17.4.4 RTC prescaler division factor register (**RTC_DIVH/RTC_DIVL**)

In each cycle of TR_CLK, the value of the counter in the RTC prescaler will be reset to the value of the RTC_PRL register. use
Users can read the RTC_DIV register to obtain the current value of the prescaler counter without stopping the work of the prescaler counter, thereby obtaining a precise
Accurate time measurement. This register is a read-only register. After the value in the RTC_PRL or RTC_CNT register changes, the
The pieces are reloaded.

RTC prescaler frequency division factor register high bit (**RTC_DIVH**)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve												RTC_DIV[19:16]			
												r	r	r	r

Bit 15: 4	Reserved, forced to 0 by hardware
Bit 3: 0	RTC_DIV[19:16]: RTC clock divider high (RTC clock divider high)

RTC prescaler division factor register low bit (**RTC_DIVL**)

Offset address: 0x14

Reset value: 0x8000

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

Bit 15:0	RTC_DIV[15:0]: RTC clock divider low (RTC clock divider low)

17.4.5 RTC counter register (**RTC_CNTH/RTC_CNTL**)

The RTC core has a 32-bit programmable counter that can be accessed through two 16-bit registers. The counter is generated by the prescaler The TR_CLK time base is the reference for counting. The RTC_CNT register is used to store the count value of the counter. They are affected by the bit of RTC_CR RTOFF write protection, only when the RTOFF value is "1", write operation is allowed. In the high or low register (RTC_CNTH or RTC_CNTL) The write operation on the above can be directly loaded to the corresponding programmable counter and reload the RTC prescaler. In the read operation, directly Returns the count value (system time) in the counter.

RTC counter register high bit (**RTC_CNTH**)

Offset address: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:0	RTC_CNT[31:16]: RTC counter high (RTC counter high) The high part of the current value of the RTC counter can be obtained by reading the RTC_CNTH register. To write to this register Before operation, you must first enter the configuration mode.

RTC counter register low bit (**RTC_CNTL**)

Offset address: 0x1C

Reset value: 0x0000

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

Bit 15:0	RTC_CNT[15:0]: RTC counter low (RTC counter low) The low part of the current value of the RTC counter can be obtained by reading the RTC_CNTL register. To write to this register Before operation, you must first enter the configuration mode.

17.4.6 RTC alarm register (RTC_ALRH/RTC_ALRL)

When the value of the programmable counter is equal to the 32-bit value in RTC_ALR, an alarm event is triggered and the RTC alarm is generated. Off. This register is write-protected by the RTOFF bit in the RTC_CR register. Only when RTOFF=1, write operations are allowed.

RTC alarm register high bit (**RTC_ALRH**)

Offset address: 0x20

Write only (see section 17.3.4)

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 15:0	RTC_ALR[31:16]: RTC alarm high (RTC alarm high) This register is used to protect the high part of the alarm time written by software. To write to this register, it must be advanced

Enter configuration mode

RTC counter register low bit (**RTC_ALRL**)

Offset address: 0x24

Write only (see section 17.3.4)

Reset value: 0xFFFF

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

Bit 15:0	RTC_ALR[15:0]: RTC alarm low (RTC alarm low) This register is used to protect the lower part of the alarm time written by software. To write to this register, it must be advanced Enter configuration mode

18. I2C interface

18.1 Introduction to I2C

The I2C (Inter-Chip) bus interface connects the microcontroller and the serial I2C bus. It provides multi-master functions to control all I2C bus specific time Order, agreement, arbitration and timing.

The I2C bus is a two-wire serial interface, in which the two-wire bit serial data (SDA) and serial clock (SCL) lines are connected to the bus devices Send message. Each device has a unique address identification, and can be used as a transmitter or receiver. In addition to the transmitter and receiver In addition, the device can also be regarded as a master or a slave when performing data transfers. The host initializes the data transmission of the bus and generates the allowable transmission The clock signal of the device. At this time, any device that is addressed is considered a slave.

I2C can work in standard mode (data transmission rate is 0~100Kb/s), fast mode (data transmission rate is up to 400Kb/s) And high-speed mode (maximum data transfer rate is 3.4Mb/s).

18.2 Main features of I2C

- Parallel bus I2C bus protocol converter
- Half-duplex synchronous operation
- Support master-slave mode
- Support 7-bit address and 10-bit address
- Support standard mode 100Kbps, fast mode 400Kbps and high-speed mode 3.4Mbps
- Generate Start, stop, resend start, and acknowledge Acknowledge signal detection
- Only supports one host in main mode
- There are 2 bytes of transmit and receive buffers respectively
- Added glitch-free circuits on scli and sdai
- Support DMA operation
- Support interrupt and query operations

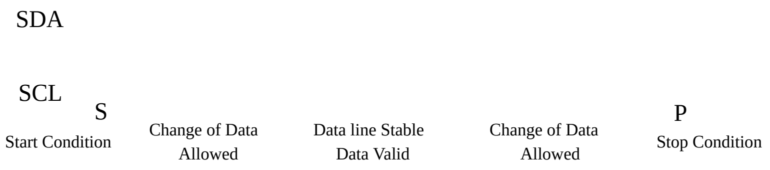
18.3 I2C protocol

18.3.1 Start and stop conditions

When the bus is in the idle state, SCL and SDA are simultaneously pulled high by the external pull-up resistor. When the host initiates data transfer, it must

First generate a starting condition. When the SCL line is high, the SDA line switches from high to low to indicate the initial condition. When the host ends the transmission A stop condition is sent when inputting. The SCL line is high, and the SDA line switches from low to high to indicate a stop condition. The image below shows the beginning And the timing diagram of the stop condition. During data transmission, when SCL is 1, SDA must remain stable.

Figure **148.** Start and stop conditions



Page 288

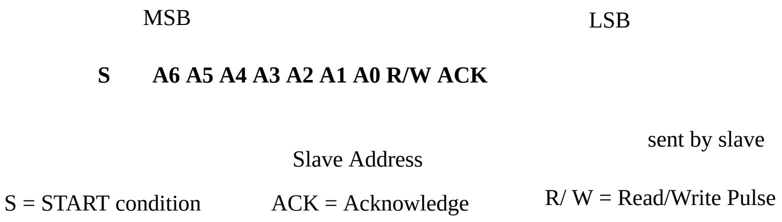
18.3.2 Slave Addressing Protocol

I2C has two address formats: 7-bit address format and 10-bit address format

7 -bit address format

The figure below shows that the first 7 bits (bit 7:1) of a byte sent after the start condition (S) are the slave address, and the lowest bit (bit 0) is the data side. To bit, when bit 0 is 0, it means that the master writes data to the slave, and 1 means that the master reads data from the slave.

Figure **149. 7** -bit address format

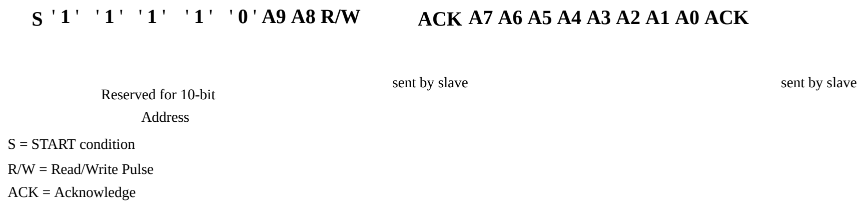


10 -bit address format

In the 10-bit address format, 2 bytes are sent to transmit the 10-bit address. The bit description of the first byte sent is as follows: the first The 5 bits (bit 7: 3) are used to inform the slave that the next 10 bits are to be transmitted. The last two bytes of the first byte (bit 2: 1) are the bit of the slave address 9: 8, the lowest bit (bit 0) is the data direction bit (R/W). The second byte transmitted is the lower eight bits of the 10-bit address.

The details are shown in the figure below:

Figure **150. 10** -bit address format


S ' 1 ' ' 1 ' ' 1 ' ' 1 ' ' 0 ' A9 A8 R/W
ACK A7 A6 A5 A4 A3 A2 A1 A0 ACK
sent by slave
sent by slave
Reserved for 10-bit Address
S = START condition
R/W = Read/Write Pulse
ACK = Acknowledge

The following table defines the special purpose and reserved address of the first byte of I2C:

Table **33. I2C** First Byte

Slave address	R/W bit	describe
0000 000	0	Broadcast call address. I2C puts the data into the receiving buffer and generates a broadcast call interrupt
0000 000	1	Start byte
0000 001	X	CBUS address. I2C interface ignores the access
0000 010	X	Reserve
0000 011	X	Reserve
0000 1xx	X	High-speed mode host code
1111 1xx	X	Reserve
1111 0xx	X	10-bit slave addressing

18.3.3 Sending and receiving protocol

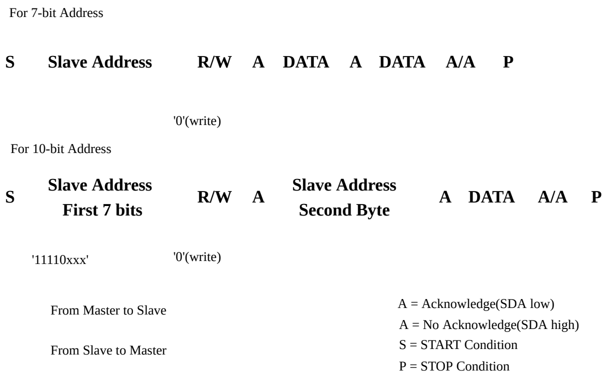
The host initiates data transmission and sends or receives data from the bus, acting as a master sending or receiving. The slave responds to the request of the master Send or receive data as a slave sender or slave receiver.

Master sending and slave receiving

All data is transmitted in byte format, and there is no limit to the number of bytes transmitted each time. When the host sends the address and R/W bit or the host sends To send a byte of data to the slave, the slave receiver must generate a response signal (ACK). When the slave receiver cannot generate an ACK response signal, The host will generate a stop condition to abort the transmission. When the slave fails to respond, it must release SDA to a high level to make the master generate a stop bar. Pieces.

When the master transmitter transmits data as shown in the figure below, the slave receiver generates an ACK after each byte received to respond to the master transmitter.

Figure **151.** Master Sending Protocol

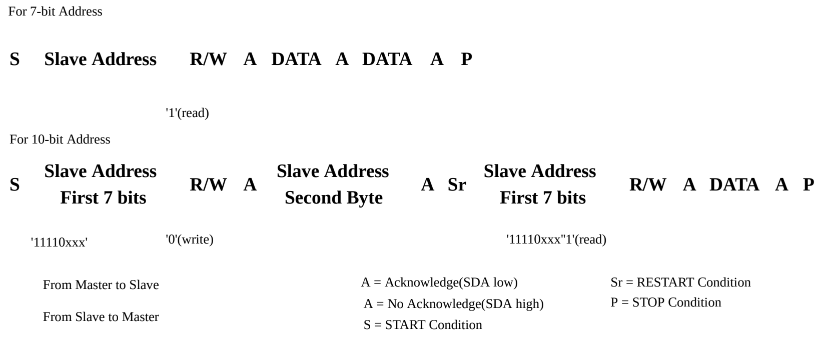


Master receive and slave send

When the host receives data as shown in the figure below, the host must respond to the slave transmitter every time it receives a byte of data, except for the last word

Festival. In this way, the master receiver can inform the slave transmitter whether it is the last byte. The slave transmitter must release when it detects a NACK SDA, so that the host can generate a stop condition.

Figure **152.** Master receiving protocol



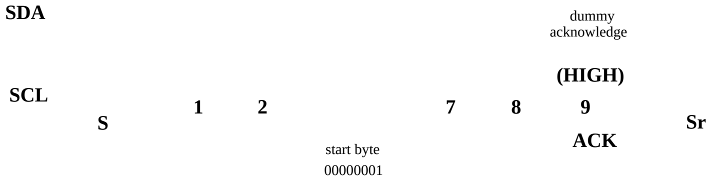
When the host does not want to generate a stop condition and releases the bus, it can generate a repeated start condition. The repeated start condition is the same as the start condition, It's just that it is actually generated after the ACK. Working in the master mode, the I2C interface can use different transmission directions to communicate with the same slave.

18.3.4 Start byte transfer protocol

The start byte transmission protocol is used for systems that do not have a dedicated I2C hardware module. When the I2C module is used as the host, The output start can generate the start byte output for the required slave.

The protocol consists of 7 zeros and one 1 as shown in the figure below. The processor can use low-speed sampling 0 to query the bus during the address phase. one Once 0 is detected, the processor can switch from low-speed sampling to the normal rate of the host.

Figure **153.** Start byte transfer



The start byte program flow is as follows:

1. The host generates an initial condition
2. The host sends the start byte (0000 0001)
3. The host sends an ACK clock pulse (ACK)
4. No slave responds to ACK signal
5. The host generates a repeated start condition (RESTART)

The hardware I2C receiver does not need to respond to the start byte, because this is a reserved address and the address will be reset after RESTART.

18.3.5 Sending buffer management and generation of start, stop and repeated start conditions

When working in master mode, the I2C module generates a stop condition on the bus whenever the transmission is empty. If you repeat the start-to-generate function Enable (IC_RESTART_EN = 1), the repeated start condition will be generated when the transfer direction changes from read to write or write to read. If it is not enabled Repeating the start condition will generate a start condition after the stop condition.

The following figure shows the bits of the IC_DATA_CMD register.

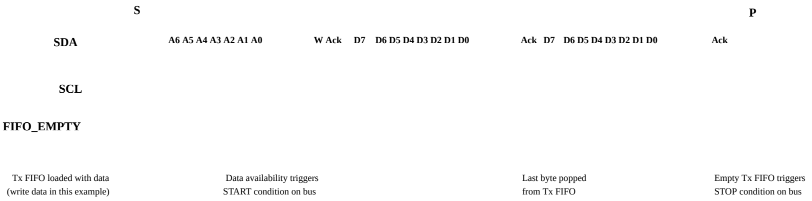
Figure **154. IC_DATA_CMD** register

IC_DATA_CMD
CMD
DATA
8
7
0

DATA —Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field.
CMD —Write-only field; this bit determines whether transfer to be carried out is read (CMD=1) or Write (CMD=0)

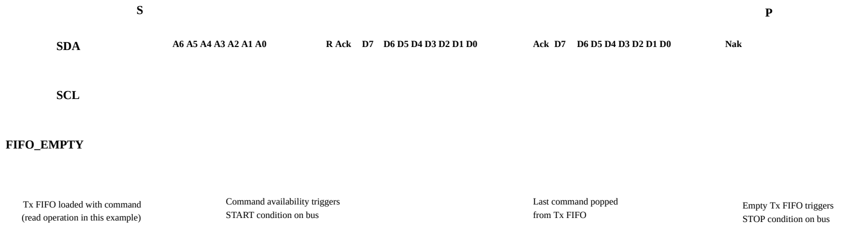
The following timing diagram describes the behavior of the I2C module when the Tx FIFO becomes empty in the main transmit mode.

Figure **155.** Master Transmit- **Tx FIFO** is empty



The following timing diagram describes the behavior of the I2C module when the Tx FIFO becomes empty in the master receiving mode.

Figure **156.** Master Receive- **Tx FIFO** is empty



18.3.6 Multiple host arbitration

The I2C bus is a multi-master bus. Arbitration is a process where multiple masters try to control the bus at the same time, but only one of them is allowed to control the bus. The process of making the message uncorrupted. Once one of the masters has taken control of the bus, then until the master sends a stop condition and

When the bus is released to the idle state, other hosts can control the bus.

When the SCL line is high, arbitration takes place on the SDA line. If two or more masters try to send information to the bus, the other masters

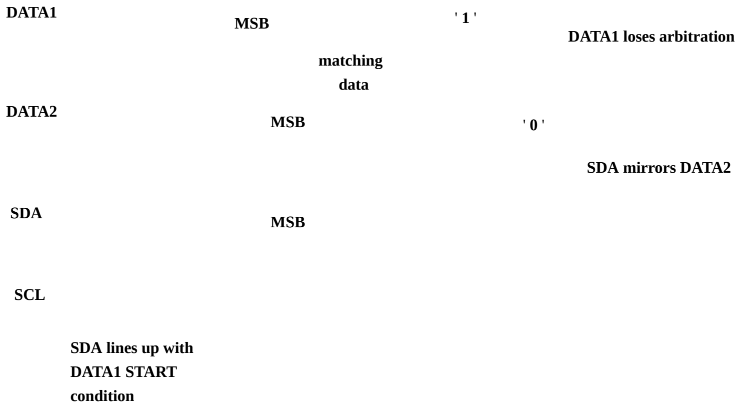
In the case of both "0"s, the host that first generates a "1" will lose the arbitration. The host that loses arbitration can continue to generate clock pulses.

To the end of byte transfer. If every master tries to address the same device, arbitration will continue in the data phase.

After the loss of arbitration is detected, the I2C interface will stop generating the SCL signal.

The following figure shows the bus timing of the arbitration of two masters

Figure **157.** Multiple host arbitration



For the high-speed mode, each host has a unique host code, so the arbitration will not enter the data phase. The 8-bit host code is composed of

The software writes to the high-speed host mode code register (IC_HS_MADDR). Since each host code is different, when transmitting a high-speed host

Only one host after the code can win arbitration.

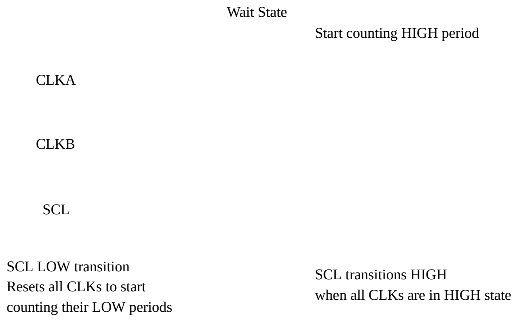
18.3.7 Clock synchronization

When two or more masters try to transmit information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All hosts generate themselves

Clock to transmit messages. Data is only valid at the high level of the clock. Clock synchronization is performed through the "AND" connection of the SCL signal. Be the master The machine changes the SCL clock to 0, and the host calculates the time of the SCL low level, and starts to change the SCL clock to 1 in the next clock cycle. but, If another master keeps SCL as 0, then this master will enter the waiting state until the SCL clock becomes 1.

All hosts will calculate their high level time, and the host with the shortest high level time will change SCL to 0. Next, the host will calculate low Level time, the host with the longest low level time will force other hosts to enter the waiting state. This generates a synchronized SCL clock, such as As shown in the figure below.

Figure **158.** Multiple master clock synchronization


Wait State
Start counting HIGH period
CLKA
CLKB
SCL
SCL LOW transition
Resets all CLKs to start
counting their LOW periods
SCL transitions HIGH
when all CLKs are in HIGH state

18.4 I2C working mode

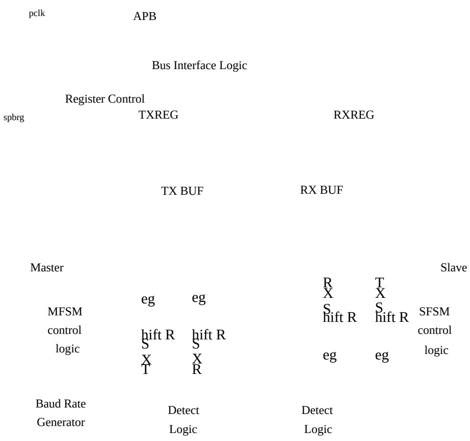
The I2C interface can operate in one of the following four ways:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

Note: The *I2C* interface module can only work in master mode or slave mode, but cannot work in both modes at the same time. So make sure to deposit Is *IC_CON* median . *6* (*IC_SLAVE_DISABLE*) and bit *0* (*IC_MASTER_MODE*) are not set to *0* and *1* (or *1* and *0* respectively).

The I2C functional block diagram is as follows:

Figure **159. I2C** functional block diagram



scls sdai sdaos

sdaom sdai sclm

18.4.1 Slave Mode

The following describes the program flow chart of the slave mode

Initial configuration

1. Write 0 to bit 0 of IC_ENABLE register to disable I2C.
2. Configure the slave address by initializing the IC_SAR register. This address is the address responded by the I2C interface.
3. Configure the IC_CON register to specify the address format (set bit 3 to select 7-bit or 10-bit address format). Write 0 to the register Bit 6 (IC_SLAVE_DISABLE) and write 0 to bit 0 (MASTER_MODE) of the IC_CON register.
4. Write 1 to bit 0 in the IC_ENABLE register to enable the I2C interface module.

Single byte operation from send

When the I2C interface is addressed by other I2C masters and requests data, the I2C interface works in slave sending mode, the steps are as follows:

1. Other I2C master devices initiate I2C transmission, and the sending address matches the slave address in the IC_SAR register.
2. The I2C interface responds to the sent address and recognizes that the direction of the transmission is working in the slave sending mode.
3. The I2C interface generates an RD_REQ interrupt (register IC_RAW_INTR_STAT bit 5) and pulls the SCL line low. bus It remains in the waiting state until the software responds.

If the RD_REQ interrupt is masked (register IC_INTR_MASK[5]=0), it is recommended that the CPU check IC_RAW_INTA_STAT regularly register.

1. Setting bit 5 of IC_RAW_INTR_STAT is equivalent to generating an RD_REQ interrupt.
2. The software must meet the requirements of I2C transmission.
3. The time interval is usually about 10 SCL clock cycles. For example, for 400kb/s, the time interval is 25us.
4. If there is still data in the Tx FIFO before receiving the read request, the I2C interface will generate a TX_ABRT interrupt (IC_RAW_INTR[6]), clear the data in Tx FIFO.
5. The software writes data to the IC_DATA_CMD register (bit 8 is set to 0).
6. The software must first clear the IC_RAW_INTA_STAT register RD_REQ and TX_ABRT interrupts (bit5, 6 respectively)
7. The I2C interface releases SCL and sends data bytes.
8. The host device sends a repeated start condition to control the bus or sends a stop condition to release the bus.

Single byte operation received from

When other host devices address the I2C interface and send data, the I2C interface works in slave receiving mode. The steps are as follows:

1. Other I2C master devices initiate I2C transmission, and the sending address matches the slave address in the IC_SAR register.
2. The I2C interface responds to the sent address and recognizes that the direction of the transmission is working in the slave receiving mode.
3. The I2C interface receives the data sent by the host and stores the data in the receiving buffer.
4. The I2C interface generates RX_FULL interrupt (IC_RAW_INTR_STAT[2]).

If the RX_FULL interrupt is masked (IC_INTR_MASK[2]=0), it is recommended that the software check the IC_STATUS register periodically. read When bit 3 (RFNE) of IC_STATUS register is 1, it is equivalent to RX_FULL interrupt generation.

5. The software obtains the received data by reading bit 7:0 in the IC_DATA_CMD register.
6. The host device sends a repeated start condition to control the bus or sends a stop condition to release the bus.

Block transfer operation from slave

In the standard I2C protocol, all data processing is a single byte processing. The program writes a byte to the Tx FIFO of the slave. Responding to the host's read request. When a slave (slave sending) receives a read request (RD_REQ) from the master (master receiving), there is at least one number

The data is placed in the Tx FIFO sent from the slave. This I2C interface module can handle multiple data in the Tx FIFO, so the next read request does not need To generate an interrupt to fetch data. Ultimately, this greatly reduces the waiting time caused by each data interruption.

This mode only exists when the I2C interface is used as a slave sending mode. If the master sends a response to the data transmitted from the send, the slave TX FIFO Without data, the I2C interface will pull down the SCL line of the I2C bus until the read request interrupt (RD_REQ) is generated and the TX FIFO data is ready Release the SCL line after completion.

If the RX_REQ interrupt is masked (IC_INTR_STAT[5]=0), the software can query and read the IC_RAW_INTR_STAT register periodically. Memory. When IC_RAW_INTR_STAT[5] is read, returning 1 is equivalent to generating an RX_REQ interrupt.

The RD_REQ interrupt is generated by a read request and must be cleared when exiting the interrupt service routine (ISR) like an interrupt. In the interrupt service routine Intermediate (ISR) can write one or more bytes of data to TX FIFO. During the transmission of these bytes to the host, if the host responds to the most After the next byte, the slave will have to generate an RD_REQ interrupt request again. This is because the host requires more data.

If the master has received n bytes from the I2C interface, but the number of data written by the program into the Tx FIFO is greater than n, the slave needs to finish After the requested n bytes of data are sent, the Tx FIFO will be cleared and the extra bytes will be ignored.

18.4.2 Main Mode

Initial configuration

1. Disable the I2C interface by setting IC_ENABLE [0]=0
2. Configure bit 2: 1 of the IC_CON register to set the rate mode (standard mode, fast mode, and high-speed mode) of I2C work. Also make sure that bit 6 (IC_SLAVE_DIASBLE) is 1, and bit 0 (MASTER_MODE) is 1.
3. Write the I2C device address to the IC_TAR register. Setting this register can be configured as a broadcast address or start byte command.
4. Only for high-speed mode transmission: write data to IC_HS_MADDR to configure the I2C interface host code. The host code is determined by the software itself righteous.
5. Set IC_ENABLE[0] to enable the I2C interface.
6. Write the transferred data and transfer direction into the IC_DATA_CMD register. If configured before enabling the I2C interface IC_DATA_CMD register, data and commands will be lost, this is because the buffer is cleared when the I2C interface is disabled.

The above steps will cause the I2C interface to generate a start condition and send the address byte data to the I2C bus.

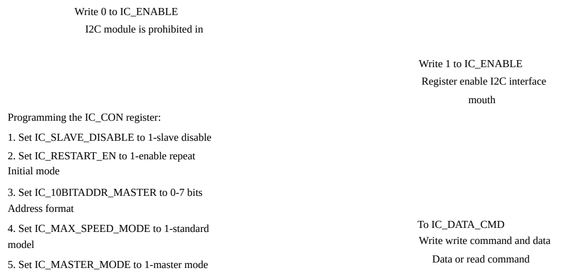
Main send and main receive

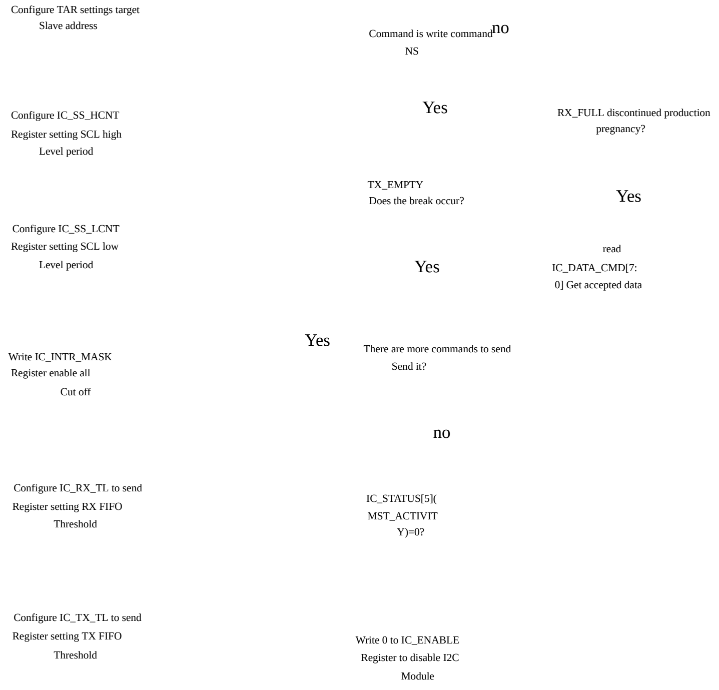
The I2C interface supports dynamic switching of reading and writing. When sending data, write data to the low byte of I2C RX/TX data buffer and command register In (IC_DATA_CMD), configure the CMD bit to 0 to generate a write operation. The next read command does not need to set the IC_DATA_CMD register Only need to ensure that the CMD bit is 1. If the transmit FIFO is empty, the I2C module pulls down SCL until the next command is written to transmit FIFO.

Program flow chart

The following flow chart is an example of a program using the I2C interface as a host:

Figure **160. I2C** interface host flow chart


Write 0 to IC_ENABLE
I2C module is prohibited in
Write 1 to IC_ENABLE
Register enable I2C interface
mouth
Programming the IC_CON register:
1. Set IC_SLAVE_DISABLE to 1-slave disable
2. Set IC_RESTART_EN to 1-enable repeat
Initial mode
3. Set IC_10BITADDR_MASTER to 0-7 bits
Address format
4. Set IC_MAX_SPEED_MODE to 1-standard
model
5. Set IC_MASTER_MODE to 1-master mode
To IC_DATA_CMD
Write write command and data
Data or read command



18.4.3 I2C abort transmission

The ABRT control bit in the IC_ENABLE register allows the software to abandon the I2C bus before finishing transmitting the command in the TX FIFO. As
In response to the ABORT request, the I2C module sends a stop condition to the I2C bus and clears the TX FIFO at the same time. The abort transfer operation value is allowed in Main mode.

Procedure flow chart

1. Stop writing new commands to Tx FIFO (IC_DATA_CMD)

2. If working in DMA mode, set TDMAE=0 to prohibit sending DMA.
3. Set the ABRT bit of the IC_ENABLE register to 1
4. Wait for TX_ABRT interrupt

18.5 Communication using **DMA**

The I2C interface supports DMA to send and receive data. DMA can be turned on separately by setting the corresponding bit in the IC_DMA_CR register
Send or DMA receive. When the data register becomes empty when sending or the data register becomes full when receiving, a DMA request is generated. DMA request must Must be responded before the end of the current byte transmission.

Use **DMA to** send

The DMA transmission mode can be activated by setting the TDMAE bit in the IC_DMA_CR register. After allocating DMA channels for I2C, when
When sending data, the DMA controller will load the data from the preset storage area into the IC_DATA_CMD register.

Use **DMA to** receive

The DMA receiving mode can be activated by setting the RDMAE bit in the IC_DMA_CR register. After allocating DMA channels for I2C, when
Each time a data byte is received, the DMA controller will transfer the data from the IC_DATA_CMD register to the preset storage area.

18.6 I2C interrupt

The following table lists the interrupt bits of I2C and their setting and clearing methods. Some bits are set by hardware and cleared by software; other bits
Set and cleared by hardware.

Table **34.** Setting and clearing of interrupt bits

Interrupt bit	Hardware set/software clear	Hardware set and clear
GEN_CALL	√	x
START_DET	√	x
STOP_DET	√	x
ACTIVITY	√	x

RX_DONE	√	x
TX_ABRT	√	x
RD_REQ	√	x
TX_EMPTY	x	√
TX_OVER	√	x
RX_FULL	x	√
RX_OVER	√	x
RX_UNDER	√	x

The following figure describes the operation of the interrupt bit set by hardware and cleared by software in the interrupt register

Figure **161.** Interrupt mechanism

18.7 I2C register description

18.7.1 I2C Control Register (IC_CON)

Offset address: 0x00

Reset value: 0x0011

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve							TX_EMPTY_CTRL	STOP_DET_IFADDRESSED	IC_SLAVE_DISABLE	IC_RESTART_EN	IC_10BITADDR_MASTER	IC_10BITADDR_SLAVE	SPEED		MASTER_MODE
							rw	rw	rw	rw	r	rw	rw	rw	rw

Bit 31: 9	Reserved, always read as 0
Bit 8	**TX_EMPTY_CTRL** : This bit controls TX_EMPTY interrupt generation. For details, refer to IC_RAW_INTR_STAT register. (This bit controls the generation of the TX_EMPTY interrupt, as described in the IC_RAW_INTR_STAT register.)
Bit 7	**STOP_DET_IFADDRESSED** : In slave mode, whether to generate STOP_DET interrupt. 1- STOP_DET interrupt is generated when the address matches (In the slave mode, 1'b1-issues the STOP_DET interrrupt only when it is addressed) 0-Regardless of whether the address matches, a STOP_DET interrupt (In the slave mode, 1'b0 – issues the STOP_DET irrespective of whether it's addressed or not)

This bit only applies to slave mode

Note: When the broadcast address is addressed, if this bit is set, the slave will not generate a STOP_DET interrupt. STOP_DET interrupt only when sending

It is generated when the address matches the slave address.

Bit 6	**IC_SLAVE_DISABLE** : This bit controls whether I2C has its slave disabled) 0: slave enable 1: Slave is prohibited
Bit 5	**IC_RESTART_EN** : When acting as a host, this bit controls whether to send RESTART conditions (Determines whether RESTART conditions may be sent when acting as a master) 0: prohibited 1: enable When RESTART is disabled, the following functions cannot be performed when the I2C interface is used as a host: Send start byte Perform work in high-speed mode Change transmission direction in combined format mode 10-bit address format read operation Replacing the RESTART condition is to send the stop condition first and then the start condition. If the above operation is executed, it will be set Bit 6 of the IC_RAW_INTR_STAT register (TX_ABRT)
Bit 4	**IC_10BITADDR_MASTER** : I2C address format when acting as a host (Address mode when acting as a master) 0: 7-bit address format 1: 10-bit address format
Bit 3	**IC_10BITADDR_SLAVE** : When acting as a slave, this bit controls the response to a 10-bit or 7-bit address (When acting as a slave, this bit controls whether the DW_apb_i2c responds to 7- or 10-bit addresses) 0: 7-bit addressing address. The I2C interface ignores handling 10-bit addressing. For 7-bit addressing, only compare the low of IC_SAR register 7 bits. 1: 10-bit addressing address. I2C only responds to 10-bit addressing, the receiving address is compared with the 10-bit IC_SAR
Bit 2: 1	**SPEED** : These bits control at which speed the DW_apb_i2c operates) This setting is only valid when the I2C interface is working in host mode. 1: Standard mode (0~100Kb/s) 2: Fast mode (≤400Kb/s) 3: High-speed mode (≤3.4Mb/s)
Bit 0	**MASTER_MODE** : This bit controls whether the DW_apb_i2c master is enabled) 0: Host prohibited 1: Host enable

The IC_SLAVE_DISABLE (bit6) and MASTER_MODE (bit0) configurations are listed in the following table

Table **35. IC_SLAVE_DISABLE** (**bit 6**) and **MASTER_MODE** (**bit 0**) configuration

IC_SLAVE_DISABLE (IC_CON[6])	MASTER_MODE IC_CON[0]	state
0	0	Slave device
0	1	Configuration error
1	0	Configuration error
1	1	Host device

18.7.2 I2C Target Address Register (**IC_TAR**)

Offset address: 0x04

Reset value: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve				SPEC IAL	GC_O R_ST ART	IC_TAR[9:0]									
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 12	Reserved, always read as 0
Bit 11	**SPECIAL** : This bit indicates whether the software is executing a special command (broadcast call or start byte command) (This bit indicates whether software performs a General Call or START BYTE command) 0: Ignore bit 10 GC_OR_START, use IC_TAR bit normally 1: Execute special I2C commands as described by GC_OR_START bit
Bit 10	**GC_OR_START** : If bit 11 is set, this bit shows whether I2C is performing a general call or start byte (If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the DW_apb_i2c) 0: General call address. Only write operations can be performed when sending a general call address. The I2C interface has been working in the broadcast address mode. The value to SPECIAL (bit11) is cleared. 1: Start byte command
Bit 9:0	**IC_TAR** : The target address of the main operation (This is the target address for any master transaction) When sending a broadcast address, these bits can be ignored. To generate a start byte command, the CPU only needs to write to these bits once.

18.7.3 I2C Slave Address Register (**IC_SAR**)

Offset address: 0x08

Reset value: 0x55

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

Bit 31: 12	Reserved, always read as 0
	IC_SAR : When the I2C interface is working in slave mode, these storage slave addresses (The IC_SAR holds the slave

Bit 9:0	address when the I2C is operating as a slave) For the 7-bit address format, only IC_SAR[6:0] is valid.

