AC7801x Reference Manual

Version: 1.5

Release date: 2022-03-17

Document Revision History

Revision	Date	Author	Description
1.0	2020-09-25	AutoChips	Initial Version
1.1	2020-11-02	AutoChips	1. Modify the description of SPI master mode fault in chapter 18.4.6 2. Modify the description of RS485 delay in chapter 8.6.16 3. Modify the description of stop mode for UART in chapter 8.4.7 4. Modify the description of the accuracy for square root in chapter 24.4
1.2	2020-12-21	AutoChips	1. Modify WDT to WDG 2. Modify Clock control diagram in Figure 4-1 3. Reorder the annotation labels in the module functions in the low-power mode of Table 5-1 4. Modify the register of bit width in chapter 8.6.15 5. Modify the ADC calibration description in chapter 9.4.6 6. Modify the RDY flag to RDYF 7. Add description of 32pin does not support i2c1 in chapter 16.3.2 32 8. Update the description of GPIO_ODR register in table16-5 9. Update the description of GPIO_PINMUX register in chapter 16.5.9 10. Add the description of disable DMA in circular mode in chapter 19.6.11 11. Modify EOP_INT to EOP in chapter 22.6.5 12. Modify ECC error address in chapter 23.4.2
1.3	2021-01-28	AutoChips	1. Update the way of exit BUSOFF: module (SRST_CAN0) reset or receive 128 consecutive groups of 11 recessive bits (recovery sequence), CAN node can return to active error state in chapter 7.3.2 2. Add description of TDC and SSPOFF in chapter 7.3.12 3. Modify operation mode sequence figure error in chapter 9 4. Modify rising edge to effective edge in chapter 11.4.18 5. Add input event description in chapter 11.5.20 6. Add comment PA12 and PA15 is configured XOSC function, switching to other multi-function is not supported in chapter 16.3.2 7. Add description of typical value for pull-up, pull-down resistor, in chapter 16.6.6 and 16.6.7

			8. Modify SAMPLE_CNT_DIV to SAMPLE_CNT in chapter 17.6.3 9. Modify STEP_CNT_DIV to STEP_CNT in chapter 17.6.4 10. Modify RTC_CLKOUT pin PA9 to PA13 in chapter 21.4.3 11. Modify RTCO change to RTC_CLKOUT in chapter 21.6.1
1.4	2021-06-15	AutoChips	1. Update the description of KOER in chapter 7.3.9.3 2. Update the description of the 9th bit of register CAN_CTRL0 in chapter 7.4.2 3. Update the description of the DR bit in UART_LSR1 in chapter 8.6.9 4. Update the description of operation in chapter 11.4.20 5. Add PWM0_FLT0 function for PA12 in Table 16-2 6. Add detailed description of address operation in chapter 17.3.1 7. Add description of host synchronization in chapter 17.4.1 8. Add detailed description of Bnd behavior in chapter 17.6.9 9. Delete DMA peripheral to peripheral transfer in chapter 19.2 10. Change register default value revise to 0x1F8000 in chapter 20.6.4 11. Add description the minimum programming bit width is 32 bits, and the programming address needs to be aligned with 4 bytes in chapter 22.2.
1.5	2022-03-17	AutoChips	1. Add description of ISP download pin about "ISP download pins are PA7 (TX) and PA8 (RX) of UART" in chapter 2.2.11. 2. Update description of BUSOFF bit in chapter 7.4.2. 3. Add description of UART baudrate range in chapter 8.2. 4. Add description of AMOHR/AMOLR unit in chapter 9.4.4. 5. Add AMO edge mode range description in chapter 9.4.4.2. 6. Add application suggestion of CALEN in chapter 9.5.3. 7. Add description about the numbering rule of SPTx in chapter 9.5.4 and 9.5.5. 8. Modify Figure 11-23 Dead-time insertion in chapter 11.4.8.4. 9. Add chapter 11.4.19 Registers updated from write buffers. 10. Add chapter 11.4.21 Features priority.

			11. Add description about "MDIS can pause and restart counting of 4 timers at the same time" in chapter 13.5.1. 12. Add the conversion formula from CVAL to time in chapter 13.5.3. 13. Modify description of the GPIO output state in chapter 16.2. 14. Add chapter 16.3 Block diagram for GPIO. 15. Add the description of GPIO pull-down/pull-up in chapter 16.6.6 and 16.6.7. 16. Update I2C rate to 100KHz, 400KHz, 1MHz in chapter 17.2. 17. Modify description of the I2C_ADDR1[RAD] bit in chapter 17.6.2. 18. Add description of SPI baudrate range in chapter 18.2. 19. Add description of SPI_CMD[SPI TXEIE] and SPI_CMD[RXFIE] bits in chapter 18.6.3. 20. Modify description of DMA priority in chapter 19.4.3. 21. Modify description of DMA end address in chapter 19.5.3 and 19.6.9. 22. Add description of read/write protection in chapter 22.4.2. 23. Modify description of CMD_ST bit in chapter 22.6.4. 24. Add instruction for clearing the error status bit of the EFLASH_SR0 register in chapter 22.6.5.

Legal Notice

This reference manual contains information that is confidential to AUTOCHIPS Inc. Unauthorized use or disclosure of the information contained herein is prohibited. You may be held responsible for any loss or damages suffered by Autochips Inc. for your unauthorized disclosure hereof, in whole or in part.

Information herein is subject to change without noticed. AUTOCHIPS Inc. does not assume any responsibility for any use of, or reliance on, the information contained herein.

THIS REFERENCE MANUAL AND ALL INFORMATION CONTAINED HEREIN IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. AUTOCHIPS INC. SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER DOES AUTOCHIPS INC. PROVIDE ANY WARRANTY WHATSOEVER WITH RESPECT TO THE SOFTWARE OF ANY THIRD PARTY WHICH MAY BE USED BY, INCORPORATED IN, OR SUPPLIED WITH THIS REFERENCE MANUAL, AND USER AGREES TO LOOK ONLY TO SUCH THIRD PARTY FOR ANY WARRANTY CLAIM RELATING THERETO. AUTOCHIPS INC. SHALL ALSO NOT BE RESPONSIBLE FOR ANY AUTOCHIPS DELIBERABLES MADE TO USER'S SPECIFICATION OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Table of Contents

List of Figures

List of Tables

Abbreviation

AAI	Auto Address Increment
AHB	Advanced High Performance Bus
APB	Advanced Peripheral Bus
CKGEN	Clock Generator
ECC	Error Checking and Correction
FBKDIV	Feedback Divider
HSE	High Speed External Clock
HSI	High Speed Internal Clock
ISP	In-System Programming
LRU	Least Recently Used
LSI	Low Speed Internal Clock
LVD	Low Voltage Detect
MMDIVSQRT	Memory-Mapped Division and Square Root
NMI	Non Maskable Interrupt
OSC	Oscillator
PLL	Phase Locked Loop
POR	Power On Reset
POSDIV	Post Divider
PREDIV	Previous Divider
PVD	Programmable Voltage Detect
SPM	System Power Manager
SRAM	Static Random-Access Memory
SYSPLL	System Phase Locked Loop
VCO	Voltage Controlled Oscillator
XOSC	External Crystal Oscillator

1 Introduction

1.1 Overview

AC7801x is a high-performance, low-power MCU using ARM Cortex™-M0+ core.

- Up to 48 MHz CPU frequency (some chip models up to 72MHz)
- Temperature range (ambient): -40°C to +125°C
- Voltage range: 2.7 V to 5.5 V

1.2 Module description

The following sections describe the modules assigned to each category in more detail.

Table 1-1 AC7801x module description

Module	Description
ARM Cortex™-M0+ core	• 32-bit MCU core from ARM Cortex™-M0+ class • Up to 48 MHz CPU frequency (some chip models up to 72 MHz)
Memories	• Internal memories include: – Up to 128 Kbyte flash memory – Up to 20 Kbyte SRAM
Clocks	• External crystal oscillator or resonator – Range: 4 MHz to 30 MHz • External square wave input clock • Internal oscillator – 8 MHz oscillator – 32 kHz oscillator
Analog	• One 12-bit analog-to-digital converter(ADC) with up to 12 external channels and 2 internal channels • One analog comparators (ACMP) with internal 6-bit digital-to-analog converter (DAC)
Timers	• Two 8-channel PWM controllers • Four 32bit general timers(Timer) • Real time clock (RTC) • Two pulse width detection timers (PWDT) • System tick timer (SysTick)

Module	Description
Communications	• Two serial peripheral interfaces (SPI) • Two inter-integrated circuit (I2C) modules • Three universal asynchronous receiver/transmitter (UART) modules, only UART0 and UART1 supporting LIN break • One controller area network (CAN),supporting CANFD
Human-Machines Interfaces	• General purpose input/output (GPIO) controller • Interrupt (IRQ)
Debug Interfaces	• Serial Wire Debug(SWD) interface
Algorithm Modules	• One MMDIVSQRT(Memory-Mapped Division And Square Root) module

2 Memory and Bus Architecture

2.1 System architecture

The main system of AC7801x consists of:

- Two masters
  - Cortex™-M0+ core AHB-Lite bus.
  - DMA (general-purpose DMA).
- Four slaves
  - Internal SRAM.
  - Internal Flash memory.
  - Fast IO, CRC and GPIO.
  - AHB to APB bridges (AHB_APB), which connect all the APB peripherals.

These are interconnected using a multilayer AHB bus architecture as shown in Figure 2-1.

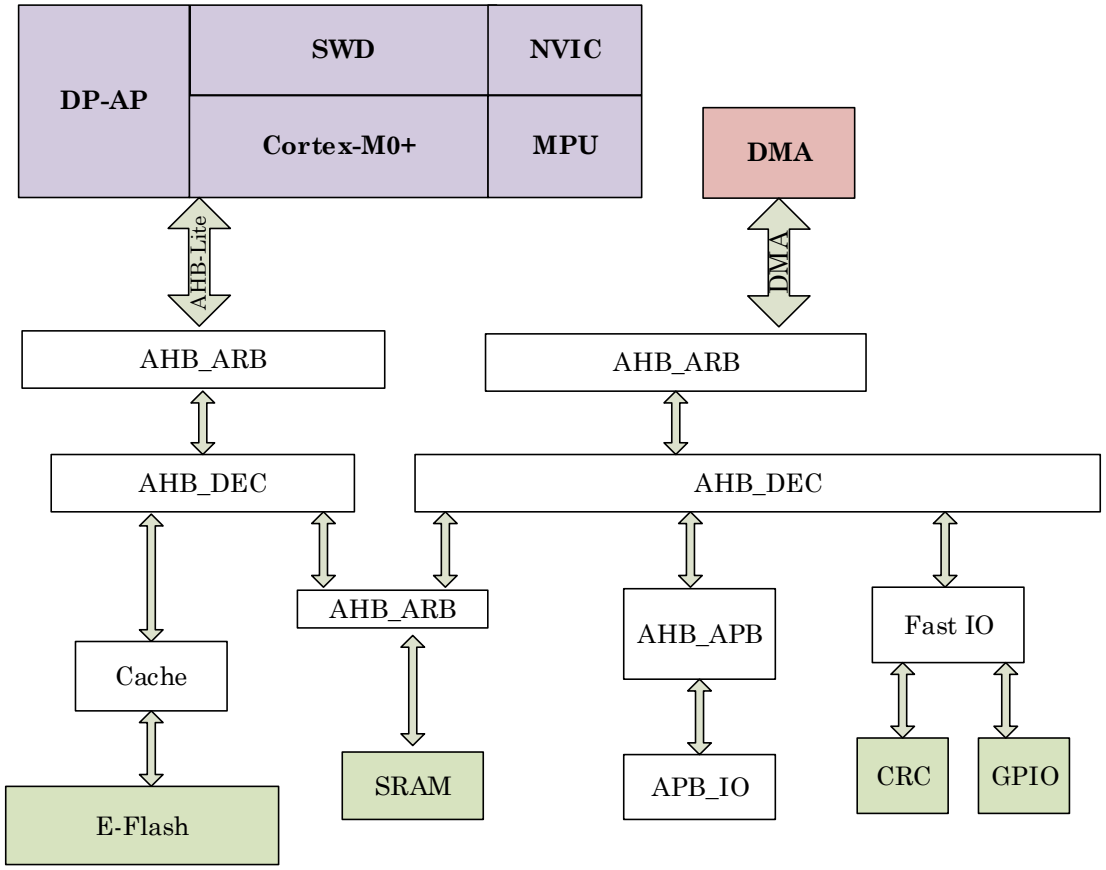

Figure 2-1 System architecture

AHB-Lite bus

This bus sends instruction and data requests of the Cortex™-M0+ core and accesses memories and peripherals.

AHB_ARB

The AHB arbitration module is used for the arbitration of the AHB bus and the DMA bus.

AHB_DEC

The AHB Decode module.

The arbitrated address will be decoded according to the address range, and then the decoded address can access different slavers.

MPU

AC7801x supports MPU function, which is disabled by default. Please enable relevant configuration when using, please refer to the ARM Cortex™-M0+ Technical Reference Manual.

DMA bus

This bus connects the AHB master interface of the DMA to the BusMatrix, which manages the access of CPU and DMA to SRAM and peripherals.

BusMatrix

The BusMatrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm.

Internal flash can only be accessed by the AHB-Lite bus.

AHB peripherals are connected to a BusMatrix through AHB-Lite to allow DMA access.

AHB2APB bridge (APB)

The AHB2APB bridge provides full synchronous connections between the AHB and the APB buses(AHB_APB unit in Figure 2-1). APB is limited to ½ frequency of AHB frequency.

DP-AP

DP-AP is the debug access port, which is composed of DP (debug port) and AP (access port).

The DP interface module (AC7801x only supports SW-DP) is to convert external signals into 32-bit debug bus signals.

The AP interface module is equivalent to a bus bridge, which is used to convert debug bus commands into data on the AHB bus and then transmit them.

Through the cooperation of DP and AP, SWD can access the address of AC7801x.

2.2 Functional description

2.2.1 Memory organization

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

For example, the storage method of the 16-bit wide number(0x1234) in the little-endian format CPU memory (assuming the storage start address is 0x4000) is:

Table 2-1 Memory organization in Little Endian format

Memory address	0x4000	0x4001
Data	0x34	0x12

For the detailed mapping of peripheral registers, please refer to the related chapters about peripheral. The addressable memory space is divided into 8 main blocks, each of 512 Mbyte.

2.2.2 Internal SRAM

There is a 20 Kbytes of static SRAM. It can be accessed as bytes, half- words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

2.2.3 Fast IO memory map

The Fast IO bridge provides a channel to access CRC and GPIO through AHB bus with more efficiency, each has 4Kbyte address range.

Fast GPIO address range: 0x2008 0000 to 0x2008 0FFF.

CRC address range: 0x2008 1000 to 0x2008 1FFF.

2.2.4 Memory map

Table 2-2 is the AC7801x high-level device memory map, which includes three different memory map tables based on different boot up configuration, such as flash memory boot up, ISP boot up and SRAM boot up.

All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved".

Table 2-2 High-level device memory map

0xE010 0000	Reserved
0xE00F FFFF …… 0xE000 0000	Cortex™-M0+'s internal peripherals
0xDFFF FFFF …… 0x6000 0000	Reserved
0x5FFF FFFF …… 0x4010 0000	Reserved
0x400F FFFF …… 0x4000 0000	Peripheral APB Address
0x3FFF FFFF …… 0x2008 2000	Reserved
0x2008 1FFF …… 0x2008 0000	AHB Fast IO
0x2007 FFFF 0x2001 0000	Reserved
0x2000 FFFF …… 0x2000 0000	AHB SRAM
0x1FFF FFFF …… 0x0804 3000	Reserved
0x0804 27FF …… 0x0804 2000	Reserved
0x0804 1FFF …… 0x0804 0800	ISP Firmware
0x0804 002F …… 0x0804 0000	option byte
0x0801 FFFF …… 0x0800 0000	Flash memory
0x07FF FFFF ……	Reserved

0xE010 0000	Reserved
0xE00F FFFF …… 0xE000 0000	Cortex™-M0+'s internal peripherals
0xDFFF FFFF …… 0x6000 0000	Reserved
0x5FFF FFFF …… 0x4010 0000	Reserved
0x400F FFFF …… 0x4000 0000	Peripheral APB Address
0x3FFF FFFF …… 0x2008 2000	Reserved
0x2008 1FFF …… 0x2008 0000	AHB Fast IO
0x2007 FFFF …… 0x2001 0000	Reserved
0x2000 FFFF …… 0x2000 0000	AHB SRAM
0x1FFF FFFF …… 0x0804 3000	Reserved
0x0804 27FF …… 0x0804 2000	Reserved
0x0804 1FFF …… 0x0804 0800	ISP Firmware
0x0804 002F …… 0x0804 0000	option byte
0x0801 FFFF …… 0x0800 0000	Flash memory
0x07FF FFFF ……	Reserved

0xE010 0000	Reserved
0xE00F FFFF …… 0xE000 0000	Cortex™-M0+'s internal peripherals
0xDFFF FFFF …… 0x6000 0000	Reserved
0x5FFF FFFF …… 0x4010 0000	Reserved
0x400F FFFF …… 0x4000 0000	Peripheral APB Address
0x3FFF FFFF …… 0x2008 2000	Reserved
0x2008 1FFF …… 0x2008 0000	AHB Fast IO
0x2007 FFFF …… 0x2001 0000	Reserved
0x2000 FFFF …… 0x2000 0000	AHB SRAM
0x1FFF FFFF …… 0x0804 3000	Reserved
0x0804 27FF …… 0x0804 2000	Reserved
0x0804 1FFF …… 0x0804 0800	ISP Firmware
0x0804 002F …… 0x0804 0000	option byte
0x0801 FFFF …… 0x0800 0000	Flash memory
0x07FF FFFF ……	Reserved

0x0004 0000	
0x0003 FFFF	Flash memory
……	
0x0000 0000	
Flash memory boot up	

0x0000 1800	
0x0000 17FF	ISP Firmware
……	
0x0000 0000	
ISP boot up	

0x0001 0000	
0x0000 FFFF	AHB SRAM
……	
0x0000 0000	
SRAM boot up	

2.2.5 Internal flash memory

The high-performance Flash memory module has the following key features:

- Density up to 128 Kbytes.
- Memory organization: the flash memory is organized as a main block and an information block.
  - Size of main memory block: up to 32 K × 32 bits divided into 64 pages of 2 Kbytes each.
  - Size of information block: 48byte.

The Flash memory controller features:

- Flash Program / Erase operation.
- Read / Write protection.
- Erase and blank check.
- Cache controller to improve read efficiency, the maximum efficiency is zero wait.

2.2.6 Read internal flash memory

Flash memory instructions and data access are performed through the AHB bus.

2.2.7 Chip Model

The model Type information can be accessed by users through the eFlash read operation interface. The chip model information is stored in the continuous space (1*32Bit) of 0x40002028~0x4000202B. Among them, the upper 8 bits are fixed 0XFF, and the lower 24 bits are chip model information.

2.2.8 Chip UUID

The UUID has a total of 128Bit, which is generated by a random number and can be used as the unique identification mark of the chip. It can be accessed through the eFlash read operation interface. The UUID information is stored in a continuous space (4*32Bit) from 0x4000202C to 0x40002038.

2.2.9 AHB to APB bridge

The AHB to APB bridge translates the AHB protocol to the APB protocol and get low frequency interface. Most peripherals are APB interface. For the detail address assignment of each peripheral, refer to Table 2-5.

2.2.10 Nested Vectored Interrupt Controller (NVIC)

Features

- 32 maskable interrupt channels (not including the 16 interrupt lines of Cortex™-M0+).
- 4 programmable priority levels (2 bits of interrupt priority are used).
- Low-latency exception and interrupt handling.
- Power management control.
- Implementation of System Control Registers.

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming see Chapter 5 Exceptions & Chapter 8 Nested Vectored Interrupt Controller of ARM Cortex™-M3 Technical Reference Manual.

Below is the AC7801x interrupt table:

Table 2-3 AC7801x interrupt table

Interrupt vector number	Priority	Type of priority	Acronym	Description
-15	-3	fixed	Reset	System reset
-14	-2	fixed	NMI	Non-markable interrupt
-13	-1	fixed	Hard Fault	All class of fault
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
-5	0	settable	SVCall	Exception caused by executing the System Service Call Instruction(SVC) via SWI instruction
			Reserved	Reserved
			Reserved	Reserved

Interrupt vector number	Priority	Type of priority	Acronym	Description
-2	0	settable	Pend SV	Pendable request for system service
-1	0	settable	SysTick	System tick timer
0	0	settable	PWDT0	PWDT0 interrupt
1	0	settable	PWDT1	PWDT1 interrupt
2	0	settable	PWM0	PWM 0 interrupt
3	0	settable	PWM1	PWM 1 interrupt
4	0	settable	ACMP0	ACMP0 interrupt
5	0	settable	UART0	UART0 interrupt
6	0	settable	UART1	UART1 interrupt
7	0	settable	UART2	UART2 interrupt
8	0	settable	WDG	Watch dog timer interrupt
9	0	settable	SPI0	SPI0 global interrupt
10	0	settable	SPI1	SPI1 global interrupt
11	0	settable	I2C0	I2C0 interrupt
12	0	settable	I2C1	I2C1 interrupt
13	0	settable	DMA0_CHANNEL0	DMA channel 0 global interrupt
14	0	settable	DMA0_CHANNEL1	DMA channel 1 global interrupt
15	0	settable	DMA0_CHANNEL2	DMA channel 2 global interrupt
16	0	settable	DMA0_CHANNEL3	DMA channel 3 global interrupt
17	0	settable	TIMER_CHANNEL0	Timer 0 interrupt
18	0	settable	TIMER_CHANNEL1	Timer 1 interrupt
19	0	settable	TIMER_CHANNEL2	Timer 2 interrupt
20	0	settable	TIMER_CHANNEL3	Timer 3 interrupt
21	0	settable	RTC	RTC interrupt
22	0	settable	PVD	Power detect interrupt
23	0	settable	SPM	System power manager interrupt
24	0	settable	CAN0	CAN 0 interrupt
25	0	settable	ADC0	ADC0 interrupt
26	0	settable	ECC_SRAM	ECC SRAM error detection interrupt
27	0	settable	EXTI0	EXTI 0 interrupt
28	0	settable	EXTI1	EXTI 1 interrupt
29	0	settable	EXTI2	EXTI 2 interrupt
30	0	settable	EXTI3_8	EXTI 3_8 interrupt
31	0	settable	EXTI9_15	EXTI 9_15 interrupt

2.2.11 Boot configuration

Three different boot modes can be selected through BOOT(PA6), PA1 and PA0 pins as shown in Table 2-4 .