18.7.4 **I2C** high-speed mode host code address register (**IC_HS_MADDR**)

Offset address: 0x0C

Reset value: 0x1

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

Bit 31: 3	Reserved, always read as 0
Bit 2: 0	**IC_HS_MAR** : **the I** the high-speed mode of the host code 2C (This holds Field The 'bit The value of the I 2 C Master the HS MODE code) The host code of HS mode is reserved for 8 bits (00001xxx), this host code is not for slave addressing or other functions. Every host Has its own independent host code. Up to 8 hosts in I2C high-speed mode can be connected to the same I2C bus. The valid value is 0~7. When the I2C interface is working in standard (1) or fast (2) mode, reading this bit will return 0.

18.7.5 **I2C** Data Command Register (**IC_DATA_CMD**)

Offset address: 0x10

Reset value: 0x1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve							CMD	DAT[7:0]							
							w	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 9	Reserved, always read as 0

Bit 8	**CMD** : This bit controls whether a read or a write is performed) 1: Read 0: write When a command enters the TX FIFO, this bit is used to distinguish read and write commands. In the slave receiving mode, the write operation of this bit is ignored. In the slave sending mode, write 0 to indicate that the data in the IC_DATA_CMD register is ready to be sent.
Bit 7:0	**DAT** : The data to be transmitted or received on the I2C bus (This register contains the data to be transmitted or received on the I 2 C bus)

18.7.6 Standard Mode **I2C** Clock High **Level** Count Register (**IC_SS_SCL_HCNT**)

Offset address: 0x14

Reset value: 0x190

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

Bit 31: 16	Reserved, always read as 0
Bit 15:0	**IC_SS_SCL_HCNT** : SCL clock high level period in I2C interface standard mode (This register sets the SCL clock high-period count for standard speed) Note: The configurable value of this register is between 6 and 65525. This is because the I2C interface uses a 16-bit timer. When the counter value is equal to IC_SS_SCL_HCNT+10, it indicates that the I2C bus is in an idle state.

18.7.7 Standard Mode **I2C** Clock Low Level Count Register (**IC_SS_SCL_LCNT**)

Offset address: 0x18

Reset value: 0x1D6

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

Bit 31: 16	Reserved, always read as 0
Bit 15:0	**IC_SS_SCL_LCNT** : SCL clock low period in I2C interface standard mode (This register sets the SCL clock low period count for standard speed) The minimum value is 8.

18.7.8 Fast mode **I2C** clock high level count register (**IC_FS_SCL_HCNT**)

Offset address: 0x1C

Reset value: 0x036

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

Bit 31: 16	Reserved, always read as 0
Bit 15:0	**IC_FS_SCL_HCNT** : SCL clock high period in I2C interface fast mode (This register sets the SCL clock high-period count for fast mode or fast mode plus) Used to send host code and start byte or broadcast address in high-speed mode. When I2C works in standard mode, this register is read-only and the return value is 0. The minimum value is 6.

18.7.9 Fast Mode **I2C** Clock Low Count Register (**IC_FS_SCL_LCNT**)

Offset address: 0x20

Reset value: 0x082

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

Bit 31: 16	Reserved, always read as 0
Bit 15:0	**IC_FS_SCL_LCNT** : SCL clock low period in I2C interface fast mode (This register sets the SCL clock low period count for fast mode or fast mode plus) At the same time, it is used to send host code or start byte or broadcast address in high-speed mode. When I2C works in standard mode, this register is read-only and the return value is 0. The minimum value is 8.

18.7.10 High-speed mode **I2C** clock high level count register (**IC_HS_SCL_HCNT**)

Offset address: 0x24

Reset value: 0x006

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

Bit 31: 16	Reserved, always read as 0

Bit 15:0	**IC_HS_SCL_HCNT** : SCL clock high period in I2C interface high-speed mode (This register sets the SCL clock high period count for high speed) SCL high level sequence is related to the load of the bus. For example, 100pF load, SCL high level duration is 60ns. For 400pF Load, the duration of SCL high level is 120ns. When I2C is working in high-speed mode, this register is read-only and the return value is 0. The minimum value is 6.

18.7.11 High-speed mode **I2C** clock low count register (**IC_HS_SCL_LCNT**)

Offset address: 0x28

Reset value: 0x010

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

Bit 31: 16	Reserved, always read as 0
Bit 15:0	**IC_HS_SCL_LCNT** : SCL clock low period in I2C interface high-speed mode (This register sets the SCL clock low period count for high speed) SCL low time sequence is related to the load of the bus. For example, 100pF load, SCL low level duration is 160ns. For 400pF Load, SCL low level duration is 320ns. When I2C is working in high-speed mode, this register is read-only and the return value is 0. The minimum value is 8.

18.7.12 I2C Interrupt Status Register (**IC_INTR_STAT**)

Offset address: 0x2C

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve		D	R_RESTART_DET	R_GEN_CALL	R_START_DET	R_STOP_DET	R_ACTIVITY	R_RX_DONE	R_TX_ABRT	R_RD_REQ	R_TX_EMPTY	R_TX_OVER	R_RX_FULL	R_RX_OVER	R_RX_UNDER
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31: 16	Reserved, always read as 0
Bit 15:0	For specific description of each bit, please refer to IC_RAW_INTR_STAT register (See “IC_RAW_INTR_STAT” for a detailed description of these bits)

18.7.13 **I2C** interrupt mask register (**IC_INTR_MASK**)

Offset address: 0x30

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve				M_GEN_CALL	M_START_DET	M_STOP_DET	M_ACTIVITY	M_RX_DONE	M_TX_ABRT	M_RD_REQ	M_TX_EMPTY	M_TX_OVER	M_RX_FULL	M_RX_OVER	M_RX_UNDER
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 12	Reserved, always read as 0
Bit 11:0	Each bit shields the corresponding bit of IC_INTR_STAT. (These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register)

18.7.14 **I2C RAW** Interrupt Register (**IC_RAW_INTR_STAT**)

Offset address: 0x34

Reset value: 0x000

The difference between IC_RAW_INTR_STAT and IC_INTR_STAT is that the former will not be masked.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve				GEN_CALL	START_DET	STOP_DET	ACTIVITY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31: 16	Reserved, always read as 0
Bit 11	**GEN_CALL** : General call Set when the general call address is received. Disable the I2C interface or clear it when the CPU reads the IC_CLR_GEN_CALL register. I2C stores the received data in the receiving Buffering.
Bit 10	**START_DET** : Start condition detection Regardless of whether the I2C interface is working in the master or slave, this bit is set once the start or repeated start conditions on the I2C interface are detected
Bit 9	**STOP_DET** : Stop condition detection (Stop condition detection) The status of this bit is based on the status of STOP_DET_IFADDRESSED in the IC_CON register When STOP_DET_IFADDRESSED=0 Regardless of whether the I2C interface is working in the master or slave, this bit is set once a stop condition on the I2C interface is detected. In slave mode, A STOP_DET interrupt will be generated regardless of whether the addressing matches or not When STOP_DET_IFADDRESSED=1 In master mode (MASTER_MODE=1), this bit shows whether a stop condition occurs in the I2C interface In the slave mode (MASTER_MODE=0), a STOP_DET interrupt is generated only when the slave address matches successfully.

Bit 8	**ACTIVITY** : I2C interface is activated, this bit is used to capture the activity status of the I2C module (This bit captures DW_apb_i2c activity and stays set until it is cleared) After being set, it can only be cleared in the following four ways: Disable I2C interface Read IC_CLR_ACTIVITY register Read IC_CLR_INTR register System reset Once set, it can only be cleared by the above method. Even if I2C is in an idle state, the bit will remain high until it is cleared.
Bit 7	**RX_DONE** : From the end of the transmission (T ransmit done) When I2C is used as a slave to send, if the host does not respond after sending a byte of data, this bit will be set.

	This situation occurs in the last byte of the transfer, indicating the end of the transfer.
Bit 6	**TX_ABRT** : Transmit abort When the I2C interface is used as a transmitter, it is set when the data in the buffer cannot be sent. Note: Abort sending will clear the receiving and sending buffers in the I2C interface. The send buffer will be in a refresh state until read IC_CLR_TX_ABRT register. Once the read operation is performed, the sender can receive new data on the APB bus.
Bit 5	**RD_REQ** : Read request When I2C is used as a slave, it is set when other masters try to read data from the I2C interface. The I2C interface keeps the bus in a waiting state (SCL=0) until the interrupt is processed. This means that the I2C interface is The other host succeeded in addressing and requested to send data. The processor must respond to the interrupt and write data to the IC_DATA_CMD register. In the memory. This bit is cleared when the processor reads the IC_CLR_RD_REQ register.
Bit 4	**TX_EMPTY** : Transmit buffer empty (Transmit Buffer empty) The status of this bit depends on the TX_EMPTY_CTRL status in the IC_CON register: When TX_EMPTY_CTRL=0, set when the transmit buffer is empty Set when TX_EMPTY_CTRL=1, the transmit buffer is empty and the internal shift register ends When the sending buffer is not empty, it is automatically cleared by hardware.
Bit 3	**TX_OVER** : Transmit Buffer overload (Transmit Buffer over) Set when the send buffer is full when the processor writes new data and causes an overflow.
Bit 2	**RX_FULL** : Receive buffer not empty Set when the receive buffer is not empty. It is cleared by hardware when the receive buffer is empty.
Bit 1	**RX_OVER** : Receive buffer over Set when the receive buffer is full and new data is received. At this time, the I2C interface will respond, but the new data will be lost.
Bit 0	**RX_UNDER** : Receive buffer under Set when the processor reads the IC_DATA_CMD register when the RX FIFO is empty.

18.7.15 I2C receive threshold (IC_RX_TL)

Offset address: 0x38

Reset value: 0x000

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

Bit 31: 8	Reserved, always read as 0
Bit 7:0	**RX_TL[7 : 0]** : Receive FIFO threshold level Control RX_FULL interrupt triggering.

18.7.16 I2C transmit threshold (IC_TX_TL)

Offset address: 0x3C

Reset value: 0x000

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

Bit 31: 8	Reserved, always read as 0
Bit 7:0	**TX_TL[7 : 0]** : Transmit FIFO threshold level Control TX_EMPTY interrupt triggering.

18.7.17 I2C combination and independent interrupt clear register (IC_CLR_INTR)

Offset address: 0x40

Reset value: 0x000

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

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_INTR** : Reading this register will clear all combined interrupts and independent interrupts (Read this register to clear the combined interrupt, all individual interrupts) This bit does not clear interrupts that can be automatically cleared by hardware, only clearing software can clear interrupts.

18.7.18 I2C clear RX_UNDER interrupt register (IC_CLR_RX_UNDER)

Offset address: 0x44

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ RX_U NDER r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_RX_UNDER** : Read this register to clear the RX_UNDER interrupt (IC_RAW_INTR_STAT[0]) (Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register)

18.7.19 I2C clear RX_OVER interrupt register (IC_CLR_RX_OVER)

Offset address: 0x48

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ RX_ OVER r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_RX_OVER** : Read this register to clear the RX_OVER interrupt (IC_RAW_INTR_STAT[1]) (Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register)

18.7.20 I2C clear TX_OVER interrupt register (IC_CLR_TX_OVER)

Offset address: 0x4C

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ TX_ OVER
															r

Bit 31:1	Reserved, always read as 0

Page 309

Bit 0	**CLR_TX_OVER** : Read this register to clear the TX_OVER interrupt (IC_RAW_INTR_STAT[3]) (Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register)

18.7.21 I2C clear RD_REQ interrupt register (IC_CLR_RD_REQ)

Offset address: 0x50

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ RD_ REQ
															r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_RD_REQ** : Read this register to clear the RD_REQ interrupt (IC_RAW_INTR_STAT[5]) (Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register)

18.7.22 I2C clear TX_ABRT interrupt register (IC_CLR_TX_ABRT)

Offset address: 0x54

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ TX_ ABRT
															r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_TX_ABRT** : Read this register to clear the TX_ABRT interrupt (IC_RAW_INTR_STAT[6]) (Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register) At the same time, the TX FIFO is released from the refresh/reset state in order to receive the written data.

Page 310

18.7.23 I2C clear RX_DONE interrupt register (IC_CLR_RX_DONE)

Offset address: 0x58

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ RX_ DONE
															r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_RX_DONE** : Read this register to clear the RX_DONE interrupt (IC_RAW_INTR_STAT[7]) (Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register)

18.7.24 I2C clears the ACTIVITY interrupt register (IC_CLR_ACTIVITY)

Offset address: 0x5C

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ ACTIV ITY
															r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_ACTIVITY** : If the I2C bus is not active, read this register to clear the ACTIVITY interrupt (IC_RAW_INTR_STAT[8]) (Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore) If I2C is still active, the ACTIVITY interrupt will continue to be set. When the I2C module is disabled or the I2C bus is no longer active This bit is cleared by hardware. The ACTIVITY (bit 8) in IC_RAW_INTR_STAT can be obtained by reading this register state.

Page 311

18.7.25 I2C clears the STOP_DET interrupt register (IC_CLR_STOP_DET)

Offset address: 0x60

Reset value: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															CLR_ STOP

_DET

r

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_STOP_DET** : Read this register to clear the STOP_DET interrupt (IC_RAW_INTR_STAT[9]) (Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register)

18.7.26 I2C clears the **START_DET** interrupt register (**IC_CLR_START_DET**)

Offset address: 0x64

Reset value: 0x000

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

Bit 31:1	Reserved, always read as 0
Bit 0	**CLR_START_DET** : Read this register to clear the START_DET interrupt (IC_RAW_INTR_STAT[10]) (Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register)

18.7.27 I2C clear **GEN_CALL** interrupt register (**IC_CLR_GEN_CALL**)

Offset address: 0x68

Reset value: 0x000

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

Bit 31:1	Reserved, always read as 0

Bit 0	**CLR_GEN_CALL** : Read this register to clear the GEN_CALL interrupt (IC_RAW_INTR_STAT[11]) (Read this register to clear the GEN_CALL interrupt (bit 11) of IC_RAW_INTR_STAT register)

18.7.28 I2C Enable Register (**IC_ENABLE**)

Offset address: 0x6C

Reset value: 0x000

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

Bit 31:1	Reserved, always read as 0
Bit 1	**ABORT** : I2C transfer abort (I2C transfer abort) 0: The suspension did not occur or has ended 1: Abort operation is in progress When the I2C module is set as the host, the I2C transmission can be stopped by software. Once set, it cannot be cleared immediately. I2C module after set The control logic will generate a STOP condition and clear the send buffer after the current transfer is completed, and then generate after the operation is aborted TX_ABRT interrupt.

	The ABRT bit will be automatically cleared after the abort operation is over.
Bit 0	**ENABLE** : I2C module enable (I2C mode enable) 0: Disable I2C module (send and receive buffers are kept in erased state) 1: Enable I2C module

18.7.29 I2C Status Register (**IC_STATUS**)

Offset address: 0x70

Reset value: 0x006

This register is read-only and indicates the current transmission and buffering status. The status bit does not generate interrupts.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve									SLV_ACTIVITY	MST_ACTIVITY	RFF	RFNE	TFE	TFNF	ACTIVITY
									r	r	r	r	r	r	r

Bit 31: 7	Reserved, always read as 0
Bit 6	**SLV_ACTIVITY** : Slave FSM activity status bit (Slave FSM activity status) 0: The slave state machine is in the IDLE state, so the I2C slave part is not active 1: The slave state machine is not in the IDLE state, so the I2C slave is partially active

Bit 5	**MST_ACTIVITY** : Master FSM activity status bit (Master FSM activity status) 0: The host state machine is in the IDLE state, so the I2C host part is not active 1: The host state machine is not in the IDLE state, so the I2C host is partially active
Bit 4	**RFF** : Receive FIFO completely full 0: The receiving buffer is not full 1: The receiving buffer is full
Bit 3	**RFNE** : Receive FIFO not empty 0: receive buffer empty 1: The receive buffer is not empty
Bit 2	**TFE** : Transmit FIFO completely empty 0: The sending buffer is not empty 1: Send buffer empty
Bit 1	**TFNF** : Transmit FIFO not full 0: send buffer full 1: Send buffer is not full
Bit 0	**ACTIVITY** : I2C bit activity status (I2C activity status) The result of ORing the MST_ACTIVITY bit and the SLV_ACTIVITY bit.

18.7.30 I2C transmit buffer level register (**IC_TXFLR**)

Offset address: 0x74

Reset value: 0x006

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

Bit 31: 2	Reserved, always read as 0
Bit 1: 0	**TXFLR** : The number of valid data in the transmit buffer (0～2) (Transmit FIFO level. Contains the number of valid data entries in the transmit FIFO)

18.7.31 I2C Receive Buffer Level Register (**IC_RXFLR**)

Offset address: 0x78

Reset value: 0x006

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

Bit 31: 2	Reserved, always read as 0

Bit 1: 0	**RXFLR** : The number of valid data in the receive buffer (0～2) (Receive FIFO level. Contains the number of valid data entries in the receive FIFO)

18.7.32 I2C SDA Hold Time Register (IC_SDA_HOLD)

Offset address: 0x7C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve								IC_SDA_RX_HOLD							
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC_SDA_TX_HOLD															
														r	r

Bit 31: 24	Reserved, always read as 0
Bit 23: 16	**IC_SDA_RX_HOLD** : When the I2C device is used as a receiver, the SDA hold time, the unit is APB1 system clock cycle (Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a reciever)
Bit 15:0	**IC_SDA_TX_HOLD** : When the I2C device is used as a transmission, the SDA hold time, the unit is APB1 system clock cycle (Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a transmitter)

18.7.33 I2C DMA control register (IC_DMA_CR)

Offset address: 0x88

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve														TDMA E	RDMA E
														rw	rw

Bit 31: 2	Reserved, always read as 0
Bit 1	**TDMAE** : Transmit DMA enable (Transmit DMA enable) 0: Send DMA prohibited 1: Send DMA enable
Bit 0	**RDMAE** : Receive DMA enable (Receive DMA enable) 0: Receiving DMA is disabled 1: Receive DMA enable

18.7.34 I2C SDA Setup Time Register (IC_SDA_SETUP)

Offset address: 0x94

Reset value: 0x0064

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

Bit 31: 8	Reserved, always read as 0
Bit 7:0	**SDA_SETUP** : SDA setup time (SDA setup) If required, the delay time is recommended to be 1000ns, and when the APB1 clock frequency is 10MHZ, it is recommended that this register be set to 11. Should send The minimum value of the register is 2

18.7.35 I2C General Call ACK Register (IC_ACK_GENERAL_CALL)

Offset address: 0x98

Reset value: 0x0001

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve															ACK_ GEN_ CALL
															rw

Bit 31:1	Reserved, always read as 0
Bit 0	**ACK_GEN_CALL** : ACK (ACK general call) 1: Respond to ACK after receiving a broadcast call. 0: No response after receiving a broadcast call, and no interrupt is generated.

19. I2S

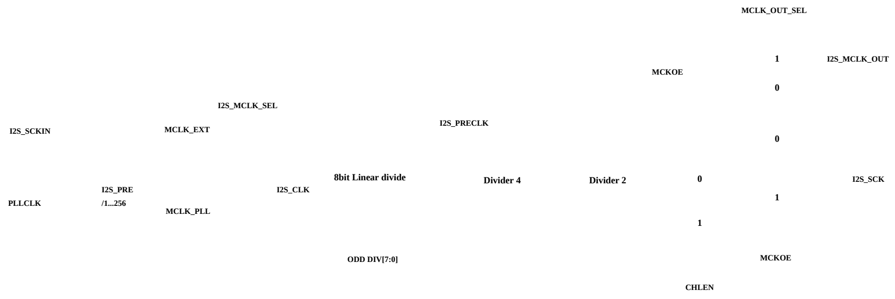
19.1 I2S features

- AMBA 2.0 APB bus interface
- Only support main mode operation, provide SCK and WS, send or receive SD

- The sampling rate of each channel can be configured as 16bit or 24bit, and the length of a single channel only supports 32bit
- Support audio stereo (stereo) or mono (mono) sending and receiving
- Support continuous or non-continuous sending and receiving
- Support left and right alignment
- Only need to select and configure one channel in mono mode
- SD, WS latch can be configured to be on the rising or falling edge of SCK
- The data format can be switched between delay (Philips mode) and non-delay (non-Philips mode)
- FIFO data empty or full can trigger an interrupt
- The embedded TX_FIFO and RX_FIFO are 32x16bit

19.2 I2S function

19.2.1 I2S clock generator



19.2.2 I2S data format

When the data format is 0 and the sampling rate is 16bit, the data format is as follows:

(1) Mono data, for one channel

31 16 15 0

Second 16bit mono channel data	First 16bit mono channel data
Fourth 16bit mono channel data	Third 16bit mono channel data

(2) Stereo data, for two channel

31 16 15 0

First 16bit channel2 data	First 16bit channel1 data
Second 16bit channel2 data	Second 16bit channel1 data

When the data format is 1 and the sampling rate is 16bit, the data format is as follows:

(1) Mono data, for one channel

31 16 15 0

16'h0	First 16bit mono channel data
16'h0	Second 16bit mono channel data

(2) Stereo data, for two channel

31 16 15 0

16'h0	First 16bit channel1 data
16'h0	First 16bit channel2 data
16'h0	Second 16bit channel1 data
16'h0	Second 16bit channel2 data

Note: When the sampling rate is 24bit, the data format is similar to 16bit

19.2.3 I2S timing information

(1) I2S bus with delay mode, left alignment (lr_align=0) Note: N is 16 or 24.

(2) I2S bus without delay mode, left alignment (lr_align=0) Note: N is 16 or 24.

(3) **I2S bus with delay mode, right alignment (lr_align=1) Note: N is 16 or 24.**

(4) I2S bus without delay mode, right alignment (lr_align=1) Note: N is 16 or 24.

19.3 I2S register description:

19.3.1 I2S write data register (I2S_WR)

Register description: I2S Write Data Register

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
I2S_WR															

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2S_WR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	I2S_WR: Write Data to I2S

19.3.2 I2S Receive Enable Register (I2S_RD)

Register description: I2S Read Data Register

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
I2S_RD															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2S_RD															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31:0	I2S_RD: Read Data from I2S

19.3.3 I2S Status Register (I2S_CSR)

Register description: I2S Current Status Register

Address offset: 0x8

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													TX FULL	RX AVL	TX EPT
													r	r	r

Bit 31: 3	Reserve
Bit 2	TXFULL: Transmission FIFO full status bit 1: Transmission FIFO is full 0: Transmission FIFO is not full
Bit 1	RXAVL: received valid data status bit 1: There is valid data in the receive FIFO 0: The receive FIFO is empty
Bit 0	TXEPT: Transmission empty status bit 1: Transmission FIFO is empty 0: Transmission FIFO is not empty

19.3.4 **I2S** Global Control Register (**I2S_GCR**)

Register description: I2S Global Control Register

Address offset: 0xC

Reset value: 0x0001 0011

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
Reserve														nFs_sel	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
txfifo _flush	rxfifo _flush	i2sen	dma mode	int_en	fill datas	txen	rxen	Reserve		txtlf		Reserve		rxtlf	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Bit 31: 18 Reserve

Bit 17: 16
nFs_sel: Fs and MCLK configuration
00: MCLK=128Fs
01: MCLK=256Fs
10: MCLK=512Fs
11: MCLK=1024Fs
Note: SCK=64Fs, this configuration is only useful when clk_bypass=1

Bit 15
txfifo_flush: txfifo refresh
1: Refresh txfifo
0: txfifo works normally

Bit 14
rxfifo_flush: rxfifo flush
1: Refresh rxfifo
0: rxfifo works normally

Bit 13
i2sen: I2S enable
1: Enable I2S
0: Disable I2S

Bit 12
dmamode: DMA mode selection
1: Enable DMA mode
0: Disable DMA mode

Bit 11
int_en: interrupt enable
1: Enable I2S interrupt
0: Disable I2S interrupt

Bit 10
filldatas: fill data selection when transmission is under load
1: Transfer previous data
0: Transmit all 0s

Bit 9
txen: TXFIFO enable
1: Enable TXFIFO
0: Disable TXFIFO

Bit 8
rxen: RXFIFO enable
1: Enable RXFIFO
0: Disable RXFIFO

Bit 7: 6 Reserve

Bit 5: 4	txtlf: TXFIFO trigger depth 01: Less than or equal to 8 words of valid data in TXFIFO other: TXFIF is not full When DMA is enabled, txtlf is set to 01
Bit 3: 2	Reserve
Bit 1: 0	rxtlf: RXFIFO trigger depth 01: 8 words or more valid data in RXFIFO other: RXFIF is not empty When DMA is enabled, rxtlf is set to 01

19.3.5 I2S Data Format Register (**I2S_DFR**)

Register description: I2S Data Format Register

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							data_ format	lr_ align	width	ch_switch		delay m	edgem	wssel	stereo
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 9	Reserve
Bit 8	data_format: data format transmitted or received by I2S 1: The memory is non-continuous valid audio data. When width=0, the low 16-bit data of a word is valid. When width=1, the lower 24 bits of a word are valid, and the other bits are filled with 0. 0: continuous valid audio data in the memory Note: Please refer to section 1.22 for specific data
Bit 7	lr_align: alignment 1: Right aligned 0: Left justify
Bit 6	width: sampling rate selection 1: 24bit 0: 16bit

Bit 5: 4	ch_switch: transmission or reception channel selection 00: No channel is valid 01: The first channel is valid 10: The second channel is valid 11: Dual channel is valid Note: Invalid channel transmission or reception all 0
Bit 3	delaym: Delay mode selection 1: SD and WS change at the same time (Non-Philips mode) 0: SD is delayed by one SCK (Philips mode) compared to WS
Bit 2	edgem: Edge mode selection 1: SD, WS change on the rising edge of SCK 0: SD, WS change on the falling edge of SCK
Bit 1	wssel: WS polarity selection 1: Transmit the first data when WS is high 0: Transmit the first data when WS is low
Bit 0	Stereo: stereo, mono data transmission or reception 1: Select stereo

0: select mono

19.3.6 **I2S** Interrupt Status Register (**I2S_ISR**)

Register description I2S Interrupt Status Register

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													underrun_intf	rxoerr_intf	rx_intf
													r	r	r

Bit 31: 4	Reserve
Bit 3	underrun_intf: Transmission underrun error interrupt flag 1: Underload error 0: No underload error
Bit 2	rxoerr_intf: Receive overload error interrupt flag 1: Overload error 0: No overload error

Bit 1	rx_intf: Receive valid data interrupt flag 1: When rxtlf=01, RXFIFO has received 8 data; when rxtlf=other, RXFIFO is not empty 0: When rxtlf=01, RXFIFO does not receive 8 data; when rxtlf=other, RXFIFO is empty
Bit 0	tx_intf: TXFOFO null interrupt flag 1: When txtlf=01, there are less than 8 data in TXFIFO; when txtlf=other, TXFIFO is not full 0: When txtlf=01, the data in TXFIFO is greater than 8; when txtlf=other, TXFIFO is full

19.3.7 **I2S** interrupt enable register (**I2S_IER**)

Register description: I2S Interrupt Enable Register

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												underrun_inten	rxoerr_inten	rx_inten	tx_inten
												rw	rw	rw	rw

Bit 31: 4	Reserve
Bit 3	underrun_inten: transmission underrun error interrupt enable 1: Underload error interrupt enable 0: Underload error interrupt disabled
Bit 2	rxoerr_inten: Receive overload error interrupt enable 1: Overload error interrupt enable

	0: Overload error interrupt disabled
Bit 1	rx_inten: receive valid data interrupt enable 1: Receive valid data interrupt enable 0: Receiving valid data interrupt is prohibited
Bit 0	tx_inten: TXFOFO null interrupt enable 1: TXFIFO empty interrupt enable 0: TXFIFO empty interrupt disabled

19.3.8 I2S Interrupt Clear Register (I2S_ICR)

Register description: I2S Interrupt Clear Register

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												underrun_intclr	rxoerr_intclr	rx_intclr	tx_intclr
												w	w	w	w

Bit 31: 4	Reserve
Bit 3	underrun_intclr: transmission underrun error interrupt clear 1: Underload error interrupt clear 0: Underload error interrupt is not cleared
Bit 2	rxoerr_intclr: Receive overload error interrupt clear 1: Overload error interrupt clear 0: Overload error interrupt is not cleared
Bit 1	rx_intclr: Receiving valid data interrupt clear 1: Receiving valid data interrupt clear 0: Receive valid data interrupt and not clear
Bit 0	tx_intclr: TXFOFO empty interrupt clear 1: TXFIFO empty interrupt clear 0: TXFIFO empty interrupt is not cleared

19.3.9 I2S Frequency Divider Register (I2S_PRE)

Register description: I2S Prescale Register

Address offset: 0x24

Reset value: 0x0000 0C02

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			mclk_out_sel	clk_bypass	chlen	mckoe	odd	div							

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 12	mclk_out_sel: master clock output selection 1: The master clock output is I2S_CLK 0: master clock output is I2S_PRECLK
Bit 11	SCK bypass 1: SCK comes from Linear Prescaler frequency division 0: SCK=64Fs, configured as 2/4/8/16 frequency division of I2S_CLK by setting nFs_sel
Bit 10	chlen: channel length selection 1: 32bit 0: 16bit Only supports 32bit
Bit 9	mckoe: master clock output enable 1: master clock output enable 0: master clock output disabled
Bit 8	odd: I2S odd division coefficient 1: The frequency division coefficient is div*2+1 0: The frequency division coefficient is div*2 This bit needs to be configured when I2S is disabled
Bit 7:0	div: I2S div division coefficient The actual frequency division factor is div*2+odd div[7:0] cannot be configured as 0 or 1, otherwise there will be no clock output This bit needs to be configured when I2S is disabled

20. Serial Peripheral Interface (**SPI**)

20.1 Brief description of **SPI**

The SPI interface is widely used for board-level communication between different devices, such as microprocessors, ADCs, and so on. In fact, SPI has become an entire industry Acceptable guidelines for the industry. Many IC manufacturers produce devices that are compatible with SPI.

SPI allows MCU to communicate with external devices in full-duplex, synchronous, and serial mode. The application software can query the status or SPI interrupt to Communication.

20.2 Main features

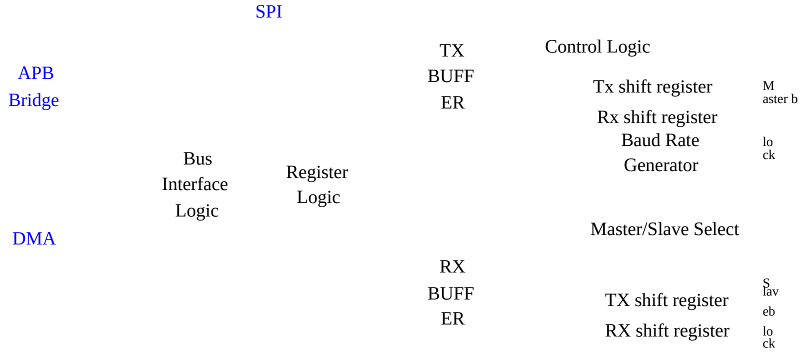
- Fully compatible with Motorola's SPI specification
- Support DMA request
- Full duplex synchronous transmission
- 16-bit programmable baud rate generator
- Support master mode and slave mode
- SPI is the fastest SPI clock in the master mode as high as pcllk/2 (pclk is the APB clock), as the SPI in the slave mode The fastest clock can be as high as pclk/4.
- Programmable clock polarity and phase
- Programmable data sequence, MSB first or LSB first
- Support multiple slave operations from one master
- Support sending and receiving 7-bit or 8-bit data at the same time
- Support 9-bit and 8-bit data transmission with configurable length
- Support the hardware to automatically send the same data repeatedly in the main mode
- With 8 bytes of transmit buffer and receive buffer each
- Interrupt driven operation
  - The sender is empty and the sender overflows
  - The received data is valid, and the data at the receiving end overflows
  - Complete reception in SPI master mode, the sender is empty
  - The hardware automatically repeats the end of the transmission.

20.3 SPI function description

20.3.1 Overview

The block diagram of SPI is shown in the figure below

Figure **162. SPI** block diagram



SPI supports receiving and sending 7 or 8 bits of data at the same time, and supports sending and receiving of configurable 4-16bit data length. SPI can
To be configured as a slave mode or as a master mode in a host environment. Can be selected by configuring the clock polarity CPOL and phase CPHA
Four possible timing relationships. Programmable data sequence, MSB first or LSB first.

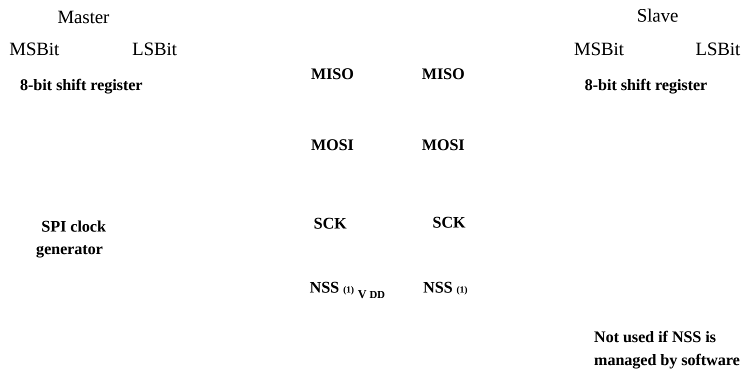
The sending and receiving parts use the same clock. Data is output on the rising or falling edge of the clock, and latched on the opposite valid edge of SCLK
data. Because SPI is used to exchange data, the data must be read after the transfer, even if the data is not valid. In SPI mode
Under the following conditions, the clock phase and polarity of the master and the slave communicating with it must be the same.

Usually SPI is connected to external devices through 4 pins:

- MISO: Master input/slave output pin. This pin sends data in slave mode and receives data in master mode.
- MOSI: Master device output/slave device input pin. This pin sends data in master mode and receives data in slave mode.
- SCK: Serial port clock, as the output of the master device and the input of the slave device
- NSS: Select from the device. This is an optional pin to select the master/slave device. Its function is to be used as a'chip select pin',
  Allow the master device to communicate with specific slave devices separately to avoid data line conflicts. The NSS pin of the slave device can be used by the master device
  Driven as a standard IO. Once enabled, the NSS pin can also be used as an output pin and set as the master mode in SPI
  At this time, all NSS pins connected to the SPI device of the master device's NSS pin will detect a low level.

The figure below is an example of single-master and single-slave interconnection.

Figure **163.** Single master and single slave applications



The MOSI pins are connected to each other, and the MISO pins are connected to each other. In this way, data is transmitted serially between the master and the slave (MSB bit first).

Communication is always initiated by the master device. The master device sends data to the slave device through the MOSI pin, and the slave device sends data back through the MISO
This means that the data output and data input of full-duplex communication are synchronized with the same clock signal; the clock signal is provided by the master device through the SCK pi

Data transmission register

SPI defines two data transmission registers: 32bit SPI_TXREGA and 16bit SPI_TXREGB (SPI_TXREGB
Divided into two register definitions: 8bit SPI_TXREGBL and 8bit SPI_TXREGH).

When data is written into SPI_TXREGA, the data is sent directly; when data is written into SPI_TXREGBH or SPI_TXREGBL alone
No data is sent, only after writing to the SPI_TXREGH register and then writing to the SPI_TXREGBL register, the 16-bit number is sent together
according to.

Note: The TXREG_SEL bit is defined in SPI_GCTL to select whether to use the SPI_TXREGA register or
SPI_TXREGB register.

The hardware automatically sends data repeatedly

SPI defines the hardware automatic data transmission control bit SER_TRANF_EN in the SPI_CCTL register. After the enable is turned on, at this time The data written into the sending register will be automatically transmitted n times by the hardware, and the number of transmissions is defined by the SPI_TX_NUM register.

Note: The value written in SPI_TX_NUM is i, then i-1 times are transmitted.

Phase and polarity of the clock signal

The CPOL and CPHA bits of the SPI_CCTL register can be combined into four possible timing relationships. CPOL (clock polarity) position control It controls the idle state level of the clock when there is no data transmission. This bit is valid for devices in both master mode and slave mode. If CPOL is cleared to '0', The SCK pin remains low in the idle state; if CPOL is set to '1', the SCK pin remains high in the idle state.

If the CPHA (clock phase) bit is set to '1', the second edge of the SCK clock (the falling edge when the CPOL bit is 0, the CPOL bit When it is 1, it is the rising edge.) The data bit is sampled, and the data is latched on the second clock edge. If the CPHA bit is cleared to '0', the SCK clock The first edge (the falling edge when the CPOL bit is 0 is the falling edge, and the rising edge when the CPOL bit is 1) is used to sample the data bits, and the data is in the first The clock edge is latched.

The combination of CPOL clock polarity and CPHA clock phase selects the clock edge for data capture. Figure 164 shows the 4 types of SPI transmission Combination of CPHA and CPOL bits. This figure can be interpreted as the master or the direct connection of the SCK pin, MISO pin, and MOSI pin of the master device and slave device. From the timing diagram.

High-speed transmission

For the sensitivity to board-level delay in high-speed transmission mode, the TXEDGE and RXEDGE control bits are paired in the SPI_CCTL register. Send phase and receive samples for time adjustment.

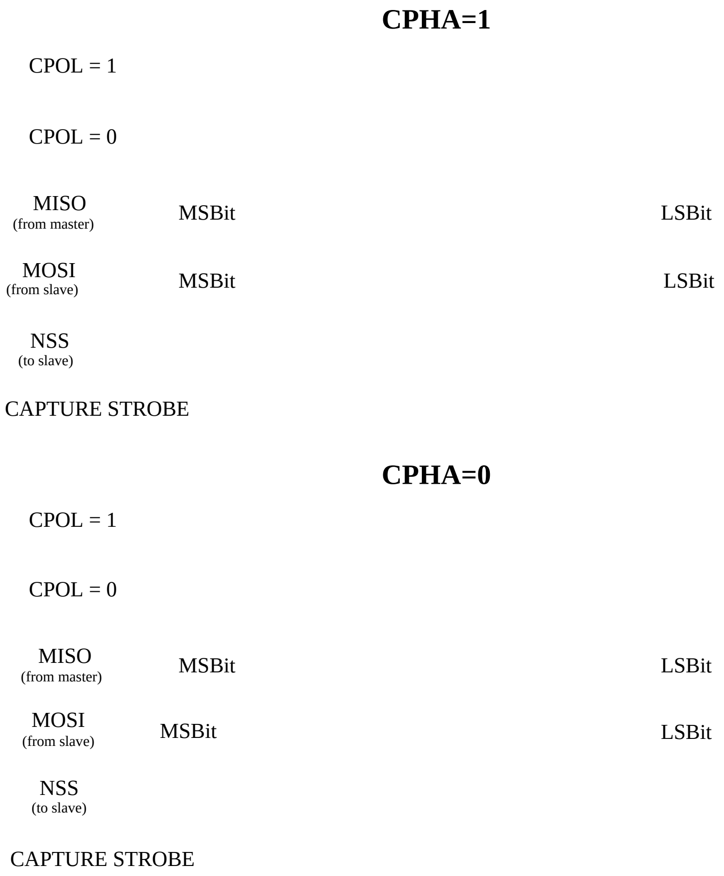
- In slave mode, when TXEDGE is 1, the sending data is sent to the data bus immediately, when used in high-speed mode (SPBRG = 4); When it is 0, the transmission data is sent to the data bus after a valid clock edge, which is used in low-speed mode (SPBRG> 4).
- In the master mode, when RXEDGE is 1, the data is sampled in the middle of the transmission data bit; when it is 0, the data is sampled at the tail clock of the transmission data bit Edge sampling data (used in high-speed mode).

Note: *1.* Before changing the *CPOL/CPHA* bit, the *SPIEN* bit must be cleared to disable *SPI* .

2. The master and slave must be configured in the same timing mode.

3. The idle state of *SCK* must be consistent with the polarity specified by the *SPI_CCTL* register (when *CPOL* is *1* , *SCK* should be pulled up when idle High level; when *CPOL* is *0* , *SCK* should be pulled down to low level when idle).

Figure **164.** Data clock timing diagram



Data frame format

According to the LSBFE bit in the SPI_CCTL register, the output data bit can be MSB first or LSB first. According to SPI_GCTL The DATA_SEL of the register, the SPILEN bit of the SPI_CCTL register and the SPI_EXTLEN register, each data frame can be 4 Any bit from bit to 16 bit. The selected data frame format is valid for both sending and receiving.

20.3.2 SPI Slave Mode

In the slave configuration, the SCK pin is used to receive the serial clock from the master device. The setting in the SPI_SPBRG register does not affect the data Transmission rate.

Configuration steps

1. Set the SPILEN bit, DATA_DEL bit, and SPI_EXTLEN bit to define the data frame format.

2. Select the CPOL and CPHA bits to define the phase relationship between data transmission and serial clock. To ensure correct data transmission, from The CPOL and CPHA bits of the device and the master device must be configured in the same way.
3. The frame format (MSB first or LSB first depends on the LSBFE bit in the SPI_CCTL register) must be the same as the master device.
4. Clear the MM bit and set the SPIEN bit to make the corresponding pin work in SPI mode. In this configuration, the MOSI pin is the data Input, MISO pin is data output.

Data sending process

In a write operation, data words are written to the transmit buffer in parallel.

When the slave device receives the clock signal and the first data bit appears on the MOSI pin, the sending process starts and the first bit is sent go out. The remaining bits are loaded into the shift register. When the data in the transmit buffer is transferred to the shift register, the SPI_INTSTAT register The TX_INTF flag is set. If the TXIEN bit on the SPI_INTEN register is set, an interrupt will be generated.

Data receiving process

For the receiver, when the data reception is complete:

- The data in the shift register is transferred to the receive buffer, and the RX_INTF flag in the SPI_INTSTAT register is set.
- If the RXIEN bit in the SPI_INTEN register is set, an interrupt is generated.

After the last sampling clock edge, the RXNE bit is set to '1', and the data byte received in the shift register is transferred to the receive buffer. When reading the SPI_RXREG register, the SPI device returns this value.

20.3.3 SPI Master Mode

In the main configuration, the serial clock is generated on the SCK pin.

Configuration steps

1. Define the serial clock baud rate through the SPI_SPBRG register.
2. Select the CPOL and CPHA bits to define the phase relationship between data transmission and serial clock.
3. Set the SPILEN bit, DATA_SEL bit, and SPI_EXTLEN bit to define the data frame format.
4. Configure the LSBFE bit of the SPI_CCTL register to define the frame format.
5. If you only receive data without sending data, configure the SPI_RNDNR register to define the number of bytes that need to be received.
6. The MM and SPIEN bits must be set.

In this configuration, MOSI pin is data output, MISO pin is data input, and NSS is slave device selection signal output.

Data sending process

When a byte is written into the transmit buffer, the transmission process begins. When sending the first data bit, the data word is Line) into the shift register, and then serially shifted out to the MOSI pin; MSB first or LSB first depends on the SPI_CCTL register The LSBFE bit in the device. The TX_INTF flag will be set when data is transferred from the transmit buffer to the shift register. If SPI_INTEN is set The TXIEN bit in the register will generate an interrupt.

Data receiving process

For the receiver, when the data transmission is complete:

- The data in the shift register is transferred to the receive buffer, and the RX_INTF flag in the SPI_INTSTAT register is set.
- If the RXIEN bit in the SPI_INTEN register is set, an interrupt is generated.

After the last sampling clock edge, the RXNE bit is set to '1', and the data byte received in the shift register is transferred to the receive buffer. When reading the SPI_RXREG register, the SPI device returns this value.

If only receiving but not sending data, after receiving the number of bytes defined by RXDNR, the RXMATCH_INTF bit is set to '1', indicating that all After the data is received, the clock signal is no longer sent in the master mode.

20.3.4 Status Flag

For the convenience of software operation, the application program can monitor the status of the SPI bus through 4 current status flags and 7 interrupt status flags. state. The current status flag is read-only and is automatically set and cleared by hardware. The interrupt status flag is set when an event occurs, and when the interrupt is enabled A CPU interrupt is generated, which is cleared by software.

There is an 8-byte transmit buffer and receive buffer inside the SPI. According to the setting of the DATA_SEL bit of SPI_GCTL, the CPU 1 or 4 bytes can be read and written at a time. According to the setting of DATA_SEL, the sending and receiving buffers have a byte or a valid number respectively According to the status flag.

Table **36. SPI** Status

Classification	Status flag	Buffer and signal status
Interrupt state	TX_INTF	According to the DATA_SEL setting, there is at least one space for valid data, Able to complete a write operation of sending data register
	RX_INTF	According to the DATA_SEL setting, there is at least one valid data data, Able to complete a read operation of the receive data register
	UNDERRUN_INTF	Send buffer empty and repeat sending
	RXOERR_INTF	The receive buffer is not empty and is covered
	RXMATCH_INTF	Not empty, the last data is transferred to the receive buffer
	RXFULL_INTF	The receive buffer is full and no more data can be received
	TXEPT_INTF	The send buffer is empty and can no longer send
	SER_TRANF_INTF	Repeat sending ends, no more sending
Current state	SER_TRANF_END	No repeat sending action
	RXAVL_4BYTE	The receive buffer has more than 4 bytes of valid data
	TXFULL	Send buffer full
	TXEPT	Send buffer empty
	RXAVL	The receive buffer is not empty, at least one byte can be received

NOTE: When *SPI_GCTL* register *TXTLF* to *00* , the transmission buffer has not less than . *1* when idle data space *TX_INTF* Set; *TXTLF* to *01* , the transmission buffer has more than half of the free space *TX_INTF* set.

When *SPI_GCTL* register *RXTLF* to *00* , the receiving buffer has not less than . *1* when valid data, *RX_INTF* set; *When RXTLF* is *01* , *RX_INTF is* set when there is more than half of the valid data in the receive buffer .

20.3.5 Baud rate setting

The baud rate is the frequency of the generated SCLK, which is generally the frequency division of PCLK. BRG is a 16-bit baud rate generator. SPBREG The register controls the counting cycle of the 16-bit counter.

Provide the desired baud rate and f $_{pclk}$ (the frequency of the APB module), and assign the approximate value calculated by the formula shown in the following table to SPBRG register. The X in the following table is equal to the value of the SPBRG register (2 ~ 65535).

Table **37.** Baud rate formula

model	formula
SPI mode	Baud rate = f $_{pclk}$ /X

20.3.6 using **DMA** of **SPI** communication

In order to achieve the maximum communication speed, it is necessary to fill the SPI send buffer with data in time, and the data in the receive buffer must also be read in time Walk to prevent overflow. In order to facilitate high-rate data transmission, SPI implements a simple request/response DMA mechanism.

When the DMAEN bit on the SPI_GCTL register is set, the SPI module can issue a DMA transfer data request. Send buffer Both the DMA request of the receiver and the receive buffer are enabled by DMAEN.

- When transmitting, when the TXTLF of the SPI_GCTL register is 00, when the transmit buffer has 1 free data space or more, that is Make a DMA transfer request; when TXTLF is 01, the DMA request is made when there is more than half of the free space in the transmit buffer. Only one DMA transfer is performed per request. The size of each DMA transfer data and the size of each data in the transmit buffer are determined by DATA_SEL is the decision.
- When receiving, when the RXTLF of the SPI_GCTL register is 00, when the receiving buffer has more than or equal to 1 valid data, it will proceed. DMA transfer request; when RXTLF is 01, the DMA request is made when there is more than half of the valid data in the receive buffer. Each Request only one DMA transfer. The size of each DMA transfer data and the size of each data in the receive buffer are determined by DATA_SEL For decision.

20.4 Description of Register File and Memory Map

20.4.1 Transmit Data Register (**SPI_TXREGA**)

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
TXREG [31: 16]															

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

Bit 31:0

TXREG : Transmit data register

The valid data bit is controlled by data_sel.

0: Only the lower 8 bits are valid

1: TXREG[31:0] are all valid

20.4.2 Transmit Data Register (**SPI_TXREGBL**)

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

Bit 31: 8 Reserve

Bit 7:0 **TXREG** : Transmit data register B lower 8 bits (Transmit data register)

20.4.3 Transmit Data Register (**SPI_TXREGBH**)

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

Bit 31: 8 Reserve

Bit 7:0 **TXREG** : Transmit data register B high 8 bits (Transmit data register)

20.4.4 Receive Data Register (**SPI_RXREG**)

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
RXREG [31:16]															

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

Bit 31:0	**RXREG** : Receive data register The valid data bit is controlled by data_sel. 0: Only the lower 8 bits are valid 1: RXREG[31:0] are all valid This register is readable but not writable.

20.4.5 Current Status Register (**SPI_CSTAT**)

Offset address: 0x10

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												RX AVL_ 4BYTE	TX FULL	RX AVL	TX EPT
												r	r	r	r

Bit 31: 4	Reserve.
Bit 3	**RXAVL_4BYTE** : The valid data in the receive buffer reaches 4 bytes flag (Receive available 4 byte data message) 1 = There are more than 4 bytes in the receive buffer 0 = The data in the receive buffer is less than 4 bytes
Bit 2	**TXFULL** : Transmitter FIFO full status bit 1 = Transmit buffer is full 0 = The transmit buffer is not full
Bit 1	**RXAVL** : Receive available byte data message This bit is set when the receiving buffer receives a complete byte of data. 1 = The receiving buffer has received a valid byte of data 0 = Receiver buffer is empty This bit is read-only and is automatically set and cleared by hardware.
Bit 0	**TXEPT** : Transmitter empty bit 1 = The transmit buffer and transmit shift register are empty 0 = The sender is not empty This bit is read-only and is automatically set and cleared by hardware.

20.4.6 Interrupt Status Register (**SPI_INTSTAT**)

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								SER_TRANF_INTF	TXEPT_INTF	RXFULL_INTF	RXMATCH_ INTF	RXOERR_INTF	UNDERRUN_INTF	RX_INTF	TX_INTF
								R	r	r	r	r	r	r	r

Bit 31: 8	Reserve
Bit 7	SER_TRANF_INTF: Continuous transmission data end interrupt flag bit (Serial transform over interrupt flag bit) 1: End of continuous data transmission 0: Continuous transmission in progress
Bit 6	**TXEPT_INTF** : Transmitter empty interrupt flag bit Automatically set by hardware, write TXEPT_ICLR bit in INTCLR register to clear 1=The transmit buffer and TX shift register are empty 0=The sender is not empty; Note: This bit is the interrupt status signal, TXEPT is the status signal
Bit 5	**RXFULL_INTF** : RX FIFO full interrupt flag bit (RX FIFO full interrupt flag bit) Automatically set by hardware, write RXFULL_ICLR bit in INTCLR register to clear 1=RX buffer full 0=RX buffer is not full
Bit 4	**RXMATCH_INTF** : Receive data match the RXDNR interrupt flag bit (Receive data match the RXDNR number, the receive process will be completed and generate the interrupt) Automatically set by hardware, write RXMATCH_ICLR bit of INTCLR register to clear 1 = The number of bytes specified by the RXDNR register has been received 0 = The number of bytes specified by the RXDNR register is not completed
Bit 3	**RXOERR_INTF** : Receive overrun error interrupt flag bit Automatically set by hardware, write INTCLR register RXOERR_ICLR bit to clear 1=overflow error 0=no overflow error
Bit 2	**UNDERRUN_INTF** : SPI underrun interrupt flag bit in slave mode Automatically set by hardware, write UNDERRUN_ICLR bit in INTCLR register to clear 1=underflow error 0=no underflow error

Bit 1	**RX_INTF** : Receive data available interrupt flag bit (Receive data available interrupt flag bit) Automatically set by hardware, clear by writing RX_ICLR bit of INTCLR register When the receiver buffer receives a complete byte of data 1=The receiving buffer has valid byte data 0=The receiving end buffer is empty
Bit 0	**TX_INTF** : Transmit buffer valid interrupt flag bit (one byte of data is sent) (Transmit FIFO available interrupt flag bit) Automatically set by hardware, write TX_ICLR bit of INTCLR register to clear 1=The sender buffer is valid 0=The sender buffer is invalid

20.4.7 Interrupt Enable Register (**SPI_INTEN**)

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								SER_ TRAN F_IEN	TX EPT_ IEN	RX FULL _IEN	RX MATCH _IEN	RXO ERR_ IEN	UND ERR UN_ IEN	RX_ IEN	TX_ IEN
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve
Bit 7	**SER_TRANF_IEN** : Continuous transmission data end interrupt flag bit (Serial transform over interrupt enable bit) 1: Interrupt enable 0: Disable interrupt
Bit 6	**TXEPT_IEN** : Transmit empty interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 5	**RXFULL_IEN** : Receive FIFO full interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 4	**RXMATCH_IEN** : Receive data complete interrupt enable bit (Receive data complete interrupt enable bit) 1 = interrupt enable 0 = Disable interrupt
Bit 3	**RXOERR_IEN** : Overrun error interrupt enable bit at the receiving end 1=Interrupt enable 0=disable interrupts

Bit 2	**UNDERRUN_IEN** : SPI slave mode underflow interrupt enable bit (SPI slave mode) (Transmitter underrun interrupt enable bit(SPI slave mode only)) 1=Interrupt enable 0=disable interrupt
Bit 1	**RX_IEN** : Receive FIFO interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 0	**TX_IEN** : Transmit FIFO empty interrupt enable bit 1=Interrupt enable 0=disable interrupt

20.4.8 Interrupt Clear Register (**SPI_INTCLR**)

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								SER_ TRAN F_ICL R	TX EPT_ ICLR	RX FULL _ICLR	RX MATCH _ ICLR	RXO ERR_ ICLR	UND ERR UN_ ICLR	RX_ ICLR	TX_ ICLR
								w	w	w	w	w	w	w	w

Bit 31: 8

Reserve

Bit 7

SER_TRANF_ICLR : Continuous data transmission end interrupt clear bit (Serial transform over interrupt clear bit)

1: Interrupt clear

0: The interrupt is not cleared

Bit 6

TXEPT_ICLR : Transmitter empty interrupt clear bit

1=Interrupt clear

0=The interrupt is not cleared

Bit 5

RXFULL_ICLR : Receiver buffer full interrupt clear bit

1=Interrupt clear

0=The interrupt is not cleared

Bit 4

RXMATCH_ICLR : Receive completed interrupt clear bit (Receive completed interrupt clear bit)

1=Interrupt clear

0=The interrupt is not cleared

Bit 3

RXOERR_ICLR : Overrun error interrupt clear bit at the receiving end

1=Interrupt clear

0=The interrupt is not cleared

Bit 2

UNDERRUN_ICLR : SPI slave mode underflow interrupt clear bit (SPI slave mode) (Transmitter underrun interrupt clear bit(SPI slave mode only))

1=Interrupt clear

0=The interrupt is not cleared

Bit 1

RX_ICLR : Receive interrupt clear bit

1=Interrupt clear

0=The interrupt is not cleared

Bit 0

TX_ICLR : Transmitter FIFO empty interrupt clear bit

1=Interrupt clear

0=The interrupt is not cleared

20.4.9 Global Control Register (**SPI_GCTL**)

Offset address: 0x20

Reset value: 0x0000 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		TXREG_SEL	DATA_SEL	NSS_SEL	DMAEN	TXTLF		RXTLF		RXEN	TXEN	MM	INT_EN	SPIEN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 13

Reserve

Bit 12

TXREG_SEL : transmit data register selection (TXREG select)

0: TXREGA

1: TXREGB

Bit 11

DATA_SEL : Send and receive data register valid data selection (Valid byte or double-word data select signal)

0: Only the lower 8 bits are valid

1: 32-bit data are valid

Note: Whether it is through CPU or DMA, it must be accessed with the specified data format.

Bit 10

NSS_SEL : NSS output in hardware or software control master mode (NSS select signal that from software or hardware)

0: Controlled by the value of the NSSR register

	1: Hardware automatic control during data transmission
Bit 9	**DMAEN** : DMA mode enable for receiving and sending (DMA access mode enable) 0: DMA mode disabled 1: DMA mode is enabled

Bit 8: 7	**TXTLF** : transmit buffer trigger DMA request edge selection (TX FIFO trigger level bit) 00: DMA request or sending interrupt request will be made when the sending buffer has more than 1 free data space 01: When there is more than half of the free space in the transmit buffer, a DMA request or an interrupt request is sent 1x: reserved Note: When DATA_SEL is 0, one data space represents 1 byte; when it is 1, one data space represents 4 bytes.
Bit 6: 5	**RXTLF** : Receive buffer trigger DMA request edge selection (RX FIFO trigger level bit) 00: DMA request or receive interrupt request is performed when the receive buffer has more than or equal to 1 valid data 01: When the receiving buffer has more than half of the valid data, it will make a DMA request or receive an interrupt request 1x: reserved Note: When DATA_SEL is 0, a valid data represents 1 byte; when it is 1, a valid data represents 4 bytes.
Bit 4	**RXEN** : Receive enable bit 1=receive enabled 0=Receive is disabled. RX buffer can be cleared at the same time Note: When SPI only works in host receiving mode, txen must be set to 0
Bit 3	**TXEN** : Transmit enable bit 1=send enable 0=Send is prohibited. The TX buffer can be cleared at the same time Note: when sending and receiving occur at the same time in host mode
Bit 2	**MM** : Master mode bit 1=Master mode (Serial clock generated by internal BRG) 0=Slave mode (serial clock comes from external host)
Bit 1	**INT_EN** : SPI interrupt enable bit (SPI interrupt enable bit) 1=Enable SPI interrupt 0=Disable SPI interrupt
Bit 0	**SPIEN** : SPI select bit 0=SPI disabled (reset state) 1=SPI enable

20.4.10 General Control Register (**SPI_CCTL**)

Offset address: 0x24

Reset value: 0x0000 0008

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										TX EDGE	RX EDGE	SPI LEN	LSB FE	CPOL	CPHA
										rw	rw	rw	rw	rw	rw

Bit 31: 6	Reserve

Bit 5	**TXEDGE** : transmit data phase adjustment bit (slave mode) (Transmit data edge select) 1=Sending data is sent to the data bus immediately, Can be used in high-speed mode (SPBRG=4) 0=Send data is sent to the data bus after a valid clock edge, Can be used in low speed mode (SPBRG>4)
Bit 4	**RXEDGE** : Receive data sampling clock edge select bit (master mode) (Receive data edge select) 1=Sampling data at the tail clock edge of the transmitted data bit (used in high-speed mode) 0=Sampling data in the middle of the transmission data bit
Bit 3	**SPILEN** : SPI data width bit (SPI character length bit) 1=8-bit data (default) 0=7-bit data
Bit 2	**LSBFE** : LSB first enable bit (LSI first enable bit) 1=The lowest bit of data transmission or reception is first 0=The most significant bit of data transmission or reception is first
Bit 1	**CPOL** : Clock polarity select bit 1=Clock is high in idle state 0=The clock is low in idle state
Bit 0	**CPHA** : Clock phase select bit 1=Data sampling starts from the first clock edge 0=Data sampling starts from the second clock edge

20.4.11 Baud rate generator (**SPI_SPBRG**)

Offset address: 0x28

Reset value: 0x0000 0002

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
SPBRG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserve
Bit 15:0	**SPBRG** : SPI baud rate control register is used to generate baud rate (SPI baud rate control register for baud rate) Baud rate formula: Baud rate = f $_{pclk}$ /SPBRG (F $_{pclk}$ / is the APB clock frequency) Note: Do not write 0 and 1 to this register.

20.4.12 Received Data **Number** Register (**SPI_RXDNR**)

Offset address: 0x2C

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
RXDNR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**RXDNR** : This register is used to store the number of bytes to be received in the next reception process (The register is used to hold a count of to be received bytes in next receive process) The value of this register is valid when SPI is the host receiving mode. The default value is 1. The value of this register is changed by MCU write value. Note: Do not write a value of '0' to this register.

20.4.13 Slave Chip Select Register (**SPI_SCSR**)

Offset address: 0x30

Reset value: 0x0000 00FF

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

Bit 15: 8	Reserve
Bit 7:0	**CSN** : Chip select output signal in master mode. Active low, this bit is invalid in slave mode (Chip select output signal in Master mode) 0: The slave device is selected 1: Slave device is not selected

20.4.14 Data Control Register (**SPI_EXTCTL**)

Offset address: 0x34

Reset value: 0x0000 0008

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											EXTLEN[4:0]				
											rw	rw	rw	rw	rw

Bit 31: 5	Reserve
Bit 4: 0	**EXTLEN** : Control the length of SPI data 0 0100: 4-bit 0 0101: 5-bit 0 0110: 6-bit 0 0111: 7-bit 0 1000: 8-bit 0 1001: 9-bit 0 1010: 10-bit 0 1011: 11-bit 0 1100: 12-bit 0 1101: 13-bit 0 1110: 14-bit 0 1111: 15-bit 1 0000: 16-bit

20.4.15 Repeated Send Data Quantity Control Register (**SPI_TX_NUM**)

Offset address: 0x38

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
TX_NUM[15:0]															
											rw	rw	rw	rw	rw

Bit 31: 16	Reserve
Bit 15:0	**TX_NUM** : the number of repeated data

20.4.16 Transfer Mode Register (**SPI_TRANSF_MODE**)

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
Reserve														MODE_SEL	
														rw	rw

Bit 31: 2	Reserve
Bit 1: 0	**MODE_SEL** : Mode selection 00: Standard mode 01: Serial mode (continuous mode)

21. Serial Peripheral Interface (**QSPI**)

21.1 Brief description of **QSPI**

The SPI interface is widely used for board-level communication between different devices, such as microprocessors, DACs, ADCs, etc. In fact, SPI has become Acceptable guidelines for the entire industry. Many IC manufacturers produce devices that are compatible with SPI.

SPI allows MCU to communicate with external devices in full-duplex, synchronous, and serial mode. The application software can query the status or SPI interrupt to Communication.

21.2 Main features

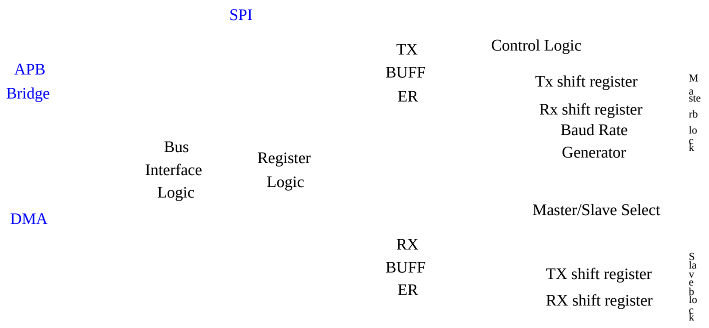
- Fully compatible with Motorola's SPI specification
- Support DMA request
- Full duplex synchronous transmission
- 16-bit programmable baud rate generator
- Support master mode and slave mode
- When SPI is used as the master mode, the SPI clock can be as fast as pcllk/2 (pclk is the APB clock). When used as the slave mode, the SPI clock can be as high as pcllk/2. The fastest clock can be as high as pclk/4.
- Programmable clock polarity and phase
- Programmable data sequence, MSB first or LSB first
- Support multiple slave operations from one master
- With 8 bytes of transmit buffer and receive buffer each
- Interrupt driven operation
  - The sender is empty and the sender overflows
  - The received data is valid, and the data at the receiving end overflows
  - In the SPI master mode, it is completely received and the sender is empty.

21.3 **SPI** function description

21.3.1 Overview

The block diagram of SPI is shown in the figure below

Figure **165. SPI** block diagram



SPI supports receiving and sending 7 or 8 bits of data at the same time. SPI can be configured as a slave mode or in a host environment
Main mode. Four possible timing relationships can be selected by configuring the clock polarity CPOL and phase CPHA. Programmable data sequence, MSB
First or LSB first.

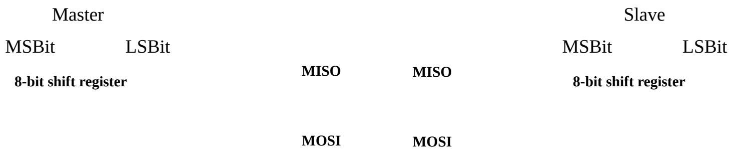
The sending and receiving parts use the same clock. Data is output on the rising or falling edge of the clock, and latched on the opposite valid edge of SCLK
data. Because SPI is used to exchange data, the data must be read after the transfer, even if the data is not valid. In SPI mode
Under the following conditions, the clock phase and polarity of the master and the slave communicating with it must be the same.

Usually SPI is connected to external devices through 4 pins:

- MISO: Master input/slave output pin. This pin sends data in slave mode and receives data in master mode.
- MOSI: Master device output/slave device input pin. This pin sends data in master mode and receives data in slave mode.
- SCK: Serial port clock, as the output of the master device and the input of the slave device
- NSS: Select from the device. This is an optional pin to select the master/slave device. Its function is to be used as a "chip select pin",
  Allow the master device to communicate with specific slave devices separately to avoid data line conflicts. The NSS pin of the slave device can be used by the master device
  Driven as a standard IO. Once enabled, the NSS pin can also be used as an output pin and set as the master mode in SPI
  At this time, all NSS pins connected to the SPI device of the master device's NSS pin will detect a low level.

The figure below is an example of single-master and single-slave interconnection.

Figure **166.** Single-master and single-slave applications



SPI clock generator

SCK **SCK**

NSS (1) V DD **NSS (1)**

Not used if NSS is managed by software

The MOSI pins are connected to each other, and the MISO pins are connected to each other. In this way, data is transmitted serially between the master and the slave (MSB bit first).

Communication is always initiated by the master device. The master device sends data to the slave device through the MOSI pin, and the slave device sends data back through the MISO
This means that the data output and data input of full-duplex communication are synchronized with the same clock signal; the clock signal is provided by the master device through the SCK pi

Phase and polarity of the clock signal

The CPOL and CPHA bits of the SPI_CCTL register can be combined into four possible timing relationships. The CPOL (clock polarity) bit is controlled in
The idle state level of the clock when there is no data transmission. This bit is valid for devices in both master mode and slave mode. If CPOL is cleared to '0',
The SCK pin remains low in the idle state; if CPOL is set to '1', the SCK pin remains high in the idle state.

If the CPHA (clock phase) bit is set to '1', the second edge of the SCK clock (the falling edge when the CPOL bit is 0, the CPOL bit
When it is 1, it is the rising edge.) The data bit is sampled, and the data is latched on the second clock edge. If the CPHA bit is cleared to '0', SCK is
The first edge of the clock (the falling edge when the CPOL bit is 0, the rising edge when the CPOL bit is 1) performs data bit sampling, and the data is in the first
The clock edge is latched.

The combination of CPOL clock polarity and CPHA clock phase selects the clock edge for data capture. Figure 164 shows the 4 types of SPI transmission
Combination of CPHA and CPOL bits. This figure can be interpreted as the master or the direct connection of the SCK pin, MISO pin, and MOSI pin of the master device and slave device.
From the timing diagram.

High-speed transmission

For the sensitivity to board-level delay in high-speed transmission mode, the TXEDGE and RXEDGE control bits are paired in the SPI_CCTL register.
Send phase and receive samples for time adjustment.

- In slave mode, when TXEDGE is 1, the sending data is sent to the data bus immediately, when used in high-speed mode (SPBRG=4);
When it is 0, the sending data is sent to the data bus after a valid clock edge, which is used in low-speed mode (SPBRG>4).
- In the master mode, when RXEDGE is 1, the data is sampled in the middle of the transmission data bit; when it is 0, the data is sampled at the tail clock of the transmission data bit
Edge sampling data (used in high-speed mode).

Note: *1.* Before changing the *CPOL/CPHA* bit, the *SPIEN* bit must be cleared to disable *SPI* .

2. The master and slave must be configured in the same timing mode.

3. The idle state of *SCK* must be consistent with the polarity specified by the *SPI_CCTL* register (when *CPOL* is *1* , *SCK* should be pulled up when idle
It is high level; when *CPOL* is *0* , *SCK* should be pulled down to low level when idle).

Figure **167.** Data clock timing diagram

CPHA=1

CPOL = 1

CPOL = 0

MISO (from master) MSBit LSBit

MOSI (from slave) MSBit LSBit

NSS (to slave)

CAPTURE STROBE

CPHA=0

CPOL = 1

CPOL = 0

MISO (from master) MSBit LSBit

MOSI (from slave) MSBit LSBit

NSS (to slave)

CAPTURE STROBE

Data frame format

According to the LSBFE bit in the SPI_CCTL register, the output data bit can be MSB first or LSB first. According to SPI_CCTL The SPILEN bit of the register, each data frame can be 8-bit or 7-bit. The selected data frame format is valid for both sending and/or receiving.

21.3.2 SPI master mode

In the main configuration, the serial clock is generated on the SCK pin.

Configuration steps

1. Define the serial clock baud rate through the SPI_SPBRG register.
2. Select the CPOL and CPHA bits to define the phase relationship between data transmission and serial clock.

3. Set the SPILEN bit to define the 8 or 7-bit data frame format.
4. Configure the LSBFE bit of the SPI_CCTL register to define the frame format.
5. If you only receive data without sending data, configure the SPI_RNDNR register to define the number of bytes that need to be received.
6. The MM and SPIEN bits must be set.

In this configuration, MOSI pin is data output, MISO pin is data input, and NSS is slave device selection signal output.

Data sending process

When a byte is written into the transmit buffer, the transmission process begins. When sending the first data bit, the data word is paralleled (via the internal bus) Into the shift register, and then serially shifted out to the MOSI pin; MSB first or LSB first depends on the SPI_CCTL register The LSBFE bit. The TX_INTF flag will be set when data is transferred from the transmit buffer to the shift register. If the SPI_INTEN register is set The TXIEN bit in the device will generate an interrupt.

Data receiving process

For the receiver, when the data transmission is complete:

- The data in the shift register is transferred to the receive buffer, and the RX_INTF flag in the SPI_INTSTAT register is set.
- If the RXIEN bit in the SPI_INTEN register is set, an interrupt is generated.

After the last sampling clock edge, the RXNE bit is set to '1', and the data byte received in the shift register is transferred to the receive buffer Device. When reading the SPI_RXREG register, the SPI device returns this value.

If only receiving but not sending data, after receiving the number of bytes defined by RXDNR, the RXMATCH_INTF bit is set to '1', indicating that all After some data is received, the clock signal is no longer sent in the master mode.

21.3.3 Status flag

For the convenience of software operation, the application program can monitor the status of the SPI bus through 4 current status flags and 7 interrupt status flags. state. The current status flag is read-only and is automatically set and cleared by hardware. The interrupt status flag is set when an event occurs, and when the interrupt is enabled A CPU interrupt is generated, which is cleared by software.

There is an 8-byte transmit buffer and receive buffer inside the SPI. According to the setting of the DATA_SEL bit of SPI_GCTL, the CPU 1 or 4 bytes can be read and written at a time. According to the setting of DATA_SEL, the sending and receiving buffers have a byte or a valid number respectively According to the status flag.