Table 2-4 Boot configuration

PIN name	BOOT(PA6)	PA1	PA0
eflash boot	0	x	x
ISP boot	1	0	0
SRAM boot	1	1	0

Note: x means do not care, the ISP download pin is UART0: PA7 (TX) and PA8 (Rx).

The values on the boot up configuration pins are latched on the 8th rising edge of 8MHz clock after a Reset. It is up to the user to set these pins to select the required boot mode, user should keep these pins stable before latched.

Due to its fixed memory map, the code area starts from the address 0x0000 0000. The Cortex™-M0+ CPU always fetches the reset vector on the AHB-Lite bus, which implies to have the boot space available only in the code area (typically, Flash memory). This AC7801x microcontrollers implement a special mechanism to be able to boot not only from the main Flash memory and ISP firmware, but also from SRAM.

Depending on the selected boot mode, the main Flash memory, ISP, SRAM are accessible as follows:

- **Boot from main Flash memory:** the main Flash memory is aliased in the boot memory space (0x0000 0000), but still is accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
- **Boot from ISP:** the ISP is aliased in the boot memory space (0x0000 0000), but still can be accessible from its original memory space (0x0804 0800) .
- **Boot from the embedded SRAM:** the SRAM is aliased in the boot memory space (0x0000 0000), but still can be accessible from its original memory space (0x2000 0000).

2.3 Address assignment of Peripherals

Table 2-5 Address assignment of each peripheral

APB Memory Map	Base_Address	Size(byte)
CKGEN	0x40000000	4K
GPIO	0x40001000 / 0x20080000	4K
Embedded Flash Controller	0x40002000	4K
ADC0	0x40003000	4K
ACMP0	0x40005000	4K
CAN	0x40007800	1K
SPM	0x40008000	1K
RTC	0x40008400	1K
WDG	0x4000b000	4K
SPI_0	0x4000c000	4K
SPI_1	0x4000d000	4K
IIC_0	0x4000e000	4K
IIC_1	0x4000f000	4K
Cortex™-M0+ controller	0x40010000	2K
TIMER	0x40011000	4K
DMA	0x40012000	4K
PWM0	0x40013000	4K
PWM1	0x40014000	4K
CTU	0x40016000	4K
PWDT0	0x40017000	2K
PWDT1	0x40017800	2K
UART_0	0x40018000	4K
UART_1	0x40019000	4K
UART_2	0x4001a000	4K
CRC	0x20081000	4K

3 Reset

3.1 Features

The following reset sources are supported in this MCU.

POR reset:

- POR_rst: IC power-on reset.

System reset:

- External Reset: external pin reset from IC pad, low active.
- LVD Reset: low voltage detection reset, low active.
- Software Reset: Cortex™-M0+ software reset.
- ECC 2 Bit Error Reset: the ECC detects a 2-BIT error reset, default disabled.
- Lock up Reset: Cortex™-M0+ lock up reset.
- PLL Unlock Reset: PLL unlock reset, default disabled.
- Watchdog Reset Normal Mode: watch dog timer reset, active in normal mode.
- Watchdog Reset Stop Mode: watch dog timer reset, active in stop mode.
- XOSC Lost Reset: when an external XOSC clock is detected to be abnormal, a maskable system reset will be generated.

Each of the system reset sources has an associated bit in the status register 0x4000 0010.

3.2 Block diagram

The block diagram is described in Figure 3-1. Each Reset signal is high by default, and Low-level reset signal triggers the system reset.

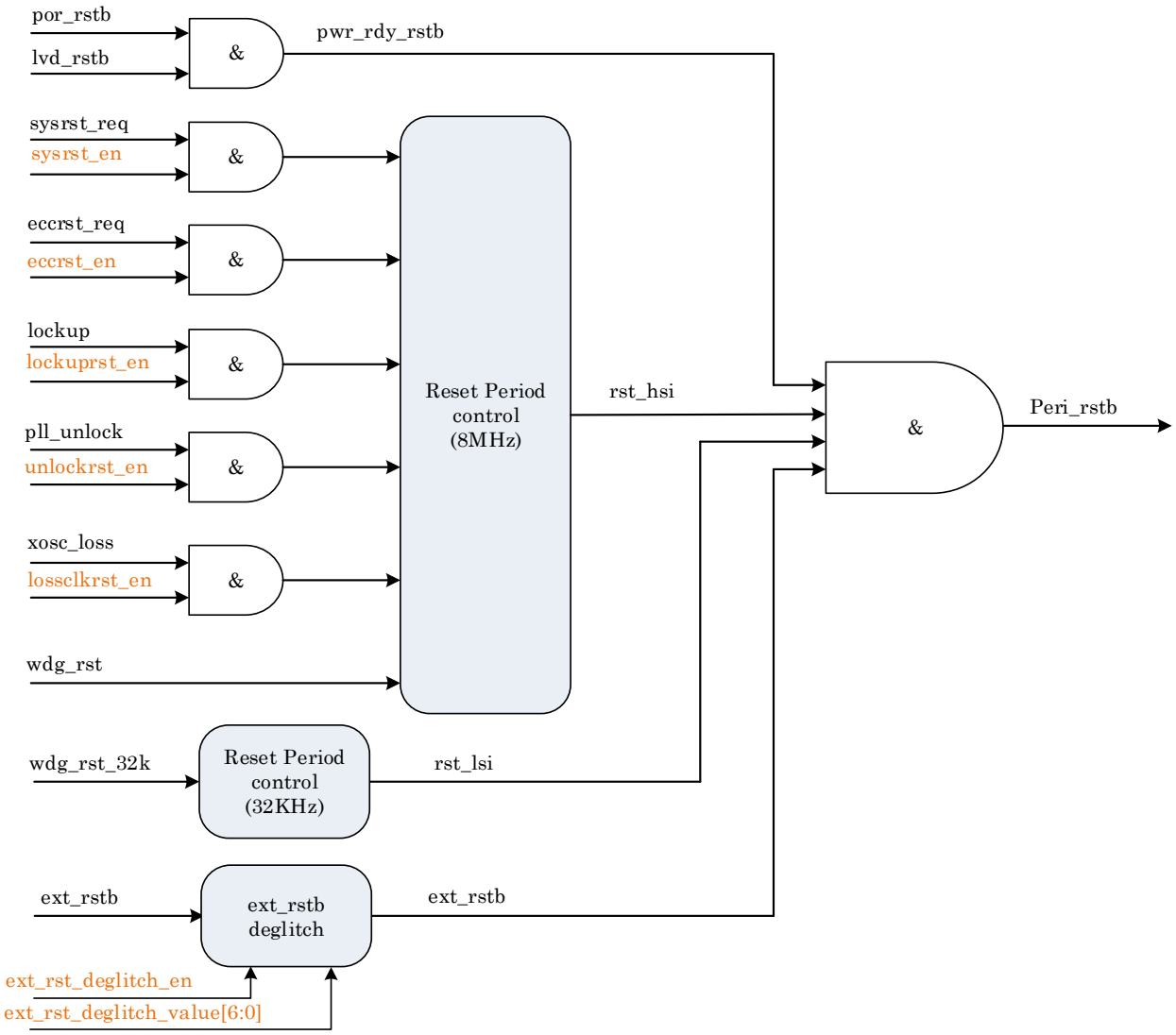

Figure 3-1 Reset block diagram

3.3 Function description

3.3.1 Power On Reset (POR)

When power is initially applied to the MCU or when the supply voltage drops below the power-on reset voltage level (VPOR), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold (V_{LVDL}), please refer to Table 6-2 LVD / POR / AVDD voltage reset specifications of ATC_AC7801x_Datasheet_EN.

3.3.2 System reset

System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it performs the following:

- Read the initial value of SP (SP_main) from vector-table offset 0.
- Read the initial value of program counter (PC) from vector-table offset 4.
- The Link Register (LR) is set to 0xFFFF_FFFF.

3.3.2.1 External pin reset

There is a dedicated pin in the MCU, and it is used to reset the whole MCU function and restart.

As it is low active, suggest to add pull up function in external PCB to prevent noise.

3.3.2.2 Low voltage detection

This device includes a system to protect against low-voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. This system consists of a power-on reset (POR) circuit, and a LVD circuit with a user selectable trip voltage, either high (V_{LVDH}) or low (V_{LVDL}).

For details, please refer to 5.1.1 DC characteristics of ATC_AC7801x_Datasheet_EN.

3.3.2.3 ECC 2 bits' error reset

Setting RESET_CTRL[23] to 1 will enable ECC 2 bits' Error Reset. When ECC detects a 2 bits' error, it will issue a system reset request to generate a system reset.

3.3.2.4 PLL unlock reset

Setting RESET_CTRL[22] to 1 will enable PLL Unlock Reset. When the PLL detects an unlock error, it will issue a system reset request to generate a system reset.

3.3.2.5 Watchdog timer reset

The watchdog timer (WDG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing (or refreshing) the watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset.

Please refer to 20 WDG for detail.

3.3.2.6 XOSC monitor

XOSC monitor system can be activated by setting CKGEN_SRC_SEL[16] to 1. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped. If a failure is detected on the HSE oscillator clock, this oscillator is automatically disabled, XOSC loss status flag is active and NMI interrupt is generated to inform the software about the failure allowing the MCU to perform rescue operations.

3.3.2.7 Lock up reset

The lock_up gives immediate indication of seriously errant from kernel software. This is the result of the core being locked because of an unrecoverable exception following the activation of the processor's built in system state protection hardware.

When a lock up occurs, a reset can be automatically generated to restore the system.

3.4 Register Definition

Table 3-1 Reset register map

Address	Name	Width (in bit)	Description
0x4000000C	RESET_CTRL	32	chip reset control
0x40000010	RESET_STATUS	32	chip reset status

3.4.1 RESET_CTRL

Table 3-2 RESET_CTRL register

RESET_CTRL **chip reset control** **Reset:0x01418004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name						XOSC_LOSS_RST_EN	CPU_LOCKUP_RST_EN	CPU_SYSRST_EN	ECC2_RST_EN	PLL_UNLOCK_RST_EN						
Type						RW	RW	RW	RW	RW						
Reset						0	0	1	0	1						
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									EXT_RST_DEGLITCH_VALUE							EXT_RST_DEGLITCH_EN
Type									RW							RW
Reset									0	0	0	0	0	1	0	0

Bit (s)	Description
26 XOSC_LOSS_RST_EN	**XOSC loss reset enable** 1 : can generate reset for IC 0 : cannot generate reset for IC
25 CPU_LOCKUP_RST_EN	**CPU lock up reset enable** 1 : can generate reset for IC 0 : cannot generate reset for IC
24 CPU_SYSRST_EN	**CPU system reset enable** 1 : can generate reset for IC 0 : cannot generate reset for IC
23 ECC2_RST_EN	**ECC2 reset enable** 1 : can generate reset for IC 0 : cannot generate reset for IC
22 PLL_UNLOCK_RST_EN	**PLL unlock reset enable** 1 : can generate reset for IC 0 : cannot generate reset for IC
7:1 EXT_RST_DEGLITCH_VALUE	**External reset deglitch value** With 32kHz as the period of the counting clock source
0 EXT_RST_DEGLITCH_EN	**External reset deglitch enable** 1 : enable 0 : disable

3.4.2 RESET_STATUS

Table 3-3 RESET_STATUS register

RESET_STATUS **chip reset status** **Reset:0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																CLEAR_RESET_STATUS
Type																RW
Reset																0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							XOSC LOSS STATUS	PLL_UNLOCK_RST_STATUS	CPU_LOCKUP_RST_STATUS	CPU_SYSRESET_STATUS	WDG_RESET_STATUS	WDG_32K_RESET_STATUS	ECC_RESET_STATUS	EXT_RESET_STATUS	LVD_RESET_STATUS	POR_RESET_STATUS
Type							R	R	R	R	R	R	R	R	R	R
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Description
16 CLEAR_ RESET_STATUS	**Clear reset status** 1 : clear all reset status 0 : allow reset status to update
9 XOSC LOSS STATUS	**XOSC LOSS status** 1 : valid 0 : invalid
8 PLL_UNLOCK_RST_STATUS	**PLL unlock reset status** 1 : valid 0 : invalid
7 CPU_LOCKUP_ RST_STATUS	**CPU lock up reset status** 1 : valid 0 : invalid
6 CPU_SYSRESET_STATUS	**CPU system reset status** 1 : valid 0 : invalid
5 WDG_RESET_STATUS	**Watchdog normal reset status** 1 : valid 0 : invalid

Bit(s)	Description
4 WDG_32K_RESET_STATUS	**Watchdog reset status in low power mode** 1 : valid 0 : invalid
3 ECC_RESET_STATUS	**ECC 2bit ERROR reset status** 1 : valid 0 : invalid
2 EXT_RESET_STATUS	**External pin reset status** 1 : valid 0 : invalid
1 LVD_RESET_STATUS	**LVD reset status** 1 : valid 0 : invalid
0 POR_RESET_STATUS	**POR reset status** 1 : valid 0 : invalid

4 Clock

4.1 Introduction

The clock control module provides clock source choices for the MCU. The module contains a phase-locked loop (PLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this PLL clock or either of the internal or external reference clocks as a source for the MCU system clock.

There are configuration control signals to generate all modules' clock source and frequency.

4.2 Block diagram

4.2.1 Clock control diagram

This device contains the following on-chip clock sources:

- High speed internal RC(HSI): the internal RC OSC to provide 8MHz clock source.
- High speed external RC(HSE): the external OSC to provide 4MHZ ~30MHz crystal and oscillator.
- Low speed internal RC(LSI): the internal low speed RC OSC to provide 32kHz clock source.
- Phase-lock loop (SYSPLL): the PLL provides high speed clock up to 48MHz.

Each peripheral has dedicated clock enable signal to control clock on/off, please refer to register control (4.3) to know the detail address.

Note:

- System clock up to 48MHz, some chip models up to 72MHz (please refer to AutoChips MCUs Selection Guide)
- HCLK (AHB) up to 48MHz, some chip models up to 72MHz (please refer to AutoChips MCUs Selection Guide)
- PCLK (APB) up to 48MHz. when HCLK is 48MHz, APBCLK_DIV can be configured as 1; When HCLK is 72MHz, APBCLK_DIV minimum value is 2.

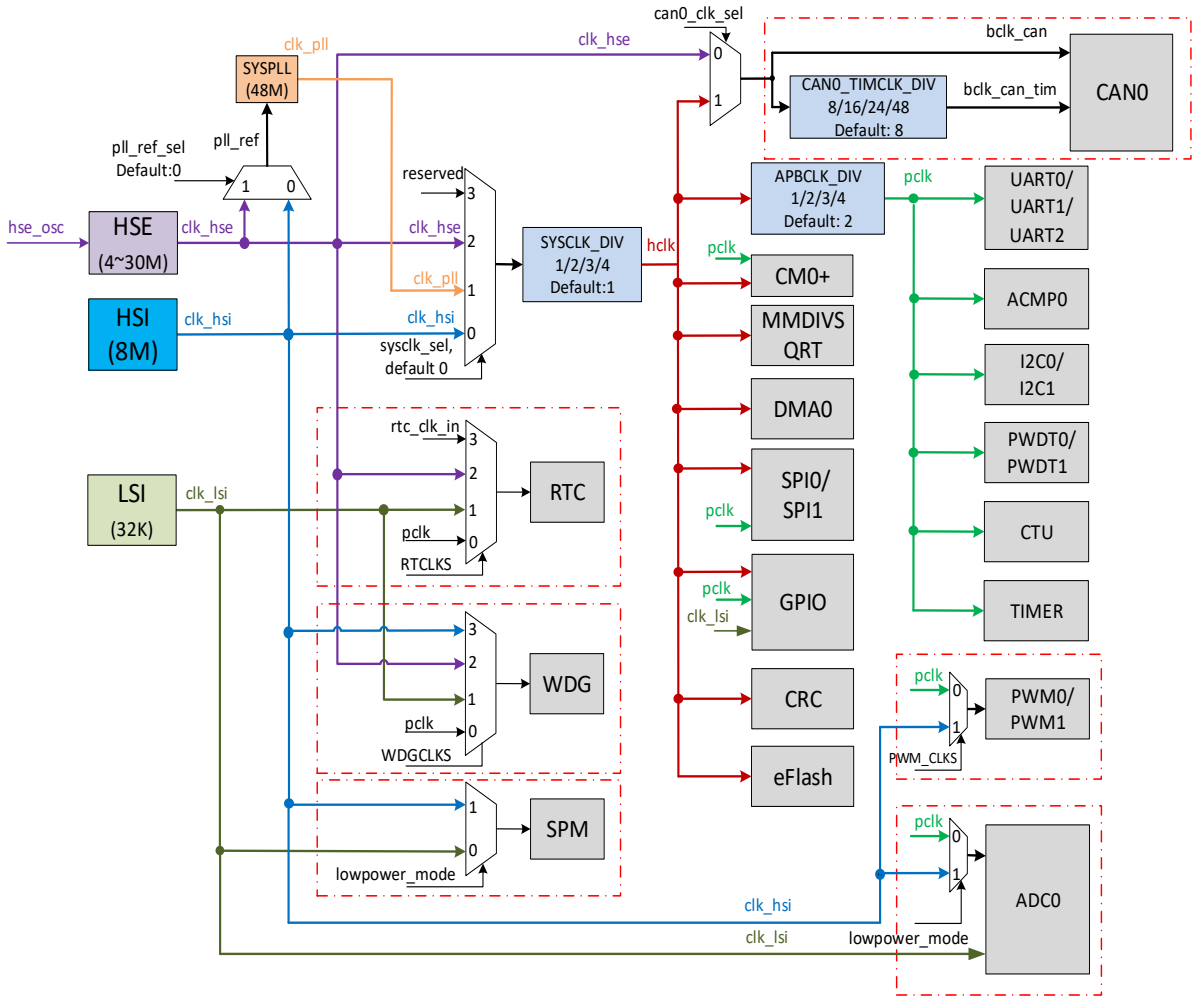

Figure 4-1 Clock control diagram

4.2.2 System clock diagram

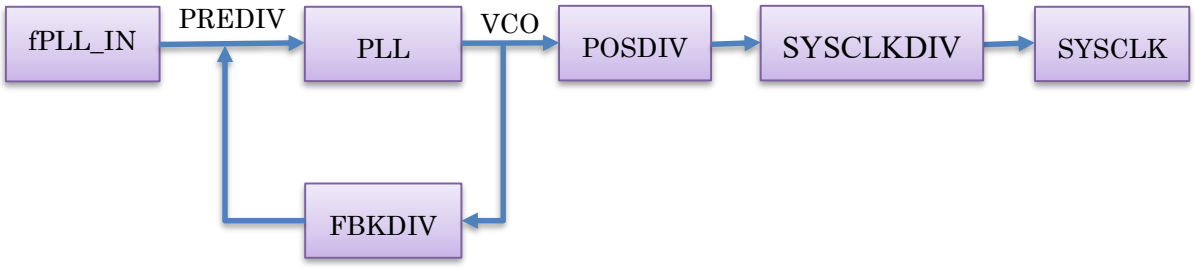

Figure 4-2 System clock diagram

fPLL_IN: Frequency Input, supporting 4 MHz to 30 MHz

VCO = FIN * FBKDIV / PREDIV

System Clock = VCO / POSDIV/SYSCLKDIV.

Note:

1. The frequency of VCO divided by POSDIV cannot be greater than 400 MHz, and the frequency range of VCO is 0.5 GHz to 1.5 GHz.
2. The FBKDIV recommendation is 48, 64, 96, 192.
3. The external crystal can be 4M~30M, and the recommendation input frequency of PLL is less than 8M.
4. PREDIV, FBKDIV, and POSDIV in the table are frequency/octavo values, not register values.

Table 4-1 Typical PLL configuration reference

fPLL_IN	PREDIV	FBKDIV	POSDIV	Frequency	Sysclk_div	System Clock
8	1	108	6	144	2	72
8	1	96	16	48	1	48
8	1	96	24	32	1	32
8	1	96	16	48	2	24
8	1	96	16	48	3	16
8	1	96	16	48	4	12
4	1	216	12	72	1	72
4	1	192	16	48	1	48
4	1	192	24	32	1	32
4	1	192	16	48	2	24
4	1	192	16	48	3	16
4	1	192	16	48	4	12
30	2	48	10	72	1	72
30	2	64	20	48	1	48
30	2	64	30	32	1	32
30	2	64	20	48	2	24
30	2	64	20	48	3	16
30	2	64	20	48	4	12
12	2	144	12	72	1	72
12	2	96	12	48	1	48
12	2	96	18	32	1	32
12	2	96	24	24	1	24
12	2	96	18	32	2	16
12	2	96	24	24	2	12
16	2	144	16	72	1	72
16	2	96	16	48	1	48
16	2	96	24	32	1	32
16	2	96	16	48	2	24
16	2	96	24	32	2	16
16	2	96	16	48	4	12

Note

In the above table, the red part indicates the default software configuration.