Table **38. SPI** Status

Classification	Status flag	Buffer and signal status
Interrupt state	TX_INTF	According to the DATA_SEL setting, there is at least one space for valid data, which can send data once Register write operation
	RX_INTF	According to the DATA_SEL setting, there is at least one valid data data, which can complete a data reception Register read operation
	UNDERRUN_INTF	Send buffer empty and repeat sending
	RXOERR_INTF	The receive buffer is not empty and is covered
	RXMATCH_INTF	Not empty, the last data is transferred to the receive buffer
	RXFULL_INTF	The receive buffer is full and no more data can be received
	TXEPT_INTF	The send buffer is empty and can no longer send
Current state	RXAVL_4BYTE	The receive buffer has more than 4 bytes of valid data
	TXFULL	Send buffer full
	TXEPT	Send buffer empty
	RXAVL	The receive buffer is not empty, at least one byte can be received

NOTE: When *SPI_GCTL* register *TXTLF* to *00* , the transmission buffer has not less than . *1* when idle data space *TX_INTF* Set; *TXTLF* to *01* , the transmission buffer has more than half of the free space *TX_INTF* set.

When *SPI_GCTL* register *RXTLF* to *00* , the receiving buffer has not less than . *1* when valid data, *RX_INTF* set; *When RXTLF* is *01* , *RX_INTF is* set when there is more than half of the valid data in the receive buffer .

21.3.4 Baud rate setting

The baud rate is the frequency of the generated SCLK, which is generally the frequency division of PCLK. BRG is a 16-bit baud rate generator. SPBREG The register controls the counting cycle of the 16-bit counter.

Provide the desired baud rate and f $_{pclk}$ (the frequency of the APB module), and assign the approximate value calculated by the formula shown in the following table to SPBRG register. The X in the following table is equal to the value of the SPBRG register (2~65535).

Table **39.** Baud rate formula

model	formula
SPI mode	Baud rate = f $_{pclk}$ /X

21.3.5 using **DMA** of **SPI** communication

In order to achieve the maximum communication speed, it is necessary to fill the SPI send buffer with data in time, and the data in the receive buffer must also be read in time Walk to prevent overflow. In order to facilitate high-rate data transmission, SPI implements a simple request/response DMA mechanism.

When the DMAEN bit on the SPI_GCTL register is set, the SPI module can issue a DMA transfer data request. Send buffer Both the DMA request of the receiver and the receive buffer are enabled by DMAEN.

- When transmitting, when the TXTLF of the SPI_GCTL register is 00, when the transmit buffer has 1 free data space or more, that is Make a DMA transfer request; when TXTLF is 01, the DMA request is made when there is more than half of the free space in the transmit buffer. Only one DMA transfer is performed per request. The size of each DMA transfer data and the size of each data in the transmit buffer are determined by DATA_SEL is the decision.

- When receiving, when the RXTLF of the SPI_GCTL register is 00, when the receiving buffer has more than or equal to 1 valid data, it will proceed. DMA transfer request; when RXTLF is 01, the DMA request is made when there is more than half of the valid data in the receive buffer. Each Request only one DMA transfer. The size of each DMA transfer data and the size of each data in the receive buffer are determined by DATA_SEL For decision.

21.4 Description of Register File and Memory Map

21.4.1 Transmit Data Register (**QSPI_TXREG**)

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
TXREG [31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXREG [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**TXREG** : Transmit data register The effective data bit is controlled by data_sel 0: Only the lower 8 bits are valid 1: TXREG[31:0] are all valid

21.4.2 Receive Data Register (**QSPI_RXREG**)

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
RXREG [31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXREG [15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31:0	**RXREG** : Receive data register (Receive data register) The effective data bit is controlled by data_sel 0: Only the lower 8 bits are valid 1: RXREG[31:0] are all valid This register is readable but not writable.

21.4.3 Current Status Register (**QSPI_CSTAT**)

Offset address: 0x08

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												RXAVL_4BYTE	TXFULL	RXAVL	TXEPT
										_	_	r	r	r	r

Bit 31: 4	Reserve
Bit 3	**RXAVL_4BYTE** : The valid data in the receive buffer reaches 4 bytes flag (Receive available 4 byte data message) 1=There are more than 4 bytes in the receive buffer 0=The data in the receive buffer is less than 4 bytes
Bit 2	**TXFULL** : Transmitter FIFO full status bit 1=Transmit buffer is full 0=The send buffer is not full
Bit 1	**RXAVL** : Receive available byte data message Set this bit when the receiving buffer receives a complete byte of data 1=The receiving buffer has received a valid byte of data 0=The receiving end buffer is empty This bit is read-only and is automatically set and cleared by hardware
Bit 0	**TXEPT** : Transmitter empty bit 1=The sending buffer and the sending shift register are empty. 0=The sender is not empty This bit is read-only and is automatically set and cleared by hardware

21.4.4 Interrupt Status Register (**QSPI_INTSTAT**)

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									TXEPT_INTF	RXFULL_INTF	RXMATCH_INTF	RXOERR_INTF	UNDERRUN_INTF	RX_INTF	TX_INTF
									r	r	r	r	r	r	r

Bit 31: 7	Reserve
Bit 6	**TXEPT_INTF** : Transmitter empty interrupt flag bit Automatically set by hardware, write TXEPT_ICLR bit in INTCLR register to clear 1=The transmit buffer and TX shift register are empty 0=The sender is not empty; Note: This bit is the interrupt status signal, TXEPT is the status signal
Bit 5	**RXFULL_INTF** : RX FIFO full interrupt flag bit (RX FIFO full interrupt flag bit) Automatically set by hardware, write RXFULL_ICLR bit in INTCLR register to clear 1=RX buffer full 0=RX buffer is not full
Bit 4	**RXMATCH_INTF** : Receive data match the RXDNR interrupt flag bit (Receive data match the RXDNR number, the receive process will be completed and generate the interrupt) Automatically set by hardware, write RXMATCH_ICLR bit of INTCLR register to clear 1 = The number of bytes specified by the RXDNR register has been received 0 = The number of bytes specified by the RXDNR register is not completed
Bit 3	**RXOERR_INTF** : Receive overrun error interrupt flag bit Automatically set by hardware, write INTCLR register RXOERR_ICLR bit to clear

	1=overflow error 0=no overflow error
Bit 2	**UNDERRUN_INTF** : SPI underrun interrupt flag bit in slave mode Automatically set by hardware, write UNDERRUN_ICLR bit in INTCLR register to clear 1=underflow error 0=no underflow error
Bit 1	**RX_INTF** : Receive data available interrupt flag bit (Receive data available interrupt flag bit) Automatically set by hardware, clear by writing RX_ICLR bit of INTCLR register When the receiver buffer receives a complete byte of data 1=The receiving buffer has valid byte data 0=The receiving end buffer is empty
Bit 0	**TX_INTF** : Transmit buffer valid interrupt flag bit (one byte of data is sent) (Transmit FIFO available interrupt flag bit) Automatically set by hardware, write TX_ICLR bit of INTCLR register to clear 1=The sender buffer is valid 0=The sender buffer is invalid

21.4.5 Interrupt Enable Register (**QSPI_INTEN**)

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									TXEPT_IEN	RXFULL_IEN	RXMATCH_IEN	RXOERR_IEN	UNDERRUN_IEN	RX_IEN	TX_IEN
									rw	rw	rw	rw	rw	rw	rw

Bit 31: 7	Reserve
Bit 6	**TXEPT_IEN** : Transmit empty interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 5	**RXFULL_IEN** : Receive FIFO full interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 4	**RXMATCH_IEN** : Receive data complete interrupt enable bit (Receive data complete interrupt enable bit) 1 = interrupt enable 0 = Disable interrupt
Bit 3	**RXOERR_IEN** : Overrun error interrupt enable bit at the receiving end 1=Interrupt enable 0=disable interrupt
Bit 2	**UNDERRUN_IEN** : SPI slave mode underflow interrupt enable bit (SPI slave mode) (Transmitter underrun interrupt enable bit(SPI slave mode only)) 1=Interrupt enable 0=disable interrupt

Bit 1	**RX_IEN** : Receive FIFO interrupt enable bit 1=Interrupt enable 0=disable interrupt
Bit 0	**TX_IEN** : Transmit FIFO empty interrupt enable bit 1=Interrupt enable 0=disable interrupts

21.4.6 Interrupt Clear Register (**QSPI_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									TXEPT_ICLR	RXFULL_ICLR	RXMATCH_ICLR	RXOERR_ICLR	UNDERRUN_ICLR	RX_ICLR	TX_ICLR
									w	w	w	w	w	w	w

Bit 31: 7	Reserve
Bit 6	**TXEPT_ICLR** : Transmitter empty interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 5	**RXFULL_ICLR** : Receiver buffer full interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 4	**RXMATCH_ICLR** : Receive completed interrupt clear bit (Receive completed interrupt clear bit) 1=Interrupt clear 0=The interrupt is not cleared
Bit 3	**RXOERR_ICLR** : Overrun error interrupt clear bit at the receiving end 1=Interrupt clear 0=The interrupt is not cleared
Bit 2	**UNDERRUN_ICLR** : SPI slave mode underflow interrupt clear bit (SPI slave mode) (Transmitter underrun interrupt clear bit(SPI slave mode only)) 1=Interrupt clear 0=The interrupt is not cleared
Bit 1	**RX_ICLR** : Receive interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 0	**TX_ICLR** : Transmitter FIFO empty interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared

Page 356

21.4.7 Global Control Register (**QSPI_GCTL**)

Offset address: 0x18

Reset value: 0x0000 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				DATA _SEL	NSS_ SEL	DMAE N	TXTLF		RXTLF		RXEN	TXEN	MM	INT_E N	SPIE N
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 12	Reserve
Bit 11	**DATA_SEL** : Send and receive data register valid data selection (Valid byte or double-word data select signal) 0: Only the lower 8 bits are valid 1: 32-bit data are valid Note: Whether it is through CPU or DMA, it must be accessed with the specified data format.
Bit 10	**NSS_SEL** : NSS output in hardware or software control master mode (NSS select signal that from software or hardware) 0: Controlled by the value of the NSSR register 1: Hardware automatic control during data transmission
Bit 9	**DMAEN** : DMA mode enable for receiving and sending (DMA access mode enable) 0: DMA mode disabled 1: DMA mode is enabled
Bit 8: 7	**TXTLF** : transmit buffer trigger DMA request edge selection (TX FIFO trigger level bit) 00: DMA request or sending interrupt request will be made when the sending buffer has more than 1 free data space 01: When there is more than half of the free space in the transmit buffer, a DMA request or an interrupt request is sent 1x: reserved Note: When DATA_SEL is 0, one data space represents 1 byte; when it is 1, one data space represents 4 bytes.
Bit 6: 5	**RXTLF** : Receive buffer trigger DMA request edge selection (RX FIFO trigger level bit) 00: DMA request or receive interrupt request is performed when the receive buffer has more than or equal to 1 valid data 01: When the receiving buffer has more than half of the valid data, it will make a DMA request or receive an interrupt request 1x: reserved Note: When DATA_SEL is 0, a valid data represents 1 byte; when it is 1, a valid data represents 4 bytes.
Bit 4	**RXEN** : Receive enable bit 1=receive enabled 0=Receive is disabled. RX buffer can be cleared at the same time Note: When SPI only works in host receiving mode, txen must be set to 0
Bit 3	**TXEN** : Transmit enable bit 1=send enable 0=Send is prohibited. The TX buffer can be cleared at the same time Note: when sending and receiving occur at the same time in host mode

Page 357

Bit 2	**MM** : Master mode bit 1=Master mode (Serial clock generated by internal BRG) 0=Slave mode (serial clock comes from external host)
	INT_EN : SPI interrupt enable bit (SPI interrupt enable bit)

Bit 1	1=Enable SPI interrupt 0=Disable SPI interrupt
Bit 0	**SPIEN** : SPI select bit 0=SPI disabled (reset state) 1=SPI enable

21.4.8 General Control Register (**QSPI_CCTL**)

Offset address: 0x1C

Reset value: 0x0000 0008

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										TXEDGE	RXEDGE	SPILEN	LSBFE	CPOL	CPHA
										rw	rw	rw	rw	rw	rw

Bit 31: 6	Reserve
Bit 5	**TXEDGE** : transmit data phase adjustment bit (slave mode) (Transmit data edge select) 1=Sending data is sent to the data bus immediately, Can be used in high-speed mode (SPBRG=4) 0=Send data is sent to the data bus after a valid clock edge, Can be used in low speed mode (SPBRG>4)
Bit 4	**RXEDGE** : Receive data sampling clock edge select bit (master mode) (Receive data edge select) 1=Sampling data at the tail clock edge of the transmitted data bit (used in high-speed mode) 0=Sampling data in the middle of the transmission data bit
Bit 3	**SPILEN** : SPI data width bit (SPI character length bit) 1=8-bit data (default) 0=7-bit data
Bit 2	**LSBFE** : LSB first enable bit (LSI first enable bit) 1=The lowest bit of data transmission or reception is first 0=The most significant bit of data transmission or reception is first
Bit 1	**CPOL** : Clock polarity select bit 1=Clock is high in idle state 0=The clock is low in idle state
Bit 0	**CPHA** : Clock phase select bit 1=Data sampling starts from the first clock edge 0=Data sampling starts from the second clock edge

21.4.9 Baud Rate Generator (**QSPI_SPBRG**)

Offset address: 0x20

Reset value: 0x0000 0002

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

Bit 31:0	**SPBRG** : SPI baud rate control register is used to generate baud rate (SPI baud rate control register for baud rate) Baud rate formula: Baud rate = fpclk/SPBRG

(fpclk/ is the APB clock frequency)
Note: Do not write 0 and 1 to this register

21.4.10 Received Data **Number** Register (**QSPI_RXDNR**)

Offset address: 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
Reserve															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDNR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	**RXDNR** : This register is used to store the number of bytes to be received in the next reception process (The register is used to hold a count of to be received bytes in next receive process) The value of this register is valid when SPI is the host receiving mode. The default value is 1. The value of this register is changed by MCU write value. Note: Do not write "0" value to this register.

21.4.11 Slave Chip Select Register (**QSPI_SCSR**)

Offset address: 0x28

Reset value: 0x0000 00FF

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

Bit 15: 8	Reserve
Bit 7:0	**CSN** : Chip select output signal in master mode. Active low, this bit is invalid in slave mode (Chip select output signal in Master mode). 0: The slave device is selected 1: Slave device is not selected

21.4.12 Slave Chip Select Register (**QSPI_MODE**)

Offset address: 0x2C

Reset value: 0x0000 000C

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												IO3_POL_SEL	IO2_POL_SEL	TRANF_MODE_SEL	
														rw	rw

Bit 31: 2	Reserve

Bit 3	**IO3_POL_SEL** : IO3 output polarity selection bit 0: The output status is 0 1: The output status is 1 Note: only need to be selected in STANDDARD, DUAL mode
Bit 2	**IO2_POL_SEL** : IO2 output polarity selection bit 0: The output status is 0 1: The output status is 1 Note: only need to be selected in STANDDARD, DUAL mode
Bit 1: 0	**TRANF_MODE_SEL** : SPI mode select (mode select). 00: Standard SPI mode (only master) 01: Dual SPI mode 10: Quad SPI mode 11: reserved

22. Universal Asynchronous Receiver Transmitter (**UART**)

22.1 Introduction to **UART**

The Universal Asynchronous Receiver Transmitter (UART) provides a flexible method to compare with external devices that use the industry standard NRZ asynchronous serial data for Full-duplex data exchange between. UART uses a fractional baud rate generator to provide a wide range of baud rate options. It supports synchronous one-way communication and Half-duplex single-wire communication, and modem (CTS/RTS) operation.

High-speed data communication can be realized by using the DMA mode of multi-buffer configuration.

22.2 Main Features of **UART**

- Support RS-232S protocol in asynchronous mode, in line with industry standard 16550.
- Support DMA request
- Full-duplex asynchronous operation
- Built-in 16-bit programmable baud rate generator.
- Separate transmit and receive buffer registers
- Built-in one byte sending and receiving buffer
- Send and receive data low order first
- Start with a start bit, followed by data bits, the output data length can be 5 bits, 6 bits, 7 bits, 8 bits, and the end is a stop bit.
  In addition, you can choose whether to add a parity bit. The parity bit is after the data bit and before the stop bit.
- Support hardware odd or even parity generation and detection
- Line disconnection generation and detection
- Support hardware automatic flow control
- The following interrupt sources are supported:
  - The sender BUFFER is empty
  - Data at the receiving end is valid
  - Receive buffer buffer overflow
  - Frame error
  - Parity error
  - Disconnect error

22.3 Overview of **UART** Function

Any UART two-way communication requires at least two pins: receive data input (RX) and transmit data output (TX).

RX: Receive data serial output. The over-sampling technique is used to distinguish data and noise, thereby recovering data.

TX: Send data output. When the transmitter is disabled, the output pin reverts to its I/O port configuration. When the transmitter is activated, and When not sending data, the TX pin is at a high level.

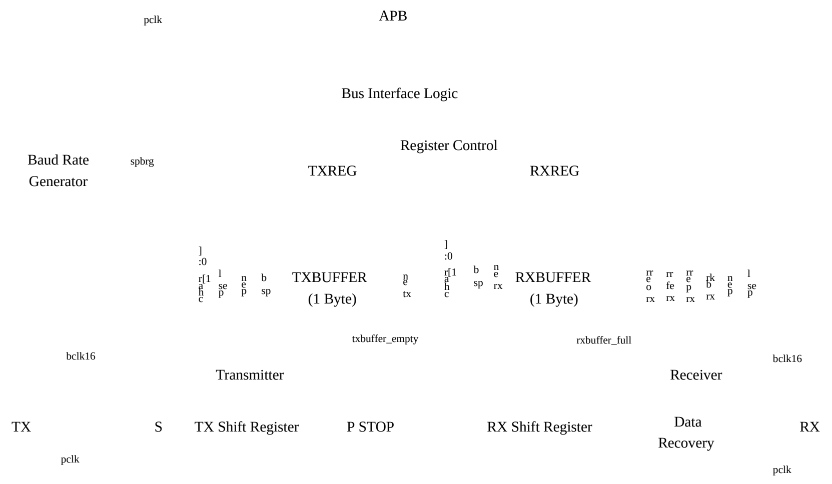
- The bus should be idle before sending or receiving
- A start bit
- One data word (5, 6, 7 or 8 bits), with the least significant bit first
- 1.5, 2 stop bits, which indicate the end of the data frame
- Use 16-bit baud rate generator

The following pins are required in hardware flow control mode:

- nCTS: Clear to send, if it is high, the next data transmission will be blocked at the end of the current data transmission.

- nRTS: Send request. If it is low, it indicates that the UART is ready to receive data.

Figure **168. UART** block diagram



22.3.1 UART feature description

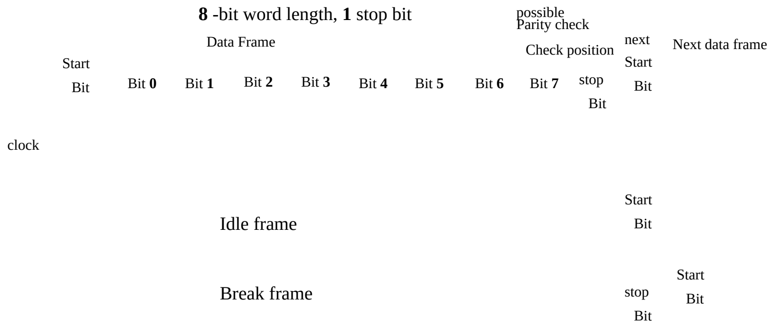
The word length can be selected from 5 to 8 bits by programming the CHAR bit in the UART_CCR register. During the start bit, the TX pin is in low power
It is high during the stop bit period.

The idle symbol is regarded as a complete data frame composed entirely of '1', followed by the start bit of the next frame containing the data ('1'
The number of digits also includes the number of stop bits).

The disconnection symbol is regarded as receiving all '0's in one frame period (including the stop bit period, which is also '0'). At the end of the disconnected frame, send
The transmitter inserts 1 or 2 stop bits ('1') to respond to the start bit.

Transmission and reception are driven by a shared baud rate generator. When the enable bits of the transmitter and receiver are set respectively, they will be generated respectively. clock.

Figure **169. UART** timing



22.3.2 Transmitter

The transmitter sends 5 to 8-bit data words according to the status of the CHAR bit. When the transmit enable bit (TXEN) is set, the transmit shift register
The data in is output on the TX pin, and the corresponding clock pulse is output on the SCLK pin.

Character sending

During UART transmission, the least significant bit of data is first shifted out on the TX pin. In this mode, the UART_TDR register contains
There is a buffer between the internal bus and the transmit shift register.

There is a low-level start bit before each character; the number of stop bits that follow is configurable.

Note: The *TE* bit cannot be reset during data transmission , otherwise the data on the *TX* pin will be destroyed because the baud rate counter stops counting. Being
The current data transferred will be lost.

Configurable stop bit

The number of stop bits sent with each character can be programmed through the SPB bit.

The break frame is a 10-bit low level followed by a stop bit; or an 11-bit low level followed by a stop bit. It is impossible to transmit longer disconnect frames (length
Greater than 10 or 11 bits), otherwise the RXBRK_INTF bit of the interrupt status register will be set.

Configuration steps

1. Activate the UART by setting the UARTEN bit in the UART_GCR register.
2. Program the CHAR bit of UART_CCR to define the word length.
3. SPB program the number of stop bits in UART_CCR.
4. Set the TXEN bit in UART_GCR.
5. Use the UART_BRR register to select the required baud rate.
6. Write the data to be sent into the UART_TDR register (this action clears the TX_INTF bit). In the case of only one buffer,
   Repeat step 6 for each data to be sent.

Single byte communication

Clearing the TX_INTF bit is always done by writing to the data register. The TX_INTF bit is set by hardware, it indicates:

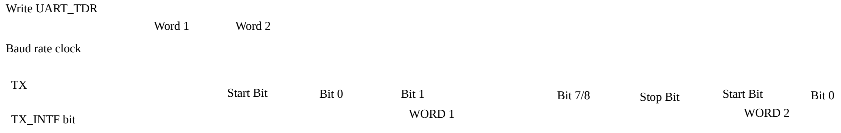
- The data has been transferred from the TDR to the shift register, and the data transmission has started
- TDR register is cleared

- The next data can be written into the UART_TDR register without overwriting the previous data.

If the TXIEN bit is set, this flag will generate an interrupt. If the UART is sending data at this time, send UART_TDR to
The write operation of the register stores the data into the TDR register, and copies the data into the shift register when the current transfer ends.

If the UART is not sending data at this time and is in an idle state, the write operation to the UART_TDR register directly puts the data in the shift
Bit register, when data transmission starts, TX_INTF bit is set immediately. At the same time, TXBUF_EMPTY of UART_CSR will also be set. when
When a frame is sent (after the stop bit is sent), and no new data is written to UART_TDR (TDR register is empty), TXC will be set
Bit, indicating that all transmissions have been completed.

Figure **170.** Status bit changes during transmission



WORD 1
Transmit Shift Reg.

TXDONE bit

WORD 2
Transmit Shift Reg.

Note: This timing diagram shows two consecutive transmissions.

Disconnect symbol

Set BRK to send a break symbol. If BRK=1 is set, after the current data transmission is completed, a break will be sent on the TX line Open symbol. The software must set BRK=0 when the transmission of the break character is completed (in the stop bit of the break symbol). UART in the last disconnected frame Insert a logic '1' at the end to ensure that the start bit of the next frame can be identified.

22.3.3 Receiver

Character reception

During USART reception, the least significant bit of data is first shifted in from the RX pin. In this mode, the UART_RDR register contains The buffer is located between the internal bus and the receive shift register.

Configuration steps:

1. Set UARTEN in the USART_GCR register to 1 to activate the UART.
2. Program the CHAR bit of USART_CCR to define the word length.
3. SPB program the number of stop bits in UART_CCR.
4. Use the UART_BRR register to select the required baud rate.
5. Set the RXEN bit of USART_GCR. Activate the receiver so that it starts looking for the start bit.

When a character is received,

- The RX_INTF bit is set. It indicates that the contents of the shift register are transferred to RDR. In other words, the data has been received and can be To be read (including error flags related to it).
- If the RXIEN bit is set, an interrupt is generated.
- If a frame error or an overflow error is detected during reception, the error flag will be set.
- Software reads the UART_RDR register. The RX_INTF bit must be cleared before the end of the next character reception.

Note: When receiving data, the *RXEN* bit should not be reset. If the *RXEN* bit is cleared during reception, the reception of the current byte is lost.

Disconnect symbol

When a disconnect frame is received, the UART will set the RXBRK_INTF interrupt.

Overflow error

If another character is received before UART_RDR is read, an overflow error will occur.

When an overflow error occurs:

- The RXOERR_INTF bit is set.
- RDR content will not be lost. Reading the UART_RDR register can still get the previous data.
- The previous content in the shift register will be overwritten. All subsequent data received will be lost.
- If the RXOERREN bit is set, an interrupt is generated.

Frame error

A frame error is detected when the stop bit is not received and recognized at the expected time. When a frame error is detected:

- The RXFERR_INTF bit is set by hardware.
- Invalid data will not be transferred from the shift register to the UART_RDR register.
- If the RXFERREN bit is set, an interrupt is generated.

22.3.4 Baud Rate Generator (**BRG**)

BRG is a dedicated 16-bit baud rate generator. The UART_BRR register controls the counting period of the 16-bit free-running counter.

Provide the desired baud rate and Fosc (APB clock frequency), use the value calculated by the formula shown in the table below to calculate an approximate integer value and assign it t UART_BRR (SPBRG) register. The X in the following table is equal to the value of the UART_BRR (SPBRG) register (1~65535). same The error of the baud rate can also be calculated at the time.

Table **40.** Baud rate formula

model	Formula

UART mode	Baud rate=Fosc/16X

X = SPBRG register value (1 to 65535)

Example **4-1** The following is an example of calculating the baud rate error:

Fosc = 100 MHz

Expected baud = 9600

Expected baud rate = Fosc/16X

9600 = 100000000 / 16X

X = 651.04 = 651

Calculated baud rate = 100000000/16 * 651 = 9600.6

Error = (calculated baud rate-expected baud rate) / expected baud rate

= (9600.6-9600) / 9600

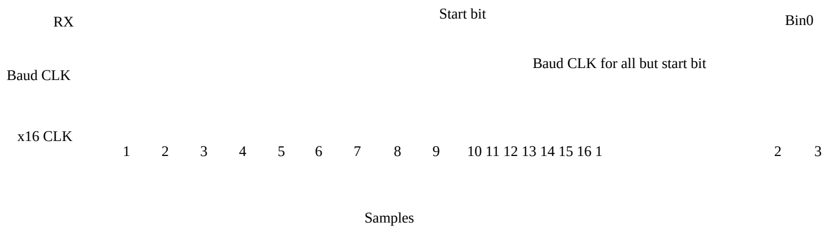
= 0.006%

Writing a new value to the SPBRG register will reset (or clear) the BRG counter. This feature ensures that BRG does not wait until the next The new baud rate will only be generated after the count overflows.

22.3.5 Sampling

Since there is no separate clock for asynchronous operation, the receiver needs a method that is synchronized to the receiver. In order to be able to get at the receiving pin "RX" For correct character data, the UART has a detection circuit. UART uses 16 times the data baud rate "bclk16" clock for sampling RX pin The data of the pin, each data has 16 clock samples, and the sampling value of the falling edge of the 7th, 8th and 9th in the middle is taken.

Figure **171. RX** pin sampling scheme



22.3.6 Check Control

Parity control (a parity bit is generated during transmission and parity check is performed during reception) by setting the PEN on the UART_CCR register Bit to activate. If a parity error occurs, invalid data will not be transferred from the shift register to the UART_RDR register.

Even parity: The parity bit makes the data in a frame and the number of '1's in the parity bit an even number.

For example: data=00110101, there are 4 '1's, if even parity is selected (PSEL=0 in UART_CCR), the parity bit will be It is '0'.

Odd parity: This parity bit makes the data in a frame and the number of '1' in the parity bit an odd number.

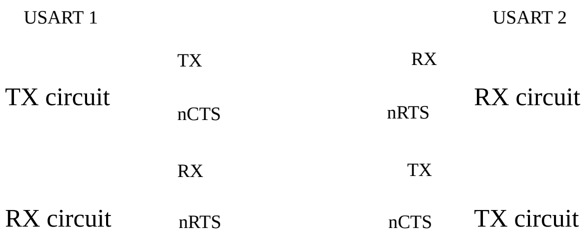
For example: data=00110101, there are 4 '1's, if odd parity is selected (PSEL=1 in UART_CCR), the parity bit will be it's 1'.

Transmission mode: If the PEN bit of UART_CCR is set, the MSB bit of the data written into the data register is sent after the check bit is replaced Send out (if you choose even parity and even number of '1', if you choose odd parity and odd number of '1'). If the parity check fails, UART_ISR The RXPERR_INTF flag in the register is set to '1', and if RXPERREN is set in advance, an interrupt is generated.

22.3.7 Hardware flow control

Use nCTS input and nRTS output to control the serial data flow between 2 devices. The figure below shows how to connect in this mode 2 Devices.

Figure **172.** Hardware flow control between two **USARTs**

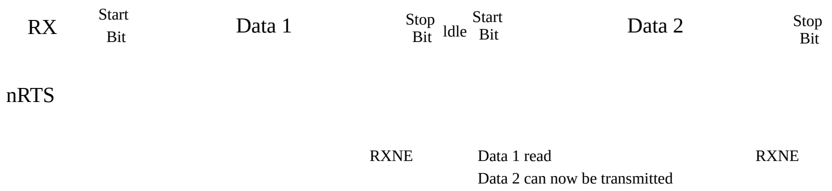


By setting AUTOFLOWEN in UASRT_GCR, RTS and CTS flow control can be enabled.

RTS flow control

If RTS flow control is enabled, as long as the UART receiver is ready to receive new data, nRTS becomes valid (connected to low level). when When data arrives in the receiving register, nRTS is released, which indicates that it is hoped to stop data transmission at the end of the current frame. The picture below is an enable An example of RTS flow control communication.

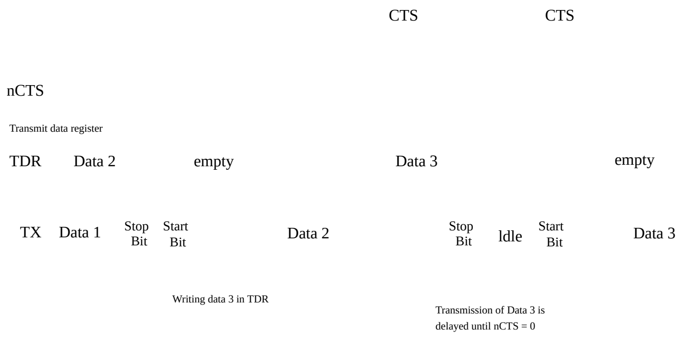
Figure **173. RTS** flow control



CTS flow control

If CTS flow control is enabled, the transmitter checks the nCTS input before sending the next frame. If nCTS is valid (pulled to low level), Then the next data is sent (assuming that data is ready to be sent), otherwise the next frame of data is not sent. If nCTS is It becomes invalid and stops sending after the current transmission is completed. The figure below is an example of communication with CTS flow control enabled.

Figure **174. CTS** flow control



22.3.8 Communication using **DMA**

UART can use DMA to communicate.

Use **DMA to** send

When using DMA for transmission, first configure the address of the UART_TDR register as the DMA transmission address on the DMA control register Destination address, configure the memory address as the source address of DMA transfer, and configure the amount of data transferred. By setting the UART_GCR register DMAMODE bit to activate DMA mode. When the TXEN bit is set to '1', DMA will transfer data from the designated SRAM area to UART_TDR register.

Use **DMA to** receive

When using DMA to receive, first configure the address of the UART_RDR register as the DMA transfer address on the DMA control register Source address, configure the memory address as the destination address of DMA transfer, and configure the amount of data transferred. By setting the UART_GCR register DMAMODE bit to activate DMA mode. When the RXEN bit is enabled, every time a byte is received, the DMA transfers the data from the UART_RDR The register is transferred to the designated SRAM area.

22.4 UART Interrupt Request

Table **41. UART** Interrupt Request

Interrupt event	Interrupt state	Enable bit
Send buffer empty	TX_INTF	TXIEN
Received valid data	RX_INTF	RXIEN
Receive overflow error	RXOERR_INTF	RXOERREN
Parity error	RXPERR_INTF	RXPERREN
Frame error	RXFERR_INTF	RXFERREN
UART receive disconnect frame	RXBRK_INTF	RXBRKEN

If the corresponding interrupt enable control bit is set, these settings can generate their corresponding interrupts.

Page 368

22.5 UART register description

22.5.1 UART Transmit Data Register (**UART_TDR**)

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								TXREG[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserved, always read as 0.
Bit 7:0	**TXREG** : Transmit data register

22.5.2 **UART** Receive Data Register (**UART_RDR**)

Offset address: 0x04

Reset value: 0x0000

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								RXREG[7:0]							
								r	r	r	r	r	r	r	r

Bit 31: 8	Reserved, always read as 0
Bit 7:0	**RXREG** : UART receive data register (Receive data register) This register is read-only.

22.5.3 **UART** Current Status Register (**UART_CSR**)

Offset address: 0x08

Reset value: 0x0009

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												TXBUF_EMPTY	TXFULL	RXAVL	TXC
												r	r	r	r

Bit 31: 4	Reserved, always read as 0.
Bit 3	**TXBUF_EMPTY** : Transmit buffer enpty flag bit 1 = Send buffer is empty 0 = Send buffer is not empty
Bit 2	**TXFULL** : Transmit buffer full flag bit 1 = Send buffer is full 0 = Send buffer is not full
Bit 1	**RXAVL** : Receive valid data flag bit This bit is set when the receive buffer has received a complete byte of data. 1 = Receive buffer has received a complete and valid byte data 0 = receive buffer is empty
Bit 0	**TXC** : Transmit complete flag bit 1 = Both transmit buffer and transmit shift register are empty

0 = send not empty

22.5.4 **UART** Interrupt Status Register (**UART_ISR**)

Offset address: 0x0C

Reset value: 0x0000

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									RXBRK_INTF	RXFERR_INTF	RXPERR_INTF	RXOERR_INTF	Reserve	RX_INTF	TX_INTF
									r	r	r	r		r	r

Bit 31: 7 Reserved, always read as 0.

Page 370

Bit 6	**RXBRK_INTF** : UART receive frame break interrupt flag bit (Receive frame break interrupt flag bit) After the abnormal stop bit, the RX pin receives 10 or more low levels within a period of time 1=Detect broken frame 0=No disconnected frame
Bit 5	**RXFERR_INTF** : Frame error interrupt flag bit Framing errors occur when abnormal stop bits are detected. 1 = Detect a frame error 0 = no framing error
Bit 4	**RXPERR_INTF** : Parity error interrupt flag bit 1 = Parity error detected 0 = no parity error
Bit 3	**RXOERR_INTF** : Receive overflow error interrupt flag bit Set only when autoflowen=0 1 = Receive overflow error 0 = no overflow error
Bit 2	Reserve
Bit 1	**RX_INTF** : Receive valid data interrupt flag bit (Receive valid data interrupt flag bit) This bit is set when the receive buffer has received a complete byte of data. 1 = Receive buffered valid byte data 0 = receive buffer is empty
Bit 0	**TX_INTF** : Transmit buffer enpty interrupt flag bit 1 = send buffer empty 0 = Send buffer is not empty

22.5.5 **UART** Interrupt Enable Register (**UART_IER**)

Offset address: 0x10

Reset value: 0x0000

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									RXBRKEN	RXFERREN	RXPERREN	RXOERREN	TIMEOUTEN	RXIEN	TXIEN
									rw	rw	rw	rw	rw	rw	rw

Bit 31: 7 Reserve

Bit 6	**RXBRKEN** : UART receive frame break interrupt enable bit (Receive frame break interrupt enable bit) 1=Interrupt enable 0=Interrupt disabled

Bit 5	**RXFERREN** : Frame error interrupt enable bit 1=Interrupt enable 0=Interrupt disabled
Bit 4	**RXPERREN** : Parity error interrupt enable bit 1=Interrupt enable 0=Interrupt disabled
Bit 3	**RXOERREN** : Receive overflow error interrupt enable bit 1=Interrupt enable 0=Interrupt disabled
Bit 2	**TIMEOUTEN** : Receive timeout interrupt enable bit (Receive timeout interrupt enable bit) 1=Interrupt enable 0=Interrupt disabled
Bit 1	**RXIEN** : Receive buffer interrupt enable bit 1=Interrupt enable 0=Interrupt disabled
Bit 0	**TXIEN** : Transmit buffer enpty interrupt enable bit 1=Interrupt enable 0=Interrupt disabled

22.5.6 **UART** Interrupt Clear Register (**UART_ICR**)

Offset address: 0x14

Reset value: 0x0000

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									RXBRKCLR	RXFERRCLR	RXPERRCLR	RXOERRCLR	TIMEOUTCLR	RXICLR	TXICLR
									w	w	w	w	w	w	w

Bit 31: 7	Reserve
Bit 6	**RXBRKCLR** : Receive frame break interrupt clear bit (Receive frame break interrupt clear bit) 1=Interrupt clear 0=The interrupt is not cleared
Bit 5	**RXFERRCLR** : Frame error interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 4	**RXPERRCLR** : Parity error interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared

Bit 3	**RXOERRCLR** : Receive overflow error interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 2	**TIMEOUTCLR** : Receive timeout interrupt clear bit The CPU must read RXREG before clearing the interrupt 1=Interrupt clear 0=The interrupt is not cleared
Bit 1	**RXICLR** : Receive interrupt clear bit 1=Interrupt clear 0=The interrupt is not cleared
Bit 0	**TXICLR** : Transmit buffer empty interrupt clear bit 1 = Interrupt clear 0=The interrupt is not cleared

22.5.7 **UART** Global Control Register (**UART_GCR**)

Offset address: 0x18

Reset value: 0x0000

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											TXEN	RXEN	AUTO FLOW EN	DMA MOD E	UART EN
											rw	rw	rw	rw	rw

Bit 31: 5	Reserve
Bit 4	**TXEN** : Enable transmit 1=send enable 0=Send is prohibited. Can clear TX BUFFER
Bit 3	**RXEN** : Enable receive 1 = receive enable 0 = Reception is disabled. You can clear RX BUFFER.
Bit 2	**AUTOFLOWEN** : Automatic flow control enable bit 1 = Automatic flow control is enabled 0 = automatic flow control disabled
Bit 1	**DMAMODE** : DMA mode selection bit 1 = select DMA method 0 = select normal mode
Bit 0	**UARTEN** : UART module selection bit (UART mode selection bit) 1 = UART module is enabled 0 = UART module disabled

22.5.8 **UART** General Control Register (**UART_CCR**)

Offset address: 0x1C

Reset value: 0x0030

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										CHAR		BRK	SPB	PSEL	PEN
										rw	rw	rw	rw	rw	rw

Bit 31: 6	Reserve
Bit 5: 4	**CHAR** : UART data bit width bit (UART width bit) 00 = 5-bit data 01 = 6-bit data 10 = 7-bit data 11 = 8-bit data (default)
Bit 3	**BRK** : UART transmit frame break 1 = Serial forced output logic "0" (disconnect frame) 0 = Disable disconnection
Bit 2	**SPB** : Stop bit selection Set the number of send stop bits. The receiver usually checks for a stop bit. 1 = 2 stop bits (when the data bit is 5 bits, the stop bit is 1, 5 other cases are 2 stop bits) 0 = 1 stop bit
Bit 1	**PSEL** : Parity selection bit When the check is enabled, this bit is used to select whether to use even check or odd check. 1 = even parity 0 = odd parity
Bit 0	**PEN** : Parity enable bit 1 = Send and receive enable check 0 = disable verification

22.5.9 UART Baud Rate Register (UART_BRR)

Offset address: 0x20

Reset value: 0x0001

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
SPBRG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 15	Reserve
Bit 15:0	**SPBRG** : UART baud rate control register (UART baud rate control register) is used to generate the baud rate Baud. Baud rate = Fosc / 16SPBRG (Fosc is the APB clock frequency) Special note: When SPBRG is 0x1, the baud rate = Fosc/16X2.

23. Secure digital input / output interface (**SDIO**)

23.1 Main Features of **SDIO**

SD/SDIO MMC card host interface (SDIO) provides APB2 peripheral bus and multimedia card (MMC), SD card and SDIO Interface between card devices.

SDIO has the following characteristics:

- Fully compatible with multimedia card system specification version 2.0-4.2. The card supports two different data bus modes: 1 bit (default),
- 4 bit
- Fully compatible with previous versions of multimedia cards (forward compatibility)
- Fully compatible with SD memory card specification version 1.0
- Fully compatible with SD memory card specification version 1.1 (high speed)
- Fully compatible with SD memory card specification version 2.0 (SDHC)
- Fully compatible with SD I/O card specification version 1.1.0: The card supports two different data bus modes: 1-bit (default) and 4-bit
- Support standard MMC model interface
- Programmable clock frequency
- Automatic command/response CRC generation/detection
- Automatic data CRC generation/detection

23.2 SDIO Register

The device communicates with the system through a 32-bit control register (accessible through APB2).

Peripheral registers must be accessed in words (32 bits).

23.2.1 SDIO mmc_ctrl

Offset address: 0x00

Reset value: 0x0045

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve					RDWTEN	INTEN	MDEN	DATWT	SPM	CLKSP			OUTM	SelSM	OPMSel
					rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 31: 11	Reserved, always read as 0.
Bit 10	RDWTEN: SDIO read wait enable signal (SDIO read wait enable signal) 1: SDIO read wait enable

	0: SDIO read waiting forbidden
Bit 9	INTEN: SDIO interrupt enable signal (SDIO interrupt enable signal) 1: SDIO interrupt enable 0: SDIO interrupt is disabled
Bit 8	MDEN: SDIO mode selection bit (SDIO mode enable) 1: SDIO mode 0: SD/MMC mode

Bit 7	DATWT: Define the bus width of SD/MMC/SDIO port data transmission width bit (Define the bus width of SD/MMC/SDIO port DAT line) 1: 4bit 0: 1bit
Bit 6	SelPTSM: SD/MMC/SDIO port transfer rate selection bit (Select SD/MMC/SDIO port transfer speed mode) 1: SD/MMC/SDIO port high-speed transmission mode 0: SD/MMC/SDIO port low-speed transmission mode
Bit 5: 3	CLKSP: SD/MMC/SDIO port clock CLK rate selection bit (SD/MMC/SDIO port CLK line speed selection) 000: 1/2 base clock 001: 1/4 base clock 010: 1/6 base clock 011: 1/8 base clock 100: 1/10 base clock 101: 1/12 base clock 110: 1/14 base clock 111: 1/16 base clock
Bit 2	OUTM: SD/MMC/SDIO port CMD output driver selection bit (SD/MMC/SDIO port CMD line output driver mode selection) 1: Open drain output 0: Push-pull output
Bit 1	SelSM: Select Signal mode (Select Signal mode) 1: SD/MMC/SDIO port automatic transmission mode 0: SD/MMC/SDIO port uses mmc_port register
Bit 0	OPMSel: SD/MMC/SDIO port operation mode select bit (SD/MMC/SDIO port operation mode select) 1: SD/MMC/SDIO mode 0: SPI mode

23.2.2 SDIO mmc_io

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve						CMDAF	CMDCH	AUTOCLKG	ENRRESP	PCLKG	CID/CSDRD	RESPCMDSEL	AUTOTR	TRANSFDIR	AUTODATTR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 10	Reserved, always read as 0.

Bit 9	CMDAF: SDIO cmd12/IO Abort flag (SDIO cmd12/IO Abort flag) 1: The mark is currently a cmd/IO abort command 0: The flag is not currently a cmd/IO abort command
Bit 8	CMDCH: SDIO command character (SDIO command character) 1: Mark the current command followed by the data block 0: Mark that there is neither a data block nor a response after the current command
Bit 7	AUTOCLKG: Enable automatic generation of 8 empty clock bits after response/command/single data block (Enable auto gnerate 8 null clock after response/command or single block data) 1: enable 0: prohibited
Bit 6	ENRRESP: Enable auto receive response after command 1: enable 0: prohibited
Bit 5	PCLKG: the SD / the MMC / CLK on the SDIO port bit clock (SD 8 space-time / the MMC / null the SDIO Port CLK. 8 Line clocks generation) 1: 8 empty clocks are generated 0: Receive response/transmit command selection via bit 3
Bit 4	CID/CSDRD: CID/CSD read control bit (CID and CSD read) 1: The read CID/CSD is stored in buffer[135:8] 0: invalid
Bit 3	RESPCMDSEL: When bit 5 is 0, the response/command selection bit (Response/Command selection when bit[5] is '0') 1: Receive response 0: Transmission command
Bit 2	AUTOTR: Set 8-bit null clock/command/response automatic transfer bit (Set auto 8null/command/response transfer) 1: Enable 8-bit empty clock/command/response automatic transmission 0: Disable automatic transmission of empty clock/command/response
Bit 1	TRANSFDIR: Set data transfer direction bit 1: Read data 0: write data
Bit 0	AUTODATTR: Set auto data transfer bit (Set auto data transfer) 1: Enable automatic data transmission 0: Prohibit automatic data transmission When the data transfer is complete, this bit will be cleared automatically.

23.2.3 SDIO mmc_bytecntl

Offset address: 0x08

Reset value: 0x0200

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

Bit 31: 16	Reserved, always read as 0.
Bit 15:0	Data transfer byte count register

23.2.4 SDIO mmc_tr_blockcnt

Offset address: 0xC

Reset value: 0x0000

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

Bit 31: 16	Reserved, always read as 0.
Bit 15:0	When multiple data blocks are transferred, the transfer is completed is the counter value (When multiple block transfer, transfer block count that are finished)

23.2.5 SDIO mmc_crcctl

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								CMD_CRCEN	DAT_CRCEN	ENCHK	ENRDMB	DAT_CRCS		CMD_CRCE	DAT_CRCE
								rw	rw	rw	rw	rw		r	r

Bit 31: 8	Reserved, always read as 0.
Bit 7	CMD_CRCEN: CRC calculation control bit of SD/MMC/SDIO port CMD (SD/MMC/SDIO port CMD Line CRC circuit enable) 1: enable 0: prohibited
Bit 6	DAT_CRCEN: CRC calculation control bit of SD / MMC/SDIO port DAT (SD / MMC/SDIO port DAT Line CRC circuit enable) 1: enable 0: prohibited

Bit 5	ENCHK: Enable auto check crc_status[2:0] enable bit (Enable auto check crc_status[2:0]) 1: Enable. If crc_status[2:0] != 3'b010, a crc error status interrupt is generated, and the write data transfer will be stopped by a stop command. Order to terminate and clear mmc_io[0] and mmc_io_mbctl[2:0] 0: prohibited Note: please refer to mmc_sig register for crc_status[2:0] bits
Bit 4	ENRDMB: Enable read multiple block data before receiving response (Enable read multiple block data before response) 1: enable 0: prohibited
Bit 3: 2	DAT_CRCS: DAT CRC selects dat_crcl and dat_crch register to store CRC value control bit (DAT CRC selection) 00: CRC value of SD/MMC/SDIO DAT0 or MMC DAT 01: CRC value of SD/MMC/SDIO DAT1 10: CRC value of SD/MMC/SDIO DAT2 11: CRC value of SD/MMC/SDIO DAT3
Bit 1	CMD_CRCE: CMD CRC check flag (CMD CRC Error). Read only.
Bit 0	DAT_CRCE: DAT CRC check mark bit (DAT CRC Error). Read only.

23.2.6 SDIO cmd_crc

Offset address: 0x14

Reset value: 0x0000

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

Bit 31: 7	Reserved, always read as 0.
Bit 6:0	CMD_CRCV: CMD CRC value (CMD CRC register value)

23.2.7 SDIO dat_crcl

Offset address: 0x18

Reset value: 0x0000

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

Bit 31: 8	Reserved, always read as 0.
Bit 7:0	DAT_CRCLV: The DAT CRC low register value

Page 380

23.2.8 SDIO dat_crch

Offset address: 0x1C

Reset value: 0x0000

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

Bit 31: 8	Reserved, always read as 0.
Bit 7:0	DAT_CRCHV: The DAT CRC high register value

23.2.9 SDIO mmc_port

Offset address: 0x20

Reset value: 0x007f

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								PCLKS	PCMDS	PDATS	AUTONTEN	NTCR			
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserved, always read as 0.
Bit 7	PCLKS: SD/MMC/SDIO port CLK signal (SD/MMC/SDIO port CLK line signal)
Bit 6	PCMDS: SD/MMC/SDIO port CMD signal (SD/MMC/SDIO port CMD line signal)
Bit 5	PDATS: SD/MMC/SDIO port DAT signal (SD/MMC/SDIO port DAT line signal)
Bit 4	AUTONTEN: Auto Ncr Timer out enable bit (Auto Ncr Timer out enable) 1: Enable automatic check of Ncr timeout 0: prohibit automatic checking of Ncr timeout
Bit 3: 0	NTCR: Ncr timeout counter (SD/MMC/SDIO clock count) (Ncr Timeout count register(SD/MMC/SDIO clock number))

23.2.10 SDIO mmc_int_mask

Offset address: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve							D1INTM	CRCINTM	CRTINTM	MBTINTM	MBDINTM	CMDEINTM	DATEINTM	DATDINT	CMDDINTM
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 9	Reserved, always read as 0.
Bit 8	D1INTM: SDIO data1 line interrupt mask (SDIO data1 line interrupt mask) 1: Open request 0: Block request
Bit 7	CRCINTM: CRC status token err interrupt mask 1: Open request 0: Block request
Bit 6	CRTINTM: Command and response Ncr timeout interrupt mask bit (Cmd and Resp Ncr Timeout interrupt mask) 1: Open request 0: Block request
Bit 5	MBTINTM: Multi Block Timeout interrupt mask (Multi Block Timeout interrupt mask) 1: Open request 0: Block request
Bit 4	MBDINTM: Multi Block done interrupt mask (Multi Block done interrupt mask) 1: Open request 0: Block request
Bit 3	CMDEINTM: CMD CRC error interrupt mask (CMD CRC error interrupt mask) 1: Open request 0: Block request
Bit 2	DATEINTM: DAT CRC error interrupt mask (DAT CRC error interrupt mask) 1: Open request 0: Block request
Bit 1	DATDINTM: DAT done interrupt mask (DAT done interrupt mask) 1: Open request 0: Block request
Bit 0	CMDDINTM: CMD done interrupt mask bit (CMD done interrupt mask) 1: Open request 0: Block request

Note: When other interrupts are generated, the CRC status bit, Ncr timeout, CMD CRC error and DAT CRC error interrupt are invalid.

23.2.11 SDIO clr_mmc_int

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							D1MC	CRCEMC	CRNTMC	MBTMC	MBDMC	CMDEMC	DATEMC	DATDMC	CMDDMC
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 9	Reserved, always read as 0.

Bit 8	D1MC: SDIO data1 line interrupt mask/clear (SIDO data1 line interrupt mask/clear) W (write): clear SDIO data1 line interrupt flag R (read): SDIO data1 line interrupt flag
Bit 7	CRCEMC: CRC status token err interrupt mask/clear W (write): Clear the CRC status error flag interrupt flag R (read): CRC status error flag interrupt flag
Bit 6	CRNTMC: Command and response Ncr timeout interrupt mask/clear (Cmd and Resp Ncr Timeout interrupt mask/clear) W (write): Clear the command and respond to the Ncr timeout interrupt flag R (read): Command and response Ncr timeout interrupt flag
Bit 5	MBTMC: Multi Block Timeout interrupt mask/clear W (write): Clear the multi-block transmission timeout interrupt flag R (read): Multi-block transmission timeout interrupt flag
Bit 4	MBDMC: Multi Block Done interrupt mask/clear W (write): Clear the multi-block transfer complete interrupt flag R (read): Multi-block transfer complete interrupt flag
Bit 3	CMDEMC: CMD CRC error interrupt mask/clear W (write): Clear the CMD CRC error interrupt flag R (read): CMD CRC error interrupt flag
Bit 2	DATEMC: DAT CRC error interrupt mask/clear W (write): Clear the DAT CRC error interrupt flag R (read): DAT CRC error interrupt flag
Bit 1	DATDMC: DAT done interrupt mask/clear W (write): clear the DAT completion interrupt flag R (read): DAT complete interrupt flag
Bit 0	CMDDMC: CMD done interrupt mask/clear W (write): Clear the CMD completion interrupt flag R (read): CMD complete interrupt flag

23.2.12 SDIO mmc_cardsel

Offset address: 0x2C

Reset value: 0x0040

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

Bit 31: 8	Reserved, always read as 0.
Bit 7	CTREN: SD/MMC/SDIO controller enable bit (SD/MMC/SDIO controller enable)
Bit 6	ENPCLK: Enable SD/MMC/SDIO port CLK clock (Enable SD/MMC/SDIO port CLK line for card)

Bit 5:0	TSCALE: SD / MMC/SDIO clock division factor (based on 1Mhz) (SD / MMC/SDIO Time scale base (1Mhz) coefficient) 1MHz = Fpclk/((mmc_cardssel[5:0] + 1)*2)

23.2.13 SDIO mmc_sig

Offset address: 0x30

Reset value: 0x00ff

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								PCMDS				PDAT3S	PDAT2S	PDAT1S	PDAT0S
								r	r	r	r	r	r	r	r

Bit 31: 8	Reserved, always read as 0.
Bit 7	PCMDS: SD / MMC/SDIO port CMD Line signal (SD / MMC/SDIO port CMD Line signal)
Bit 6: 4	CRC status[2:0] CRC status token when writing data (CRC status[2:0] when write data CRC status token)
Bit 3	PDAT3S: SD / MMC/SDIO port DAT3 Line signal (SD / MMC/SDIO port DAT3 Line signal)
Bit 2	PDAT2S: SD / MMC/SDIO port DAT2 Line signal (SD / MMC/SDIO port DAT2 Line signal)
Bit 1	PDAT1S: SD / MMC/SDIO port DAT1 Line signal (SD / MMC/SDIO port DAT1 Line signal)
Bit 0	PDAT0S: SD / MMC/SDIO port DAT0 line signal (SD / MMC/SDIO port DAT0 Line signal)

Note: When the master device reads the register, the SD/MMC/SDIO controller will generate a periodic pulse on the CLK line of the SD/MMC/SDIO port Chong, and SD / MMC / SDIO port state signal will be latched into the register as rising SD / MMC / SDIO terminal CLK line.

23.2.14 SDIO mmc_io_mbctl

Offset address: 0x34

Reset value: 0x0010

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								NTSSel		BTSSel		PCLKP	PAUTOTR	SMBDTD	SPMBDTR
								rw		rw		rw	rw	rw	rw

Bit 31: 8	Reserved, always read as 0.
Bit 7: 6	NTSSel: SD / MMC/SDIO Nac timeout level selection bit (SD / MMC/SDIO N AC timeout scale selection) 00: 1us 01: 100us 10: 10ms 11: 1s

Bit 5: 4	BTSSel: SD / MMC/SDIO Busy timeout level selection bit (SD / MMC/SDIO Busy timeout scale selection) 00: 1us 01: 100us 10: 10ms 11: 1s
Bit 3	PCLKP: SD/MMC/SDIO port CLK line polarity selection bit (SD/MMC/SDIO port CLK line polarity) 1: Clock falling edge push, rising edge pull 0: clock rising edge push, falling edge pull
Bit 2	PAUTOTR: Enable SD / MMC/SDIO port fully automatic command and multi-data block transmission bit (Set SD / MMC/SDIO port full auto cmd and multiple block data transferring) 1: enable 0: prohibited Note: If mmc_io[7:6]==11, this position will trigger SD/MMC/SDIO commands, responses, 8 empty clocks, and more data Piece.
Bit 1	SMBDTD: Select multiple block data transfer direction 1: Read data 0: write data
Bit 0	SPMBDTR: Enable SD / MMC/SDIO port automatic multi-block data transmission bit (Set SD / MMC/SDIO port auto multiple block data transfer) 1: enable 0: prohibited

Note: This position one will trigger SD/MMC/SDIO multi-block data transmission, and the block counter is defined by the mmc_blockcnt register. when

After the data transfer is complete, this bit will be cleared automatically.

23.2.15 SDIO mmc_blockcnt

Offset address: 0x38

Reset value: 0x0001

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

Bit 31: 16	Reserved, always read as 0.
Bit 15:0	Data block number register In the multi-block transmission mode, configure these bits to define the number of data blocks to be transmitted.

23.2.16 SDIO mmc_timeoutcnt

Offset address: 0x3C

Reset value: 0x0040

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

Bit 31: 8	Reserved, always read as 0.
Bit 7:0	DTCNT: Data transfer timeout count register Time = Scale* bit[7:0] Note: Scale is defined according to mmc_io_mbctl[7:6]/[5:4].

23.2.17 SDIO cmd_bufx(x = 0..15)

Offset address: 0x40-0x7C

Reset value: 0x0000

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

Bit 31: 8	Reserved, always read as 0.
	cmd_buf byte x, mapped to command [(15+8x): 8(x+1)] bits cmd_buf byte 0, mapped to command [15:8] bits cmd_buf byte 1, mapped to command [23:16] bits cmd_buf byte 2, mapped to command [31:24] bits cmd_buf byte 3, mapped to command [39:32] bits cmd_buf byte 4, mapped to command [47:40] bits cmd_buf byte 5, mapped to command [55:48] bits cmd_buf byte 6, mapped to command [63:56] bits

Bit 7:0	cmd_buf byte 7, mapped to command [71:64] bits cmd_buf byte 8, mapped to command [79:72] bits cmd_buf byte 9, mapped to command [87:80] bits cmd_buf byte 10, mapped to command [95:88] bits cmd_buf byte 11, mapped to command [103:96] bits cmd_buf byte 12, mapped to command [111:104] bits cmd_buf byte 13, mapped to command [119:112] bits cmd_buf byte 14, mapped to command [127:120] bits cmd_buf byte 15, mapped to command [135:128] bits

Page 386

23.2.18 SDIO buf_ctl

Offset address: 0x80

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBFEN	DRM		DFIFOSM	SBAD	DMAHEN	DBML								DBE	DBF
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 16	Reserved, always read as 0.
Bit 15	DBFEN: Data Buf flush enable bit (Data Buf flush enable) 1: Trigger to clear the data buff 0: invalid Note: When this bit is 1, it will be cleared automatically after one clock cycle.
Bit 14	DRM: DMA request mask bit (Dma Requst mask) 1: Open request 0: Block request Note: mask this bit before enabling DMA configuration. After enabling DMA, this bit is set to 1, and DMA should be started.
Bit 13	Reserve
Bit 12	DFIFOSM: Data FIFO status signal mask bit 1: Activate display data FIFO status 0: Default value, mask data FIFO status Note: Data FIFO status bit, active is high. Active meaning, read card access, FIFO is full; write card access, FIFO is empty.
Bit 11	SBAD: Set buff access direction bit (Set buff access direction) 1: Write data 0: read data
Bit 10	DMAHEN: DMA hardware interface enable bit (DMA hardware interface enable) 1: DMA hardware interface handshake 0: normal APB access buff data Note: When using the DMA interface, this bit will be automatically reset when the block (single block transfer) or multi-block data transfer is completed.
Bit 9: 2	DBML: Data buff mark, this bit is valid only when buf_ctl[10] = 1. (Data buff data water mark level)
Bit 1	DBE: buff the data empty state (the Data buff empty). Read only.
Bit 0	DBF: Data buff full state (Data buff full). Read only.

23.2.19 SDIO data_buf

Offset address: 0x100-0x2FF

Reset value: 0x0000

31	30	29	28	27	26	25 24		twenty three	twenty two	twenty one	20	19	18	17	16
DB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:0	DB: data buff (Data buffer) Note: All visits within this range will be regarded as address AMBA read visits.

24. Controller Area Network (**CAN**)

24.1 Introduction to **CAN**

Its design goal is to efficiently process a large number of received messages with minimal CPU load. It also supports the priority requirements for message transmission (excellent

Advanced features can be configured by software).

For safety-critical applications, bxCAN provides all the hardware functions required to support the time-triggered communication mode.

24.2 Main Features of **CAN**

- Support 2.0A and 2.0B of CAN protocol
- Extended receive buffer (64 bytes, first in first out FIFO)
- Support both 11-digit and 29-digit identification codes
- Bit rate up to 1Mbits/s
- PeliCAN mode extended functions
  - Error counter for read/write access
  - Programmable error alarm limit
  - Last error code register
  - Interrupt for every CAN bus error
  - Arbitration lost interrupt controlled by specific control bit
  - Single transmission (no retransmission)
  - Listen only mode (no confirmation, no active error flags)
  - Software bit rate detection
  - Acceptance filter extension (4-byte code, 4-byte mask)
  - Self-reception (self-reception request)

24.3 General description of **CAN** controller

In today's CAN applications, the number of CAN network nodes is increasing, and multiple CANs are often connected through gateways. The number of messages in a CAN network (each node needs to be processed) has increased dramatically. In addition to application layer messages, network management and diagnostic mess Introduce.

An enhanced filtering mechanism is needed to handle various types of messages.

In addition, application layer tasks require more CPU time, so the degree of real-time response required for message reception needs to be reduced.

The receiving FIFO scheme allows the CPU to spend a long time processing application layer tasks without losing messages.

The high-level protocol software built on the low-level CAN driver requires an efficient interface with the CAN controller.

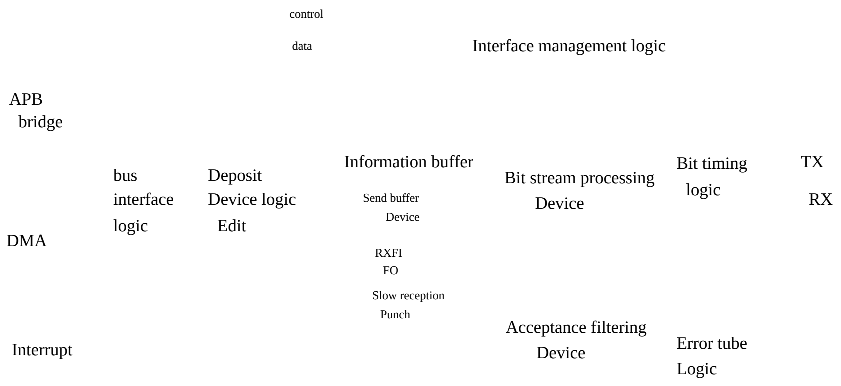
Figure **175. CAN** network topology

24.3.1 CAN 2.0B Active Core

The CAN module can automatically receive and send CAN messages; and fully supports standard identifiers (11 bits) and extended identifiers (29 Bit).

24.3.2 CAN block diagram

Figure **176. CAN** structure block diagram



24.3.3 Interface Management Logic (**IML**)

The interface management logic interprets the commands from the CPU and controls the addressing of the CAN register to provide interrupt information and status information to the ma

24.3.4 Transmit Buffer (**TXB**)

The transmit buffer is the interface between the CPU and the BSP (Bit Stream Processor), which can store the complete information sent to the CAN network. slow
The punch is 13 bytes long, which is written by the CPU and read by the BSP.

24.3.5 Receive buffer (**RXB** , **RXFIFO**)

The receiving buffer is the interface between the acceptance filter and the CPU to store the information received and received from the CAN bus. The receiving buffer
(RXB, 13 bytes) As a window of the receive FIFO (RXFIFO, 64 bytes long), it can be accessed by the CPU.

With the support of this FIFO, the CPU can receive other information while processing information.

24.3.6 Acceptance Filter (**ACF**)

The acceptance filter compares the data in it with the content of the received identification code to determine whether to receive the information. Pure reception test
In, all information is stored in RXFIFO.

24.3.7 Bit Stream Processor (**BSP**)

The bit stream processor is a program device that controls the data flow between the transmit buffer, RXFIFO and CAN bus. It's still in the CAN total
Perform error detection, arbitration, filling, and error handling online.

24.3.8 bit sequential logic (**BTL**)

The bit timing logic monitors the CAN bus of the serial port and processes the bit timing related to the bus. It is at the beginning of the message'weak-dominant' bus transmission
Synchronize the CAN bus bit stream (hard synchronization), and resynchronize the next transmission when receiving information (soft synchronization). BTL also provides programmable tim
Segment to compensate for propagation delay time, phase conversion, and to define the sampling point and the number of samples within one bit time.

24.3.9 Error Management Logic (**EML**)

EML is responsible for the error control of the transport layer module. It receives error reports from the BSP and informs the BSP and IML to perform error statistics.

24.4 CAN operating mode

The CAN controller has 2 main operating modes:

- BasicCAN mode
- PeliCAN mode

The default mode when the system is reset is BasicCAN mode.

PeliCAN mode is a new operating mode, which can handle all CAN 2.0B standard frame types. And it also provides some enhancements
It can be applied to a wider area.

Register CAN_CDR.7 defines the CAN mode. If CDR.7 is 0, the CAN controller works in BasicCAN mode. otherwise,
The CAN controller works in PeliCAN mode.

24.4.1 The difference between **Basic CAN** and **PeliCAN** modes

In Peli CAN mode, the CAN controller has a reorganized register with many functions. Peli CAN mode supports CAN 2.0B protocol
All functions specified by the protocol (29-byte identification code). The following are the main new features of PeliCAN mode:

- Reception and transmission of standard and extended frames
- Receive FIFO (64 bytes)
- There are single/double acceptance filters in standard and extended formats (including shielding and code registers)
- Error counter for read/write access
- Programmable error limit alarm
- Last bit error register
- Error interrupt for every CAN bus error
- Arbitration lost and detailed bit position
- One-time transmission (no retransmission in case of error or loss of arbitration)
- Listen only mode (CAN bus monitor, no response, no error flag)

24.5 CAN function description

24.5.1 Basic CAN mode

Reset mode

The reset mode is the initialization mode. Reset the request bit (CAN_CR.0) when the hardware is started or the bus status is set to '1' (bus off)
It is set to '1' (current). If these bits are accessed by software, their values will change and will affect the next rising edge of the internal clock. Reset
When the request bit changes, the read reset request bit synchronized with the internal frequency clock can reflect this synchronization state. The reset mode is mainly used for CAN communi
Parameter configuration, the kernel has different access rights to CAN registers in different working modes.

After the reset request bit is set to '0', the CAN controller will wait:

a) A bus idle signal (11 weak bits), if the previous reset request was a hardware reset or CPU initial reset.

b) 128 buses are idle, if the previous reset request was caused by the CAN controller initializing the bus before re-entering the bus-on mode
It must be noted that if the reset request bit is set, the value of some registers will be changed.

Operating mode

After the reset mode is completed, the software should let the hardware enter the normal mode in order to receive and send messages normally. In reset mode, once to
The reset bit transmits the falling edge of '1-0', and the CAN controller will return to working mode to send and receive messages.

Sleep mode

When the sleep mode bit is set to 1 (sleep), the CAN controller will enter sleep mode; there is no bus activity or interrupt waiting. Destroy at least these two
One of these situations will cause the sleep mode to generate a wake-up interrupt. The bus enters an active state or interrupts after the sleep mode bit is set to low (wake up)
Activated. After wake-up, the clock is started and a wake-up interrupt is generated. Wake up due to bus activity until 11 consecutive hidden (weak) detected
This message can be received only after the bit (bus idle sequence). Note that the sleep mode bit cannot be set in the reset mode. After clearing the reset mode,
When the bus is detected to be idle again, the setting of the sleep mode bit becomes effective.

Basic CAN mode register permission allocation table:

CAN address offset (HEX)	part	Operating mode		Reset mode	
		read	Write	read	Write
00	control	control	control	control	control
04		(FFH)	Order	(FFH)	Order
08		state	—	state	—
0C		(FFH)	—	Interrupt	—
10		(FFH)	—	Acceptance code	Acceptance code
14		(FFH)	—	Acceptance shield	Acceptance shield
18		(FFH)	—	Bus timing 0	Bus timing 0
1C		(FFH)	—	Bus timing 1	Bus timing 1
20		(FFH)	—	—	—
twenty four		test	test	test	test
28	Send buffer Device	Identification code (10 ~ 3)	Identification code (10 ~ 3)	(FFH)	—
2C		Identification code (2 ~ 0) RTR and DLC	Identification code (2 ~ 0) RTR And DLC	(FFH)	—
30		DATA1	DATA1	(FFH)	—

CAN address offset (HEX)	part	Operating mode		Reset mode	
		read	Write	read	Write
34	Receive buffer Device	DATA2	DATA2	(FFH)	—
38		DATA3	DATA3	(FFH)	—
3C		DATA4	DATA4	(FFH)	—
40		DATA5	DATA5	(FFH)	—
44		DATA6	DATA6	(FFH)	—
48		DATA7	DATA7	(FFH)	—
4C		DATA8	DATA8	(FFH)	—
50		Identification code (10 ~ 3)	Identification code (10 ~ 3)	Identification code (10 ~ 3)	Identification code (10 ~ 3)
54		Identification code (2 ~ 0) RTR and DLC	Identification code (2 ~ 0) RTR And DLC	Identification code (2 ~ 0) RTR and DLC	Identification code (2 ~ 0) RTR and DLC
58		DATA1	DATA1	DATA1	DATA1
5C		DATA2	DATA2	DATA2	DATA2
60		DATA3	DATA3	DATA3	DATA3
64		DATA4	DATA4	DATA4	DATA4
68		DATA5	DATA5	DATA5	DATA5
6C		DATA6	DATA6	DATA6	DATA6
70		DATA7	DATA7	DATA7	DATA7
74		DATA8	DATA8	DATA8	DATA8
78		(FFH)	—	(FFH)	—
7C		Clock divider	Clock divider	Clock divider	Clock divider

NOTE: ' (*FFH*) '' representative of the read data are all *1* , ' - *'* represents no write authorization, represents the remaining operable. Offset address *0x7C* ' clock The divider ' is used to select *BasicCAN* and *PeliCAN* .

24.5.2 Peli CAN mode

Reset mode

The reset mode is the initialization mode. When the hardware reset or the bus status bit is '1' (bus off), the reset mode bit is set to '1' (current). If this bit is accessed through software, the value will change and the rising edge of the next internal clock (frequency 1/2 of the external oscillator) will be valid. complex The change of the bit request bit is synchronized with the internal frequency clock. Reading the reset request bit can reflect this synchronization state. After the reset mode bit is '0', CAN The controller will wait:

a) A bus idle signal (11 hidden (weak) bits), if the last reset was a hardware reset or CPU initial reset.
b) 128 buses are idle, if the last reset was when the CAN controller initialized the reset before re-entering the bus and turning on.

Operating mode

After the reset mode is completed, the software should let the hardware enter the normal mode in order to receive and send messages normally. In reset mode, once checked It is detected that the RM bit of the CAN_MOD register has a falling edge of '1-0', and the CAN controller will return to the working mode to send and receive messages. receive.

Sleep mode

When the sleep mode bit (CAN_MOD.4) is set to 1 (sleep), the CAN controller will enter sleep mode; there is no bus activity or interruption, etc. treat. Damage to at least one of these two conditions will cause the sleep mode to generate a wake-up interrupt. After the sleep mode bit is set to low (wake up), the bus enters Enter the active state or the interrupt is activated. After wake-up, the clock is started and a wake-up interrupt is generated. Waking up due to bus activity until 11 detected

This message can only be received after a continuous hidden (weak) bit (bus idle sequence). Note that the sleep mode bit cannot be set in the reset mode of. After the reset mode is cleared, the setting of the sleep mode bit becomes effective when the bus is detected to be idle again.