4.3 Register Definition

Table 4-2 Clock register mapping

Address	Name	Width (in bit)	Description
0x40000000	CKGEN_CTRL	32	clock control
0x40000004	CKGEN_PERI_CLK_EN_0	32	peripheral clock enable control 0
0x40000008	CKGEN_PERI_CLK_EN_1	32	peripheral clock enable control 1
0x40000018	CKGEN_PERI_SFT_RST0	32	peripheral software reset control 0
0x4000001C	CKGEN_PERI_SFT_RST1	32	peripheral software reset control 1
0x40008890	CKGEN_SYSPLL1_CFG0	32	SYSPLL1 configuration register 0
0x40008894	CKGEN_SYSPLL1_CFG1	32	SYSPLL1 configuration register 1

4.3.1 CKGEN_CTRL

Table 4-3 CKGEN_CTRL register

CKGEN_CTRL **Clock generator ctrl** **Reset: 0x00000100**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name						CAN0_CLK_SEL				CAN0_TIMCLK_DIV		PLL_REF_SEL				XOSC_MON_EN
Type						RW				RW		RW				RW
Reset						0				0	0	0				0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							APBCLK_DIV				SYSCLK_DIV				SYSCLK_SEL	
Type							RW				RW				RW	
Reset							0	1			0	0			0	0

Bits	Description
26 CAN0_CLK_SEL	**CAN0 clock source select** 0 : external oscillator clock 1: AHB divided clock

Bits	Description
22:21 CAN0_TIMCLK_DIV	**CAN0 time stamp clock divided by the CAN clock source** 00 : divider by 8 01 : divider by 16 10 : divider by 24 11 : divider by 48
20 PLL_REF_SEL	**PLL reference clock select** 1 : reference clock is external oscillator 0 : reference clock is internal oscillator
16 XOSC_MON_EN	**XOSC monitor enable** 1 : monitor enable 0 : monitor disable
9:8 APBCLK_DIV	**APB clock divider by system clock** 00 : divider by 1 01 : divider by 2 10 : divider by 3 11 : divider by 4
5:4 SYSCLK_DIV	**System clock divider** 00 : divider by 1 01 : divider by 2 10 : divider by 3 11 : divider by 4
1:0 SYSCLK_SEL	**System clock source select** 00 : internal oscillator 01 : PLL output 10 : external oscillator 11 : reserved

4.3.2 CKGEN_PERI_CLK_EN_0

Table 4-4 CKGEN_PERI_CLK_EN_0 register

CKGEN_PERI_CLK_EN_0 peripheral clock enable control 0 Reset: 0x02800001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name				CAN0_EN		CRC_EN	WDG_EN		GPIO_EN		DMA0_EN	RTC_EN	TIMER_EN			
Type				RW		RW	RW		RW		RW	RW	RW			
Reset				0		0	1		1		0	0	0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				PWM1_EN	PWM0_EN	PWDT0_EN	I2C1_EN	I2C0_EN	SPI1_EN	SPI0_EN				UART2_EN	UART1_EN	UART0_EN
Type				RW	RW	RW	RW	RW	RW	RW				RW	RW	RW
Reset				0	0	0	0	0	0	0				0	0	1

Note

Before writing to the peripheral register, the clock of the corresponding peripheral needs to be enabled, otherwise Hardfault will occur.

Bits	Descriptions
28 CAN0_EN	**CAN0 clock enable** 1 : clock enable 0 : clock disable
26 CRC_EN	**CRC clock enable** 1 : clock enable 0 : clock disable
25 WDG_EN	**WDG APB clock enable** 1 : clock enable 0 : clock disable
23 GPIO_EN	**GPIO AHB clock enable** 1 : clock enable 0 : clock disable
21 DMA0_EN	**DMA0 clock enable** 1 : clock enable 0 : clock disable

Bits	Descriptions
20 RTC_EN	**RTC clock enable** 1 : clock enable 0 : clock disable
19 TIMER_EN	**TIMER clock enable** 1 : clock enable 0 : clock disable
12 PWM1_EN	**PWM1 timer clock enable** 1 : clock enable 0 : clock disable
11 PWM0_EN	**PWM0 timer clock enable** 1 : clock enable 0 : clock disable
10 PWDT0_EN	**PWDT0 clock enable** 1 : clock enable 0 : clock disable
9 I2C1_EN	**IIC1 clock enable** 1 : clock enable 0 : clock disable
8 I2C0_EN	**IIC0 clock enable** 1 : clock enable 0 : clock disable
7 SPI1_EN	**SPI1 clock enable** 1 : clock enable 0 : clock disable
6 SPI0_EN	**SPI10 clock enable** 1 : clock enable 0 : clock disable
2 UART2_EN	**UART2 clock enable** 1 : clock enable 0 : clock disable
1 UART1_EN	**UART1 clock enable** 1 : clock enable 0 : clock disable
0 UART0_EN	**UART0 clock enable**

Bits	Descriptions
	1 : clock enable 0 : clock disable

4.3.3 CKGEN_PERI_CLK_EN_1

Table 4-5 CKGEN_PERI_CLK_EN_1 register

CKGEN_PERI_CLK_EN_1 peripheral clock enable control 1 Reset:0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											PWDT1_EN		ACMP0_EN	ADC0_EN	CTU_EN	
Type											RW		RW	RW	RW	
Reset											0		0	0	0	

Note

Before writing to the peripheral register, the clock of the corresponding peripheral needs to be enabled, otherwise Hardfault will occur.

Bits	Description
5 PWDT1_EN	**PWDT1 clock enable** 1 : clock enable 0 : clock disable
3 ACMP0_EN	**ACMP0 clock enable** 1 : clock enable 0 : clock disable
2 ADC0_EN	**ADC0 core clock enable** 1 : clock enable 0 : clock disable
1 CTU_EN	**CTU APB clock enable** 1 : clock enable 0 : clock disable

4.3.4 CKGEN_PERI_SFT_RST0

Table 4-6 CKGEN_PERI_SFT_RST0 register

CKGEN_PERI_SFT_RST0 peripheral software reset control 0 Reset:0x02900001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name				SRST_CAN0		SRST_CRC	SRST_WDG		SRST_GPIO		SRST_DMA0	SRST_RTC	SRST_TIMER			
Type				RW		RW	RW		RW		RW	RW	RW			
Reset				0		1	1		1		0	1	0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				SRST_PWM1	SRST_PWM0	SRST_PWDT0	SRST_I2C1	SRST_I2C0	SRST_SPI1	SRST_SPI0				SRST_UART2	SRST_UART1	SRST_UART0
Type				RW	RW	RW	RW	RW	RW	RW				RW	RW	RW
Reset				0	0	0	0	0	0	0				0	0	1

Bits	Description
28 SRST_CAN0	**CAN0 software reset** 0 : reset active 1 : reset inactive
26 SRST_CRC	**CRC software reset** 0 : reset active 1 : reset inactive
25 SRST_WDG	**Watch dog timer software reset** 0 : reset active 1 : reset inactive
23 SRST_GPIO	**GPIO AHB software reset** 0 : reset active 1 : reset inactive
21 SRST_DMA0	**DMA0 software reset** 0 : reset active 1 : reset inactive
20 SRST_RTC	**RTC software reset** 0 : reset active 1 : reset inactive

Bits	Description
19 SRST_TIMER	**TIMER software reset** 0 : reset active 1 : reset inactive
12 SRST_PWM1	**PWM1 software reset** 0 : reset active 1 : reset inactive
11 SRST_PWM0	**PWM0 software reset** 0 : reset active 1 : reset inactive
10 SRST_PWDT0	**PWDT0 software reset** 0 : reset active 1 : reset inactive
9 SRST_I2C1	**IIC1 software reset** 0 : reset active 1 : reset inactive
8 SRST_I2C0	**IIC0 software reset** 0 : reset active 1 : reset inactive
7 SRST_SPI1	**SPI1 software reset** 0 : reset active 1 : reset inactive
6 SRST_SPI0	**SPI0 software reset** 0 : reset active 1 : reset inactive
2 SRST_UART2	**UART2 software reset** 0 : reset active 1 : reset inactive
1 SRST_UART1	**UART1 software reset** 0 : reset active 1 : reset inactive
0 SRST_UART0	**UART0 software reset** 0 : reset active 1 : reset inactive

4.3.5 CKGEN_PERI_SFT_RST1

Table 4-7 CKGEN_PERI_SFT_RST1 register

CKGEN_PERI_SFT_RST1 peripheral software reset control 1 Reset:0x00000010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											SRST_PWDT1	SRST_ANA_REG	SRST_ACMP0	SRST_ADC0	SRST_CTU	
Type											RW	RW	RW	RW	RW	
Reset											0	1	0	0	0	

Bits	Description
5 SRST_PWDT1	**PWDT1 software reset** 0 : reset active 1 : reset inactive
4 SRST_ANA_REG	**ANA register software reset** 0 : reset active 1 : reset inactive
3 SRST_ACMP0	**ACMP0 software reset** 0 : reset active 1 : reset inactive
2 SRST_ADC0	**ADC0 software reset** 0 : reset active 1 : reset inactive
1 SRST_CTU	**CTU software reset** 0 : reset active 1 : reset inactive

4.3.6 CKGEN_SYSPLL1_CFG0

Table 4-8 CKGEN_SYSPLL1_CFG0 register

CKGEN_SYSPLL1_CFG0 **SYSPLL1 configuration register 0** **Reset: 0x10301030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SYSPLL1_PREDIV		SYSPLL1_POSDIV							SYSPLL1_FBKDIV						
Type	RW		RW							RW						
Reset	0	0	0	1	0	0	0			0	1	1	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							SYSPLL1_MONREF_EN									
Type							RW									
Reset	0						0									

Bits	Description
31:30 SYSPLL1_PREDIV	**Pre-divider ratio** 00: Fref = Fin/1 01: Fref = Fin/2 1X: Fref = Fin/4
29:25 SYSPLL1_POSDIV	**Frequency division ratio of single-end clock output** 00000: none 00001: VCO/2 00010: VCO/4 00011: VCO/6 … 11110: VCO/60 11111: VCO/62
22:15 SYSPLL1_FBKDIV	**Feedback frequency division ratio** 8'd6: /6 8'd255: /255
9 SYSPLL1_MONREF_EN	**PLL monitor reference clock enable** 0 : disable 1 : enable

4.3.7 CKGEN_SYSPLL1_CFG1

Table 4-9 CKGEN_SYSPLL1_CFG1 register

CKGEN_SYSPLL1_CFG1 SYSPLL1 configuration register 1 Reset:0x00F000EE

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								SYSPLL1_LD_EN					SYSPLL1_LD_DLY_SEL			
Type								RW					RW			
Reset								0					1	1	1	

Bits	Description
8 SYSPLL1_LD_EN	**PLL Lockup Detector enable** 0: disable 1: enable
3:1 SYSPLL1_LD_DLY_SEL	**Lockup Detector Comparison time selection** 000: 300ps 001: 600ps 010: 900ps 011: 1200ps 100: 2200ps 101: 3200ps 110: 4200ps 111: 5200ps

5 Power Modes

5.1 Introduction

This chapter describes the AC7801x chip power modes and functionality of each module in these modes.

5.2 Power modes

The device supports Run, Sleep, Stop, and Standby modes which are easy to use for customers both from different power consumption level and functional requirement. I/O states are held in Run, Sleep, Stop modes. In Standby mode, the I/O is off, and the pin state is determined by the external circuit. It is recommended to add pull up/pull down to determine its status in Standby mode.

- Run mode—CPU clocks can be run at full speed.
- Sleep mode—CPU enters into the sleep mode, system clocks and bus clock are running.
- Stop mode— CPU enters into the deep sleep mode and some modules can wake up CPU.
- Standby mode—CPU and individual modules are shut down, the RTC and NMI pin can wake up CPU.

5.3 Application notes

5.3.1 Entering and exiting power modes

1. Enable the wakeup source required using the **SPM_EN_PERIPH_WKUP.**
2. Set the power mode with the **SPM_PWR_MGR_CFG0[SLEEP_MODE]** before WFI, the configuration is set as follows:
   1) 2'b01: stop mode
   2) 2'b1x: standby mode
3. Call WFI to enter the low-power mode.
4. The processor exits the low-power mode via an interrupt.
5. Re-enable the module

5.3.2 Module operation in low-power modes

The following table illustrates the functionality of each module while the chip is in each of the low-power modes. The standard behavior is shown with some exceptions.

Table 5-1 Module functionality in low-power modes

Module	Sleep mode	Stop mode	Standby mode
CM0+	Standby	Standby	off
MMDIVSQRT	on	off	off
SRAM	on	Standby	off
Embedded Flash	on	off	off
I2C	on	Optional on [1]	off
SPI	on	Optional on [2]	off
WDG	on	Optional on	off
PWDT	on	off	off
UART	on	Standby[3]	off
DMA	on	off	off
TIMER	on	off	off
PWM	on	off	off
CRC	on	off	off
CTU	on	off	off
CAN	on	Standby[4]	off
RTC	on	Optional on	Optional on
SPM	on	Always on	Always on
PLL	on	off	off
XOSC	on	off	off
HSI(8 MHz)	on	Optional on [5]	off
LSI(32 kHz)	on	Always on	Always on
GPIO	on	Always on [6]	off[6]
ADC	on	Optional on [5]	off
LVD	Optional on	Optional on	Optional on
PVD	Optional on	Optional on [7]	off
ACMP	Optional on	Optional on [8]	off
DAC	Optional on	Optional on	off
T-sensor	Optional on	off	off
DIGLDO	on	Low power	Low power, CM0+ off
FLHLDO	on	off	off
POR	on	on	on
BG	on	Optional on [9]	Optional on

Note:

1. Supports address match wake-up in Stop mode.
2. Supports slave mode receive and wake-up in Stop mode.
3. Supports the edge wake-up in Stop mode (UART pin goes low directly to SPM).
4. Supports wake-up on edge in Stop mode, open the filter is optional.

5. Supports analog monitoring wake-up in Stop mode.
6. The I/O state is maintained in Stop mode, supports all GPIO interrupt wake-ups, and the standby mode only supports the NMI pin wakeup.
7. Supports PVD Warning interrupt wake-up in Stop mode.
8. Supports ACMP setting voltage comparison wake-up in Stop mode.
9. Enables ADC to wake up, or enables LVD, PVD, BG is turned on, otherwise turn off BG in stop mode.

Note:

- **on:** both Power and Clock of the module are provided normally.
- **Standby:** Power of the module is normal, and the Clock is turned off.
- **off:** both Power and Clock of the module are turned off.

6 System Power Management

6.1 Introduction

SPM (System Power Management) provides the max flexibility for software developers to develop the system housekeeping task. It aims for low level sleep/wakeup task such as MTCMOS power domain, SRAM and analog module power control.

6.2 Features

- Supports power management in Stop mode.
- Supports power management in Standby mode.

6.3 Application notes

6.3.1 SPM power control program guide

Stop mode and standby mode are supported in AC7801x. For the working status and the wake-up source of each module in Stop mode, please refer to Table 5-1. But in standby mode, all digital modules are power off except the module RTC and SPM.

The WFI instruction invokes stop and standby modes for the chip, and processor exits the low-power mode via an interrupt.

Program sequence:

1. Configure the wake-up source to work normally and generate the interrupt normally.
2. Set the wake-up source: SPM_EN_PERIPH_WKUP.
3. Enable the SPM power control: PWR_EN.
4. Program the SPM configuration registers to determine the power mode: SLEEP_MODE.
5. WFI.

6.3.2 XOSC/SYSPLL power control

XOSC/SYSPLL are off at default. When needed, XOSC/SYSPLL can be powered on by configuring SPM register bits: SPM_PWR_MGR_CFG1.

SPM register SPM_PWR_MGR_CFG1:

- **XOSC_HSEON**: external high-speed clock enables.
- **XOSC_HSEBYP**: external high-speed clock bypass.

- **SYSPLL_ON**: SYSPLL enable.

When the corresponding bit is set to 1'b1, SPM will power XOSC/PLL on following the power on sequence and it may take some time. So you should be waiting for XOSC/SYSPLL power on finish and clock ready before use it. XOSC/PLL power on states can be determined by reading SPM register : SPM_PWR_MGR_CFG1.

- **XOSC_RDY:** external high-speed clock ready flag.
- **SYSPLL_RDY**: SYSPLL clock ready flag.

For example, before the clk source switches to the PLL clk, you should power SYSPLL on first, and wait for SYSPLL clock stable.

When chip wakes up from stop mode, SPM will keep XOSC/SYSPLL on or off same as state before sleep. But when wakeup from the standby mode, the XOSC/SYSPLL will be off.

6.4 Register Definition

Table 6-1 SPM register mapping

SPM: 0x40008000

Address	Name	Width (in bit)	Description
SPM+ 0x00	SPM_PWR_MGR_CFG0	32	Power Manager Configuration 0 Register
SPM+ 0x04	SPM_PWR_MGR_CFG1	32	Power Manager Configuration 1 Register
SPM+ 0x0C	SPM_PERIPH_SLEEP_ACK_STATUS	32	Peripheral Sleep Ack Status
SPM+ 0x10	SPM_EN_PERIPH_SLEEP_ACK	32	Peripheral Sleep Ack Waiting Enable Register
SPM+ 0x14	SPM_EN_PERIPH_WKUP	32	Peripheral Wakeup Enable Register
SPM+ 0x1C	SPM_WAKEUP_IRQ_STATUS	32	SPM Wakeup IRQ Flags Status Register

6.4.1 SPM_PWR_MGR_CFG0

Table 6-2 SPM_PWR_MGR_CFG0 register

SPM_PWR_MGR_CFG0 Power Manager Configuration 0 Register Reset:0x00000118

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							SLEEP_MODE		EN_IO_SUS		EN_CAN0_FILTER	EN_LVD	EN_DPWRLVD	EN_PVD	EN_FAST_BOOT	PWR_EN
Type							RW		RW		RW	RW	RW	RW	RW	RW
Reset							0	1	0		0	1	1	0	0	0

Bits	Description
9:8 SLEEP_MODE	**sleep mode** 01: stop mode 1x: standby mode
7 EN_IO_SUS	**enable IO suspend in stop mode** 1: I/O suspend when enter stop mode this ctrl bit does not affect in standby mode, hardware auto suspends IO. 0: I/O hold when enter stop mode
5 EN_CAN0_FILTER	**enable CAN0 wakeup interrupt filter** 1: enable when enable SPM, it will use interrupt after analog filter as CAN0 wakeup int. 0: disable
4 EN_LVD	**chip lower voltage detect control bit** 1: enable detect chip VCC voltage, if VCC under voltage, trigger LVD reset. 0: disable
3 EN_DPWRLVD	**chip ldo voltage detect control bit** 1: enable detect the LDO voltage inside the chip. If the LDO is under voltage, the LVD reset will also be triggered. 0: disable

Bits	Description
2 EN_PVD	**enable program voltage detect control bit** 1:enable detect chip VCC voltage, if VCC under voltage, trigger PVD interrupt. 0:disable
1 EN_FAST_BOOT	**enable fast boot mode** 1:enable fast boot mode: to save wakeup time, chip will stop sleep sequence and wake up immediately when receive a wakeup interrupt. 0:disable
0 PWR_EN	**SPM power control enable** 1:enable SPM power control 0:disable

6.4.2 SPM_PWR_MGR_CFG1

Table 6-3 SPM_PWR_MGR_CFG1 register

SPM_PWR_MGR_CFG1 **Power Manager Configuration 0 Register** **Reset:0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	XOSC_RDY	SYSPLL_RDY	XOSC_HSEON	XOSC_HSEBYP	SYSPLL_ON											
Type	R/W	R/W	R/W	R/W	R/W											
Reset	0	0	0	0	0											
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													LVDPVD			
Type													R/W			
Reset													0	0	0	0

Bits	Description
31 XOSC_RDY	**XOSC clock ready flag** 1: ready 0: unready
30 SYSPLL_RDY	**PLL clock ready flag** 1: ready 0: unready
29 XOSC_HSEON	**External high-speed clock enable** 1: enable XOSC 0: disable XOSC

Bits	Description
28 XOSC_HSEBYP	**External high-speed clock bypass** 1: bypassing the oscillator with an external clock 0: disable bypassing the oscillator with an external clock
27 SYSPLL_ON	**SYSPLL enable** 1: enable SYSPLL 0: disable SYSPLL
3:0 LVDPVD	**LVDPVD setting** 0000: Reserved 0011: VLVDL = 2.60±0.1V VPVDL = 2.95±0.1V 1011: VLVDH = 4.2±0.15V VPVDH = 4.6±0.15V

6.4.3 SPM_PERIPH_SLEEP_ACK_STATUS

Table 6-4 SPM_PERIPH_SLEEP_ACK_STATUS register

SPM_PERIPH_SLEEP_ACK_STATUS Peripheral Sleep Ack Status Reset:0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name														EFLASH		ADC0
Type														R		R
Reset														0		0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMA0				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
Type	R				R	R	R		R		R	R	R	R		R
Reset	0				0	0	0		0		0	0	0	0		0

Bits	Description
18 EFLASH	**EFLASH idle status** 1: idle 0: busy
16 ADC0	**ADC0 sleep ack status** 1: ACK 0: not ACK
15 DMA0	**DMA0 sleep ack status** 1: ACK 0: not ACK
11 UART2	**UART2 sleep ack status** 1: ACK 0: not ACK

Bits	Description
10 UART1	**UART1 sleep ack status** 1: ACK 0: not ACK
9 UART0	**UART0 sleep ack status** 1: ACK 0: not ACK
7 CAN0	**CAN0 sleep ack status** 1: ACK 0: not ACK
5 SPI1	**SPI1 sleep ack status** 1: ACK 0: not ACK
4 SPI0	**SPI0 sleep ack status** 1: ACK 0: not ACK
3 I2C1	**I2C1 sleep ack status** 1: ACK 0: not ACK
2 I2C0	**I2C0 sleep ack status** 1: ACK 0: not ACK
0 ACMP0	**ACMP0 sleep ack status** 1: ACK 0: not ACK

6.4.4 SPM_EN_PERIPH_SLEEP_ACK

Table 6-5 SPM_EN_PERIPH_SLEEP_ACK register

SPM_EN_PERIPH_SLEEP_ACK Peripheral Sleep Ack Waiting Enable Reset:0x00058EBD

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name														EFLASH		ADC0
Type														R/W		R/W
Reset														1		1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMA0				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
Type	R/W				R/W	R/W	R/W		R/W		R/W	R/W	R/W	R/W		R/W
Reset	1				1	1	1		1		1	1	1	1		1

Bits	Description
18 EFLASH	**Enable eflash Sleep ACK Waiting** 1: enable 0: disable
16 ADC0	**Enable ADC0 Sleep ACK Waiting** 1: enable 0: disable
15 DMA0	**Enable DMA0 Sleep ACK Waiting** 1: enable 0: disable
11 UART2	**Enable UART2 Sleep ACK Waiting** 1: enable 0: disable
10 UART1	**Enable UART1 Sleep ACK Waiting** 1: enable 0: disable
9 UART0	**Enable UART0 Sleep ACK Waiting** 1: enable 0: disable
7 CAN0	**Enable CAN0 Sleep ACK Waiting** 1: enable 0: disable
5 SPI1	**Enable SPI1 Sleep ACK Waiting** 1: enable 0: disable
4 SPI0	**Enable SPI0 Sleep ACK Waiting** 1: enable 0: disable
3 I2C1	**Enable I2C1 Sleep ACK Waiting** 1: enable 0: disable
2 I2C0	**Enable I2C0 Sleep ACK Waiting** 1: enable 0: disable