Self-test mode

This mode is mainly used for testing. Set the self-test mode bit (CAN_MOD.2) to 1, enter the self-test mode. This mode can detect all There are nodes, but no active nodes use self-receiving commands; even if there is no response, the CAN controller will send it successfully.

Listen only mode

This is mainly used for testing. Set the listen-only mode bit (CAN_MOD.1) to 1 to enter the listen-only mode. This mode of operation makes the CAN controller Enter a negative state of error. Information transfer is impossible. Software-driven bit rate detection can use listen-only mode. All other functions can be like Use the same in normal working mode.

In this mode, the CAN controller cannot write dominant bits on the CAN bus. The activation error flag or the overload flag cannot be written at all, and the connection is successful. The response signal after receipt will not be given either.

Note: Before entering the listening-only mode, you must enter the reset mode.

Peli CAN mode register permission allocation table:

CAN address offset Move (HEX)	Operating mode				Reset mode	
	read		Write		read	Write
00	model		model		model	model
04	(00H)		Order		(00H)	Order
08	state		—		state	—
0C	Interrupt		—		Interrupt	—
10	Interrupt enable		—		Interrupt enable	Interrupt enable
14	(00H)		—		(00H)	—
18	Bus timing 0		—		Bus timing 0	Bus timing 0
1C	Bus timing 1		—		Bus timing 1	Bus timing 1
20	Reserve		—		—	—
twenty four	Detect		Detect		Detect	Detect
28	Reserve		—		Reserve	—
2C	Arbitration lost capture		—		Arbitration lost capture	—
30	Error code capture		—		Error code capture	—
34	False alarm limit		—		False alarm limit	False alarm limit
38	RX error counter		—		RX error counter	RX error counter
3C	TX error counter		—		TX error counter	TX error counter
40	RX frame information SFF	RX frame information EFF	TX frame information SFF	TX frame information EFF	Acceptance code 0	Acceptance code 0
44	RX identification code 1	RX identification code 1	TX identification code 1	TX identification code 1	Acceptance code 1	Acceptance code 1

CAN address offset Move (HEX)	Operating mode				Reset mode	
	read		Write		read	Write
48	RX identification code 2	RX identification code 2	TX identification code 2	TX identification code 2	Acceptance code 2	Acceptance code 2
4C	RX data 1	RX identification code 3	TX data 1	TX identification code 3	Acceptance code 3	Acceptance code 3
50	RX data 2	RX identification code 4	TX data 2	TX identification code 4	Acceptance mask 0	Acceptance mask 0
54	RX data 3	RX data 1	TX data 3	TX data 1	Acceptance shield 1	Acceptance shield 1
58	RX data 4	RX data 2	TX data 4	TX data 2	Acceptance shield 2	Acceptance shield 2
5C	RX data 5	RX data 3	TX data 5	TX data 3	Acceptance shield 3	Acceptance shield 3
60	RX data 6	RX data 4	TX data 6	TX data 4	Reserve	—
64	RX data 7	RX data 5	TX data 7	TX data 5	Reserve	—
68	RX data 8	RX data 6	TX data 8	TX data 6	Reserve	—
6C	(FIFO RAM)	RX data 7	—	TX data 7	Reserve	—
70	(FIFO RAM)	RX data 8	—	TX data 8	Reserve	—
74	RX information counter		—		RX information counter	—

78	RX buffer start address	—	RX buffer start site	RX buffer start site
7C	Clock divider	Clock divider	Clock divider	Clock divider
80	Internal RAM address 0 (FIFO)	—	Internal RAM address 0	Internal RAM address 0
84	Internal RAM address 1 (FIFO)	—	Internal RAM address 1	Internal RAM address 1
…	…		…	…
17C	Internal RAM address 63 (FIFO)	—	Internal RAM address 63	Internal RAM address 63
180	Internal RAM address 64 (TX buffer Device)	—	Internal RAM address 64	Internal RAM address 64
	…		…	…
1B0	Internal RAM address 76 (TX buffer Device)	—	Internal RAM address 76	Internal RAM address 76
1B4	Internal RAM address 77 (free)	—	Internal RAM address 77	Internal RAM address 77
1B8	Internal RAM address 78 (free)	—	Internal RAM address 78	Internal RAM address 78
1BC	Internal RAM address 79 (free)	—	Internal RAM address 79	Internal RAM address 79
1C0	(00H)	—	(00H)	—
	…		…	
1FC	(00H)	—	(00H)	—

24.5.3 Send processing

According to the CAN protocol specification, the transmission of messages is independently completed by the CAN controller. Microcontroller sets identifier, data length and to be sent Data; then the'send request' position '1' of the command register is sent to request transmission. When the CAN controller is sending a message, send the buffer It is write locked. Therefore, before preventing a new message from reaching the transmit buffer, the microcontroller must check the "transmit buffer status" flag of the status register. History (TBS).

Setting the command bits CMR.0 and CMR.1 will immediately generate a message transmission. When the transmission is wrong or the arbitration is lost, it will not be retransmitted (si Sent). Only set the command bit CMR.0. If data transmission fails, it will be retransmitted. In the self-test mode, set the command bits CMR.4 and CMR.1 Will immediately produce a self-receiving information transmission.

Suspend

For a message that has been requested to be sent, you can execute the “stop sending” by setting the corresponding bit of the command register bit, and send it to CAN_CMR. The AT bit in the register is '1', and the sending request can be aborted.

When the CPU needs to wait for the current request to be sent, for example, when sending an urgent message first. But the transfer currently being processed is not stop. If you want to know whether the source information is successfully sent, you can check it through the transfer complete status bit. However, this should be in the transmit buffer status bi Set to '1' or after generating a transmit interrupt.

It should be noted that even if the message is aborted because the transmit buffer status bit becomes "released", a transmit interrupt will be generated.

If the send request is set to '1' in the previous command, it cannot be cancelled by setting the send request bit to '0', but should be canceled by the abort sending bit Cancel for '0'.

24.5.4 Receiving management

The received message is independently completed by the CAN controller. The received message is placed in the receive buffer. Messages that can be sent to the microcontroller by The receiving buffer status flag "RBS" and the receiving interrupt flag "RI" of the status register are marked.

- Query control reception

The microcontroller reads the status register of the CAN controller and checks the receive buffer status (RBS) to see if a message has been received.

When it reads that the RBS bit is 1, it means that one or more messages have been received, and the microcontroller gets the message from CAN, and then sets the command register The response flag bit'RRB' sends a release receive buffer command.

- Interrupt control reception

The interrupt enable flag is located in the CAN controller register (for BasicCAN mode) or in the interrupt enable register (for PeliCAN mode). If the CAN controller has received a message, and the message has passed the acceptance filter and placed in the receive FIFO, then Then a receive interrupt will be generated. Enter the interrupt service routine, the microcontroller fetches the message, and then sets the response flag bit'RRB' of the command register Send a release receive buffer command.

overflow

When the receive FIFO is full but other messages are received, it will cause an overflow, and set the data overload status bit in the status register at the same time (If enabled) Notify the microcontroller that there is a data overflow, the CMR.3 bit is '1' to clear the overflow status.

Valid message

According to the CAN protocol, when the message is received correctly (there is no error until the last bit of the EOF field), and the identifier filter is passed, Then the message is considered to be a valid message.

The RRB bit in the CAN_CMR register is used to clear the data overflow condition indicated by the data overflow status bit.

24.5.5 Identifier filtering

In the CAN protocol, the identifier of the message does not represent the address of the node, but is related to the content of the message. Therefore, the sender The form sends the message to all receivers. When the node receives a message-according to the value of the identifier-decides whether the software needs the message; if If needed, it will be copied to SRAM; if not needed, the message will be discarded without software intervention.

The standalone CAN controller is equipped with a multifunctional acceptance filter that allows automatic checking of identifiers and data bytes. use These effective filtering methods can prevent messages or message groups that are invalid for a node from being stored in the receiving buffer. So reduce the micro control The processing load of the controller.

The filter is controlled by the acceptance code register and the mask register according to a given algorithm. The received data will be The values are compared bit by bit. The receive mask register defines the position relative to the comparison (0 = relevant, 1 = not relevant). Only corresponding to the received message If the corresponding bits in the acceptance code register are the same, the message will be received.

Acceptance filter in **BasicCAN** mode

The filter is controlled by two registers-Acceptance Code Register (ACR) and Acceptance Mask Register (AMR). CAN message identifier The upper 8 bits of the are compared with the values in these registers. Several group identifiers can be defined to be received by any node.

example:

Acceptance Code Register (ACR) includes:

	MSB LSB
Acceptance Code Register (ACR) includes:	0 1 1 1 0 0 1 0
Acceptance Mask Register (AMR) includes:	0 0 1 1 1 0 0 0
Message with 11-bit identifier is received	0 1 XXX 0 1 0 XXX
(X=irrelevant)	ID.10

At the position of '1' in the acceptance mask register, the corresponding bit of the identifier can be any value. This is the same for the three lowest bits. therefore In this example, 64 different identifiers can be received. The other bits of the identifier must be equal to the value of the corresponding bit of the acceptance code register.

Acceptance filter in **PeliCAN** mode

With the help of the acceptance filter, only when the identification bit in the received message is equal to the pre-defined value of the acceptance filter, the CAN controller will Allows the received information to be stored in the RXFIFO.

In PeliCAN mode, the acceptance filter is defined by the acceptance code register (ACRn) and the acceptance mask register (AMRn). To receive The bit pattern of the information is defined in the acceptance code register. The corresponding acceptance mask register allows certain bits to be defined as'do not affect' (that is, any value).

There are two different filtering modes that can be selected in bit 3 of the mode register:

- Single filter mode (1)
- Double filter mode (0)

Single filter configuration

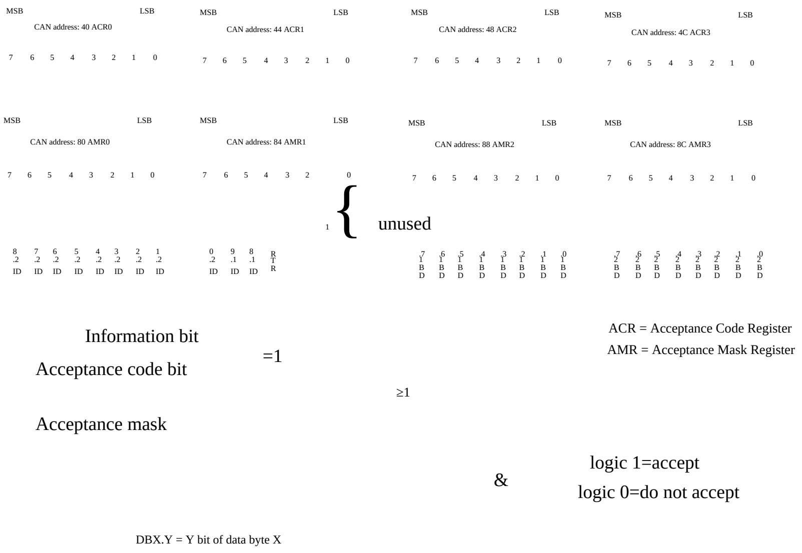
This filter configuration can define a long filter (4 bytes). The bit correspondence between the filter byte and the information byte depends on the current Frame format before receiving.

Standard frame: If the information received is the standard frame format, only the first two data bytes are used in the acceptance filter to store the RTR bit The complete identification code. If there is no data byte due to the setting of the RTR bit, or there is no data byte because of setting the corresponding data length code. There is only one data byte, and the information will also be received. For a successfully received message, all individual bits must be sent and received after the comparison

Signal.

Note: The lower four *bits of ACR1* and *AMR1* are not used. In order to be compatible with future products, these bits can be set by setting *AMR1.3* , *AMR1.2* , *AMR1.1* , *AMR1.0* to *1* and as *'* do not affect *'* .

Figure **177.** Single filter configuration when receiving standard structure information

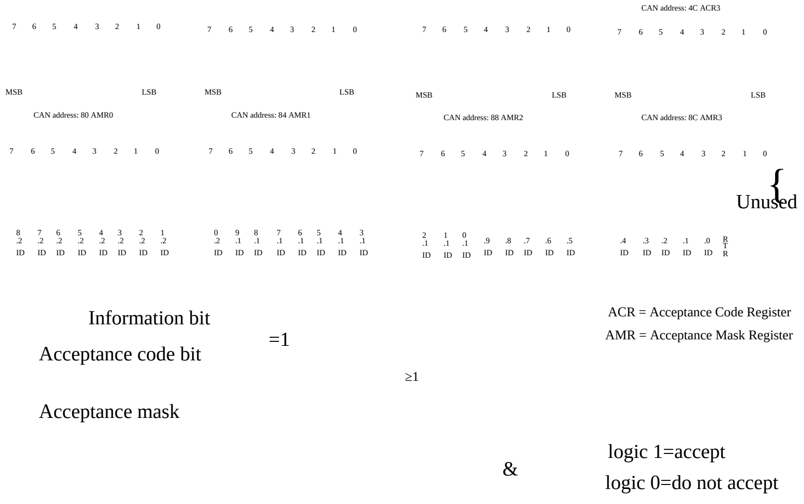


Extended frame: If the received information is in an extended frame format, all identification codes including RTR bits will be used by the reception filter.

In order to successfully receive information, a receive signal must be sent after each bit is compared.

Note: *AMR3* minimum two and *ACR3* is not used. These bits should be set by *AMR3.1* and *AMR3.0* to as ' do not affect *'* .

Figure **178.** Single filter configuration, receiving extended frame information


CAN address: 4C ACR3
MSB
LSB
CAN address: 80 AMR0
CAN address: 84 AMR1
CAN address: 88 AMR2
CAN address: 8C AMR3
Unused
ID.28 ID.27 ID.26 ID.25 ID.24 ID.23 ID.22 ID.21
ID.20 ID.19 ID.18 ID.17 ID.16 ID.15 ID.14 ID.13
ID.12 ID.11 ID.10 ID.9 ID.8 ID.7 ID.6 ID.5
ID.4 ID.3 ID.2 ID.1 ID.0 RTR
ACR = Acceptance Code Register
AMR = Acceptance Mask Register
Information bit
Acceptance code bit
=1
≥1
Acceptance mask
&
logic 1=accept
logic 0=do not accept

Dual filter configuration

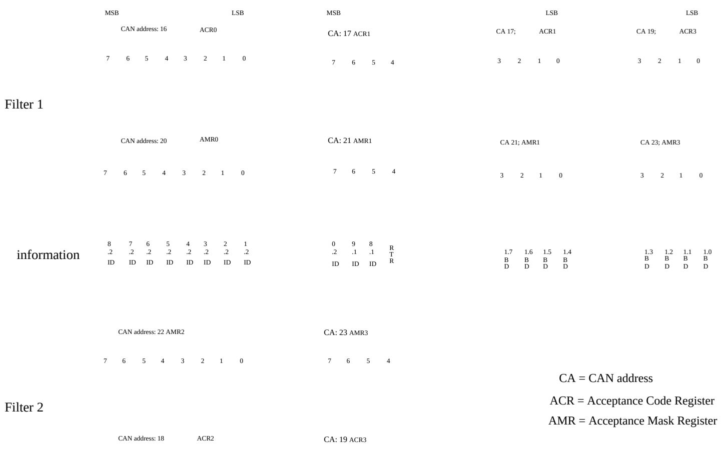
This configuration can define two short filters. A piece of received information is compared with two filters to determine whether to put it in the receiving buffer. At least one filter sends out the received signal for the received information to be valid. The bit correspondence between the filter byte and the information byte depends on the current The previously received frame format.

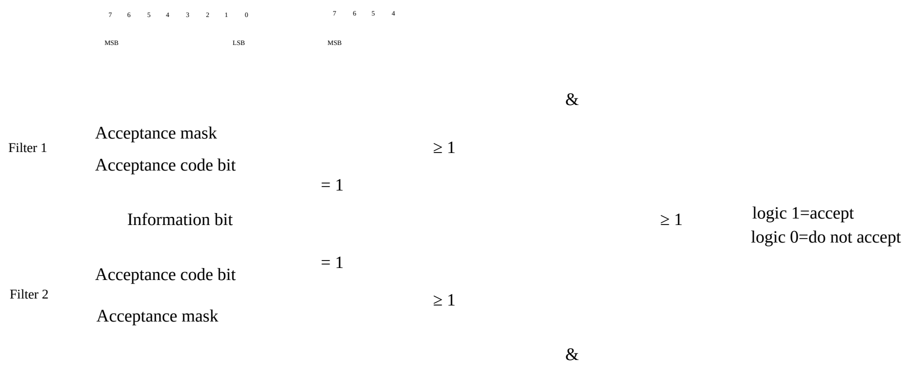
Standard frame: If the received standard frame information, the two defined filters are different. The first filter comparison includes the RTR bit The first data byte of the entire standard identification code and information. The second filter only compares the entire standard identification code including RTR bits.

In order to receive information successfully, there should be at least one filter to indicate reception when comparing all individual bits. RTR bit position or data length code When it is 0, it means that there is no data byte. In any case, as long as the part from the beginning to the RTR bit is indicated as received, the information can pass the filter Wave device 1.

If no data byte filtering is requested from the filter, the lower four bits of AMR1 and AMR3 must be set to '1' (not affected). When using the package When including the entire standard identification code of the RTR bit, both filters work the same.

Figure **179.** Dual filter configuration when receiving standard structure information

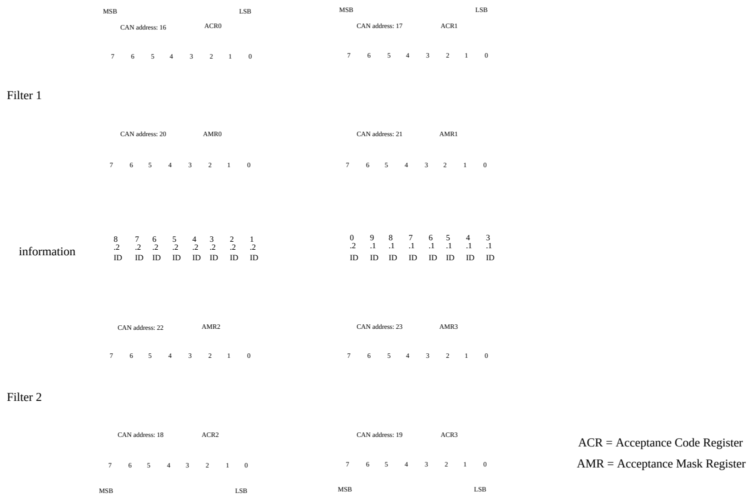


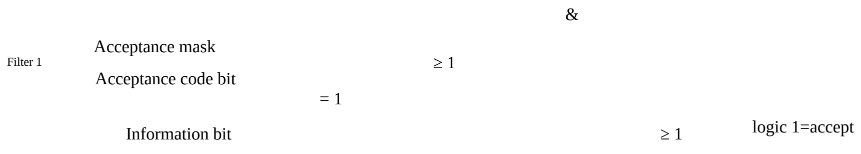


Extended frame: If the extended frame information is received, the two defined filters are the same. Both filters only compare the front of the extended identification code Two bytes.

In order to receive information successfully, at least one filter indicates the reception when comparing all individual bits.

Figure **180.** Dual filter configuration to receive extended frame information





Filter 2 Acceptance code bit = 1 logic 0=do not accept

Acceptance mask ≥ 1

&

Example 1: Assuming that 61 standard frame messages are to be filtered in PeliCAN mode, it can be done by using a long filter (single filter mode Mode).

Acceptance code register (ACRn) and acceptance mask register (AMRn) include:

n	0	1 (high four)	2	3
ACRn	01XX X010	XXXX	XXXX XXXX	XXXX XXXX
AMRn	0011 1000	1111	1111 1111	1111 1111
Received message (ID28 ~ ID.18, RTR)		01xx x010 xxxx		

('X' = irrelevant,'x' = any value, only the upper four bits of ACR1 and AMR1 are used.)

Example 2: Assume that the following two messages with standard frame identifiers are received without further decoding of the identifiers. Data and remote frames must be Accept it. Data bytes do not require acceptance filtering.

Message 1: (ID.28) 1011 1100 101 (ID.18)

Message 2: (ID.28) 1111 0100 101 (ID.18)

Using the single filter mode, four messages can be received instead of only the required two:

n	0	1 (high four)	2	3
ACRn	1X11 X100	101X	XXXX XXXX	XXXX XXXX
AMRn	0100 1000	0001	1111 1111	1111 1111
		1011 0100 101x		
Received message		1111 0100 101x	(Message 2)	
(ID28 ~ ID.18, RTR)		1011 1100 101x	(Message 1)	
		1111 1100 101x		

('X' = irrelevant,'x' = any value, only the upper four bits of ACR1 and AMR1 are used.)

This result does not meet the requirement of receiving two pieces of information without further decoding.

Use dual filters to get correct results

	Filter 1			Filter 2	
n	0	1	3 Low four	2	3 High four
ACRn	1011 1100	101X XXXX	… XXXX	1111 0100	101X…
AMRn	0000 0000	0001 1111	… 1111	0000 0000	0001…
Received information (ID.28 ~ ID.18, RTR)		1011 1100 101X (Message 1)		0100 101X (Message 2)	

('X' = irrelevant,'x' = any value)

Message 1 is received by filter 1, and message 2 is received by filter 2. If the message is received by at least one of the two filters, the message is It is stored in the receive FIFO. This method can meet this requirement.

Example 3:

In this example, a long acceptance filter is used to boss a group of messages with extended frame identifiers

n	0	1	2	3 (high six)
ACRn	1011 0100	1011 000X	1100 XXXX	0011 0XXX
AMRn	0000 0000	0001 0001	0000 1111	0000 0111
Received information				

(ID.28 ~ ID.18, RTR)	1011 0100 1011 000x 1100 xxxx 0011 0x

('X' = irrelevant,'x' = any value, only the upper six bits of ACR1 and AMR1 are used.)

Example 4:

Some use the standard framing system to identify the message with only an 11-bit identifier and the first two data bytes. If the message exceeds 8 data bytes, the first two
A data byte is defined as a message header and using a segmented storage protocol will use a protocol like this. For example, DeviceNet. For this type of system,
In addition to the 11-bit identifier and RTR bit, the CAN controller can filter two data bytes in single filter mode, and can filter two data bytes in dual filter mode.
Filter a data byte (except for the 11-bit identifier and RTR bit).

The following example shows the use of dual filter mode to effectively filter messages in this system:

	Filter 1			Filter 2	
n	0	1	3 Low four	2	3 High four
ACRn	1110 1011	0010 1111	… 1001	1111 0100	XXX0…
AMRn	0000 0000	0000 0000	… 0000	0000 0000	1110…
Received information	1110 1011 0010		1111… 1001	1111 0100	xxx0
(ID.28 ~ ID.18, RTR)	Identifier + RTR		First data byte	Identifier	RTR

('X' = irrelevant,'x' = any value)

- The messages filtered by filter 1 are:
  - Identifier ' 11101011001 '
  - RTR = ' 0', which means it is a data frame, and
  - Data byte ' 11111001' (this refers to DeviceNet, for example: all segments of a message are filtered).
- Filter 2 is used to filter a group of 8 messages, among which the messages are:
  - Identifiers ' 11110100 000' to 11110100111', and
  - RTR = ' 0', which is the data frame

24.5.6 Message storage

The data to be sent on the CAN bus is loaded into the storage area of the CAN controller. This storage area is called the'transmit buffer'. From CAN bus
The received data is also stored in the storage area of the CAN controller. This storage area is called the “receiving buffer”. These buffers consist of 2, 3 or 5 words
Section identifier and frame information (depending on the mode and frame type), and can contain up to 8 data bytes.

BasicCAN mode

Buffer up to 10 bytes

- 2 identifier bytes
- Up to 8 data bytes

RX and TX buffers in BasicCAN mode

Relative CAN offset		Register name		Composition and notes
TX (Hexadecimal)	RX (Hexadecimal)	TX	RX	
28	50	CAN_TXIDR1	CAN_RXIDR1	8-bit identifier
2C	54	CAN_TXIDR2	CAN_RXIDR2	3 identifiers, 1 remote transmission request bit, 4 digits According to the length code, indicating the number of data bytes
30	58	CAN_TXDR1	CAN_RXDR1	It is indicated by the data length code, up to 8 data bytes
34	5C	CAN_TXDR2	CAN_RXDR2	
38	60	CAN_TXDR3	CAN_RXDR3	
3C	64	CAN_TXDR4	CAN_RXDR4	
40	68	CAN_TXDR5	CAN_RXDR5	
44	6C	CAN_TXDR6	CAN_RXDR6	
48	70	CAN_TXDR7	CAN_RXDR7	
4C	74	CAN_TXDR8	CAN_RXDR8	

PeliCAN mode

These buffers are 13 bytes long

- 1 byte frame information
- 2 or 4 identifier bytes (standard frame or extended frame)
- Up to 8 data bytes

Send buffer

The complete list of send buffers is shown in the figure below. Be sure to distinguish between the standard frame format (SFF) and extended frame format (EFF) configuration. Send buf The device allows to define up to 8 data bytes to send information.

Figure **181.** List of standard frame and extended frame format configuration in the transmit buffer

CAN address	40	TX frame information	CAN address	40	TX frame information
	44	TX identification code 1		44	TX identification code 1
	48	TX identification code 2		48	TX identification code 2
	4C	TX data byte 1		4C	TX identification code 3
	50	TX data byte 2		50	TX identification code 4
	54	TX data byte 3		54	TX data byte 1
	58	TX data byte 4		58	TX data byte 2
	5C	TX data byte 5		5C	TX data byte 3

60	TX data byte 6	60	TX data byte 4
64	TX data byte 7	64	TX data byte 5
68	TX data byte 8	68	TX data byte 6
6C	Unused	6C	TX data byte 7
70	Unused	70	TX data byte 8
	a. Standard frame format		b. Extended frame format

The FF bit in the frame information determines whether the CAN controller will send the extended frame format or the standard frame format.

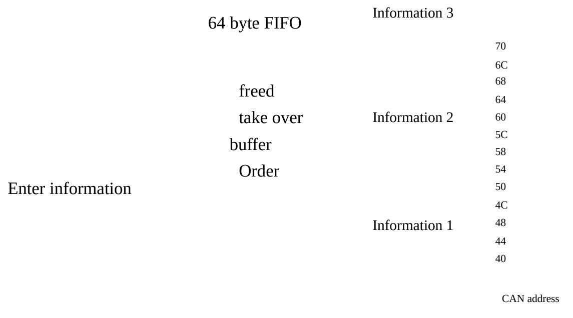
Receive buffer

The list of receive buffers is very similar to that of transmit buffers. The receive buffer is the accessible part of RXFIFO, located at 40 ~ of CAN address 70. Each piece of information is not a description area and a data area.

Note: The received byte length code in the frame information byte represents the actual data length code sent, and it may be greater than *8* (depending on the sending 器). In any case, the maximum number of received data bytes is *8* . This point should be considered when reading the information in the receive buffer.

As shown in the figure below, RXFIFO has a total of 64 information bytes. How many pieces of information can be stored at one time depends on the length of the data. if There is not enough space in the RXFIFO to store new information, the CAN controller will generate a data overflow condition, at this time the information is valid and receive detection For sure. When a data overflow occurs, the information that has been partially written into the RXFIFO will be deleted. In this case, the status register and the number According to the overrun interrupt (interrupt allowed), it is reflected to the CPU.

Figure **182.** Example of information storage in **RXFIFO**



The currently available information in the receiving buffer is information 1

24.5.7 Error Management

Based on the value of the error counter, each CAN controller can work in one of three error states: error activation, error recognition, or bus Offline. If the value of the error counter is between 0 and 127, the CAN controller is activated by error. At this time, a false activation flag (6 Dominant bit). If the value of an error counter is between 128 and 255, the CAN controller recognizes it incorrectly. At this time, an error is detected Before, a recognized error flag (6 recessive bits) is generated. If the value of the sending error counter is higher than 255, the bus offline state is reached. In this kind of In the state, the reset request is automatically set, and the CAN controller has no effect on the bus. The offline state of the bus can only be reset by the command' Bit request = 0'to exit. This will start the bus offline recovery timer and send the error counter to count 128 bus release signals. After the count is over, Both error counters are 0, and the device is again in an error active state.

Error counter

As described above, the error status of CAN is directly related to the values of the transmit error counter and the receive error counter.

In order to carefully study the error definition and support the enhanced error analysis function of the CAN controller, the CAN controller provides a readable error count Device. In addition, in reset mode, write access to the two error counters is allowed.

Error interrupt

There are three interrupt sources to send the wrong state to the microprocessor. Each interrupt can be individually enabled in the interrupt enable register.

- Bus error interrupt:

Any error detected on the CAN bus will generate an interrupt.

- Error warning interrupt:

If the error warning limit is exceeded, an error warning interrupt is generated. And it repeats before the CAN controller enters the bus offline state and again This interrupt will also be generated when entering the false activation state once. The error warning limit of the CAN controller is programmable in the reset mode. Default after reset The value is 96.

- Wrong recognition interruption:

If the error status changes from false activation to false recognition or vice versa, a false recognition interrupt will be generated.

Error code capture

The CAN controller can perform all the error definitions defined in the CAN2.0B specification. The entire process of error handling by each CAN controller is Completely automatic. However, in order to provide users with detailed information about an error, the CAN controller provides an error code capture function. No matter what Whenever a CAN bus error occurs, it will force a corresponding bus error interrupt. At the same time, the current bit position is captured into the error code Capture register. Before the main controller reads out the captured data, it will be stored in the register. Then the capture mechanism is activated again. Deposit The analyzer can distinguish four types of errors: format errors, padding errors, bit errors, and other errors. As shown in the figure below, the register also has another table It indicates whether the error occurred during the reception or transmission of the message. The five bits in this register indicate the bit position of the error in the CAN frame. More informatic For information, refer to the table and data sheet below.

Figure **183.** Example of error code capture function

The CAN specification defines: each bit on the CAN bus has only a special type of error. The following two shows the CAN message sending and receiving All errors that may occur during the period. The left part includes the location and the type of error, which are captured by the error code capture register. Per table The right part is to convert the error code into the upper-level error description, and you can know its meaning directly from the contents of the register. By using these forms, you can Get more information about the changes in the error counter and the error status of the device's transmit and receive pins. When using these tables, for example, in error In the analysis software, each error state can be analyzed in detail. Information about the type and location of CAN errors can be used for error statistics and system maintenance Or make corrections in the system optimization device.

Table **42.** Errors that may occur when receiving

Errors in the CAN bit stream Misplaced	Type of error	RX error count	Error code capture	describe
Identifier SRR, IDE and RTR Bit Reserved bit Data length code Data field CRC sequence	filling	+ 1	Receive 5 consecutive bits of the same level	-
CRC delimiter	Format filling	+ 1 + 1	RX = dominant Receive more than 5 consecutive bits with the same level	Bit must be recessive
Response bit	Bit	+ 1	TX = dominant, but RX = recessive	Cannot write dominant bit
Response delimiter	Format	+ 1	RX = dominant, or CRC error detected	Critical bus timing or bus length CRC sequence is incorrect
End of frame	Format other	+ 1 ± 0	RX = the first six bits are dominant RX = Dominance of the last digit	- Reaction: send out the overload sign, if send out The transmitter resends, the data may be duplicated
interval	other	± 0	RX = dominant	Reaction: The receiver sends out an overload sign
Activate error flag	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
Allowable dominant position	other	+ 8	RX = The first bit after the error flag is dominant RX = There are more than 7 dominant bits after the error or overload flag	
Error delimiter	Format other	+ 1 ± 0	RX = the first seven bits are dominant bits RX = The last bit of the delimiter is a dominant bit	- Send overload flag
Overload sign	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit

Table **43.** Possible errors when sending

Errors in the CAN bit stream Location	Type of error	TX error count	Error code capture	describe
Start of frame	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
Identifier	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
	filling	± 0	TX = recessive, but RX = dominant	-
SRR bit	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
	filling	± 0	TX = recessive, but RX = dominant	-
IDE and RTR bits	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
	filling	+ 8	TX = recessive, but RX = dominant	-
Reserved bit Data length code Data field CRC sequence	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
CRC delimiter	Format	+ 8	RX = dominant	Bit must be recessive
Response gap	other	+ 8	RX = recessive (error activation)	No answer
	other	± 0	RX = recessive (false recognition)	No response, the node may be alone On the bus
Response delimiter	Format	+8	RX = dominant	Critical bus timing or bus length
End of frame	Format	+ 8	RX = the first six bits are dominant bits	-
	other	+ 8	RX = the last bit is a dominant bit	The frame has been received by some nodes, Sending again may result in receiving Duplicate data in the device
interval	other	± 0	RX = dominant	From the'old' CAN controller Overload sign
Activate error flag Overload sign	Bit	+ 8	TX = dominant, but RX = recessive	Cannot write dominant bit
Dominant bit allowed	Format	+ 8	RX = Error flag is activated or over There are more than 7 dominant bits after loading the flag	----
Error delimiter	Format	+ 8	RX = the first seven bits are dominant bits	----
	other	± 0	RX = the last digit of the delimiter is Dominant bit	
Recognition error flag	other	+ 8	RX = dominant (false recognition)	No response is received, the node is not Alone on the bus.

Offline recovery

If the value of the transmission error counter is higher than 255, the bus offline state is reached. The bus status bit is set to '1'. In this state, since If the reset request bit is set automatically, the CAN controller has no effect on the bus. If the error interrupt is allowed, an error interrupt will be generated. This kind of The status will continue until the CPU clears the reset request bit. After all these are completed, the CAN controller will wait for the minimum time specified in the protocol (128 A bus idle signal).

24.5.8 bit time characteristics

The bit time characteristic logic monitors the serial CAN bus through sampling, and synchronizes with the edge of the frame start bit, and through the following The edge of the face is resynchronized to adjust its sampling point.

Its operation can be simply explained as dividing the nominal time per person into 3 segments as follows:

- Synchronization segment ($t_{SYNCSEG}$): It is usually expected that the bit change occurs within this time period. Its value is fixed at 1 time unit (1 x tCAN).
- Time period 1 (t_{TSEG1}): Define the location of the sampling point. It includes PROP_SEG and PHASE_SEG1 in the CAN standard. Its value can be programmed from 1 to 16 time units, but it can also be automatically extended to compensate for the frequency difference between different nodes in the network The positive shift of the phase caused by the difference.
- Time period 2 (t_{TSEG2}): Define the location of the sending point. It represents PHASE_SEG2 in the CAN standard. Its value can be programmed as 1 to 8 time units, but can also be automatically shortened to compensate for negative phase drift.

The period of the CAN system clock t_{SCL} is programmable and determines the corresponding bit timing.

The CAN system clock is calculated by the following formula: $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$. Here t_{CLK} = the frequency period of APB1.

The synchronization jump width (SJW) defines the upper limit of how many time units can be extended or shortened in each bit. In order to compensate for the different total
The phase shift between the clock oscillators of the line controllers, any bus controller must be resynchronized on the edge of the relevant signal currently transmitted. Synchronize
The jump width defines the maximum number of clock cycles that can be shortened or extended by resynchronization for each bit cycle.

$t_{SJW} = t_{SCL} \times (SJW + 1)$

Time period 1 (TSEG1) and time period 2 (TSEG2) determine the number of clocks for each bit and the location of the sampling point, here:

$t_{SYNCSEG} = 1 \times t_{SCL}$

$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$

$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$

A valid transition is defined as the first transition from a dominant bit to a recessive bit when CAN itself does not send a recessive bit. If in time period
1 (t_{TSEG1}) instead of detecting a valid transition in the synchronization segment ($t_{SYNCSEG}$), then the time of t_{TSEG1} is extended by up to SJW,
As a result, the sampling point is delayed.