Bits	Description
0 ACMP0	**Enable ACMP0 Sleep ACK Waiting** 1: enable 0: disable

6.4.5 SPM_EN_PERIPH_WKUP

Table 6-6 SPM_EN_PERIPH_WKUP register

SPM_EN_PERIPH_WKUP Peripheral Wakeup Enable Register **Reset:0x00028000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													PVD	NMI	GPIO	ADC0
Type													R/W	R/W	R/W	R/W
Reset													0	0	1	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTC				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
Type	R/W				R/W	R/W	R/W		R/W		R/W	R/W	R/W	R/W		R/W
Reset	1				0	0	0		0		0	0	0	0		0

Bits	Descriptions
19 PVD	**enable pvd warning wakeup** 1: enable When disable, PVD warning wakeup is not supported, and this wakeup int will be ignored. 0: disable
18 NMI	**enable NMI wakeup** 1: enable 0: disable
17 GPIO	**enable GPIO wakeup** 1: enable 0: disable
16 ADC0	**enable ADC0 wakeup** 1: enable 0: disable
15 RTC	**enable RTC wakeup** 1: enable 0: disable
11 UART2	**enable UART2 wakeup** 1: enable 0: disable

Bits	Descriptions
10 UART1	**enable UART1 wakeup** 1: enable 0: disable
9 UART0	**enable UART0 wakeup** 1: enable 0: disable
7 CAN0	**enable CAN0 wakeup** 1: enable 0: disable
5 SPI1	**enable SPI1 wakeup** 1: enable 0: disable
4 SPI0	**enable SPI0 wakeup** 1: enable 0: disable
3 I2C1	**enable I2C1 wakeup** 1: enable 0: disable
2 I2C0	**enable I2C0 wakeup** 1: enable 0: disable
0 ACMP0	**enable ACMP0 wakeup** 1: enable 0: disable

6.4.6 SPM_WAKEUP_IRQ_STATUS

Table 6-7 SPM_WAKEUP_IRQ_STATUS register

SPM_WAKEUP_IRQ_STATUS SPM Wakeup IRQ Flags Status Register Reset:0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name												OVER_COUNT	PVD	NMI	GPIO	ADC0
Type												W1C	W1C	W1C	W1C	W1C
Reset												0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTC				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
Type	W1C				W1C	W1C	W1C		W1C		W1C	W1C	W1C	W1C		W1C
Reset	0				0	0	0		0		0	0	0	0		0

Bits	Descriptions
20 OVER_COUNT	**SPM over count wake-up flag** 1: before enter stop mode, the SPM will wait for all peripheral ACK. If there is a peripheral NO ACK, the OVER_COUNT is set, and exit stop mode, resulting in an SPM interrupt. 0: invalid **Note: write "1" to clear this bit**
19 PVD	**PVD wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
18 NMI	**NMI wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
17 GPIO	**GPIO wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
16 ADC	**ADC wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**

Bits	Descriptions
15 RTC	**RTC wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
11 UART2	**UART2 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
10 UART1	**UART1 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
9 UART0	**UART0 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
7 CAN0	**CAN0 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
5 SPI1	**SPI1 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
4 SPI0	**SPI0 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**

Bits	Descriptions
3 I2C1	**I2C1 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
2 I2C0	**I2C0 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit**
0 ACMP0	**ACMP0 wake-up flag** 1 : valid 0 : invalid **Note: write "1" to clear this bit.**

7 CAN

7.1 Introduction

7.1.1 The CAN-CTRL core

The CAN-CTRL core is a serial communications controller that performs serial communication according to the CAN protocol. This CAN bus interface uses the basic CAN principle and meets all constraints of the CAN-specification 2.0B active. In addition, the CAN core can be configured as a CAN FD with a flexible data rate that meets the CAN specification.

The CAN protocol uses a multi-master bus configuration for the transfer of frames (communication objects) between nodes of the network and manages the error handling without any burden on the CPU. The CAN-CTRL bus controller enables the user to set up economic and reliable links between various components. The CAN-CTRL core is mapped as an I/O device in the microcontroller. A CPU accesses the CAN-CTRL core to control transmission or reception of frames through a two wire CAN bus system. The connection to a CAN bus is illustrated in Figure 7-1.

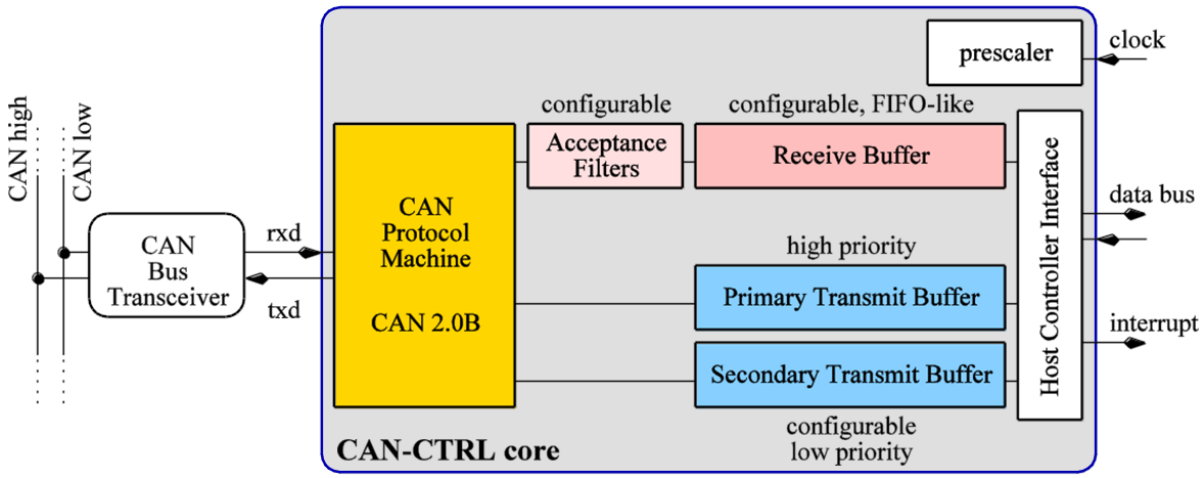

Figure 7-1 Connection to CAN bus and main features of the CAN-CTRL core

7.1.2 The CAN protocol

CAN communication is organized in frames. Two types of frames exist: standard and extended frames. For CAN 2.0, the maximum data payload is up to 8 bytes can be transmitted using one frame. For CAN FD, up to 64 bytes can be transmitted in one frame. All CAN nodes are equal in terms of bus access.

Data addressing is done by using message identifiers. In a CAN network, only one node shall transmit messages with a certain identifier. All nodes receive all messages and the node host controller has to decide if it was addressed by the appropriate message identifier. To reduce the load of a host controller, a CAN core may use acceptance filters. These filters compare all received

message identifiers to user-selectable bit patterns. Only if a message passes an acceptance filter, it will be stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are also used for bus arbitration. The CAN protocol machine stops transmission of a message with a low-priority identifier when a message with a higher priority identifier is transmitted by another CAN node. The CAN protocol machine automatically attempts to re-transmit the stopped message at the next possible transmit position.

CAN 2.0B defines data bit rates up to 1Mbit/s. There is no fixed limit for CAN FD, and the CAN FD standard defines the switching of the bit rate. If enabled, the frame payload can be transmitted at a higher speed, while the frame header is sent at a lower speed.

7.2 Features

- Supports CAN specification
  - CAN 2.0A/B (up to 8 bytes payload, verified by Bosch reference model).
  - Optional support for the CAN FD (up to 64 bytes payload, verified by the ISO 11898-1:2015 or non-ISO Bosch reference model).
- Free programmable data rate
  - CAN 2.0B defines data rates up to 1Mbit/s.
  - CAN FD supports up to 8Mbit/s (limited by the transceiver and the selected CAN-CTRL core clock frequency).
  - AHB divided clock or external oscillator clock can be chosen.
- Programmable baud rate prescaler (1 to 1/256).
- One receive buffers with FIFO-like behavior, the FIFO depth is 7.
- Two transmit buffers:
  - One Primary Transmit Buffer (PTB), the FIFO depth is 1
  - Secondary Transmit Buffer (STB) with the FIFO depth 3, operation in FIFO or priority decision mode.
- 16 Independent and programmable internal 29 bits acceptance filters.
- Extended features:
  - Single Shot Transmission Mode (for PTB and for STB).
  - Listen Only Mode.
  - Loop Back Mode (internal and external).
  - Transceiver Standby Mode.
- Extended status and error report:
  - Capturing of last occurred kind of error and of arbitration lost position.

– Programmable Error Warning Limit.

- Configurable interrupt sources.
- Time-stamping
  - CiA 603 time stamp.
- Programmable wake-up functionality with integrated low-pass filter.

7.3 Application notes

7.3.1 Message buffers

7.3.1.1 Basic concepts about the message buffers

The concept of the message buffers is illustrated in Figure 7-2. All buffer slots are big enough to store frames with the maximum length.

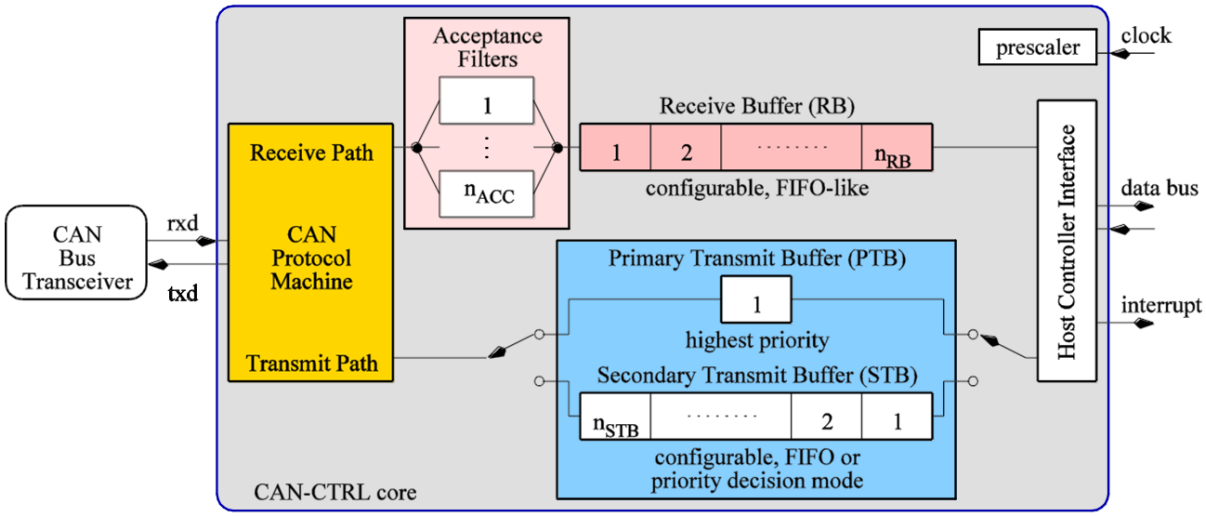

Figure 7-2 Message buffers

7.3.1.2 Receive buffer

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN-CTRL core checks the message identifier during acceptance filtering. If the received frame matches the filter criteria of one of the acceptance filters, then it will be stored in the Receive Buffer (RB), which has FIFO-like behavior.

Depending on the number of available message slots, the host controller does not need to read incoming messages immediately. The CAN-CTRL core is able to generate interrupts upon every received message. If the user has enabled the corresponding "full" interrupt enable register RFIF or "almost full" register RAFIE, when the RB is full or filled to a user-selectable "almost full" limit,

the CAN core will also generate the corresponding interrupt. Because of the FIFO-like behavior, the host controller always reads the oldest message from the RB.

Table 7-1 Receive buffer register RBUF—standard format (rw-u)

Address	Bits								Description
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1						ID(10:8)			Identifier
RBUF+2									Identifier
RBUF+3	ESI								Identifier
RBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				Control bits
RBUF+5	KOER			TX					Status bits
RBUF+8	d1(7:0)								Data byte1
RBUF+9	d2(7:0)								Data byte2
.									.
RBUF+71	d64(7:0)								Data byte64
RBUF+72	RTS(7:0)								CiA 603
.	…								.
RBUF+75	RTS(31:24)								CiA 603

Table 7-2 Receive buffer register RBUF—extended format (rw-u)

Address	Bits								Descriptions
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	ID(15:8)								Identifier
RBUF+2	ID(23:16)								Identifier
RBUF+3	ESI			ID(28:24)					Identifier
RBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control bits
RBUF+5	KOER			TX					Status bits
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.									.
RBUF+71	d64(7:0)								Data byte 64
RBUF+72	RTS(7:0)								CiA 603
.	…								.
RBUF+75	RTS(31:24)								CiA 603

The RBUF register (0x00 to 0x4f) points to the earliest received message in the receive buffer RB. All RBUF registers can be read in any order.

The KOER bit in RBUF has the same meaning as the KOER in register ERRINFO. If RBALL=1, the KOER in RBUF will become meaningful.

If the loopback mode is enabled, and the CAN core has received the data frame which sent by itself, the status bit TX will be set to 1.

The receive timestamp RTS is stored after the message, so compared with TTS, the RTS saved location is related to the RBUF slot.

7.3.1.3 Transmit buffer

For frame transmission purposes, two Transmit Buffers (TB) are offered. The Primary TB (PTB) has a higher priority, but is able to buffer only one frame. The Secondary TB (STB) has a lower priority. It can act in FIFO or in priority mode. The priority decision between PTB and STB is fixed and fully independent from the CAN bus arbitration. Bus arbitration is a priority decision based on the frame identifiers.

The STB can be commanded to transmit one or all stored frames. In FIFO mode with every transmission, the oldest frame inside this buffer is transmitted first. In priority mode, the frame with the highest priority inside this buffer (based on the frame identifier) is transmitted first.

A frame located in the PTB has always a higher priority for the CAN protocol machine than the frames in the STB regardless of the frame identifiers. A PTB transmission stops and delays an STB transmission. The STB transmission is automatically restarted after the PTB frame has been successfully transmitted.

If the STB has started sending and the sending is not completed, the PTB will stop or delay the STB sending after the current STB has finished sending this frame.

Interrupting STB transmissions using a PTB transmission may happen in the following cases:

1. The STB is commanded to output all stored frames and the host controller decides to command a PTB transmission before all STB transmissions are completed.
2. The STB is commanded to output a single frame and the host controller decides to command a PTB transmission before the STB transmission is completed.

If the host controller waits until each commanded transmission is completed, then it can easily decide which buffer shall transmit the next frame.

Table 7-3 Transmit buffer register TBUF—standard format (rw-u)

Address	Bits								Description
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF +1						ID(10:8)			Identifier
TBUF +2									Identifier
TBUF +3	TTSEN								Identifier
TBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				Control bit
TBUF +8	d1(7:0)								Data byte 1
TBUF +9	d2(7:0)								Data byte 2
.									.
TBUF +71	d64(7:0)								Data byte64

Table 7-4 Transmit buffer register TBUF—extended format (rw-u)

Address	Bits								Description
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF +1	ID(15:8)								Identifier
TBUF +2	ID(23:15)								Identifier
TBUF +3	TTSEN			ID(28:24)					Identifier
TBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control bit
TBUF +8	d1(7:0)								Data byte 1
TBUF +9	d2(7:0)								Data byte 2
.									.
TBUF +71	d64(7:0)								Data byte 64

If TBSEL=1, the TBUF register (0x50 to 0x97) points to the next empty message buffer in STB, otherwise it points to PTB. All TBUF registers can be written in any order. For STB, it is needed to set TSNEXT to mark the filled buffer and jump to the next message buffer.

Pay attention to the interval between TBUF+5 and TBUF+7 within the addressing range of the TBUF. This is done for better address segment alignment. It is possible to read and write memory cells in the gap, but it does not make sense for the CAN protocol.

TBUF is built with the real 32-bit wide memory, so write access needs to be performed as 32-bit writes.

Both RBUF and TBUF contain some independent frame control bits (as shown in Table 7-5). For RBUF, these bits indicate the status of the corresponding CAN control field bits of the received CAN frame. And for TBUF, these bits select the appropriate CAN control field bits for the frames that must be sent.

Compared with RTS that stores each received frame, TTS stores only the last frame sent. TTS has nothing to do with the TBUF buffer actually selected.

Table 7-5 RBUF and TBUF control bits

Name	Description
IDE	**Identifier extension** 0: standard frame: ID(10:0) 1: extended frame: ID(28:0)
RTR	**Remote transmission request** 0: data frame 1: remote frame Only the CAN 2.0 frames can be remote frames. The CAN FD has no remote frame. Therefore, if FDF = 1 in TBUF and RBUF, the RRS bit (the corresponding RTR bit of the CAN2.0 frame) is forced to 0. When FDF = 1 even if RRS = 1, the receiving node can still correctly receive the CAN FD frame.
FDF	**CAN FD frame** 0: CAN 2.0 frame (up to 8 bytes per frame) 1: CAN FD frame (up to 64 bytes per frame)
BRS	**Bit rate switching** 0: the whole frame is normal/low bit rate 1: Switch to data/fast bit rate, therefore if FDF=0, the BRS is forced to 0.
DLC	**Frame length, the unit is byte.** For the detailed description, refer to Table 7-6.
ESI	**Error status indicator** 0: no passive errors on other nodes of the CAN bus 1: passive errors on other nodes of the CAN bus This is the read-only status bit of RBUF, not available in TBUF. The CAN-CTRL hardware will automatically embed the correct ESI value in the transmission frame. ESI is included only in the CAN FD framework, not in the CAN 2.0 framework. For CAN 2.0 frames, the ESI bit in RBUF is always kept low. The transmission error status is indicated by the EPASS bit in the register ERRINT.

Name	Description
TTSEN	**Transmit timestamp enable** 0: don't get the transmit timestamp of the frame (the TTS value is invalid). 1: enable TTS update. In TBUF, for the CiA 603 timestamp TTS, update enable is optional.

The data length codes (DLC) in RBUF and TBUF define the length of the payload, the number of payload bytes in the frame.

Remote frames (only for the CAN 2.0 frames with FDF = 0) are always transmitted with 0 payload bytes, but the content of the DLC is sent in the frame header. Therefore, some information can be encoded into the DLC bits of the remote frame. However, if different CAN nodes are allowed to send remote frames with the same ID, it is needed to be paid attention. In this case, all transmitters need to use the same DLC, otherwise it will lead to unresolved conflicts.

Table 7-6 DLC definition (based on the CAN 2.0 and CAN FD specification)

DLC(Binary)	Frame type	Effective bytes
0000 to 1000	CAN 2.0 and CAN FD	0 to 8
1001 to 1111	CAN 2.0	8
1001	CAN FD	12
1010	CAN FD	16
1011	CAN FD	20
1100	CAN FD	24
1101	CAN FD	32
1110	CAN FD	48
1111	CAN FD	64

The TBUF register can be read and written. Therefore, if necessary, the main controller can use TBUF to sequentially prepare messages.

7.3.2 Bus off state

The "bus off" state is signaled using the status bit **BUSOFF** in register **CAN_CTRL0**. A CAN node enters the "bus off" state automatically if its transmit error counter becomes >255. Then it will not take part in further communications until it returns into the error active state again. A CAN node returns to error active state if it is reset by a module(**SRST_CAN0**) reset or 128 groups of 11 consecutive recessive bits (recovery sequence) are received.

In the "bus off" state, **RECNT** is used to count the recovery sequences while **TECNT** stays unchanged. Please note that while entering "bus off" state, **TECNT** rolls over and therefore may hold a small value 0-7. Therefore, it is recommended to use **TECNT** before the node enters bus off state and status bit **BUSOFF** afterwards.

If the node recovers from the "bus off" state, then **RECNT** and **TECNT** are automatically reset to 0.

If a frame is pending for transmission but the CAN node has entered bus off state, then this frame is kept pending. If a node returns to error-active state after bus off, then the transmission of the pending frame will be tried. If this is not desired, then the frame should be aborted by the host controller.

7.3.3 Acceptance filters

To reduce the load of the received frames for the host controller, the core uses acceptance filters. The CAN-CTRL core checks the message identifier during acceptance filtering. Therefore, the length of each acceptance filter is 29 bits.

If a message passes one of the filters, then it will be accepted. If accepted, the message will be stored into the RB and finally RIF is set if RIE is enabled. If the message is not accepted, RIF is not set and the RB FIFO pointer will not be increased. Messages that are not accepted will be discarded and overwritten by the next message. No stored valid message will be overwritten by any not accepted message.

Independently of the result of acceptance filtering, the CAN-CTRL core checks every message on the bus and sends an acknowledge or an error frame to the bus.

The acceptance mask defines which bits shall be compared while the acceptance code defines the appropriate values. Setting mask bits to 0 enables the comparison of the selected acceptance code bits with the corresponding message identifier bits. Mask bits that are set to 1 are disabled for the acceptance check and this results in accepting the message.

The identifier bits will be compared with the corresponding acceptance code bits ACODE as follows:

- Standard: ID(10:0) with ACODE(10:0).
- Extended: ID(28:0) with ACODE(28:0).

Example: If AMASK_x(0)=0 and all other AMASK_x bits are 1, then the value of the last ID bit has to be equal to ACODE(0) for an accepted message. All other ID bits are ignored by the filter.

Note

Disabling a filter by setting AE_x=0 blocks messages. In contrast to this, disabling a mask bit in AMASK_x disables the check for this bit which results in accepting messages.

The definitions of AMASK and ACODE alone do not distinguish between standard or extended frames. If bit AIDEE=1 then the value of AIDE defines with frame type is accepted. Otherwise if AIDE=0 both types are accepted.

After power-on reset, the CAN-CTRL core is configured to accept all messages. (Filter 0 is enabled by AE_0=1, all bits in AMASK_0 are set to 1 and AIDEE=0. All other filters are disabled. Filter 0

is the only filter that has defined reset values for AMASK/ ACODE while all other filters have undefined reset values.)

7.3.4 Message reception

The received data will be stored in the RB as shown in Figure 7-3. The RB is configurable by a pre-synthesis parameter and has FIFO-like behavior. Every received message that is valid and accepted sets RIF=1 if RIE is enabled. RSTAT is set depending of the fill state. When the number of filled buffers is equal to the programmable value AFWL, then RAFIF is set if RAFIE is enabled. In case, when all buffers are full, the RFIF is set if RFIE is enabled.

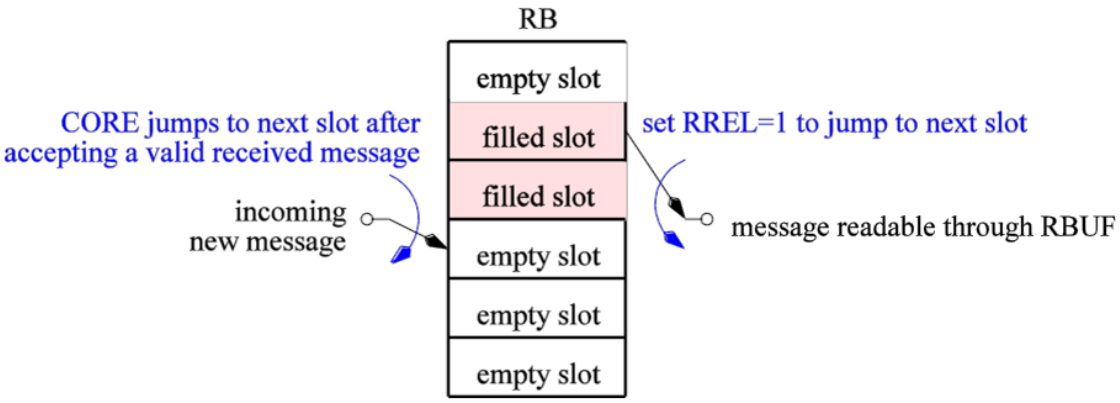

Figure 7-3 Schematic of the FIFO-like RB (example with 6 slots)

The RB always maps the message slot containing the oldest message to the RBUF registers. The maximum payload length for CAN 2.0 messages is 8. The individual length of each message is defined by the DLC. Because of this, the RB provides slots for each message and the host controller is required to set RREL to jump to the next RB slot. All RBUF bytes of the actual slot can be read in any order.

If the RB is full, the next incoming message will be stored temporarily until it passes for valid (6^{th} EOF bit). Then if ROM=0 the oldest message will be overwritten by the newest or if ROM=1 the newest message will be discarded. In both cases, ROIF is set if ROIE is enabled. If the host controller reads the oldest message and sets RREL before a new incoming message becomes valid, then no message will be lost.

7.3.5 Handling message receptions

Without acceptance filtering, the CAN-CTRL core would signal the reception of every frame and the host would be required to decide if it was addressed. This would result in quite a big load on the host controller.

It is possible to disable interrupts and use the acceptance filters to reduce the load for the host controller. For a basic operation: RIF is set to 1 if RIE is enabled and the CAN-CTRL core has received a valid message. To reduce the number of reception interrupts, it is possible to use

RAFIE/RAFIF (RB Almost Full Interrupt) or RFIE/RFIF (RB Full Interrupt) instead of RIE/RIF (Reception Interrupt). The “almost full limit” is programmable using AFWL.

The RB contains a number of RB slots, which is selectable before synthesis using a generic parameter. Reading the RB shall be done as follows:

1. Read the oldest message from the RB FIFO using the RBUF registers.
2. Release the RB slot with RREL=1. This selects the next message (the next FIFO slot), RBUF will be updated automatically.
3. Repeat these actions until RSTAT signals an empty RB.

If the RB FIFO is full and a new received message is recognized as valid (6th EOF bit), then one message will be lost (see bit ROM). Before this event, no message is lost. This should give enough time for the host controller to read at least one frame from the RB after the RB FIFO has been filled and the selected interrupt has occurred. To enable this behavior, the RB includes one more (hidden) slot than specified by the synthesis parameter RBUF_SLOTS. This hidden slot is used to receive a message, validate it and check it if it matches the acceptance filters before an overflow occurs.

7.3.6 Message transmission

Before starting any transmission, at least one of the transmit buffers (PTB or STB) has to be loaded with a message. TPE signals if the PTB is locked and TSSTAT signals the fill state of the STB. The TBUF registers provide access to both the PTB as well as to the STB.

Below is the recommended programming flow:

1. Set TBSEL to the desired value to select either the PTB or the STB.
2. Write the frame to the TBUF registers.
3. For the STB set TSNEXT=1 to finish loading of this STB slot.

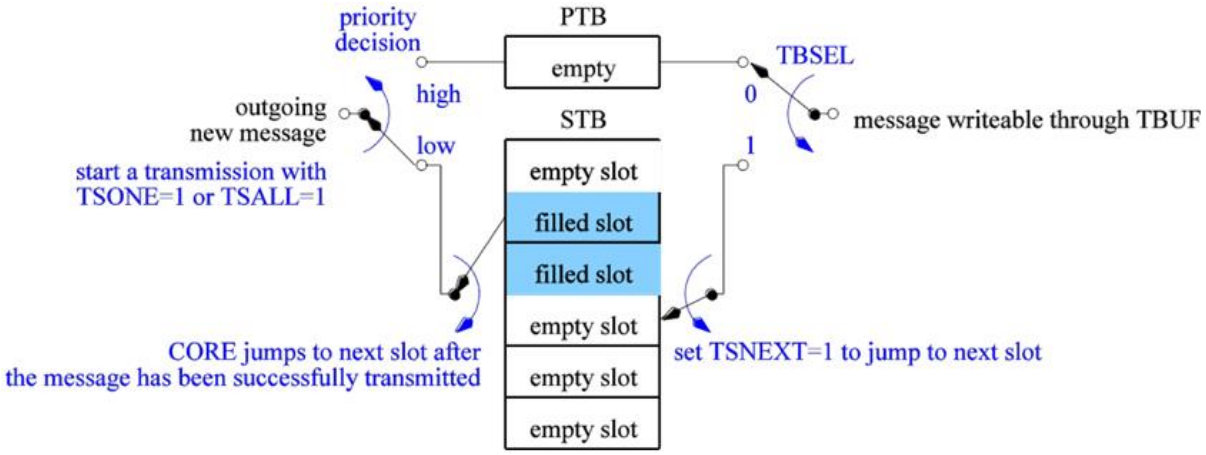

Figure 7-4 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

The maximum payload length for CAN 2.0 messages is 8 bytes. The maximum length for CAN FD is 64 bytes. The individual length of each message is defined by the DLC. For remote frames (bit RTR), the DLC becomes meaningless, because remote frames always have a data length of 0 bytes. The host controller is required to set TSNEXT to jump to the next STB slot. All TBUF bytes can be written in any order.

Setting TSNEXT=1 is meaningless if TBSEL=0 selects the PTB. In this case TSNEXT is automatically cleared and does no harm.

Bit TPE should be set to start a transmission when using the PTB. To use the STB, TSONE has to be set to start a transmission of a single message or TSALL to transmit all messages.

The PTB has always a higher priority than the STB. If both transmit buffers have the order to transmit, the PTB message will be always sent first regardless of the frame identifiers. If a transmission from the STB is already active, it will be completed before the message from the PTB is sent at the next possible transmit position (the next interframe slot). After the PTB transmission is completed or aborted, the CAN- CTRL core returns to process other pending messages from the STB.

When the transmission is completed, the following transmission interrupts are set:

- For the PTB, TPIF is set if TPIE is enabled.
- For the STB using TSONE, TSIF is set if one message has been completed and TSIE is enabled.
- For the STB using TSALL, TSIF is set if all messages have been completed and if TSIE is enabled. In other words, TSIE is set if the STB is empty.

It is meaningless to set TSONE or TSALL while the STB is empty. In such a case TSONE respectively TSALL will be reset automatically. No interrupt flag will be set and no frame will be transmitted.

7.3.7 Message transmission abort

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. To avoid such situation, the host controller can withdraw the transmission request by setting TPA or TSA respectively, if the transmission has not yet been started. Both TPA and TSA source a single interrupt flag: AIF.

The CAN protocol machine executes an abort only if it does not transmit anything to the CAN bus. Therefore, the following is valid:

- There is no abort during bus arbitration.
  - If the node loses arbitration, the abort will be executed afterwards.
  - If the node wins arbitration, the frame will be transmitted.
- There is no abort while a frame is transmitted.

- If a frame is transmitted successfully, then a successful transmission is signaled to the host controller. In this case no abort is signaled. This is done by the appropriate interrupt and status bits.
- After an unsuccessful transmission where the CAN node does not receive an acknowledge, the error counter is incremented and the abort will be executed.
- If there is at least one frame left in the STB, while the host has commanded all frames to be transmitted (TSALL=1), then both the completed frame as well as the abort is signaled to the host.

Because of these facts, aborting a transmission may take some time depending on the CAN

communication speed and frame length. If an abort is executed, this results in the following actions:

- TPA releases the PTB which results in TPE=0.

  The frame data is still stored in the PTB after releasing the PTB.
- TSA releases one single message slot or all message slots of the STB. This depends on whether TSONE or TSALL was used to start the transmission. TSSTAT will be updated accordingly. Releasing a frame in the STB results in discarding the frame data because the host cannot access it.

Setting both TPA and TSA simultaneously is not recommended. If a host controller decides to do it anyway, AIF will be set and both transmissions from PTB and STB will be aborted if possible. As already stated if one transmission will be completed before the abort can be executed, this will result in signaling a successful transmission. Therefore, the following interrupt flags may be set if enabled:

- AIF (once for both PTB and STB transmission abort)
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF (very seldom, will only happen if the host does not handle TPIF immediately)
- TPIF + TSIF + AIF (very seldom, will only happen if the host does not handle TPIF and TSIF

  immediately)

To clear the entire STB, both TSALL and TSA need to be set. In order to detect if a message cannot be sent for a long time because it loses arbitration, the host may use the ALIF/ALIE.

7.3.8 Full STB

After writing a message to the STB, TSNEXT=1 marks a buffer slot filled and jumps to the next free message slot. TSNEXT is automatically reset to 0 by the CAN-CTRL core after this operation. If the last message slot has been filled and therefore all message slots are occupied, then TSNEXT stays set until a new message slot becomes free. While TSNEXT=1, then writing to TBUF is blocked by the CAN-CTRL core.

When a slot becomes free, then the CAN-CTRL core automatically resets TSNEXT to 0. A slot becomes free if a frame from the STB is transmitted successfully or if the host requests an abort (TSA=1). If a TSALL transmission is aborted, then TSNEXT is also reset, but additionally the complete STB is marked as empty.

7.3.9 Extended status and error report

During CAN bus communication transmission errors may occur. The following features support detection and analysis of them.

7.3.9.1 Programmable error warning limit

Errors during reception/ transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in register LIMIT, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in steps of 8 errors from 8 to 128:

Limit of count of errors = (EWL+1) * 8.

The interrupt EIF will be set if enabled by EIE under the following conditions:

- The border of the error warning limit has been crossed in either direction by RECNT or TECNT
- The BUSOFF bit has been changed in either direction.

7.3.9.2 Arbitration Lost Capture (ALC)

The core is able to detect the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled. The value of ALC stays unchanged if the node is able to win the arbitration. Then ALC holds the old value of the last loss of arbitration.

The value of ALC is defined as follows: A frame starts with the SOF bit and then the first bit of the ID is transmitted. This first ID bit has ALC value 0, the second ID bit ALC value 1 and so on. Arbitration is only allowed in the arbitration field. Therefore, the maximum value of ALC is 31, which is the RTR bit in extended frames.

Additional hint: If a standard remote frame arbitrates with an extended frame, then the extended frame loses arbitration at the IDE bit and ALC will be 12. The node transmitting the standard remote frame will not notice that an arbitration has been taken place, because this node has won. It is impossible to get an arbitration loss outside of the arbitration field. Such an event is a bit error.

7.3.9.3 Kind of Error (KOER)

The core recognizes errors on the CAN bus and stores the last error event in the KOER bits. A CAN bus error can be signaled by the BEIF interrupt if it is enabled. Every new error event overwrites

the previous stored value of KOER. Therefore, the host controller has to react quickly on the error event.

7.3.9.4 Receive all data frame (RBALL)

If RBALL = 1, all received data frames will be stored, even if there are error data frames. This also applies to the loopback mode. Only data frames are stored in RBUF, error frames or overload frames are not stored.

If the CiA 603 timestamp is enabled (TIMEEN = 1) and time stamping is configured for EOF (TIMEPOS = 1). In the event of an error, the time stamp will be obtained at the beginning of the error frame.

Most possible errors will only occur when the node is the sender of the frame. In this case, if the framework activates the loopback mode, it will only be stored in RBUF.

Depending on the error type, when the other parts are unknown, the frames stored in the RBUF slot may be valid.

7.3.10 Extended features

7.3.10.1 Single shot transmission

Sometimes an automatic re-transmission is not desired. Therefore, the order to transmit a message only once can be set separately for the transmit buffers PTB by the bit TPSS and for the transmit buffer STB by the bit TSSS. In this case no re-transmission will be performed in the event of an error or arbitration lost if the selected transmission is active.

In the case of an immediate successful transmission, there is no difference to normal transmission. But in the case of an unsuccessful transmission, the following will happen.

- TPIF gets set if enabled, the appropriate transmit buffer slot gets cleared.
- In case of an error, KOER and the error counters get updated. BEIF gets set if BEIE is enabled and the other error interrupt flags will act accordingly.
- In case of a lost arbitration, ALIF gets set if ALIE is enable.

Therefore, for a single send, TPIF does not indicate whether the frame has been successfully sent or not. The single send should be used only with BEIF and ALIF.

If single shot transmission is used with TSALL and there is more than one frame in the STB then for each frame a single shot transmission is done. Regardless if any of the frames is not successfully transmitted (e.g. because of an ACK error), the CAN-CTRL advances to the next frame and stops if the STB is empty.

Therefore, in this scenario only the error counters indicate what has happened. This can be quite complex to evaluate because if one of two frames got errors, the host cannot detect which one was the successful one.

If the bus is occupied by another frame, the CAN-CTRL waits until the bus is free and then tries to send a single frame if a single transmission is started.

7.3.10.2 Listen Only Mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus.

Another application is the automatic bit rate detection where the host controller tries different timing settings until no errors occur.

Errors will be monitored (KOER, BEIF) in LOM.

In LOM, the core is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules.

- In LOM, the protocol machine acts as if in error passive mode where only recessive error flags are generated. Only the protocol machine acts as if in error passive mode. All other components including the status registers are not touched.
- In LOM, the protocol machine does not generate a dominant ACK.
- The error counters stay unchanged regardless of any error condition.

Important facts regarding ACKs for LOM:

- If a frame is transmitted by a node, then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus that is not in LOM. Then there will be no error and all nodes (also those in LOM) will receive the frame.
- If there is an active or passive error flag after an ACK error, then the node in LOM is able to detect this as ACK error.

Activation of LOM should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If LOM is enabled, then no transmission can be started.

7.3.10.3 Bus connection test

To check if a node is connected to the bus, the following steps shall be done:

- Transmit a frame. If the node is connected to the bus, then its TX bits are visible on its RX input.
- If there are other nodes connected to the CAN bus, then a successful transmission (including an acknowledge from the other nodes) is expected. No error will be signaled.
- If the node is the only node that is connected to the CAN bus (but the connection between the bus, the transceiver and the CAN-CTRL core is fine), then the first regular error situation occurs in the ACK slot because of no acknowledge from other nodes. Then a BEIF error interrupt will be generated if enabled and KOER= “100” (ACK error).

- If the connection to the transceiver or the bus is broken, then immediately after the SOF bit the BEIF error interrupt will be set and KOER= “001” (BIT error).

7.3.10.4 Loop Back Mode (LBMI and LBME)

CAN-CTRL supports two Loop Back Modes: internal (LBMI) and external (LBME). Both modes result in reception of the own transmitted frame which can be useful for self-tests.

In LBMI, CAN-CTRL is disconnected from the CAN bus and the txd output is set to recessive. The output data stream is internally fed back to the input. In LBMI, the node generates a self-ACK to avoid an ACK error.

In LBME, CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver, CAN-CTRL receives its own frame. In LBME, the node does not generate a self-ACK. Therefore, there are two possible results upon a frame transmission in LBME mode:

1. Another node receives the frame too and generates an ACK. This will result in a successful transmission and reception.
2. No other node is connected to the bus and this results in an ACK error. To avoid retransmissions and incrementing the error counters, it is recommended to use TPSS or TSSS if it is unknown if other nodes are connected.

In Loop Back Mode, the core receives its own message, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled.

LBMI can be useful for chip-internal and software tests while LBME can test the transceiver and the connections to it.

Activation of both Loop Back Modes should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If the node is connected to a CAN bus, switching back from LBMI to normal operation cannot be accomplished by simply setting LBMI to 0, as it may be that this occurs just at the moment while another CAN node is transmitting. In this case, switching back to normal operation shall be done by setting bit RESET to 1. This automatically clears LBMI to 0. Finally RESET can be disabled and the core returns back to normal operation. In contrast to this, LBME can be disabled every time.

7.3.10.5 Transceiver standby mode

Using the register bit STBY, the output signal standby is driven. It can be used to activate a standby mode for a transceiver.

Once the standby mode is activated, sending is not possible, so TPE, TSONE and TSALL cannot be set. On the other hand, CAN-CTRL does not allow STBY to be set if a transmission is already active (TPE, TSONE or TSALL is set).

If STBY is set, the transceiver enters a low-power mode. In this mode, it is unable to receive a frame at full speed but monitors the CAN bus for a dominant state. If the dominant state is active for a time which is defined in the transceivers data sheet, the transceiver will pull the rxd signal low. If rxd is low, CAN-CTRL automatically clears STBY to 0 which disables standby mode for the transceiver. This is done silently without an interrupt to the host controller.

Switching from standby mode to active mode takes some time for the transceiver and therefore the initial wakeup frame cannot be received successfully. Therefore, the node recently being in standby will not respond with an ACK. If no CAN node at the bus responds to the wakeup frame with an ACK, then this results in an ACK error for the transmitter of the wakeup frame. Then the transmitter will automatically repeat the frame. During the repetition, the transceiver will be back in active mode and CAN-CTRL will receive the frame and will respond with an ACK.

In summary, one node transmits a frame for wakeup. If all others nodes are in standby mode, the transmitter gets an ACK error and will automatically repeat the frame. During the repetition, the nodes are back in active mode and will respond with an ACK.

STBY is not affected by bit RESET.

7.3.10.6 Error counter reset

According to the CAN standard RECNT counts receive errors and TECNT counts transmit errors. After too many transmit errors, a CAN node has to go to bus off state. This will activate the bit RESET. Deactivating the bit, RESET does not modify the errors counters or bus off state. The CAN specification defines rules how to disable bus off state and to decrease the error counters. A good node will recover from all of this automatically if only a temporary error has caused the problems. The classic CAN 2.0B specification requires this automatic behavior without host controller interaction to avoid the babbling idiot problem.

Setting BUSOFF to 1 without setting EIF resets the error counters, thus forcing the node to exit the bus off state.

7.3.10.7 Low-pass filter

When the MCU enters the stop mode, the CAN wake-up can be enabled. If it is needed to filter out the glitch signal (less than 2us), set the register EN_CAN0_FILTER equal to 1.

7.3.11 Software reset

If the bit RESET in CAN_CTRL0 is set to 1, then the software reset is active. Several components are forced to a reset state if RESET=1 but no all components are touched by RESET. Some components are only sensitive to a hardware reset. The reset values of all bits are always the same for software and hardware reset.

Table 7-7 Software reset

Register	RESET	Description
ACFADR	No	-
ACODE_x	No	Register is writeable if RESET=1 and write-locked if 0.
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIE	No	-
ALIF	Yes	-
AMASK_x	No	Register is writeable if RESET=1 and write-locked if 0.
BEIE	No	-
BEIF	Yes	-
BUSOFF	No	BUSOFF and error counter can be reset by setting BUSOFF=1.
EIE	No	-
EIF	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
FD_ISO	No	Register is writeable if RESET=1 and write-locked if 0.
F_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
F_SEG_1	No	Register is writeable if RESET=1 and write-locked if 0.
F_SEG_2	No	Register is writeable if RESET=1 and write-locked if 0.
F_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
KOER	Yes	-
LBME	Yes	-
LBMI	Yes	-
LOM	No	-
RACTIVE	Yes	Reception is immediately cancelled even if a reception is active. No ACK will be generated.
RAFIE	No	-
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
RECNT	No	-
RFIE	No	-
RFIF	Yes	-
RIE	No	-

Register	RESET	Description
RIF	Yes	-
ROIE	No	-
ROIF	Yes	-
ROM	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-
RSTAT	Yes	All RB slots are marked as empty.
SACK	Yes	-
SELMASK	No	-
STBY	No	-
S_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
S_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
TACTIVE	Yes	All transmissions are immediately terminated with RESET. If a transmission is active, this will result in an erogenous frame. Other nodes will generate error frames.
TBSEL	Yes	TBUF fixed to point to PTB
TBUF	(Yes)	All STB slots are marked as empty because TBSEL TBUF points to the PTB.
TECNT	No	-
TIMEEN	No	-
TIMEPOS	No	-
TPA	Yes	-
TPE	Yes	-
TSA	Yes	-
TSALL	Yes	-
TSMODE	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIE	No	-
TPIF	Yes	-
TPSS	Yes	-
TSFF	Yes	All STB slots are marked as empty.
TSIE	No	-
TSIF	Yes	-
TSSS	Yes	-
TSSTAT	Yes	All STB slots are marked as empty.
TTS	No	-

7.3.12 CAN bit time

CAN 2.0B defines data bit rates up to 1Mbit/s. There is no fixed limit for CAN FD. For real life systems the data rates are limited by the used transceiver and the achievable clock frequency for the CAN-CTRL core which depends on the used target cell library.

The CAN-CTRL core can be programmed to arbitrarily chosen data rates only limited by the range of the bit settings in the appropriate bit timing and prescaler registers.