On the contrary, if a valid transition is detected in time period 2 (t_{TSEG2}) instead of $t_{SYNCSEG}$, then the time of t_{TSEG2} is shortened by up to SJW
So long, so the sampling point is advanced.

In order to avoid software programming errors, the bit time characteristic register (CAN_BTR) can only be set when CAN is in the initialization state.
Proceed under.

CAN baud rate = APB1/(2*(BRP+1)*(TSEG1+ 1+ TSEG2+ 1+ 1));

24.5.9 Loss of Arbitration

When arbitration is lost, the corresponding arbitration lost interrupt (interrupt enable) will be generated. At the same time, the current bit position of the bit stream processor is captured a
Arbitration lost capture register. Until the user reads this value through software, the contents of the register will not change. Subsequently, the capture mechanism was stimulated
Alive.

When reading the interrupt register, the corresponding interrupt flag bit in the interrupt register is cleared. Until the arbitration lost capture register is read once,
The new arbitration lost interrupt is valid.

The following figure shows the explanation of the arbitration lost bit

Figure 184. Example of arbitration loss interpretation

24.5.10 CAN Interrupt

There are 5 interrupts in **BasicCAN** mode:

- Receive interrupt

Generated when the receive FIFO is not empty and the receive interrupt is enabled (CAN_CR register bit 1). The RI bit of the CAN_IR register is set to '1'

- Send interrupt

Generated when the state of the transmit buffer changes from 0 to 1 (released) and the transmit interrupt is enabled (CAN_CR register bit 2)

- Error interrupt

When the error interrupt is enabled (CAN_CR register bit 3), the error status bit or the change of the bus status bit will set this bit

- Data overflow interrupt

Probably, when the data overflow interrupt enable bit (CAN_CR register bit 4) is set to '1', it will jump to the data overflow status bit '0-1'

- Wake up interrupt

This interrupt is generated when exiting sleep mode.

There are 8 different interrupts in **PeliCAN** mode:

- Receive interrupt

An interrupt is generated when the receive FIFO is not empty and the RIE bit of the interrupt register is set

- Send interrupt

An interrupt is generated when the transmit buffer status changes from '0-1' (released) and the TIE bit of the interrupt register is set

- Error alarm interrupt

An interrupt is generated when the error status bit and the bus status bit change and the EIE bit of the interrupt register is set

- Data overflow interrupt

An interrupt is generated when the data overflow status bit has a '0-1' transition and the DOIE bit of the interrupt register is set

- Wake up interrupt

An interrupt is generated when the CAN controller detects bus activity in sleep mode and the WUIE bit of the interrupt register is set to '1'

- False negative interrupt

When the CAN controller reaches the error-negative state (at least one error counter exceeds the value 127 specified in the protocol) or from the error-negative state An interrupt is generated when the error active state is entered and the EPIE bit of the interrupt register is set

- Arbitration lost interrupt

When the CAN controller loses arbitration and becomes the receiver and ALIE of the interrupt enable register is set, an interrupt is generated

- Bus error interrupt

An interrupt is generated when the CAN controller detects a bus error and the BEIE in the interrupt enable register is set

24.6 CAN register description

24.6.1 CAN Mode Register (CAN_MOD)

PeliCAN mode only

Offset address: 0x000

Reset value: 0x0001

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											SM	AFM	STM	LOM	RM
											rw	rw	rw	rw	rw

Bit 31: 5	Reserved, the value of read bit 3 is always '1'.
Bit 4	**SM** : Sleep mode 1: Sleep; when there is no CAN interrupt waiting and bus activity, the CAN controller enters sleep mode 0: wake up; wake up from sleep
Bit 3	**AFM** : Acceptance filter mode 1: Single; select a single acceptance filter (32-bit length) 0: Double; select two acceptance filters (each with 16-bit activation)
	STM : Self test mode (Self test mode)

Bit 2

1: Self-detection; this mode can detect all nodes without any active nodes using self-receiving commands; even if there is no Reply, the CAN controller will also send successfully.

0: Normal mode; a response signal is required for successful transmission

Bit 1

LOM : Listen only mode

1: Listen only; in this mode, even if the message is successfully received, the CAN controller does not send a response signal to the bus; error count The counter stops at the current value.

0: normal mode

Bit 0

RM : Reset mode

1: Reset; it is detected that the reset mode bit is set, the information currently being received/sent is aborted, and the reset mode is entered.

0: Normal; after the reset mode bit receives a transition of '1-0', the CAN controller returns to working mode.

24.6.2 **CAN** Control Register (**CAN_CR**)

BasicCAN mode only:

Offset address: 0x000

Reset value: 0x0001

The content of the control register is used to change the behavior of the CAN controller. These bits can be set or set by the microcontroller, which can To read/write the control register.

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											OIE	EIE	TIE	RIE	RR
											rw	rw	rw	rw	rw

Bit 31: 5

Reserved, the value of read bit 3 is always '1'.

Bit 4

OIE : Overflow interrupt enable

1: Enable; if the data overflow bit is set, the microcontroller receives an overflow interrupt signal

0: Disabled; the microcontroller does not receive an overflow interrupt signal from the CAN controller

Bit 3

EIE : Error interrupt enable

1: Enable; if an error occurs or the bus status changes, the microcontroller receives an error interrupt signal

0: Disabled; the microcontroller does not receive error interrupt signals from the CAN controller

Bit 2

TIE : Transmit interrupt enable

1: Enable; when the information is successfully sent or the sending buffer is accessed again (for example, after the sending command is aborted), the micro-control The controller receives a transmission interrupt signal from the CAN controller

0: Disabled; the microcontroller does not receive and send interrupt signals from the CAN controller

Bit 1

RIE : Receive interrupt enable (Receive interrupt enable)

1: Enable; when the information is received without error, a receiving interrupt signal sent by the CAN controller to the microcontroller

0: Disabled; the microcontroller does not receive interrupt signals from the CAN controller

Bit 0

RR : Reset request

1: Current; after the CAN controller detects the reset request, it suspends the current sending/receiving information and enters the reset mode

0: Vacancy; after the reset request bit receives a falling edge. CAN controller returns to working mode

24.6.3 CAN Command Register (CAN_CMR)

Offset address: 0x004

Reset value: 0x0000

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											GTS/ SRR	CDO	RRB	AT	TR
											rw	rw	rw	rw	rw

Bit 31: 5	Reserve.
Bit 4	BasicCAN mode: **GTS** : Go to sleep 1: Sleep; if there is no CAN interrupt waiting and bus activity, the CAN controller enters sleep mode. 0: Wake up; CAN controller works in normal mode. PeliCAN mode: **SRR** : Self reset request 1: Current; information can be sent and received at the same time 0: (vacant)
Bit 3	**CDO** : Clear data overrun 1: Clear; clear the data overflow status bit 0: No action Clear data overflow This command bit is used to clear the data overflow situation indicated by the data overflow status bit. Fruit data When the overflow bit is set, no data overflow interrupt will be generated. When releasing the receiving buffer command, you can issue a clear at the same time Data overflow command.
Bit 2	**RRB** : Release receive buffer (Release receive buffer) 1: Release; the memory space for storing information in the receiving buffer will be released 0: No action After reading the receive buffer, the microcontroller can release the current in the RXFIFO by setting the release receive buffer bit to 1. Information memory space. This may cause another message in the receiving buffer to be immediately valid. This will produce another The second receive interrupt (when enabled). If there is no other information available, the receiving interrupt will not be generated again, and the receiving will be slowed down. The punch status bit is cleared.
Bit 1	**AT** : Abort transmission 1: Currently; if it is not in the process of processing, the sending request waiting to be processed will be cancelled 0: Vacancy; no action The abort transfer bit is used when the CPU requires the current transfer to be suspended, for example, to transfer an emergency message. Ongoing The transmission does not stop. To check whether the original message was sent successfully, it can be checked by the transmission success status bit. However, this can only be achieved when the transmit buffer status bit is 1 (released) or the transmit interrupt is generated.

Bit 0	**TR** : Transmission request 1: current; information is sent 0: Vacancy; no action If the send request is set in the previous command. It cannot be cancelled by directly setting it to 0. but, It can be cancelled by setting the abort sending bit to 0.

24.6.4 CAN Status Register (CAN_SR)

Offset address: 0x008

Reset value: 0x0000

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					BS	ES	TS	RS	TCS	TBS	DOS	RBS
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7	**BS** : Bus status 1: Bus is closed; CAN controller exits bus activity 0: The bus is on; the CAN controller joins the bus activity When the transmission error counter exceeds the limit (255) (bus status bit '1'-bus is off), the CAN controller will Reset request bit '1' (current), if the error interrupt is allowed, an error interrupt will be generated. This state It will continue until the CPU clears the reset request bit. After all these are completed, the CAN controller will wait for the agreement Minimum (128 bus idle signals). After the bus status bit is cleared (the bus is turned on), the error status bit is set to '0' (ok), the error counter is reset and an error interrupt is generated (interrupt allowed)
Bit 6	**ES** : Error status 1: Error; at least one error counter is full or exceeded CPU alarm limit 0: ok; both error counters are below the alarm limit According to the CAN 2.0B protocol description, errors detected during reception or transmission will affect the error count. When there is at least one error When the error counter is full or exceeds the CPU warning limit (96), the error status bit is set. Where permitted, will produce Error interrupted.
Bit 5	**TS** : Transmit status 1: Send; CAN controller is transmitting information 0: idle; there is no information to send If the receiving status bit and the sending status bit are both 0, the CAN bus is idle.
Bit 4	**RS** : Receive status 1: Receive; CAN controller is receiving information 0: Idle; no information is being received If the receiving status bit and the sending status bit are both 0, the CAN bus is idle.

Bit 3	**TCS** : Transmission complete status 1: Complete; the last sending request was successfully processed 0: not completed; the current sending request has not been processed Whenever the send request bit is set to '1', the send complete bit will be set to '0' (not completed). '0' of the transmitted complete bit Will keep until the message is successfully sent.
Bit 2	**TBS** : Transmit buffer status 1: Release; CPU can write information to the sending buffer 0: locked; the CPU cannot access the sending buffer; there is information waiting to be sent or being sent If the CPU attempts to write to the transmit buffer when the transmit buffer status bit is 0 (locked), the written byte will be rejected Received and will be lost without any prompt.
Bit 1	**DOS** : Data overrun status 1: Overflow; information is lost because there is not enough space in the RXFIFO to store it 0: Vacancy; no data overflow has occurred since the last clear data overflow command was executed When the information to be received successfully passes the acceptance filter (for example, at the beginning of the arbitration), the CAN controller needs to Some space is used in the RXFIFO to store the descriptor of this information. Therefore, there must be enough space to store every received One data byte. If there is not enough space to store the information, the information will be lost and the CPU will only be notified of data overflow Condition. If the received message has no errors except the last digit, the message is valid.
Bit 0	**RBS** : Receive buffer status 1: Full; there is information available in RXFIFO 0: empty; no information available After reading the information in the RXFIFO and releasing the memory space with the release receive buffer command, this bit is cleared. If there is still information available in the FIFO, this bit will be reset in the time limit (tSCL) of the next bit.

24.6.5 CAN Interrupt Register (**CAN_IR**)

Offset address: 0x00C

Reset value: 0x0000

The interrupt register allows the identification of the interrupt source. The interrupt is activated when one or more bits of the register are set. Interrupt Register to Microcontroller The controller is a read-only memory.

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	ALI	EPI	WUI	DOI	EI	TI	RI
								r	r	r	r	r	r	r	r

Bit 31: 8	Reserved, the value of bit 7, bit 6, and bit 5 is always 1.

Bit 7	Basic mode: Reserved, the read value is 1. PeliCAN mode: **BEI** : Bus error interrupt 1: Set; this bit is set when the CAN controller detects a bus error and the BEIE in the interrupt enable register is set Bit 0: reset
Bit 6	Basic mode: Reserved, the read value is 1 PeliCAN mode: **ALI** : Arbitration lost interrupt 1: Set; when the CAN controller loses arbitration and the ALIE that becomes the receiver and interrupt enable register is set, this The bit is set. 0: reset
Bit 5	Basic mode: Reserved, the read value is 1. PeliCAN mode: **EPI** : Error passive interrupt 1: Set; when the CAN controller reaches the error negative state (at least one error counter exceeds the value 127 specified in the protocol) Or from the negative state of the error to the error active state and the EPIE bit of the interrupt register is set, this bit is set to '1' 0: reset
Bit 4	**WUI** : Wake-up interrupt 1: Set; this bit is set when exiting sleep mode 0: Reset; any read access of the microcontroller will clear this bit If when the CAN controller participates in bus activity or CAN interrupt is waiting, the CPU tries to enter sleep mode and wake up Interrupts will also be generated.
Bit 3	**DOI** : Data overrun interrupt 1: Set; when the data overflow interrupt enable bit is set to '1', it jumps to the data overflow status bit '0-1', and this bit is set. 0: Reset; any read access of the microcontroller will clear this bit The overflow interrupt bit (when the interrupt is enabled) and the overflow status bit are set at the same time.
Bit 2	**EI** : Error interrupt 1: Set; when the error interrupt is enabled, the error status bit or the change of the bus status bit will set this bit. 0: Reset; any read access of the microcontroller will clear this bit
Bit 1	**TI** : Transmit interrupt 1: Set; this bit is set when the transmit buffer status changes from 0 to 1 (released) and the transmit interrupt is enabled. 0: Reset; any read access of the microcontroller will clear this bit

Bit 0

RI : Receive interrupt

1: Set; set this bit when the receive FIFO is not empty and the receive interrupt is enabled

0: Reset; any read access of the microcontroller will clear this bit

The receive interrupt bit (when the interrupt is enabled) and the receive buffer status bit are set at the same time.

It must be noted that the receive interrupt bit is cleared when reading, even if there is other information available in the FIFO. Once released

After the receive buffer command is executed, there are other available information in the receive buffer, and the receive interrupt (when the interrupt is enabled) will be

At the next t SCL is reset.

24.6.6 CAN Interrupt Enable Register (CAN_IER)

Offset address: 0x010

Reset value: 0x0000

Only **PeliCAN** mode exists

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								BEIE	ALIE	EPIE	WUIE	DOIE	EIE	TIE	RIE
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 5

Reserve.

Bit 7

BEIE : Bus error interrupt enable

1: Enable; if a bus error is detected, the CAN controller requests the corresponding interrupt

0: prohibited

Bit 6

ALIE : Arbitration lost interrupt enable (Arbitration lost interrupt enable)

1: Enable; if the CAN controller has lost arbitration, request the corresponding interrupt.

0: prohibited

Bit 5

EPIE : Error passive interrupt enable

1: Enable; if the error state of the CAN controller changes (from passive to active or vice versa), the corresponding interrupt is requested

0: prohibited

Bit 4

WUIE : Wake-up interrupt enable

1: Enable; if the CAN controller in sleep mode is awakened, the corresponding interrupt is requested.

0: prohibited

Bit 3

DOIE : Data overrun interrupt enable (Data overrun interrupt enable)

1: Enable; if the data overflow status bit is set (see the status register), the CAN controller requests the corresponding interrupt.

0: prohibited

Bit 2

EIE : Error interrupt enable

1: Enable; if there is an error or the bus status changes (see status register), the CAN controller requests the corresponding interrupt.

0: prohibited

Bit 1

TIE : Transmit interrupt enable

1: Enable; when the information is successfully sent or the sending buffer is accessible again (for example, after the sending command is aborted), the CAN The controller requests the corresponding interrupt.

0: prohibited

Bit 0

RIE : Receive interrupt enable (Receive interrupt enable)

1: Enable; when the receiving buffer status is'full', the CAN controller requests the corresponding interrupt.

0: prohibited

Page 418

24.6.7 CAN Acceptance Code Register

BasicCAN mode: CAN_ACR

Offset address: 0x010

Reset value: 0x0000

With the help of the acceptance filter, the CAN controller can allow the RXFIFO to only receive the same identification code and the preset value in the acceptance filter. Information. The acceptance filter is defined by the acceptance code register and the acceptance mask register.

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								AC							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	**AC[7 : 0]** : (Acceptance code) When the reset request bit is set high (current), this register is accessible (Read/write). If a message passes the acceptance filter test and there is room in the receiving buffer, then describe Symbols and data will be sequentially written into RXFIFO respectively. When the information is received correctly, it will: Receiving status position is high (full) Receive interrupt enable bit is high (enable) Receive interrupt is set high (generate interrupt) The acceptance code bits (AC.7-AC.0) and the upper 8 bits of the information identification code (ID.10-ID.3) are equal, and are equal to the acceptance mask bits The corresponding phase of (AM.7-AM.0) is 1. That is, if it meets the description of the following equation, it will be accepted: [(ID.10-ID.3)≡(AC.7-AC.0)] ∨(AM.7-AM.0)≡ 11111111

PeliCAN mode: There are four acceptance code registers: CAN_ACR0, CAN_ACR1, CAN_ACR2, CAN_ACR3

CAN_ACR0: Offset address: 0x040

Reset value: 0x0000

CAN_ACR1: Offset address: 0x044

Reset value: 0x0000

CAN_ACR2: Offset address: 0x048

Reset value: 0x0000

CAN_ACR3: Offset address: 0x04C

Reset value: 0x0000

Note: For details, see the introduction of *peliCAN* mode in the identifier filter

Page 419

24.6.8 CAN Acceptance Mask Register

BasicCAN mode:

Offset address: 0x014

Reset value: 0x0000

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								AM							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	**AM[7 : 0]** : (Acceptance mask) If the reset request bit is high (currently) this register can be accessed (Read/write). The acceptance mask register defines whether the corresponding bit of the acceptance code register is'related' or'none' to the acceptance filter Affected' (can be any value).

PeliCAN mode: There are four acceptance mask registers: CAN_AMR0, CAN_AMR1, CAN_AMR2, CAN_AMR3

CAN_AMR0: Offset address: 0x050

Reset value: 0x0000

CAN_AMR1: Offset address: 0x054

Reset value: 0x0000

CAN_AMR2: Offset address: 0x058

Reset value: 0x0000

CAN_AMR3: Offset address: 0x05C

Reset value: 0x0000

Note: For details, see the introduction of *peliCAN* mode in the identifier filter

24.6.9 CAN bus timing 0 (CAN_BTR0)

Offset address: 0x018

Reset value: 0x0000

The bus timing register 0 defines the baud rate preset value (BRP) and the synchronization jump width (SJW) value. When the reset mode is valid, this The register can be accessed (read/write).

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								SJW[1:0]		BRP[5:0]					
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7: 6	**SJW[1 : 0]** : Synchronization jump width In order to compensate for the phase shift between the clock oscillators of different bus controllers, any bus controller must The transmitted edges of the relevant signal are resynchronized. The synchronization jump width defines that each bit period can be shortened by resynchronization or The maximum number of extended clock cycles. $t_{SJW} = t_{SCL} \times (SJW + 1)$
	BRP[5 : 0] : Baud rate prescaler

Bit 5:0	The period of the CAN system clock tSCL is programmable and determines the corresponding bit timing. The CAN system clock is represented by the following Formula calculation $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ Here t_{CLK} = the clock period of APB1.

24.6.10 CAN bus timing **1** (**CAN_BTR1**)

Offset address: 0x01C

Reset value: 0x0000

The bus timing register 1 defines the length of each bit period, the location of the sampling point, and the number of samples at each sampling point. In reset mode In, this register can be read/write accessed.

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								SAM	TESG2[6:4]			TESG1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.

Bit 7	**SAM** : Sampling 1: Triple; bus sampling three times; it is recommended to use on low/medium speed buses (A and B grades), this pair filters the gross on the bus Spikes are beneficial 0: Single; bus sampling once; recommended to use on high-speed bus (SAE C level)
Bit 6:0	**TSEG2[6** : **4]** , **TSEG1[3** : **0]** : Time segment 1 (Time segment 1) and time segment 2 (Time segment 2) T_{SEG1} and T_{SEG2} determine the number of clocks for each bit and the location of the sampling point, here: $t_{SYNCSEG} = 1 \times t_{SCL}$ $t_{TSEG1} = t_{SCL} \times (8 \times TSEG1.3 + 4 \times TSEG1.2 + 2 \times TSEG1.1 + TSEG1.0 + 1)$ $t_{TSEG2} = t_{SCL} \times (4 \times TSEG\ 2.2 + 2 \times TSEG2.1 + TSEG2.1 + 1)$

24.6.11 CAN Transmit Identification Code Register **0** (**CAN_TXID0**)

Only BasicCAN mode exists:

Offset address: 0x028

Reset value: 0x0000

The sending identification code register 0 defines the type and data length of the sending frame. This register can be read/write accessed only in working mode.

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								ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	CAN identifier byte 10 ~ 3 (CAN identifier byte 10 ~ 3)

24.6.12 CAN Transmit Identification Code Register **1** (**CAN_TXID1**)

Only BasicCAN mode exists:

Offset address: 0x02C

Reset value: 0x0000

The sending identification code register 1 defines the type and data length of the sending frame. This register can be read/write accessed only in working mode.

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								ID.2	ID.1	ID.0	RTR	DLC3	DLC2	DLC1	DLC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7: 5	CAN identifier 2 ~ 0 (CAN identifier byte 2 ~ 0)
Bit 4	**RTR** : Frame format (Remote transmission request) 1: Remote; CAN will send remote frames 0: data; CAN will send data frame
Bit 3: 0	Send data area length 0 ~ 8 (Data length code 0 ~ 8)

Only **BasicCAN** mode exists :

Offset address: 0x030 ~ 0x04C

Reset value: 0x0000

Send data register CAN_TXDR0 ~ 7. This register can be read/write accessed only in working mode.

The data format of the receiving buffer is the same as that of the sending buffer.

24.6.13 **CAN** Arbitration Loss Capture Register (**CAN_ALC**)

Only PeliCAN mode exists:

Offset address: 0x02C

Reset value: 0x0000

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											BITNO4				
											rw	rw	rw	rw	rw

Bit 31: 5	Reserved, the read value is 0.
Bit 4: 0	**BITNO[4** : **0]** : Refer to the table below for values and functions (Bit number) When arbitration is lost, the corresponding arbitration lost interrupt (interrupt enable) will be generated. At the same time, the current bit of the bit stream processor The settings are captured and sent to the arbitration lost capture register. Until the user reads this value through software, the contents of the register are all Will not change. Subsequently, the capture mechanism was activated again. When reading the interrupt register, the corresponding interrupt flag bit in the interrupt register is cleared. Until the arbitration is lost, the capture register is After reading it once, the new arbitration lost interrupt is valid.

Page 423

Table **44.** The function of **bit 4-bit 0** of the arbitration loss capture register

		Bit			Decimal value	Function
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	Arbitration is lost in bit1 of the identification code
0	0	0	0	1	1	Arbitration is lost in bit 2 of the identification code
0	0	0	1	0	2	Arbitration is lost in bit3 of the identification code
0	0	0	1	1	3	Arbitration is lost in bit 4 of the identification code
0	0	1	0	0	4	Arbitration is lost in bit5 of the identification code
0	0	1	0	1	5	Arbitration is lost in bit 6 of the identification code
0	0	1	1	0	6	Arbitration is lost in bit7 of the identification code
0	0	1	1	1	7	Arbitration is lost in bit 8 of the identification code
0	1	0	0	0	8	Arbitration is lost in bit9 of the identification code
0	1	0	0	1	9	Arbitration is lost in bit10 of the identification code
0	1	0	1	0	10	Arbitration is lost in bit11 of the identification code
0	1	0	1	1	11	Arbitration is lost in the SRTR bit; Note 2
0	1	1	0	0	12	Arbitration lost in IDE bit
0	1	1	0	1	13	Arbitration is lost in bit 12 of the identification code; Note 3
0	1	1	1	0	14	Arbitration is lost in bit 13 of the identification code; Note 3
0	1	1	1	1	15	Arbitration is lost in bit 14 of the identification code; Note 3
1	0	0	0	0	16	Arbitration is lost in bit 15 of the identification code; Note 3
1	0	0	0	1	17	Arbitration is lost in bit16 of the identification code; Note 3
1	0	0	1	0	18	Arbitration is lost in bit17 of the identification code; Note 3
1	0	0	1	1	19	Arbitration is lost in bit18 of the identification code; Note 3
0	1	1	0	1	13	Arbitration is lost in bit 12 of the identification code; Note 3
0	1	1	1	0	14	Arbitration is lost in bit 13 of the identification code; Note 3
0	1	1	1	1	15	Arbitration is lost in bit 14 of the identification code; Note 3
1	0	0	0	0	16	Arbitration is lost in bit 15 of the identification code; Note 3
1	0	0	0	1	17	Arbitration is lost in bit16 of the identification code; Note 3
1	0	0	1	0	18	Arbitration is lost in bit17 of the identification code; Note 3
1	0	0	1	1	19	Arbitration is lost in bit18 of the identification code; Note 3
1	0	1	0	0	20	Arbitration is lost in bit19 of the identification code; Note 3
1	0	1	0	1	twenty one	Arbitration is lost in bit 20 of the identification code; Note 3
1	0	1	1	0	twenty two	Arbitration is lost in bit 21 of the identification code; Note 3
1	0	1	1	1	twenty three	Arbitration is lost in bit 22 of the identification code; Note 3
1	1	0	0	0	twenty four	Arbitration is lost in bit23 of the identification code; Note 3
1	1	0	0	1	25	Arbitration is lost in bit24 of the identification code; Note 3
1	1	0	1	0	26	Arbitration is lost in bit25 of the identification code; Note 3
1	1	0	1	1	27	Arbitration is lost in bit26 of the identification code; Note 3
1	1	1	0	0	28	Arbitration is lost in bit27 of the identification code; Note 3

Page 424

		Bit			Decimal value	Function
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
1	1	1	0	1	29	Arbitration is lost in bit28 of the identification code; Note 3
1	1	1	1	0	30	Arbitration is lost in bit29 of the identification code; Note 3
1	1	1	1	1	31	Arbitration is lost in the RTR bit; Note 3

Note: The number of binary code structure bits lost in arbitration.

RTR bit of standard frame information .

Only used for extended frame information.

24.6.14 **CAN** error code capture (**CAN_ECC**)

Only PeliCAN mode exists:

Offset address: 0x030

Reset value: 0x0000

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								ERRC1		DIR	SEG				
								r	r	r	r	r	r	r	r

Bit 31: 8	Reserved, the read value is 0.
Bit 7: 6	**ERRC[1 : 0]** : Error code 00: bit error 01: wrong format 10: Filling wrong 11: Other errors
Bit 5	**DIR** : Direction 1: RX; error occurred when receiving 0: TX; error occurred during transmission

Bit 4: 0	**ECC** [4:0]: Segment (Error code capture) 00010: ID.28-ID.21 00011: start of frame 00100: SRTR bit 00101: IDE bit 00110: ID.20-ID.18 00111: ID.17-ID.13 01000: CRC sequence 01001: reserved bit 0 01010: Data area 01011: Data length code 01100: RTR bit 01101: reserved bit 1 01110: ID.4-ID.0 01111: ID.12-ID.5 10001: Activity error flag 10010: Discontinued 10011: Domination (control) bit error 10110: Negative error flag

10111: Error delimiter
11000: CRC delimiter

11001: Response channel

11010: end of frame

11011: response delimiter

11100: overflow flag

Other: reserved

Note: The bit setting reflects the different error events of the current structure segment.

When an error occurs on the bus, it is forced to generate a corresponding error interrupt (when the interrupt is enabled). At the same time, the current position of the bit stream processor Error code capture register. The content remains unchanged until the user reads it out through the software. After reading, the capture mechanism is activated again. access During the interrupt register, the corresponding interrupt flag bit in the interrupt register is cleared. The new bus interrupt is not available until the capture register is read out once Can be effective.

24.6.15 CAN Error Alarm Limit Register (**CAN_EWLR**)

Only PeliCAN mode exists:

Offset address: 0x034

Reset value: 0x0096

The error alarm limit is defined in this register. The default value (reset value) is 96. In reset mode, this register is for the CPU It is readable/writeable. It is read-only in working mode.

Note: Only when you enter the reset mode before, *EWLR* may be changed. No error may occur until the reset mode is cancelled again The change of the error state (see the status register) and the error alarm interrupt caused by the new register content.

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								EWL							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	**EWL[7** : **0]** : Programmable error warning limit (Programmable error warning limit) When only one error counter exceeds the error limit programmed value, the error status bit is set.

24.6.16 CAN RX Error Counting Register (**CAN_RXERR**)

Only PeliCAN mode exists:

Offset address: 0x038

Reset value: 0x0000

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								RXERR							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	**RXERR[7** : **0]** : RX error counter register Reflects the current value of the reception error counter. The register is initialized to 0 after hardware reset. In working mode, the CPU It is read-only. Only in reset mode can write access to this register. If a bus shutdown occurs, the RX error counter is initialized to 0. When the bus is off, writing to this register is no Effective. Note: Only by entering the reset mode before, can the CPU force the RX error counter to change. Until reset After the mode is cancelled, the error status changes (see status register), error alarms and caused by the new register content

The error interrupt may be effective.

24.6.17 **CAN TX** Error Counting Register (**CAN_TXERR**)

Only PeliCAN mode exists:

Offset address: 0x03C

Reset value: 0x0000

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								TXERR							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	**TXERR[7 : 0]** : TX error counter register Reflects the current value of the transmission error counter. In working mode, this register is read-only memory for the CPU. Only in reset mode can write access to this register. hard After the software is reset, the register is initialized to 0. If the bus is closed, the TX error counter is initialized to 127 to calculate the total The minimum time defined by the line (128 bus idle signals). During this time, reading the TX error counter will reflect the bus Turn off the restored status information. If the bus shutdown is active, write access to units 0 ~ 254 of TXERR will clear the bus shutdown flag, and the reset mode will be After clearing, the controller will wait for an 11-bit continuous hidden (weak) bit (bus free). Writing 255 to TXERR will initiate the bus shutdown event driven by the CPU. Only when you enter the reset mode before, you can A change in the contents of the TX error counter caused by the CPU can occur. Until the reset mode is cancelled again, error or bus status Status changes (see status register), error alarms, and error interrupts caused by new register contents are possible. effect. After leaving the reset mode, as caused by a bus error, a new TX counter content is given and the bus is shut down. The same execution. This means that the reset mode is re-entered, the TX error counter is initialized to 127, and the RX counter is Cleared to 0, all relevant status and interrupt register bits are set. Clearing the reset mode will execute the bus shutdown recovery sequence specified in the protocol (waiting for 128 bus idle signals). If the reset mode is entered before the bus shutdown recovery (TXERR> 0), the bus shutdown remains valid and TXERR being locked.

Page 428

24.6.18 CAN Send Frame Information Register (CAN_SFF)

Only PeliCAN mode exists:

Offset address: 0x040

Reset value: 0x0000

The sending frame information register sets the type and data length of the sending frame. This register can be read/write accessed only in working mode, while writing The sending register is the receiving register when read, and the data structure is the same.

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								FF	RTR	X	X	DLC.3	DLC.2	DLC.1	DLC.0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7	**FF** : Frame format 1: EFF; CAN will send/receive extended frame format 0: SFF; CAN will send/receive standard frame format
Bit 6	**RTR** : Frame format (Remote transmission request) 1: Remote; CAN will send/receive remote frames 0: data; CAN will send/receive data frames
Bit 5: 4	Reserve.
Bit 3: 0	**DLC** : Data length code bit The length of the sending data area is 0 ~ 8.

24.6.19 CAN Transmit Identification Code Register 0 (CAN_TXID0)

Only PeliCAN mode exists:

Offset address: 0x044

Reset value: 0x0000

Send identification code register 0 to set the identifier of the frame. This register can be read/write accessed only in working mode, and it is send register when writing When reading, it is the receiving register, and the data structure is the same.

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								ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.

Page 429

Bit 7:0	**IDx** : CAN identifier ID28 ~ ID21 (Identifier bit 28-21) In a standard frame, it forms an 11-bit identifier with ID20 ~ ID18.

24.6.20 CAN Transmit Identification Code Register 1 (CAN_TXID1)

Only PeliCAN mode exists:

Offset address: 0x048

Reset value: 0x0000

Send identification code register 1 to set the identifier of the frame. This register can be read/write accessed only in working mode, and it is send register when writing When reading, it is the receiving register, and the data structure is the same.

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								ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7: 5	**IDx** : CAN identifier ID20 ~ ID18 (Identifier bit 20 -18) Standard frame Meaningless
Bit 40:	Extended frame **IDx** : CAN identifier ID17 ~ ID13 (Identifier bit 17-13)

24.6.21 CAN transmit data register **0** (**CAN_TXDATA0**)

Only PeliCAN mode exists:

Offset address: 0x04C

Reset value: 0x0000

Send data register 0, used for CAN data transmission and storage. This register can be read/write accessed only in working mode, while writing is sending Send register, receive register when read, the data structure is the same.

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								DATA0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.

Page 430

Bit 7:0	Standard frame **DATA0** : CAN transmit data 0 (CAN transmit data 0) Extended frame ID12 ~ 5

24.6.22 CAN transmit data register **1** (**CAN_TXDATA1**)

Only PeliCAN mode exists:

Offset address: 0x050

Reset value: 0x0000

Send data register 1, used for CAN data sending and storage. This register can be read/write accessed only in working mode, while writing is sending Send register, receive register when read, the data structure is the same.

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	DATA1							
	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31: 8	Reserve.
Bit 7:0	Standard frame **DATA1** : CAN transmit data 1 (CAN transmit data 1) Extended frame ID4 ~ 0, the lower 3 bits are meaningless

Only PeliCAN mode exists:

Offset address: 0x054 ~ 0x070

Reset value: 0x0000

The above offset addresses are all CAN data registers. When the frame is extended, the CAN sending data 0 is stored in DATA2 and the remaining data is analogized in turn. These registers are sending registers when writing, and receiving registers when reading, with the same data structure.

24.6.23 CAN clock divider register (**CAN_CDR**)

Offset address: 0x07C

Reset value: 0x0000

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								MODE	Reserve						
								rw							

Bit 31: 8	Reserve.
Bit 7	**MODE** : CAN mode (CAN mode) If MODE is 0, the CAN controller works in BasicCAN mode. Otherwise, the CAN controller works on PeliCAN model. It can be written only in reset mode.
Bit 6:0	Reserve.

Page 432

25. **USB** full-speed device interface

25.1 Introduction to **USB**

The USB peripheral implements the interface between the USB2.0 full-speed bus and the APB1 bus.

USB peripherals support USB suspend/resume operations, and can stop the device clock to achieve low power consumption.

25.2 Main Features of **USB**

- Comply with the technical specifications of USB2.0 full-speed devices
- Supports full-speed 12M mode and low-speed 1.5M mode.
- One control transmission endpoint and four independent general-purpose endpoints are used for interrupt transmission and bulk transmission.
- Control, batch and interrupt transmission can transmit packets up to 64 bytes.
- CRC (cyclic redundancy check) generation/checking, reverse non-return-to-zero (NRZI) encoding/decoding and bit stuffing
- Support USB suspend/resume operation
- Support DMA transfer

The following figure is a block diagram of USB peripherals

Figure **185. USB** block diagram

25.3 **USB** function description

The USB module provides a communication connection that conforms to the USB specification between the PC host and the functions implemented by the microcontroller. PC host and
The data transmission between the controllers is completed through the data buffer, the USB module communicates with the PC host, and realizes token grouping according to the USB specifi
Detection, data transmission/reception processing, and handshake packet processing. The format of the entire transmission is completed by hardware, including the generation of CRC
And check.

Each endpoint has a 64-byte buffer description block, and the buffer is inside the USB module and cannot be directly accessed by the CPU.
When the USB module recognizes a valid function/endpoint token grouping, (if data needs to be transmitted and the endpoint has been configured), the relevant
Data transfer. The USB module implements data exchange between the port and the dedicated buffer through an internal register. After all data transfer is complete
Then, if necessary, according to the direction of transmission, send or receive appropriate handshake packets.