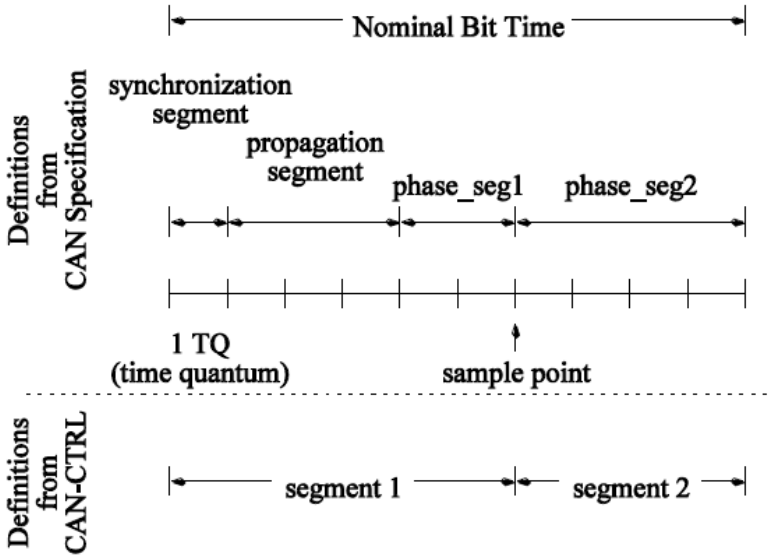

Figure 7-5 CAN bit timing

The CAN bit time BT consists of several segments as shown in Figure 7-5. Each segment consists of a number of time quanta units n_{TQ} . The duration of a time quanta TQ is:

$$TQ = n_{prescaler} / f_{CLOCK}$$

$$BT = n_{prescaler} * n_{TQ} / f_{CLOCK} = t_{Seg\_1} + t_{Segt\_2}$$

The CAN specification requires several relationships between the segment lengths (Table 7-8) which results in relationships between t_{Seg_1}, t_{Segt_2} and the maximum synchronization jump width t_{SJW}. Please note that Table 7-8 lists the minimum configuration ranges defined by the CAN specification.

Table 7-8 CAN Timing Segments

Segment	Description
SYNC_SEG	**Synchronization Segment** = 1 TQ
PROP_SEG	**Propagation Segment** [1...8] TQ CAN 2.0 bit rate CAN FD not enabled [1...48] TQ CAN 2.0 bit rate CAN FD enabled [1...48] TQ CAN FD nominal bit rate [0...8] TQ CAN FD data bit rate

Segment	Description
PHASE_SEG1	**Phase Buffer Segment 1** [1...8] TQ CAN 2.0 bit rate CAN FD not enabled [1...16] TQ CAN 2.0 bit rate CAN FD enabled [1...16] TQ CAN FD nominal bit rate [1...8] TQ CAN FD data bit rate
PHASE_SEG2	**Phase Buffer Segment 2** [2...8] TQ CAN 2.0 bit rate CAN FD not enabled [2...16] TQ CAN 2.0 bit rate CAN FD enabled [2...16] TQ CAN FD nominal bit rate [2...8] TQ CAN FD data bit rate
SJW	**Synchronization Jump Width** [1...4] TQ CAN 2.0 bit rate CAN FD not enabled [1...16] TQ CAN 2.0 bit rate CAN FD enabled [1...16] TQ CAN FD nominal bit rate [1...8] TQ CAN FD data bit rate
IPT	Information Processing Time = [0…2] TQ PHASE_SEG2 ⩾ IPT

As can be seen in Table 7-8, the CAN-CTRL core collects SYNC_SEG, PROP_SEG and PHASE_SEG1 into one group and the length of the group is configurable with t_{Seg_1}. Table 7-9 lists the available configuration ranges. Please note that the CAN-CTRL core does not check if all rules are met and offers a wider configuration range than defined by the CAN specification.

Table 7-9 CAN-CTRL Timing Settings

Setting	Requirements			
t_{seg_1}	[2…65]		TQCAN 2.0 bit rate	(slow)
	[2…65]		CAN FD nominal bit rate	(slow)
	[2…17]		CAN FD data bit rate	(fast)
t_{seg_2}	[1…8] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$	CAN 2.0 bit rate	(slow)
	[1…32] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$	CAN FD nominal bit rate	(slow)
	[1…8] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 1$	CAN FD data bit rate	(fast)
t_{SJW}	[1…16] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN 2.0 bit rate	(slow)
	[1…16] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN FD nominal bit rate	(slow)
	[1…8] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN FD data bit rate	(fast)

For CAN 2.0 bit rate nominal bit rate and CAN FD(slow) nominal bit rate, the register settings S_Seg_1, S_Seg_2, S_SJW and S_PRESC define the appropriate segment lengths. The register S_Seg_1, S_Seg_2, S_SJW and S_PRESC are invalid for CAN FD(fast) data bit rate.

$t_{Seg\_1} = (S\_Seg\_1 + 2) * TQ$ $t_{Seg\_1} = (F\_Seg\_1 + 2) * TQ$

$t_{Seg\_2} = (S\_Seg\_2 + 1) * TQ$ $t_{Seg\_2} = (F\_Seg\_2 + 1) * TQ$

$t_{SJW} = (S\_SJW + 1) * TQ$ $t_{SJW} = (F\_SJW + 1) * TQ$

$n_{prescaler} = S\_PRESC + 1$ $n_{prescaler} = F\_PRESC + 1$

The CAN FD core can switch from the low-speed nominal bit rate to the fast data bit rate by setting BRS=1, and switch back at the sampling point of the CRC delimiter bit.

For CAN FD node, the communication delay of CAN FD may be more than one bit due to the delay of transceiver. Therefore, the original SP (sample point) cannot be used to sample the correct bit value, and CAN FD specification defines an additional secondary sampling point (SSP). At this time, you can choose to enable TDC (TDCEN = 1) and configure SSPOFF register. It is recommended that SSPOFF configuration is equal to t_{Seg_1}, the user needs to configure the register according to the actual situation. If TDC is not enabled, the sending node may not be able to sample the data it sends correctly, and the sending node will detect a bit error.

The following steps need to be carried for the configuration of the CAN-CTRL core:

1. Set bit RESET=1.
2. Set registers S_Seg_1 and S_Seg_2:

   In the example the data rate on the bus 1M baud and the system clock is 48 MHz.

   The values of nTQ and $nprescaler$ have to be selected to fit BT.

   In this example $nprescaler$ = 2 and nTQ = 24 are chosen which results in a perfect match: $BT = 24TQ$.

   In the time segment definitions, $tSeg\_1 = 18TQ$; $tSeg\_2 = 6TQ$ can be chosen as suitable values which finally results in the register settings S_Seg_1=16 and S_Seg_2=5.
3. Load the acceptance code and mask registers (optional).
4. Set register S_SJW:

   With $t_{Seg\_2} \geq t_{SJW}$, one is free to choose $tSJW = 4$ which finally results in S_SJW=3.
5. Load the clock prescaler register S_PRESC: $n_{prescaler}$ = PRESC + 1 results in S_PRESC=1.
6. Set bit RESET=0.

   The given order is not mandatory. It is merely necessary to set bit RESET=1 at the beginning, as it is otherwise not possible to load the bit timing, ACODE and AMASK registers. RESET=0 is required upon completion of the configuration. The controller then waits for 8 recessive bits (frame end) and resumes its normal operation.
7. Continue with configuration of interrupts other configuration bits and execute commands.

There are some sample tables for the bit timing settings that should apply to all nodes in a CAN network.

Table 7-10 Sample settings for 48MHz can_clk

Bit rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	1	48	36	12	12
0.8	80	2	30	24	6	6
0.5	75	2	48	36	12	12
0.25	75	4	48	36	12	12
0.125	75	8	48	36	12	12
0.1	75	10	48	36	12	12

Table 7-11 Sample settings for 8MHz can_clk

Bit rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	1	8	6	2	2
0.8	80	1	10	8	2	2
0.5	75	1	16	12	4	4
0.25	75	2	16	12	4	4
0.125	75	4	16	12	4	4
0.1	75	4	20	15	5	4

Table 7-12 Sample settings for 48MHz can fd clk

Bit rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1[TQ]	Seg2[TQ]	SJW[TQ]	TDC[clk]
8	83	1	6	5	1	1	5
6	75	1	8	6	2	2	6
4	75	1	12	9	3	3	9
2	75	1	24	18	6	6	18
1	75	2	24	18	6	6	36
0.1	75	20	24	18	6	6	-
0.05	75	40	24	18	6	6	-

7.3.13 Time stamp

The time stamp in CAN-CTRL includes the time stamp of the sent frame and the time stamp of the received frame. When receiving data or sending data, CAN-CTRL copies the timer value and the frame timestamp and is stored in RTS and TTS. The user can read the register RTS or TTS to get the timestamp value.

CAN-CTRL can obtain the time stamp at the sampling point of the SOF or EOF bit of the valid frame. The sampling point position can be selected by configuring the bit TIMEPOS. The seven recessive bits after the ACK delimiter form the EOF of the CAN/CAN FD frame. Software-based time stamping, which is widely used in many systems, depends on the receiving and transmitting interrupts. Therefore, it is recommended to configure the timestamp sampling point at EOF.

CAN-CTRL only supports the time stamp of one transmission frame (TTS), but has a separate time stamp for the received frame (RTS). The time stamp for generating transmission frames can be enabled or disabled for each frame using the TTSEN bit in the TBUF slot.

CAN-CTRL contains a timer, a timestamp mechanism, a register to store TTS, and a memory to store RTS every frame. The register TIMEEN bit enables or disables time stamping. If disabled, TTS and RTS are invalid.

The following examples illustrate the steps of the CAN-CTRL sending timestamp:

1. Initialize CAN-CTRL, enable transmission interrupt and configure the clock source to 48MHZ;
2. Set the CAN-CTRL timer clock division to 48, then each count value in TTS is 1us;
3. Set the sampling point position to EOF by configuring the TIMEPOS register;
4. Enable the register TIMEEN and the register TTSEN;
5. Read the TTS value after the CAN-CTRL transmission is completed.

The following examples illustrate the steps of the CAN-CTRL receiving timestamp:

1. Initialize CAN-CTRL, enable the receive interrupt and configure the clock source to 48MHZ;
2. Set the CAN-CTRL timer clock division to 48, then each count value in RTS is 1us;
3. Set the sampling point position to EOF by configuring the TIMEPOS register;
4. Enable the register TIMEEN;
5. Read the RTS value after the CAN-CTRL reception is completed.

7.4 Register Definition

The register mapping for 32 bit interfaces is given in Table 7-13.

Table 7-13 CAN-CRTL Register mapping

CAN0 base address:0x40007800

Address	Name	Width (in bit)	Description
CAN base addr ~ CAN base addr +0x4F	CAN_RBUF	32*20	Receive Buffer Register and receive time stamp
CAN base addr +0x50 ~ CAN base addr +0x97	CAN_TBUF	32*18	Transmit Buffer Register

Address	Name	Width (in bit)	Description
CAN base addr +0x98 ~ CAN base addr +0x9F	CAN_TTSx	32*2	Transmission data frame timestamp(x=0-1), TTS1 reserved
CAN base addr +0xA0	CAN_CTRL0	32	Control register 0
CAN base addr +0xA4	CAN_CTRL1	32	Control register 1
CAN base addr +0xA8	CAN_SBITRATE	32	Low speed CAN baudrate configuration register
CAN base addr +0xAC	CAN_FBITRATE	32	High speed CAN baudrate configuration register
CAN base addr +0xB0	CAN_ERRINFO	32	CAN error type and error counter register
CAN base addr +0xB4	CAN_ACFCTRL	32	filter control register
CAN base addr +0xB8	CAN_ACF	32	filter enable register
CAN base addr +0xBC	CAN_VERSION	32	CAN core version register

7.4.1 CAN_TTSx

Table 7-14 CAN_TTSx register

CAN_TTSx **Transmission data frame time stamp** **RESET: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TTS															
Type	R															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TTS															
Type	R															
Reset	0															

Bits	Description
31: 0 TTS	**Transmission time stamp** TTSx is 32bit. TTS holds the time stamp of the last transmitted frame for CiA 603 time stamping. If TTSEN = 1, Every new frame overwrites TTS. The time-stamp is 32 bit wide, where TTS1 is reserved.

7.4.2 CAN_CTRL0

Table 7-15 CAN_CTRL0 register

CAN_CTRL0 CAN ctrl register 0 RESET: 0x00900080

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SACK	ROM	ROV	RREL	RBALL		RSTAT		FDISO	TSNEXT	TSMODE				TSSTAT	
Type	RW	RW	R	RW	RW		R		RW	RW	RW				R	
Reset	0	0	0	0	0		0		1	0	0				0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bits	Description
31 SACK	**Self Ack** 0: no self ack 1: self ack if LBME=1
30 ROM	**Receive buffer Overflow Mode** In case of a full RBUF when a new message is received, then ROM selects the following: 1: The new message will not be stored. 0: The oldest message will be overwritten.
29 ROV	**Receive buffer Overflow** 1: Overflow. At least one message is lost. 0: No Overflow. ROV is cleared by setting RREL=1.
28 RREL	**Receive buffer Release** 1: RB Release, the host has read the RB. 0: No release The host controller has read the actual RB slot and releases it. Afterwards the CAN-CTRL core points to the next RB slot. RSTAT gets updated.

Bits	Description
27 RBALL	**RB buffer stores all data frames** 0: normal operation 1: The RB buffer stores all received data frames, even if the data frame is wrong, it will be stored. This bit setting also applies to the loopback mode. Only data frames but not error frames and overload frames are stored.
25:24 RSTAT	**Receive buffer Status** 00: empty 01: > empty and < almost full (AFWL) 10: almost full (programmable threshold by AFWL) but not full and no overflow 11: full (stays set in case of overflow – for overflow signaling see ROV)
23 FDISO	**CAN FD ISO mode** 0: Bosch CAN FD(non-ISO) mode 1: ISO CAN FD mode (ISO 11898-1:2015) ISO CAN FD mode has a different CRC initialization value and an additional stuff bit count. Both modes are incompatible and must not be mixed in one CAN network. This bit has no impact to CAN 2.0B. This bit is only writeable if RESET=1.
22 TSNEXT	**Transmit buffer Secondary Next** 0: no action 1: STB slot filled, select next slot. After all frame bytes are written to the TBUF registers, the host controller has to set TSNEXT to signal that this slot has been filled. Then the CAN-CTRL core connects the TBUF registers to the next slot. Once a slot is marked as filled a transmission can be started using TSONE or TSALL. It is possible to set TSNEXT and TSONE or TSALL together in one write access. TSNEXT has to be set by the host controller and is automatically reset by the CAN-CTRL core immediately after it was set. Setting TSNEXT is meaningless if TBSEL=0. In this case TSNEXT is ignored and automatically cleared. It does not do any harm. If all slots of the STB are filled, TSNEXT stays set until a slot becomes free. TSNEXT has no meaning in TTCAN mode and is fixed to 0.

Bits	Description
21 TSMODE	**Transmit buffer Secondary operation Mode** 0: FIFO mode 1: priority decision mode In FIFO mode frames are transmitted in the order in that they are written into the STB. In priority decision mode the frame with the highest priority in the STB is automatically transmitted first. The ID of a frame is used for the priority decision. A lower ID means a higher priority of a frame. A frame in the PTB has always the highest priority regardless of the ID. TSMODE shall be switched only if the STB if empty.
17:16 TSSTAT	**Transmission Secondary Status bits** 00: STB is empty. 01: STB is less than or equal to half full 10: STB is more than half full 11 : STB is full
15 TBSEL	**Transmit Buffer Select** 0: PTB (high-priority buffer) 1: STB Selects the transmit buffer to be loaded with a message. Use the TBUF registers for access. TBSEL needs to be stable all the time the TBUF registers are written and when TSNEXT is set.
14 LOM	**Listen Only Mode** 0: disabled 1: enabled LOM should not be enabled while a transmission is active. No transmission can be started if LOM is enabled.
13 STBY	**Transceiver Standby Mode** 0: disabled 1: enabled This register bit is connected to the output signal standby which can be used to control a standby mode of a transceiver. STBY cannot be set to 1 if TPE=1, TSONE=1 or TSALL=1. If the host sets STBY to 0 then the host needs to wait for the time required by the transceiver to start up before the host requests a new transmission.

Bits	Description
12 TPE	**Transmit Primary Enable** 1: transmission enable for the message in the high-priority PTB 0: transmission for the PTB If TPE is set, the message from the PTB will be transmitted at the next possible transmit position. A started transmission from the STB will be completed before, but pending new messages are delayed until the PTB message has been transmitted. TPE stays set until the message has been transmitted successfully or it is aborted using TPA. The host controller can set TPE to 1 but cannot reset it to 0. This would only be possible using TPA and aborting the message. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0
11 TPA	**Transmit Primary Abort** 1: Aborts a transmission from PTB which has been requested by TPE=1 but not started yet. (The data bytes of the message remains in the PTB.) 0: no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TPA automatically de-asserts TPE. The host controller can set TPA to 1 but cannot reset it to 0. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1. TPA should not be set simultaneously with TPE.
10 TSONE	**Transmit Secondary One frame** 1: Transmission enable of one in the STB. In FIFO mode this is the oldest message and in priority mode this is the one with the highest priority. TSONE in priority mode is difficult to handle, because it is not always clear which message will be transmitted if new messages are written to the STB meanwhile. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0: No transmission for the STB. TSONE stays set until the message has been transmitted successfully or it is aborted using TSA. The host controller can set TSONE to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the message. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM = 1 and LBME=0).

Bits	Description
9 TSALL	**Transmit Secondary All frames** 1: Transmission enable of all messages in the STB. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0: No transmission for the STB. TSALL stays set until all messages have been transmitted successfully or they are aborted using TSA. The host controller can set TSALL to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the messages. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0).
8 TSA	**Transmit Secondary Abort** 1: Aborts a transmission from STB which has been requested but not started yet. For a TSONE transmission, only one frame is aborted while for a TSALL Transmission, all frames are aborted. One or all message slots will be released which updates TSSTAT. All aborted messages are lost because they are not accessible any more. If in priority mode a TSONE transmission is aborted, then it is not clear which frame will be aborted if new frames are written to the STB meanwhile. 0: no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TSA, automatically de-asserts TSONE or TSALL respectively. The host controller can set TSA to 1 but cannot reset it to 0. The bit will be reset to the hardware reset value if RESET=1. TSA should not be set simultaneously with TSONE or TSALL.
7 RESET	**Reset request bit** 1: the host controller performs a local reset of CAN-CTRL. 0: no local reset of CAN-CTRL Some register (e.g for node configuration) can only be modified if RESET=1. Bit RESET = 1 will forces several components to a reset state. **Note that a CAN node will participate in CAN communication after RESET is switched to 0 after 11 CAN bit times. This delay is required by the CAN standard (bus idle time).** If RESET is set to 1 and immediately set to 0, then it takes some time until RESET can be read as 0 and becomes inactive. The reason is clock domain crossing from host to CAN clock domain. RESET is held active as long as needed depending on the relation between host and CAN clock.

Bits	Description
6 LBME	**Loop Back Mode, External** 0: disabled 1: enabled LBME should not be enabled while a transmission is active.
5 LBMI	**Loop Back Mode, Internal** 0: disabled 1: enabled LBMI should not be enabled while a transmission is active.
4 TPSS	**Transmission Primary Single Shot mode for PTB** 0: disabled 1 : enabled
3 TSSS	**Transmission Secondary Single Shot mode for STB** 0: disabled 1 : enabled
2 RACTIVE	**Reception Active (Receive Status bit)** 1: The controller is currently receiving a frame. 0 : No receive activity.
1 TACTIVE	**Transmission Active (Transmit Status bit)** 1: The controller is currently transmitting a frame. 0 : No transmit activity.
0 BUSOFF	**Bus Off (Bus Status bit)** 1: The controller status is “bus off”. 0: The controller status is “bus on”. Writing a 1 will clear TECNT and RECNT and exit BUSOFF when in BUSOFF status. This should be done only for debugging.

7.4.3 CAN_CTRL1

Table 7-16 CAN_CTRL1 register

CAN_CTRL1 CAN control register 1 **RESET: 0x1B00007E**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	AFWL				EWL				EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
Type	RW				RW				R	R	RW	W1C	RW	W1C	RW	W1C
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
Type	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RW	RW	RW	RW	RW	RW	RW	R
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0

Bits	Description
31: 28 AFWL(3: 0)	**receive buffer Almost Full Warning Limit** AFWL defines the internal warning limit AFWL_i with ^{n}RB being the number of available RB slots. AFWL_i is compared to the number of filled RB slots and triggers RAFIF if equal. The valid range of AFWL_i : 1…nRB. AFWL_i = 0 is meaningless and automatically treated as 0x1. (Note that AFWL is meant in this rule and not AFWL_i.) RB AFWL_i > n_{RB} is meaningless and automatically treated as n_{RB}. AFWL_i = n_{RB} is a valid value, but note that RFIF also exists.
27: 24 EWL(3: 0)	**Programmable Error Warning Limit** = (EWL+1)*8. Possible Limit values: 8, 16, …128. The value of EWL controls EIF.
23 EWARN	**Error Warning limit reached** 1: One of the error counters RECNT or TECNT is equal or bigger than EWL 0 : The values in both counters are less than EWL.
22 EPASS	**Error Passive mode active**

Bits	Description
	0: not active (node is error active) 1 : active (node is error passive)
21 EPIE	**Error Passive Interrupt Enable**
20 EPIF	**Error Passive Interrupt Flag.** EPIF will be activated if the error status changes from error active to error passive or vice versa and if this interrupt is enabled. Set the bit 1 to clean the flag.
19 ALIE	**Arbitration Lost Interrupt Enable**
18 ALIF	**Arbitration Lost Interrupt Flag** Set the bit 1 to clean the flag.
17 BEIE	**Bus Error Interrupt Enable**
16 BEIF	**Bus Error Interrupt Flag** The bus error type corresponds to KOER. Set the bit 1 to clean the flag.
15 RIF	**Receive Interrupt Flag** 1: Data or a remote frame has been received and is available in the receive buffer. 0: No frame has been received. Set the bit 1 to clean the flag.
14 ROIF	**RB Overrun Interrupt Flag** 1: At least one received message has been overwritten in the RB. 0: No RB overwritten. In case of an overrun both ROIF and RFIF will be set. Set the bit 1 to clean the flag.
13 RFIF	**RB Full Interrupt Flag** 1: All RBs are full. If no RB will be released until the next valid message is received, the oldest message will be lost. 0: The RB FIFO is not full. Set the bit 1 to clean the flag.
12 RAFIF	**RB Almost Full Interrupt Flag** 1: number of filled RB slots ≥ AFWL 0: number of filled RB slots < AFWL Set the bit 1 to clean the flag.
11	**Transmission Primary Interrupt Flag**

Bits	Description
TPIF	
	1: The requested transmission of the PTB has been successfully completed. 0: No transmission of the PTB has been completed Set the bit 1 to clean the flag.
10 TSIF	**Transmission Secondary Interrupt Flag** 1: The requested transmission of the STB has been successfully completed 0: No transmission of the STB has been completed successfully. Set the bit 1 to clean the flag.
9 EIF	**Error Interrupt Flag** 1: The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction. 0 : There has been no change.
8 AIF	**Abort Interrupt Flag** 1: After setting TPA or TSA the appropriated message(s) have been aborted. It is recommended to not set both TPA and TSA simultaneously because both source AIF. 0: No abort has been executed. The AIF does not have an associated enable register. Set the bit 1 to clean the flag.
7 RIE	**Receive Interrupt Enable** 0: Disabled 1 : Enabled
6 ROIE	**RB Overrun Interrupt Enable** 0: Disabled 1: Enabled
5 RFIE	**RB Full Interrupt Enable** 0: Disabled 1: Enabled
4 RAFIE	**RB Almost Full Interrupt Enable** 0: Disabled 1: Enabled
3 TPIE	**Transmission Primary Interrupt Enable** 0: Disabled 1: Enabled
2 TSIE	**Transmission Secondary Interrupt Enable**

Bits	Description
	0: Disabled 1: Enabled
1 EIE	**Error Interrupt Enable** 0: Disabled 1: Enabled, the interrupt flag corresponds to EIF.
0 TSFF	**Transmit Secondary Buffer Full Flag** 1: The STB is filled with the maximal number of messages. 0: The STB is not filled with the maximal number of messages

7.4.4 CAN_SBITRATE

Table 7-17 CAN_SBITRATE register

CAN_SBITRATE Low speed CAN baudrate configuration RESET: 0x01020203

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	S_PRESC									S_SJW						
Type	RW									RW						
Reset	0x01									0x02						
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		S_SEG2							S_SEG1							
Type		RW							RW							
Reset		0x02							0x03							

Bits	Description
31: 24 S_PRESC	**Prescaler (slow speed)** The prescaler divides the CAN clock to get the time quanta clock tq_clk. Valid range S_PRESC=[0x00, 0xff] results in divider values 1 to 256.
22: 16 S_SJW	**Synchronization Jump Width (slow speed)** The Synchronization Jump Width t_{SJW} =(S_SJW+1).TQ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.
14: 8 S_Seg_2	**Bit Timing Segment 2 (slow speed)** Time t_{Seg_2} = (S_Seg_2 +1).TQ after the sample point.
7: 0 S_Seg_1	**Bit Timing Segment 1 (slow speed)** The sample point will be set to t_{Seg_1}= (S_Seg_1+ 2).TQ after start of bit time.

7.4.5 CAN_FBITRATE

Table 7-18 CAN_FBITRATE register

CAN_FBITRATE **Fast speed CAN baudrate configuration** **RESET: 0x01020203**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	F_PRESC												F_SJW			
Type	RW												RW			
Reset	0x01												0x02			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					F_SEG2							F_SEG1				
Type					RW							RW				
Reset					0x02							0x03				

Bits	Description
31: 24 F_PRESC	**Prescaler (high speed)** The prescaler divides the CAN clock to get the time quanta clock tq_clk. Valid range F_PRESC=[0x00, 0xff] results in divider values 1 to 256.
19: 16 F_SJW	**Synchronization Jump Width (high speed)** The Synchronization Jump Width t_{SJW} =(F_SJW+1).TQ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.
11: 8 F_Seg_2	**Bit Timing Segment 2 (high speed)** Time tSeg_2 =(F_Seg_2 + 1). TQ after the sample point.
4: 0 F_Seg_1	**Bit Timing Segment 1 (high speed)** The sample point will be set to t_{Seg_1} = (F_Seg_1+ 2).TQ after start of bit time.

7.4.6 CAN_ERRINFO

Table 7-19 CAN_ERRINFO register

CAN_ERRINFO CAN error type and error counter register **RESET: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TECNT								RECNT							
Type	R								R							
Reset	0								0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TDCEN	SSPOFF							KOER			ALC				
Type	RW	RW							R			R				
Reset	0	0							0			0				

Bits	Description
31: 24 TECNT	**Transmit Error Count (number of errors during transmission)** TECNT is incremented and decremented as defined in the CAN specification. TECNT can overflow. See chapter 7.3.2 for more details about TECNT and the "bus off" state.
23: 16 RECNT	**Receive Error Count (number of errors during reception)** RECNT is incremented and decremented as defined in the CAN specification. RECNT cannot overflow. RECNT signals 0xff = 255 as maximum value. See chapter 7.3.2 for more details about RECNT and the "bus off" state.
15 TDCEN	**Transmitter Delay Compensation Enable** TDC will be activated during the data phase of a CAN FD frame if BRS is active if TDCEN=1.
14: 8 SSPOFF	**Secondary Sample Point Offset** The transmitter delay plus SSPOFF defines the time of the secondary sample point for TDC. SSPOFF is given as a number of TQ.
7: 5 KOER	**Kind of Error (Error code)** 000 : no error 001: BIT ERROR 010: FORM ERROR 011: STUFF ERROR 100: ACKNOWLEDGEMENT ERROR 101: CRC ERROR 110: OTHER ERROR (dominant bits after own error flag, received active Error Flag too long, dominant bit during Passive-Error-Flag after ACK error). 111: not used KOER is updated with each new error. Therefore it stays untouched when frames are successfully transmitted or received.
4: 0 ALC	**Arbitration Lost Capture** (bit position in the frame where the arbitration has been lost)

7.4.7 CAN_ACFCTRL

Table 7-20 CAN_ACFCRTL register

CAN_ACFCTRL filter control register **RESET: 0x00010000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ACFENx															
Type	RW															
Reset	0															1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							TIME POS	TIMEN			SEL MASK		ACFADR			
Type							RW	RW			RW		RW			
Reset							0	0			0		0			

Bits	Description
31: 16 ACFENx	**Acceptance filter Enable (x=0-15)** 1: acceptance filter enabled 0: acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Only filter number 0 is enabled by default after hardware reset. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches. To accept all messages, one filter x has to be enabled by setting AE_x=1, AMASK_x=0xff and ACODE_x=0x00. This is the default configuration after hardware reset for filter x=0 while all other filters are disabled.
9 TIMEPOS	**Time-stamping Position** 0: SOF 1: EOF TIEMPOS can only be changed when TIMEEN = 0, and TIMEPOS can also be modified and set TIMEEN=1
8 TIMEEN	**Time-stamping Enable** 0: Disabled 1: Enabled
5 SELMASK	**Select acceptance Mask** 0: Registers ACF_x point to acceptance code 1: Registers ACF_x point to acceptance mask. ACFADR selects one specific acceptance filter.

Bits	Description
3: 0 ACFADR	**acceptance filter address** ACFADR points to a specific acceptance filter 0-15. The selected filter is accessible using the registers ACF_x. Bit SELMASK selects between acceptance code and mask for the selected acceptance filter.

7.4.8 CAN_ACF(ACODE)

Table 7-21 CAN_ACF(ACODE) register

CAN_ACF Acceptance CODE(ACODE) register RESET: 0x0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name				ACODE												
Type				RW												
Reset				0												
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ACODE															
Type	RW															
Reset	0															

Bits	Description
28: 0 ACODE	**Acceptance CODE, it is needed to set SELMASK to 0.** 0/1: ACC bit value to compare with ID bit of the received message ACODE_x (10:0) will be used for standard frames. ACODE_x (28:0) will be used for extended frames. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

7.4.9 CAN_ACF(AMASK)

Table 7-22 CAN_ACF(AMASK) register

CAN_ACF Acceptance MASK(AMASK) register RESET: 0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		AIDEE	AIDE	AMASK												
Type		RW	RW	RW												
Reset		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AMASK															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Description
30 AIDEE	**Acceptance mask IDE bit check enable, SELMASK is needed to set as 1.** 1: acceptance filter accepts either standard or extended as defined by AIDE 0: acceptance filter accepts both standard or extended frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.
29 AIDE	**Acceptance mask IDE bit value, SELMASK is needed to set as 1.** If AIDEE=1 then: 1: acceptance filter accepts only extended frames 0: acceptance filter accepts only standard frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.
28: 0 AMASK	**Acceptance MASK, SELMASK is needed to set as 1.** 1: acceptance check for these bits of receive identifier disabled 0: acceptance check for these bits of receive identifier enable AMASK_x (10:0) will be used for standard frames. AMASK_x (28:0) will be used for extended frames. Disabled bits result in accepting the message. Therefore, the default configuration after reset for filter 0 accepts all messages. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

7.4.10 CAN_VERSION

Table 7-23 CAN_VERSION register

CAN_VERSION **CAN core version register** **RESET: 0x0000708**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VERSION															
Type	R															
Reset	0x0708															

Bits	Description
15:0 VERSION	**CAN-CTRL Version** Version of CAN-CTRL, given as decimal value. Example: VERSION=0708 represents that the major version is 07 and the minor version is 08.

8 UART

8.1 Introduction

The UART (Universal Asynchronous Receiver/Transmitter) is a basic peripheral for serial communication. It operates to achieve many functions mainly by transmitter and receiver. In Table 8-1, main functions are classified and listed.

Table 8-1 UART function classification and configuration

Functions	LINEN	RS485EN	ILEN
BASIC UART	0	0	0
RS485	0	1	0
LIN	1	0	0

Note

1. Only UART0 and UART1 support soft LIN.
2. Hardware flow control function must be disabled when it is under RS485 mode.
3. LIN mode supports 8-bit data format and 16 times oversample. Meanwhile, if auto baud rate function is enabled (LABAUDEN=1), the synchronous field data (0x55) will be discarded.
4. DMA function must operate with FIFO enable.

8.2 Features

- Full duplex, standard NRZ format.
- Programmable baud rate (16-bit divisor).
  - The baud rate range is 600 bps to 3 Mbps, baud rate error is less than 1%
- Interrupt or polling is used to check these status flags.
  - Transmitter data register empty, transmission complete.
  - Receiver data register full.
  - Receive overflow error, frame error, parity error.
  - Idle line detect.
  - Break detect supporting LIN and optional 10/11 bits break character detection.
  - Active edge detects for wake up from stop mode.
- Supports DMA.

- Programmable 8- or 9-bit data length, 1/2 stop bit, hardware automatically generates parity bit.
- Selectable transmitter output and receiver input polarity.
- Supports hardware flow control.
- 13-28 bits break character generation.
- Supports RS485 with direction auto-control.

8.3 Block diagram

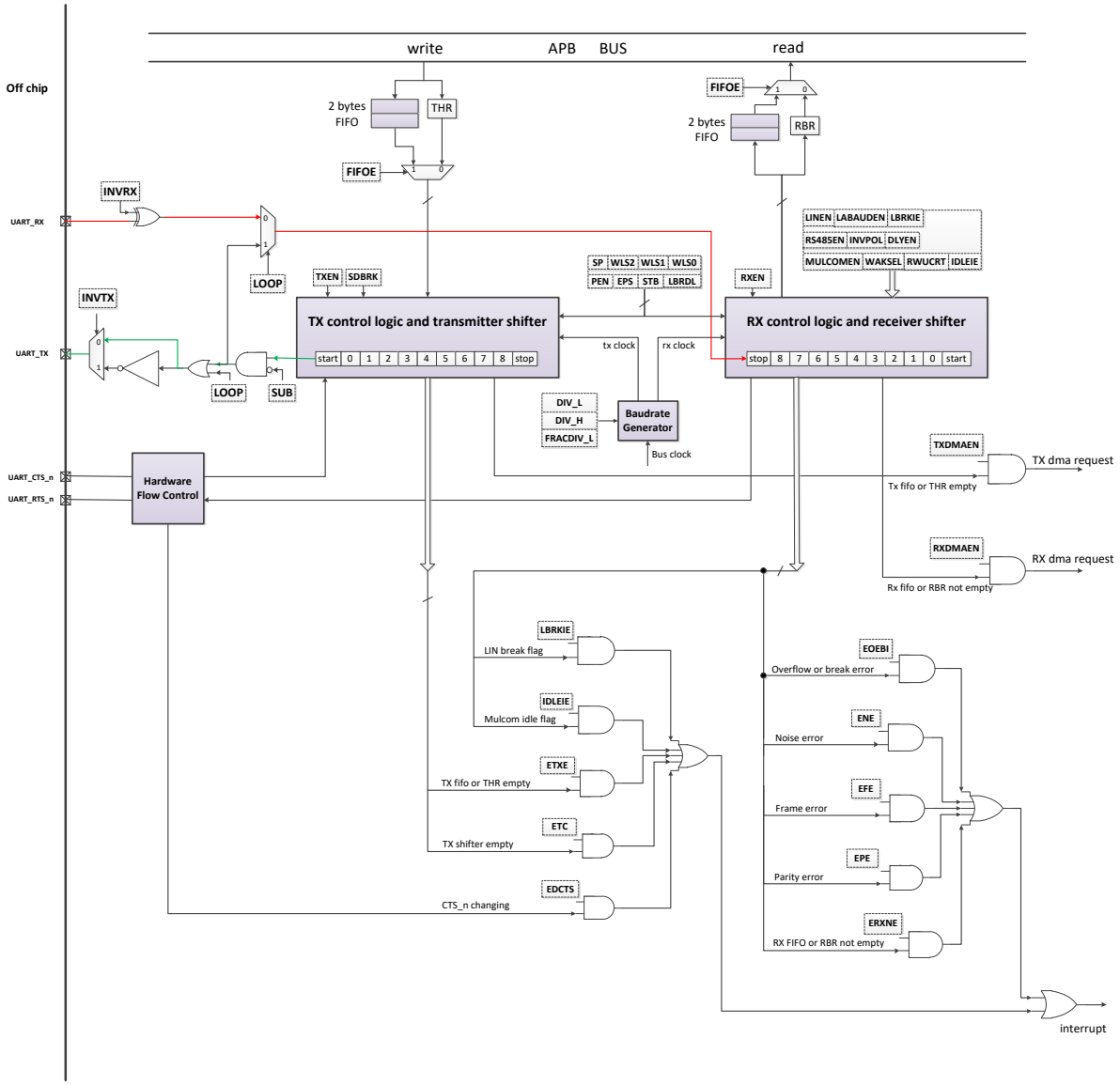

Figure 8-1 UART block diagram

8.4 Function description

Basic UART is used to transmit and receive the serial data bit by bit. In Figure 8-2 and Figure 8-3, it illustrates the full data bits including start bit, data bits, parity bit, stop bits and guard time. But the bit6, bit7, bit8, bit9, parity, stop2 bit and guard time can be configured by user described in UART_LCR0 and UART_LCR1 in detail. One bit is corresponding to one-bit time controlled by baud rate.

There exist many statuses for UART transmitting and receiving. User had better know when the statuses are generated in the transmitting or receiving process. In this way, user can use the UART function better. The status bits THRE and TXC just occur in transmitting process illustrated in Figure 8-2. During the initialization condition after global reset or power on, THRE and TXC turn to 1 just after TXEN is set to 1. But in the process of transmitting, THRE turns to 1 just after start bit and TXC turns to 1 just after the last bit, such as the guard time if GUARDEN=1.

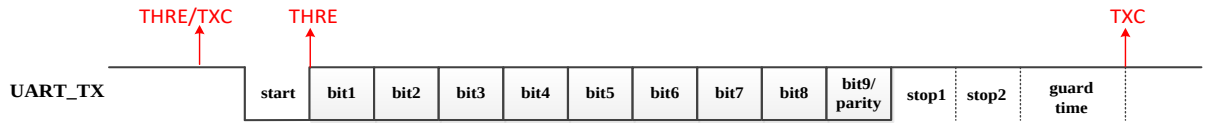

Figure 8-2 UART transmitter flow

The status bit DR, OE, PE, FE, BI and NE are set to 1 just after the stop1 bit if related events occur during receiving process as illustrated in Figure 8-3.

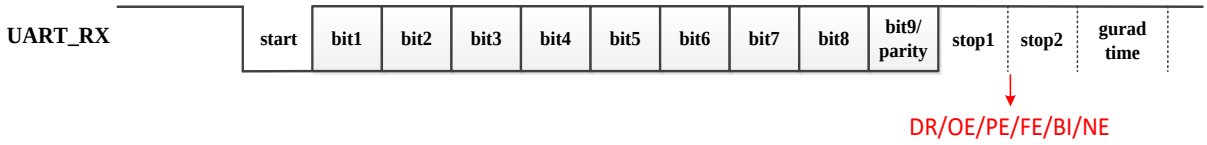

Figure 8-3 UART receiver flow

It is important to note that the statuses of PE, FE and NE just aim at the current receiving data byte and will be cleared automatically just when the next data is received completely. For other statuses, they hold the value for all the time until they are cleared by reading or writing 1.

8.4.1 Input & Output timing

Table 8-2 UART input & output timing

Pin name	Corresponding signal	Width	I/O	Pull up or not	Timing limitation
UARTx_TX	uart_tx	1	O	no	None
UARTx_RX	uart_rx	1	I	no	None
UARTx_RTS	uart_rts_n	1	O	no	None
UARTx_CTS	uart_cts_n	1	1	no	None

Note

x=0~2. Only UART0 has the full hardware flow control and its four signals (UART0_TX, UART0_RX, UART0_RTS, UART0_CTS) can all be found in PIN.

8.4.2 Noise detection

For NE status, it is generated during receiving process when the noise exists in UART_RX signal. In order to detect the noise, UART_RX is sampled three times at the middle position as illustrated in Figure 8-4.

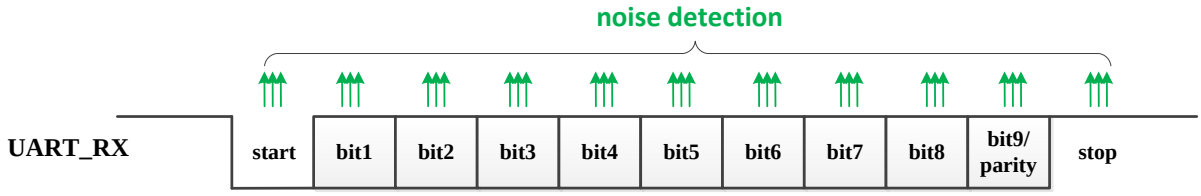

Figure 8-4 UART noise detection

To explain clearly and conveniently, the value of three times sample is called SM1, SM2 and SM3 respectively. If there are more than two '1' among SM1, SM2 and SM3 for start bit, the start bit is invalid and the receiver is reset to receive again. Other than this case for start bit, if the value of SM1, SM2 and SM3 are different for each other, the noise is detected with the NE status setting to 1.

8.4.3 Baud rate description

UART Baud rate accuracy is determined by many aspects including UART clock, oversample time and so on. So, some specific and too high baud rate is not realized or realized with large error. Table 8-3 and Table 8-4 describe the typical baud rate configurations at different system clocks and corresponding errors.

Table 8-3 Typical baud rate and error@bclock=48MHz

No.	Theoretical value (Kbps)	Practical value (bps)	DIV_MAN [15:0]	DIV_FRAC [4:0]	Oversample times	Error
1	2.4	2399.98	1250	0	16	-0.001%
2	9.6	9599.69	312	16	16	-0.003%
3	19.2	19202.22	156	8	16	0.006%
4	57.6	57603.69	52	3	16	0.006%
5	115.2	115207.38	26	1	16	0.006%
6	230.4	230414.75	13	1	16	0.006%
7	460.8	460829.50	6	16	16	0.006%

8	921.6	917431.19	3	8	16	-0.452%
9	1843.2	1834862.38	3	8	8	-0.452%
10	4200	4166666.75	1	14	8	-0.794%

Table 8-4 Typical baud rate and error @blocky=24MHz

No.	Theoretical Value (Kbps)	Practical Value (bps)	DIV_MANTI [15:0]	DIV_FRAC [4:0]	Oversample times	Error
1	2.4	2400	625	0	16	0
2	9.6	9600	156	8	16	0
3	19.2	19200	78	4	16	0
4	57.6	57600	26	1	16	0
5	115.2	115200	13	1	16	0
6	230.4	230769.23	6	16	16	0.16%
7	460.8	461538.47	3	8	16	0.16%
8	921.6	923076.94	1	20	16	0.16%
9	1843.2	-	-	-	-	-
10	4200.0	-	-	-	-	-

8.4.4 Hardware flow control function

UART hardware flow controls the communication between two UART devices by RTS_n and CTS_n in order to reduce the CPU workload. In this way, the two communication sides can handle the data one by one leisurely. The practical application connection is described in Figure 8-5.

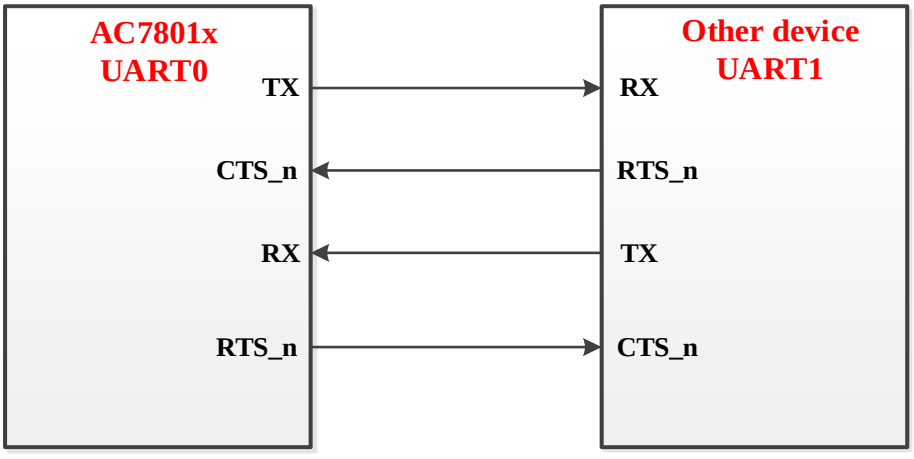

Figure 8-5 Hardware flow control connection

In Figure 8-5 , the signal RTS_n of UART0 can inform UART1 not to transmit data because UART0 RX data register or FIFO has been full. When UART0 RX data register or FIFO turns to not full by reading, the UART0 RTS_n turns to low automatically. Then UART1 can transmit data by detecting the UART1 CTS_n low. In the similar way, the UART1 RTS_n and UART0 CTS_n are used to control the UART0 transmission.

In particular, if the UART1 CTS_n turns to high during the UART1 is transmitting data, the current data will be transmitted completely. Therefore, user should usually check the CTS or CTS_n status to use the hardware flow control function and other status bits.