At the end of the data transfer, the USB module will trigger an interrupt related to the endpoint, by reading the status register and/or using different interrupts.
Processing procedures, the microcontroller can determine:

- The type of transfer requested by the host
- Which endpoint needs to be serviced
- What type of transmission is in progress
- Endpoint's response
- Is the transfer complete

Whenever the USB module is not needed, the USB module can always be placed in low-power mode by writing to the USB_POWER register.
Mode (SUSPEND mode). In this mode, no dynamic current consumption is generated, and the USB clock is also slowed down or stopped. pass through
For the detection of data transmission on the USB line, the USB module can be awakened in low power consumption mode, or the USB system can be directly awakened through software sett
System, you can also connect a specific interrupt input source directly to the wake-up pin, so that the system can immediately restore the normal clock system, and support
Start or stop the clock system directly.

25.3.1 **USB** function module description

The USB module contains an 8-bit wide 320-byte FIFO, and each endpoint has a 64-byte FIFO. On the APB1 bus, each
The memory or storage data uses the lower 8 bits of the 32-bit wide bus.

The USB module contains the following registers:

- USB register. Used to configure USB parameters and query status
- Endpoint status register
- Endpoint setting register
- Setup data register, which stores the data of SETUP packet.
- The endpoint data control register controls the data flow.

When the APB1 bus or DMA writes or reads data to the data port, the number of FIFO data will increase or decrease. Only at the endpoint
After receiving and sending data successfully, the FIFO pointer will change. Each endpoint has a FIFO data register, as a write or read FIFO
The entry address. Each endpoint also has a special register to query the number of valid data in the current FIFO.

Only endpoint 1 and endpoint 2 support DMA transfer.

25.4 Problems to be considered in programming

In the following chapters, the interaction process between the USB module and the application program will be introduced, which will help simplify the development of the application p

25.4.1 Overview of **USB** transfer

The following shows the relationship between packages, things, and transmissions.

Packet is the basic unit of information transmission in the USB system. All data is packaged and transmitted on the bus.

The basic USB packages are as follows, such as token package, data package and handshake package.

Figure **186.** The basic format of the package

The process of receiving or sending data information on the USB is called a transaction. Types of transaction processing include input
Input (IN) transaction processing, output (OUT) transaction processing, setting (SETUP) transaction processing, etc.

Figure **187. USB** transaction

The following figure describes the complete transmission process of an endpoint. For bulk/control transmission, a transmission must be completed after the end of the previous transmiss
Start.

Figure **188. USB** transfer

When the USB controller starts to work, the USB is in the IDLE state, and then waits for the bus command. When a bus command is received, such as reset,
Setting, etc., will generate an interrupt, and the CPU will query the command type. For example, if the command is reset, the CPU will clear and reset all available
The state of programming. If it is a transmission request command, the CPU will write\read the data in the USB controller FIFO. For batch transfer of inputs or
When the CPU or DMA prepares the data, the CPU will send an ACK handshake signal or STALL handshake signal to end the transfer.
Beam transmission. For batch transmission output, when the data is put into the corresponding FIFO, the USB will automatically send an ACK signal.

When the data size exceeds the maximum packet size during transmission, the data will be divided into more than one packet for transmission, otherwise only one packet will be transmi

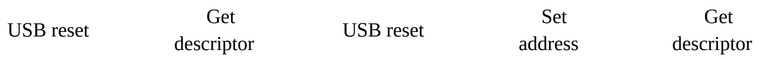
25.4.2 USB Enumeration

The host sends various requests to the device through endpoint 0 by means of control transmission, and the device restores the corresponding information after receiving the request from
Information, perform enumeration operations.

After the system is powered on and reset, first configure the USB controller:

1. Set the endpoint enable bit.
2. Turn on reset and endpoint interrupt
3. Open the connection bit (CONNECT bit in USB_TOP register)

Figure **189.** Enumeration process



When endpoint 0 receives the SETUP packet, the system will read the set 8 bytes of data to decide the next step. For SetAddress
Descriptor, the USB controller will automatically load the new address.

During the enumeration process, the first back and forth analysis.

When the device is detected, the host sends a bus reset. This reset is different from USB power-on reset and system reset. This is SIE according to the total
The line status informs the user of a kind of reset. The device generates a reset interrupt, and how to handle it is determined by the device firmware program.

The host initiates the first control transfer:

(1) Host SETUP packet (sent to address 0 and endpoint 0), host data packet (request device descriptor), device handshake packet ACK.

The device generates an endpoint 0 data output interrupt, and the firmware program should be prepared according to the host request in the data packet, here is the endpoint 0 input
The buffer is ready for the device descriptor.

(2) During the data process, the host first sends an IN token packet, and the device sends a data packet (this data has been prepared, and the SIE receives the IN order
After the card is signed, it is directly sent to the bus, and the user does not intervene at this time), and the host sends an ACK packet.

At this time, SIE generates an endpoint 0 data input interrupt, indicating that the host has taken the data prepared by the device, and the user can also use the interrupt
Do your own processing in the processing procedure. (SIE refers to the serial interface engine, which is the "core" inside all USB controllers. SIE is responsible for handling the underlying pro
The main task of SIE is to convert low-level signals into bytes for control
Controller)

At this time, the host only accepts data once, with a minimum of 8 bytes. If the user data has not been sent completely, and the buffer is entered in the control endpoint, the quasi-
When the data is prepared, the host ignores it.

(3) Status process: the host sends an OUT packet (notifies the device to output), the host sends a 0 byte status data packet (this is 0 byte, the table
It shows that it has received the device descriptor), and the device sends a handshake ACK packet.

At this time, the device will not generate endpoint 0 data output interrupt, and there is no data at this time.

During the enumeration process, the second back and forth: setting the address.

After the first round trip is successful, the host resets the bus again. Enter the address setting control transmission phase.

(1) Host SETUP packet (sent to address 0 and endpoint 0), host data packet (request to set address), device handshake packet ACK. Place
The SETUP packet will be followed by a data packet indicating the purpose of the host's SETUP, either GET or SET.

The device generates an endpoint 0 data output interrupt. The firmware program should be prepared according to the host's requirements in the data packet.
The incoming address is written into its own address control register.

(2) During the data process, there is no data in this transmission.

(3) Status process: the host sends an IN packet (notify the device to return data), the device sends a 0 byte status data packet (indicating that the address is set
After success), the host sends a handshake ACK packet (the address setting has taken effect).

At this time, the device will not generate endpoint 0 data input interrupt, and there is no data at this time.

During the enumeration process, the third round trip: the host uses the new address to obtain the complete device descriptor.

The host initiates the first control transfer with the new address:

(1) Host SETUP packet (sent to the new address endpoint 0), host data packet (request device descriptor), device handshake packet ACK.

The device generates an endpoint 0 data output interrupt, and the firmware program should be prepared according to the host request in the data packet, here is the endpoint 0 input
The buffer is ready for the device descriptor.

(2) During the data process, the host first sends an IN token packet, and the device sends a data packet (this data has been prepared, and the SIE receives the IN order
After the card is signed, it is directly sent to the bus, and the user does not intervene at this time), and the host sends an ACK packet.

At this time, SIE generates an endpoint 0 data input interrupt, indicating that the host has taken away the data prepared by the device, and the user can use this interrupt processing proce
The following processing should be done in the sequence: if the descriptor is not sent out once, the remaining content should be filled in the endpoint 0 input buffer again.

The second data transmission: the host sends another IN token packet, the device sends a data packet, and the host sends an ACK packet.

At this time, SIE generates endpoint 0 data input interrupt again, if the data has been sent out. I won't deal with it here. Enter the state process.

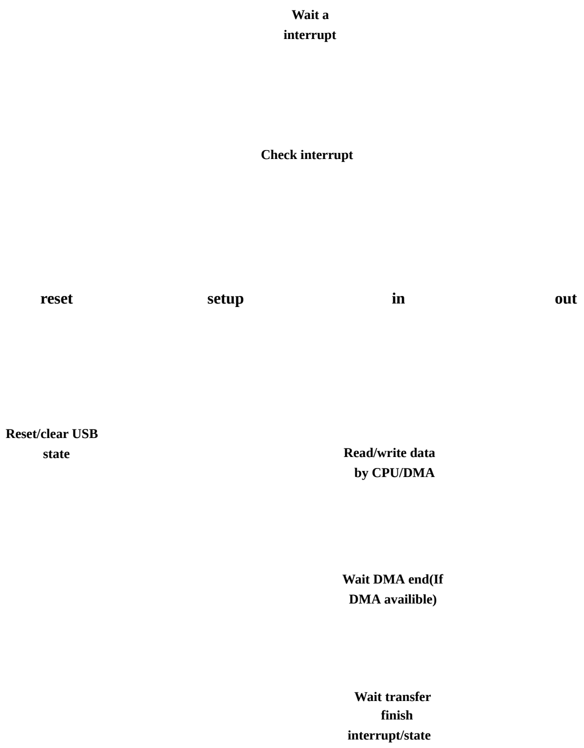
(3) Status process: the host sends an OUT packet (notifies the device to output), the host sends a 0-byte status data packet (indicating that it has received the device Descriptor), the device sends a handshake ACK packet.

25.4.3 USB transfer processing

The system needs to set up a no-operation loop to wait for the request from the USB host. The request of the USB host sets an interrupt bit, and the system communicates with By continuously querying the interrupt bit or entering the interrupt service routine, it breaks out of the loop. When the system detects the interrupt bit, it jumps to the corresponding subroutine reason.

The following figure is a flow chart of a typical USB transfer:

Figure **190. USB** transfer flow chart


Wait a interrupt
Check interrupt
reset
setup
in
out
Reset/clear USB state
Read/write data by CPU/DMA
Wait DMA end(If DMA availible)
Wait transfer finish interrupt/state

25.4.4 IN token package

When receiving an IN request from the host, if it receives a NACK response, the USB host will resend the IN request until the endpoint confirms Confirm that the request is valid. If the received address matches a configured endpoint address, you can follow the steps below:

1. If the data in the FIFO is smaller than the size set in the register EPX_CTRL[6:0], or EPX_CTRL[7] is not set to 1, The USB module will automatically send NACK until the data is ready.
2. The CPU receives the IN_NACK status.
3. The CPU writes data into the FIFO.
4. The CPU sets the size of the data to be sent and the send enable in the EPX_CTRL register.
5. The USB module automatically sends the data in the FIFO when it receives the next IN request. The last data byte has been sent After that, the calculated CRC is automatically sent.
6. The CPU will receive the IN_ACK status and will receive the END status after the transmission is completed.

7. If the endpoint is not enabled or the EP_HALT register setting is suspended, the USB module will answer IN_STALL without sending data.

Page 438

25.4.5 OUT token package

When receiving an OUT request from the host, if it receives a NACK response, the USB host will resend the OUT request until the end Click to confirm that the request is valid. If the received address matches a configured endpoint address, you can follow the steps below:

1. If the space in the FIFO is less than the size of the packet sent by the host, the USB module will automatically send a NACK until the CPU sends the data Remove from FIFO.
2. The CPU will receive the OUT_ACK status.
3. The CPU reads data from the FIFO.
4. If the endpoint is not enabled or the EP_HALT register setting is suspended, the USB module will answer OUT_STALL without sending data.

25.5 USB register description

25.5.1 USB TOP Register (USB_TOP)

Address offset: 0x00

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								ACTIVE	DP/DM STATE		SUSPEND	RESET	Reserve	CONNECT	SPEED
								rw	r	r	r	rw	rw	rw	rw

Bit 15: 8	Reserved, always read as 0
Bit 7	**ACTIVE**: USB bus is active (USB bus is active) 0: USB bus is not active 1: USB bus is active
Bit 6: 5	**DP/DM STATE** : Current USB DP/DM line state (Current USB DP/DM line state)
Bit 4	**SUSPEND** : USB SUSPEND state (USB suspend state) This bit monitors the controller status and has nothing to do with the APB_POWER register 0: The controller is working 1: The controller is in a suspended state
Bit 3	**RESET** : Reset the endpoint and FIFO of the USB controller (Reset EP and FIFO in USB controller) 0: do not reset 1: Reset Note that after setting this bit, it needs to be cleared by software.
Bit 2	Reserve
Bit 1	**CONNECT** : USB connection status (USB connection) 0: Disconnect 1: Connect
Bit 0	**SPEED** : Set the USB speed 0: Full speed transmission 1: Low-speed transmission

Page 439

25.5.2 USB Interrupt Status Register (USB_INT_STATE)

Address offset: 0x04

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve											EPINTF	SOFF	RESUMF	SUNPENDF	RSTF
											r	rc_w1	rc_w1	rc_w1	rc_w1

Bit 15: 7	Reserved, always read as 0
Bit 6: 5	Reserve
Bit 4	**EPINTF** : Endpoint interrupt flag bit (EP interrupt received) When any endpoint generates an interrupt, this bit is set to 1. Refer to EP_INT_STATE description for details. This bit is read-only.
Bit 3	**SOFF** : SOF detection flag (BUS received) When SOF is detected, this bit is set to 1. Write 1 to clear.
Bit 2	**RESUMF** : wake-up flag bit (BUS resume received) When the USB bus is activated, this bit is set to 1. Write 1 to clear.
Bit 1	**SUSPENDF** : USB bus suspend flag bit (BUS suspend received) This bit is set when the suspend state on the bus is detected. Write 1 to clear.
Bit 0	**RSTF** : USB bus reset request flag (BUS reset received) When the bus reset signal input is detected, this bit is set to 1. Write 1 to clear.

25.5.3 USB Endpoint Interrupt Status Register (EP_INT_STATE)

Address offset: 0x08

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve											EP4F	EP3F	EP2F	EP1F	EP0F
											r	r	r	r	r

Bit 15: 7	Reserved, always read as 0
Bit 4	**EP4F** : Endpoint 4 interrupt flag bit (EP4 interrupt received)
Bit 3	**EP3F** : Endpoint 3 interrupt flag bit (EP3 interrupt received)
Bit 2	**EP2F** : Endpoint 2 interrupt flag bit (EP2 interrupt received)
Bit 1	**EP1F** : Endpoint 1 interrupt flag bit (EP1 interrupt received)
Bit 0	**EP0F** : Endpoint 0 interrupt flag bit (EP0 interrupt received)

25.5.4 USB Endpoint 0 Interrupt Status Register (EP0_INT_STATE)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	SETUPF
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 15: 8	Reserved, always read as 0
	OUT-STALL : OUT packet response STALL identification (OUT-STALL received) After the endpoint receives the OUT packet from the host, the controller responds to STALL. Set EP_HALT[0] to 1 in the register and

Bit 7	When EP0_CTRL[7] is enabled, the USB controller will answer STALL. Write 1 to clear.
Bit 6	**OUT-ACK** : OUT packet response ACK identification (OUT-ACK received) After endpoint 0 receives the OUT packet from the host, there is enough space in the FIFO, and the host completes the current transmission. USB controller Acknowledge ACK automatically. Write 1 to clear.
Bit 5	**OUT-NACK** : OUT packet response NACK identification (OUT-NACK received) After the endpoint receives the OUT packet from the host, there is not enough space to store the data sent from the host at this time. USB control The controller automatically answers NACK. Write 1 to clear.
Bit 4	**IN-STALL** : IN packet response STALL identification (IN-STALL received) After the endpoint receives the IN packet from the host, the controller responds to STALL. Set EP_HALT[0] to 1 in the register and When EP0_CTRL[7] is enabled, the USB controller will answer STALL. Write 1 to clear.
Bit 3	**IN-ACK** : IN packet response ACK identification (IN-ACK received) After the endpoint receives the IN packet from the host, there is enough data in the FIFO, and the host completes the current transmission. USB controller since Acknowledge ACK automatically. Write 1 to clear. After the endpoint receives the IN packet from the host, the host completes the current transmission and sets this bit.
Bit 2	**IN-NACK** : IN packet response NACK identification (IN-NACK received) After the endpoint receives the IN packet from the host, but there is not enough data to complete the current transmission at this time. USB controller auto answer NACK. Write 1 to clear.
Bit 1	**END** : Transmission completed identification (Status stage finished) When the endpoint transfer is complete, this bit is set to 1. Write 1 to clear.
Bit 0	**SETUP** : SETUP packet received (SETUP packet received) After setting the bit, the contents of the SETUP packet can be read from the registers SETUP0〜7. Write 1 to clear.

25.5.5 USB Interrupt Enable Register (USB_INT_EN)

Address offset: 0x10

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								Reserve			EPIE	SOFIE	RESUMIE	SUNPENDIE	RSTIE
											rw	rw	rw	rw	rw

Bit 15: 8	Reserved, always read as 0
Bit 7: 5	Reserve
Bit 4	**EPINTIE** : EP interrupt enable bit
Bit 3	**SOFIE** : SOF detection interrupt enable bit (SOF interrupt enable)
Bit 2	**RESUMIE** : BUS resume interrupt enable bit (BUS resume interrupt enable)
Bit 1	**SUSPENDIE** : USB bus suspend interrupt enable bit (BUS suspend interrupt enable)
Bit 0	**RSTIE** : USB bus reset interrupt enable bit (BUS reset interrupt enable)

25.5.6 USB Endpoint Interrupt Enable Register (EP_INT_EN)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve											EP4IE	EP3IE	EP2IE	EP1IE	EP0IE

rc_w1 rc_w1 rc_w1 rc_w1 rc_w1

Bit 15: 5	Reserved, always read as 0
Bit 4	**EP4IE** : Endpoint 4 interrupt enable bit (EP4 interrupt enable)
Bit 3	**EP3IE** : Endpoint 3 interrupt enable bit (EP3 interrupt enable)
Bit 2	**EP2IE** : Endpoint 2 interrupt enable bit (EP2 interrupt enable)
Bit 1	**EP1IE** : Endpoint 1 interrupt enable bit (EP1 interrupt enable)
Bit 0	**EP0IE** : Endpoint 0 interrupt enable bit (EP0 interrupt enable)

Page 442

25.5.7 USB Endpoint 0 Interrupt Enable Register (EP0_INT_EN)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								OUT_ STALL IE	OUT-A CKIE	OUT- NACK IE	IN-ST ALLIE	IN-AC KIE	IN-NA CKIE	ENDI E	SETU PIE
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 15: 8	Reserved, always read as 0
Bit 7	**OUT-STALLIE** : OUT packet response STALL interrupt enable bit (OUT-STALL interrupt enable)
Bit 6	**OUT-ACKIE** : OUT packet response ACK interrupt enable bit (OUT-ACK interrupt enable)
Bit 5	**OUT-NACKIE** : OUT packet response NACK interrupt enable bit (OUT-NACK interrupt enable)
Bit 4	**IN-STALLIE** : IN packet response STALL interrupt enable bit (IN-STALL interrupt enable)
Bit 3	**IN-ACKIE** : IN packet response ACK interrupt enable bit (IN-ACK interrupt enable)
Bit 2	**IN-NACKIE** : IN packet response NACK interrupt enable bit (IN-NACK interrupt enable)
Bit 1	**ENDIE** : Status stage finished interrupt enable bit (Status stage finished interrupt enable)
Bit 0	**SETUPIE** : Received SETUP packet interrupt enable bit (SETUP packet interrupt enable)

25.5.8 USB endpoint X interrupt status register (EPX_INT_STATE) (X=1~4)

Address offset: 0x20~0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								OUT-S TALLF	OUT-A CKF	OUT- NACK F	IN-ST ALLF	IN-AC KF	IN-NA CKF	END reserved	
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	

Bit 15: 8	Reserved, always read as 0
Bit 7	**OUT-STALL** : OUT packet response STALL identification (OUT-STALL received) After the endpoint receives the OUT packet from the host, the controller responds to STALL. Set EP_HALT[0] to 1 in the register and When EP0_CTRL[7] is enabled, the USB controller will answer STALL. Write 1 to clear.

OUT-ACK : OUT packet response ACK identification (OUT-ACK received)

Bit 6	After the endpoint receives the OUT packet from the host, there is enough space in the FIFO, and the host completes the current transmission. USB controller since Acknowledge ACK automatically. Write 1 to clear.

Bit 5	**OUT-NACK** : OUT packet response NACK identification (OUT-NACK received) After the endpoint receives the OUT packet from the host, there is not enough space to store the data sent from the host at this time. USB control The controller automatically answers NACK. Write 1 to clear.
Bit 4	**IN-STALL** : IN packet response STALL identification (IN-STALL received) After the endpoint receives the IN packet from the host, the controller responds to STALL. Set EP_HALT[0] to 1 in the register and When EP0_CTRL[7] is enabled, the USB controller will answer STALL. Write 1 to clear.
Bit 3	**IN-ACK** : IN packet response ACK identification (IN-ACK received) After the endpoint receives the IN packet from the host, there is enough data in the FIFO, and the host completes the current transmission. USB controller since Acknowledge ACK automatically. Write 1 to clear. After the endpoint receives the IN packet from the host, the host completes the current transmission and sets this bit.
Bit 2	**IN-NACK** : IN packet response NACK identification (IN-NACK received) After the endpoint receives the IN packet from the host, but there is not enough data to complete the current transmission at this time. USB controller auto answer NACK. Write 1 to clear.
Bit 1	**END** : Transfer finished mark (Transfer finished) When the endpoint transfer is complete, this bit is set to 1. Write 1 to clear.
Bit 0	Reserve

25.5.9 USB endpoint X interrupt enable register (EPX_INT_EN) (X=1~4)

Address offset: 0x40 to 0x4C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	Reserve
								rw	rw	rw	rw	rw	rw	rw	

Bit 15: 8	Reserved, always read as 0
Bit 7	**OUT-STALLIE** : OUT packet response STALL interrupt enable bit (OUT-STALL interrupt enable)
Bit 6	**OUT-ACKIE** : OUT packet response ACK interrupt enable bit (OUT-ACK interrupt enable)
Bit 5	**OUT-NACKIE** : OUT packet response NACK interrupt enable bit (OUT-NACK interrupt enable)
Bit 4	**IN-STALLIE** : IN packet response STALL interrupt enable bit (IN-STALL interrupt enable)
Bit 3	**IN-ACKIE** : IN packet response ACK interrupt enable bit (IN-ACK interrupt enable)
Bit 2	**IN-NACKIE** : IN packet response NACK interrupt enable bit (IN-NACK interrupt enable)
Bit 1	**ENDIE** : Finished interrupt enable bit (Finished interrupt enable)
Bit 0	Reserve

25.5.10 **USB** Address Register (**USB_ADDR**)

Address offset: 0x60

Reset value: 0x0000 0000

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

Bit 15: 7	Reserved, always read as 0
Bit 6:0	**ADDR[6** : **0]** : USB address (USB address) When receiving the setting address descriptor sent by the host, the hardware automatically loads the address into this register. The hardware automatically clears the value of this register when the bus reset is received.

25.5.11 **USB** Endpoint Enable Register (**EP_EN**)

Address offset: 0x64

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve											EP4EN	EP3EN	EP2EN	EPIEN	EP0EN
											rw	rw	rw	rw	rw

Bit 15: 7	Reserved, always read as 0
Bit 4	**EP4EN** : Enable End Point 4 (Enable End Point 4)
Bit 3	**EP3EN** : Enable End Point 3 (Enable End Point 3)
Bit 2	**EP2EN** : Enable End Point 2
Bit 1	**EP1EN** : Enable End Point 1 (Enable End Point 1)
Bit 0	**EP0EN** : Enable End Point 0 (Enable End Point 0)

25.5.12 **USB** Data Toggle Control Register (**TOG_CTRL1_4**)

Address offset: 0x78

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								DTOG4EN	DTOG4	DTOG3EN	DTOG3	DTOG2EN	DTOG2	DTOG1EN	DTOG1
								w	rw	w	rw	w	rw	w	rw

Bit 15: 8	Reserved, always read as 0
Bit 7	**DTOG4EN** : End Point 4 data flip enable bit (Set End Point 4 enable) 0: Do not change the data flip bit of endpoint 4 1: Set bit 6 DTOG4 as the data flip bit of endpoint 1
Bit 6	**DTOG4** : Set End Point 4 Toggle 0: DATA0 1: DATA1
Bit 5	**DTOG3EN** : End Point 3 data flip enable bit (Set End Point 3 enable) 0: Do not change the data flip bit of endpoint 3 1: Set bit 4 DTOG3 as the data flip bit of endpoint 1
Bit 4	**DTOG3** : Set End Point 3 Toggle 0: DATA0

	1: DATA1
Bit 3	**DTOG2EN** : End Point 2 data toggle enable bit (Set End Point 2 enable) 0: Do not change the data flip bit of endpoint 2 1: Set bit 2 DTOG2 as the data toggle bit of endpoint 1
Bit 2	**DTOG2** : Set End Point 2 Toggle 0: DATA0 1: DATA1
Bit 1	**DTOG1EN** : End Point 1 data flip enable bit (Set End Point 1 enable) 0: Do not change the data flip bit of endpoint 1 1: Set bit 0 DTOG1 as the data toggle bit of endpoint 1
Bit 0	**DTOG1** : Set End Point 1 Toggle 0: DATA0 1: DATA1

25.5.13 USB Setup Packet Data Register (**SETUPX**) (**0~7**)

Address offset: 0x80 to 0x9C

Reset value: 0x0000 0000

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

Bit 15: 8	Reserved, always read as 0
Bit 7:0	**SETUPDX** : USB setup packet data bit (x = 0,1,2,..,7) (Setup Data X) The 64-bit SETUP data is automatically set by the hardware according to the data sent by the host.

25.5.14 USB transfer packet size register (**PACKET_SIZE**)

Address offset: 0xA0

Reset value: 0x40

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								SIZE0[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 15: 8	Reserved, always read as 0
Bit 7:0	**SIZE0** : USB maximum transmission packet size (USB DMA Max Packet Size) Up to 64 bytes can be set.

25.5.15 USB Endpoint **X** Effective Data Register (**EPX_AVAIL**)

Address offset: 0x100 to 0x110

Reset value: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								EPXAVAIL7:0]							
								r	r	r	r	r	r	r	r

Bit 15: 8	Reserved, always read as 0
Bit 7:0	

EPXAVIL : USB endpoint X FIFO available data number (EPX FIFO available data number)

25.5.16 USB Endpoint **X** Control Register (**EPX_CTRL**)

Address offset: 0x140 to 0x150

Reset value: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve								TRAN EN	TRANCOUNT						
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 15: 8	Reserved, always read as 0
Bit 7	**TRANEN** : USB endpoint X transfer enable bit (EPX transfer enable) If this bit is set to 1, the endpoint X will acknowledge the number of data defined in bit 6:0 after the IN transmission, otherwise the endpoint X answers NACK. If there is not enough data in the endpoint x FIFO, endpoint X will also respond with NACK. If the endpoint X HALT enable bit is set to 1, endpoint X will automatically answer STALL. This bit will automatically change to 0 at the end of the transmission.

Bit 6:0	**TRANCOUNT** : Endpoint X transfer count (EPX transfer counter) The number of data to be transmitted by endpoint X. The data is stored in the FIFO of each endpoint, and the maximum number of transfers cannot exceed the register The maximum packet size defined by PACKAGE_SIZE, the transmission quantity of the last packet may be less than the maximum packet, or even Think of it as zero, which means that an empty packet is transmitted.

25.5.17 USB Endpoint **X FIFO** Register (**EPX_FIFO**)

Address offset: 0x160 to 0x170

Reset value: 0x00

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

Bit 15: 8	Reserved, always read as 0
Bit 7:0	**EPX_FIFO** : Endpoint X FIFO data port (EPX FIFO port)

25.5.18 USB Endpoint **DMA** Enable Register (**EP_DMA**)

Address offset: 0x184

Reset value: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve														DMA2 EN	DMA1 EN
														rw	rw

Bit 15: 2	Reserved, always read as 0
Bit 1	**DMA2EN** : Endpoint 2 DMA enable bit (EP2 DMA enable)
Bit 0	**DMA1EN** : Endpoint 1 DMA enable bit (EP1 DMA enable)

Note: *the USB* controller supports only Endpoint *1* and Endpoint *2* of the *DMA* operation.

25.5.19 USB Endpoint Halt Register (**EP_HALT**)

Address offset: 0x188

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve											HALT4	HALT 3	HALT 2	HALT 1	HALT 0
											rw	rw	rw	rw	rw

Page 448

Bit 15: 5	Reserved, always read as 0
Bit 4	**HALT4** : Endpoint 4 halt bit (EP4 halt) When this bit is set to 1, the device will automatically respond to STALL after IN/OUT transmission. This bit will be hardened when a token packet is received The pieces are automatically cleared.
Bit 3	**HALT3** : Endpoint 3 halt bit (EP3 halt) When this bit is set to 1, the device will automatically respond to STALL after IN/OUT transmission. This bit will be hardened when a token packet is received The pieces are automatically cleared.
Bit 2	**HALT2** : Endpoint 2 halt bit (EP2 halt) When this bit is set to 1, the device will automatically respond to STALL after IN/OUT transmission. This bit will be hardened when a token packet is received The pieces are automatically cleared.
Bit 1	**HALT1** : Endpoint 1 halt bit (EP1 halt) When this bit is set to 1, the device will automatically respond to STALL after IN/OUT transmission. This bit will be hardened when a token packet is received The pieces are automatically cleared.
Bit 0	**HALT0** : Endpoint 0 halt bit (EP0 halt) When this bit is set to 1, the device will automatically respond to STALL after IN/OUT transmission. This bit will be hardened when a token packet is received The pieces are automatically cleared.

25.5.20 USB power control register (**USB_POWER**)

Address offset: 0x1C0

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve												WKUP retains SUSP			SUSP EN
											rw	rw	rw	rw	rw

Bit 15: 4	Reserved, always read as 0
Bit 3	**WKUP** : Enable controller wake up from suspend state 1: wake up 0: do not wake up
Bit 2	Reserve
Bit 1	**SUSP** : Suspend bit (suspend) 1: Normal working mode 0: Suspend mode
Bit 0	**SUSPEN** : BUS suspend enable bit 1: The controller directly controls whether the USB is suspended according to the status of bit 1 0: The controller controls whether to suspend the signal

Page 449

26. TK80

26.1 Features

- AHB2.0 bus interface
- Support read and write commands and read and write data operations
- Read (data or command) supports two modes of direct reading and interrupt/query reading
- Support blind reading
- Supports sequential read data operations in the memory area
- Support area filling operation
- Support DMA (only support write)
- CS_n supports two methods of automatic hardware generation and software generation
- Support read transfer completion and write transfer completion interrupt
- Two-way port half-duplex data transmission

26.2 Interface signal

Name	IO	Function and Description
AHB bus	BUS	AHB bus, specific signals are temporarily omitted
CS_n	O	Peripheral chip select signal 0, indicating that the peripheral is selected 1, indicates that the peripheral is not selected
RS	O	Command or data enable signal 0, which means that the command operation is executed, and the command is transmitted on D23-D0 1, indicates that the data operation is performed, and the data is transmitted on D23-D0
WR_n	O	Write enable Used to control the output of data to the D23-D0 port, the data on D23-D0 will be latched on the rising edge of WR
RD_n	O	Read enable Used to control the reading of data from the D23-D0 port, the data on D23-D0 will be valid after the rising edge of RD
D23-D0	I/O	Data input and output signal D23-D0 are used to output commands and data to the peripheral, or read data from the peripheral. D23 is the most significant data, D0 Is the least significant data
I80_busy	I	Peripheral busy signal
dma_req	O	DMA request
dma_ack	I	DMA response
irq	O	Interrupt

26.3 Interface Timing

26.3.1 Write command

26.3.2 Write data

26.3.3 Read Command

26.3.4 Read data

26.3.5 I80_busy

In any case, if it encounters that I80_busy is valid, stop the corresponding operation directly, and wait for I80_busy to fail, and then continue with the previous operate. The time waiting for I80_busy is not included in the control signal counting time. An example of writing data is as follows.

26.4 Timing parameters

Unit: clock cycles

T_start	See configuration register CFGR1
T_write	See configuration register CFGR1
T_read	See configuration register CFGR1
T_sample	See configuration register CFGR1
T_busy	See configuration register CFGR2
T_end	See configuration register CFGR2

26.5 Register description

26.5.1 Control Register **CR**

offset: 0x00

default: 0x000c0b02

Bit 31:22	res
Bit 21:18	narrow_num: The valid valid time of the output data shortens the cycle The actual cycle is narrow_num+1
Bit 17	narrow_valid: The valid time of output data is shortened 0: do not shorten

	1: shorten
Bit 16	burst: fill mode 1: Start filling mode 0: non-filled mode
Bit 15:12	res
Bit 11	busy_val 0: External busy signal is invalid 1: External busy signal is valid

Bit 10	busy_resp_level 0: I80_busy is high, indicating that the device is busy 1: I80_busy is low, indicating that the device is busy
Bit 9:8	read_mode: define two read data modes 00: Read command mode 2 01: Read data mode 2 10: Read command mode 1 11: Read data mode 1
Bit 7:3	res
Bit 2	CS_sel: Software CS selection signal 0: CS generated by hardware 1: Software generates CS
Bit 1	CS_soft_n: software CS, valid when CS_soft is 1, invalid when CS_soft is 0 0: Software pulls down the CS_n signal 1: Software pulls up the CS_n signal
Bit 0	dma_en: DMA enable 1: DMA enable 0: DMA disabled

26.5.2 Configuration Register **CFGR1**

offset: 0x04

default: 0x01cfcf01

Bit 31:24	T_sample, the actual value is T_sample +1, T_sample>1
Bit 23:16	T_read, the actual value is T_read +1, T_read>1
Bit 15:8	T_write, the actual value is T_write +1, T_write>1
Bit 7:0	T_start, the actual value is T_start +1, T_start>1

26.5.3 Configuration Register **CFGR2**

offset: 0x08

default: 0x00010101

Bit 31:16	res
Bit 15:8	T-Busy
Bit 7:0	T-end, the actual value is T_end+1, T_end>1

Page 454

26.5.4 Status Register **SR**

offset: 0x0C

default: 0x00000000

Bit 31:17	res
Bit 16	busy: Determine whether the TK80 is operating on the device 1: Indicates that the TK80 interface is operating on the device 0: No operation
Bit 15:12	res
Bit 11	write_single_ie Single write data completion interrupt enable
Bit 10	read_single_ie Single read data completion interrupt enable
Bit 9	write_ie Write data complete interrupt enable
Bit 8	read_ie Read data complete interrupt enable
Bit 7: 4	res
Bit 3	write_end_single: Single write data is completed, write 1 to clear
Bit 2	read_end_single Single read data is completed, write 1 to clear
Bit 1	write_end Write data complete, write 1 to clear
Bit 0	read_end Read data is complete, write 1 to clear

26.5.5 Command Input Register **CMDIR**

offset: 0x10

default: 0x00000000

31:24	res
23:0	cmdi:

26.5.6 Data Input Register **DINR**

offset: 0x14

default: 0x00000000

31:24	res
23:0	din:

The CPU transfers data to the peripheral by writing to this register, and the value of this register is transferred to the peripheral through D23-D0

Page 455

26.5.7 Command output register **CMDOR**

offset: 0x18

default: 0x00000000

31:24	res
23:0	cmdo:

26.5.8 Data output register **DOUTR**

offset: 0x20

default: 0x00000000

31:24	res
23:0	dout:

26.5.9 Blind Read Data Output Register **BRDR**

offset: 0x24

default: 0x00000000

31:24	res
23:0	brdr:

26.5.10 Configuration Register **CFGR3**

offset: 0x30

default: 0x00000001

Bit 31:0	circle_num: number of fill points 0: Single point filling N: automatic filling of N points