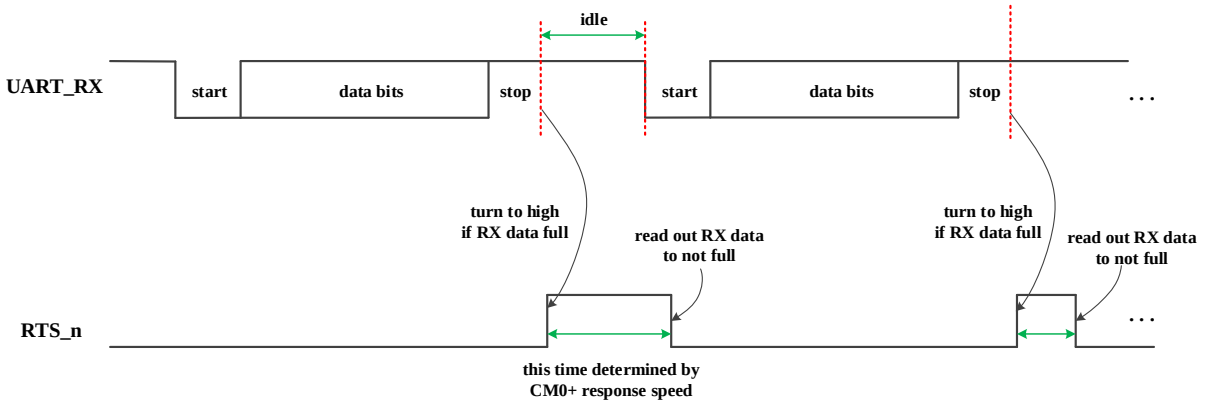

Figure 8-6 Hardware flow control principle

8.4.5 RS485 function

Compared with basic UART function, RS485 function generates an auto direction control signal UART_RTS, which is default low for receiving the data and high for transmitting data illustrated in Figure 8-7 and Figure 8-8. Because of the half-duplex, RS485 can implement one of the transmitting or receiving at the same time. In the Figure 8-7, there are two delays: delay1 is used for pulling up the UART_RTS signal before practical data transmission and guard time is used for pulling down the UART_RTS after the data bit transmission has absolutely been finished. Practical PCB routing delay may lead to UART_RTS signal turning to high later than UART_TX so that the first data bits may be damaged. Therefore, the delays help to guarantee the UART_RTS is high during the whole transmission. After transmission is finished, UART_RTS will go to low automatically to let the UART in receiving state.

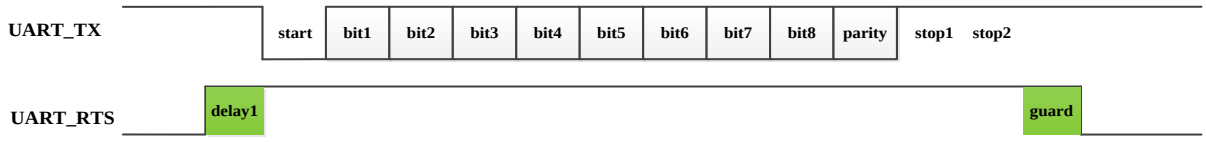

Figure 8-7 Single byte data transmission

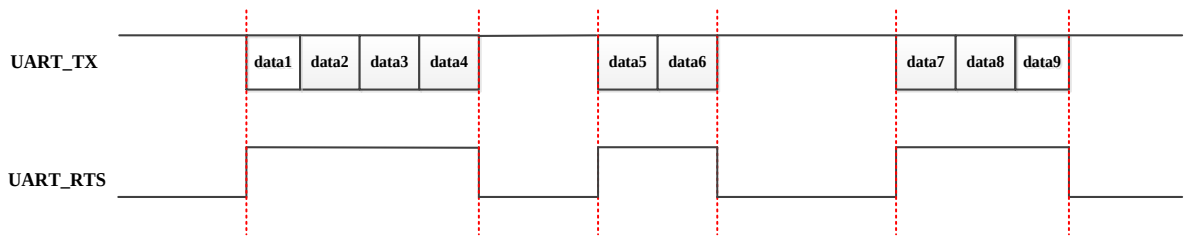

Figure 8-8 Multiple bytes data transmission

Figure 8-9 describes a typical case of the user application connection. UART_RTS acts as a direction control signal to MAX485. With this connection method, the default level of UART_RTS is low so that MAX485 is under default receiving condition. When UART wants to transmit data, the signal UART_RTS is set to high by hardware.

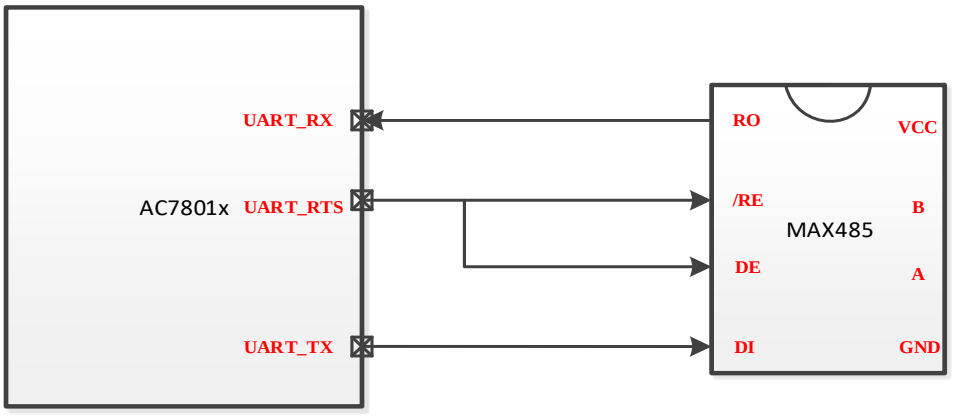

Figure 8-9 Practical circuit connection

8.4.6 LIN function

The UART LIN is just a soft LIN which transmits the break field, synchronous field and data respectively controlled by user as Figure 8-10. User can learn more details in the newest LIN protocol and Figure 8-10 just illustrates a basic frame condition. User should pay more attention to the UART_LINCR register in addition to the basic UART registers. In UART module, there exists a hardware unit for LIN detection and it is enabled when LINEN is configured to 1. It should be noted that only UART0 and UART1 support LIN function.

Firstly, receiving process are introduced based on the serial data stream appears on the UART_RX as Figure 8-10.

1. When UART receives 10 (LBRKDL=0) or 11 (LBRKDL=1) bit zero, UART LIN detection logics treat it as a valid break field for LIN frame and the LIN break flag FBRK is set to 1 by hardware to indicate a valid LIN break field has occurred. It should be noted that the LIN break is not a data and not stored into the RX data register or FIFO.
2. After delimiter bits period in Figure 8-10, synchronous field namely 0x55 acts as a normal data for receiving. If LABAUDEN is configured to 1, auto baud rate detection is in operation during synchronous field and the auto baud rate detection operation is finished just after the fifth rising edge illustrated in Figure 8-10. But the data 0x55 will not be stored into RX data register or FIFO. If LABAUDEN is configured to 0, auto baud rate detection is not performed and the data 0x55 is stored into the RX data register or FIFO.
3. After synchronous field, the data streams are the useful data for user and received by UART without difference.

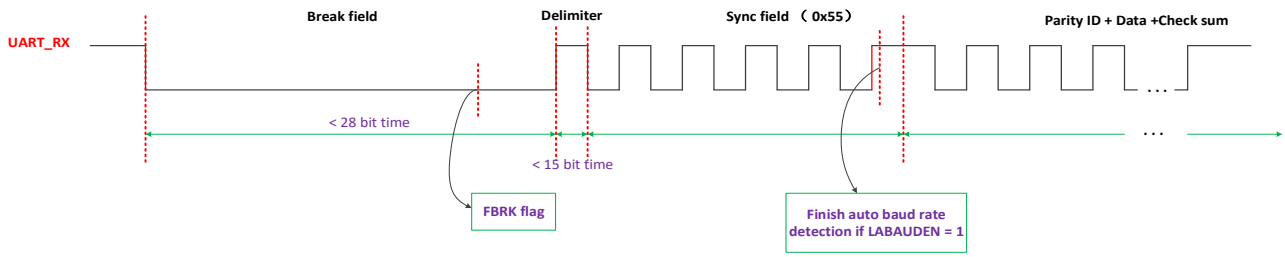

Figure 8-10 LIN frame flow

To avoid the module halting when exception condition occurs, time out mechanism is brought in, as illustrated in Figure 8-10.

1. When the break field received by the LIN slave exceeds 28 bits, the hardware will set the BRKWAK bit of the register UART_LCR1. If the BRKWAKIE bit of the register UART_LINCR is enabled, a wake-up interrupt will be generated.
2. More than 15 bits time for delimiter can lead the module to reset the receiver and LIN detection logics.
3. If the synchronization segment fails, the hardware sets the SYNERR bit of UART_LCR1. If the SYNERRIE bit of UART_LINCR is enabled, when the synchronization fails, the hardware generates a synchronization segment error interrupt.

Then, introduction for transmitting UART–LIN protocol frame is described in this paragraph.

1. As software LIN, how to transmit the break field is the key procedure. When user wants to transmit the break field, user should check the UART_LCR0[THRE] status. The length of the break field sent depends on the configuration of the register UART_BRKLGH.
2. The value of register UART_BRKLGH is 0-15, corresponding to the break field of the 13-28 bit. At this time, if the value of UART_LCR0[THRE] is 1, the user can send a break field by writing 1 to the register UART_LINCR[SBRK]. To point out, when UART_LINCR[SBRK] has been written to 1, the break field will be transmitted just after the current data has been transmitted completely or be transmitted immediately when the UART is in idle state. The UART_LINCR[SBRK] will be cleared automatically by hardware.
3. And then the synchronous field(0x55) and other following data can be written to the THR data register and sent as normal data transmitting.

8.4.7 Two power mode

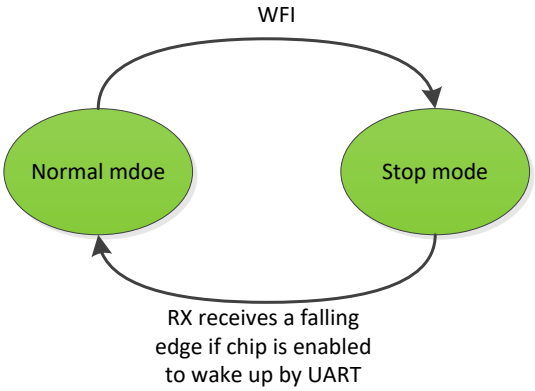

Figure 8-11 Chip normal mode and stop mode condition

The UART status in two power modes designed for the chip is introduced here. As the Figure 8-11 illustrates, user can execute the WFI instruction to make the chip go into the stop mode, in which the chip power consumption will be much less obviously and the UART module will power down and be reset by default. The register configuration of the UART module is maintained and does not need to be reconfigured after awakening.

In stop mode, the chip can wake up to normal mode by UART receiving a falling edge if the chip is enabled to wake up by UART. Because of the time required for the wake-up flow, UART can normally receive data just after the sum time of 5 ms. In details, TX1 can send 0xFF to act as a falling edge and actually other data is also ok. Specially, data will be lost in RX2 if TX1 send some data to RX2 during the wake up flow.

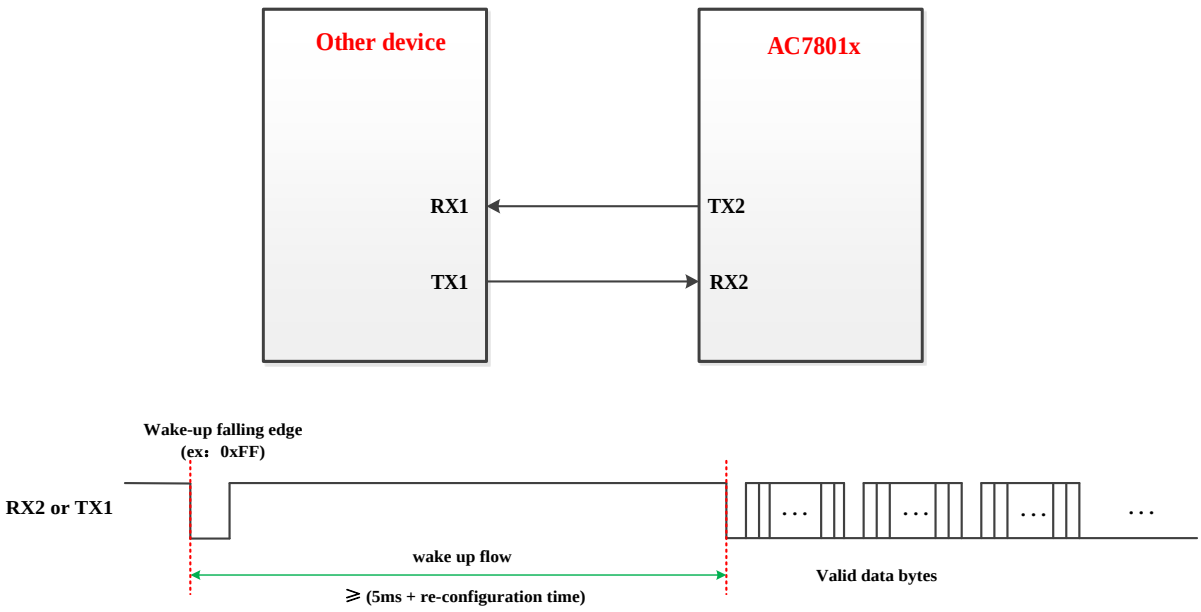

Figure 8-12 Typical flow for waking up the chip by UART

8.5 Application notes

8.5.1 Baud rate configuration notes

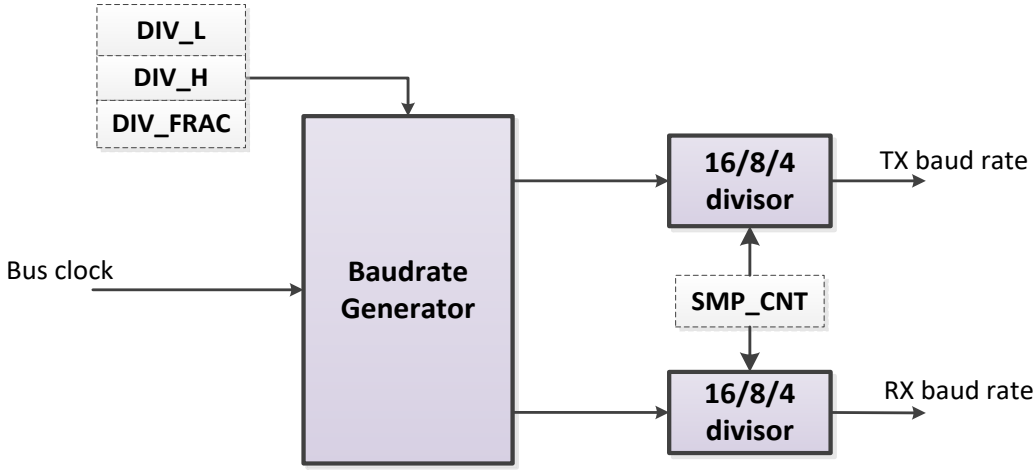

Figure 8-13 Diagram for baud rate generator

As illustrated in the Figure 8-13, the baud rate related configuration registers are as follows:

UART_SMP_CNT/UART_DIV_H/UART_DIV_L/UART_DIV_FRAC. And the detailed information is described in register map. For configuration, the formula below is used for baud rate configuration.

$$Baudrate = \frac{f_{clk}}{DIV * SMP\_CNT}$$

Note: DIV = {UART_DIV_H, UART_DIV_L} ; SMP_CNT = UART_SMP_CNT.

For example:

If user wants to get baud rate 230400 bps at 8 times sample mode with 24MHz bus frequency, so we can get the DIV configuration as follows. So, **UART_DIV_L=13, UART_DIV_H=0, UART_DIV_FRAC=32 * 0.0208=1.**

$$\mathrm{DIV} = \frac{24000000}{\mathrm{Baudrate}*\mathrm{SMP\ CNT}} = \frac{24000000}{230400*8} = 13.0208$$

Combined with the example above, UART_DIV_FRAC[4:0] can be explained clearly. In the expression above, DIV is not an integer. If user removes the fractional part, the accuracy of baud rate will decrease. Especially in high baud rate condition, the discard of DIV fractional part may reduce the accuracy to a low level so that the normal data transmission will make a mistake.

In order to keep high accuracy, UART_DIV_FRAC[4:0] is configured to represent the DIV fractional part. Because of 5 bits width, UART_DIV_FRAC ranges from 0 to 31. So, 32 multiplied by the DIV fractional part generates the configuration value for UART_DIV_FRAC[4:0].

8.5.2 UART configure notes

Configuration steps:

(1) TX/RX data storage mode: UART_FCR.

(2) Baud rate: UART_DIV_L/UART_DIV_H/ UART_DIV_FRAC/UART_SMP_CNT

(3) DMA: UART_DMA_EN

Note

DMA function must operate with FIFO.

---

(4) Data format: UARTn_LCR0/UARTn_LCR1

Note

SUB bit must be taken care of.

---

(5) Function configuration: UART_RS485CR/UART_LINCR/UART_IDLE/ UART_CNRT and so on.

Note

This step is an option for different function.

---

(6) Interrupt enable: UART_IER

(7) Transmitter and receiver enable: UART_LCR1[TXEN] / UART_LCR1[RXEN].

(8) Transmit or receive data: UART_THR/UART_RBR

Note

This step is in normal transmitting or receiving data process actually.

Notes:

1. For LIN function, data format must be configured as 8 bit with no parity check with 16 times oversample.
2. For LIN function, the sync field data(0x55) will be received and stored into FIFO or RX register when LABAUDEN=0. The sync field data(0x55) will be received and not stored into FIFO or RX register when LABAUDEN=1.
3. For RS485 function, RTS_n PIN is used as the transmitting or receiving direction controlling signal.

8.6 Register Definition

Table 8-5 UART register mapping

UART0 base addr: 0x40018000
UART1 base addr: 0x40019000
UART2 base addr: 0x4001A000

Address	Name	Width (in bit)	Description
UARTx base addr+0x00	UART_RBR/THR	32	TX holding register /RX buffer register
UARTx base addr +0x04	UART_DIV_L	32	Divisor Low 8 bits
UARTx base addr +0x08	UART_DIV_H	32	Divisor High 8 bits
UARTx base addr +0x0C	UART_LCR0	32	UART supplementary control register 0
UARTx base addr +0x10	UART_LCR1	32	UART supplementary control register 1
UARTx base addr +0x14	UART_FCR	32	FIFO control register
UARTx base addr +0x18	UART_EFR	32	Hardware flow enable register
UARTx base addr +0x1C	UART_IER	32	Interrupt enable register
UARTx base addr +0x20	UART_LSR0	32	Status register0
UARTx base addr +0x24	UART_LSR1	32	Status register 1
UARTx base addr +0x28	UART_SMP_CNT	32	UART sample counter register
UARTx base addr +0x34	UART_GUARD	32	Guard time added register
UARTx base addr +0x38	Reserved	32	
UARTx base addr +0x3C	UART_SLEEP_EN	32	Sleep enable register
UARTx base addr +0x40	UART_DMA_EN	32	DMA enable register
UARTx base addr +0x44	UART_DIV_FRAC	32	Fractional divider register
UARTx base addr +0x48	Reserved	32	
UARTx base addr +0x4C	UART_RS485CR	32	RS485 control register
UARTx base addr +0x54	UART_CNTR	32	Delay time for RS485
UARTx base addr +0x58	UART_IDLE	32	Idle interrupt enable register
UARTx base addr +0x5C	UART_LINCR	32	Software LIN control register **Note: UART2 does not support LIN, no such register.**

Address	Name	Width (in bit)	Description
UARTx base addr +0x60	UART_BRKLGH	32	Software LIN sync break length control register **Note: UART2 does not support LIN, no such register.**

8.6.1 UART_RBR/THR

Table 8-6 UART_RBR/THR register

UART_RBR/THR **RX/TX data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								RBR/THR								
Type								RW								
Reset								0								

Bits	Description
8: 0 RBR/THR	**RX/TX Data register** The received data can be read by accessing this register and the transmitting data can be written into this register. The data width is not more than 9 bits.

8.6.2 UART_DIV_L

Table 8-7 UART_DIV_L register

UART_DIV_L **Divisor Low 8 bits register** **Reset: 0x00000001**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DIV_L							
Type									RW							
Reset									1							

Bits	Description
7: 0 DIV_L	**Baud rate divisor** Divisor low 8 bits.

8.6.3 UART_DIV_H

Table 8-8 UART_DIV_H register

UART_DIV_H **Divisor High 8 bits register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DIV_H							
Type									RW							
Reset									0							

Bits	Description
7: 0 DIV_H	**Baud rate divisor** Divisor high 8 bits.

8.6.4 UART_LCR0

Table 8-9 UART_LCR0 register

UART_LCR0 **Control Register 0** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										SUB	SP	EPS	PEN	STB	WLS1_WLS0	
Type										RW	RW	RW	RW	RW	RW	
Reset										0	0	0	0	0	0	

Note

SUB bit configuration must be 0, otherwise tx transmit '0' at any time.

Bits	Description
6 SUB	**Set up Break** 0: No effect 1: SOUT signal is forced into the "0" state.
5 SP	**Stick parity** 0: No effect. 1: The parity bit is forced into a defined state, depending on the states of EPS and PEN If EPS = 1 & PEN = 1, the parity bit is set and checked = 0. If EPS = 0 & PEN = 1, the parity bit is set and checked = 1.
4 EPS	**Select even parity** 0: an odd number of ones is sent and checked. 1: an even number of ones is sent and checked.
3 PEN	**Enable parity** 0: The parity is neither transmitted nor checked. 1: The parity is transmitted and checked.
2 STB	**Number of STOP bits** 0: One STOP bit is always added. 1: Two STOP bits are added after each character is sent; unless the character length is 5 when 1 STOP bit is added.
1: 0 WLS1_WLS0	**Selects word length** 00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits **Note: if the word length is set to 9 bits, set WLS2 to 1.**

8.6.5 UART_LCR1

Table 8-10 UART_LCR1 register

UART_LCR1 **Control Register 1** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									INVTX	INVRX	WLS2	LOOP			TXEN	RXEN
Type									RW	RW	RW	RW			RW	RW
Reset									0	0	0	0			0	0

Bits	Description
7 INVTX	**Determine whether inverse the tx output, including idle, break, data bits, start bit, stop bit.** 0: don't inverse tx output 1: inverse tx output
6 INVRX	**Determine whether inverse the RX input, including idle, break, data bits, start bit, stop bit.** 0: don't inverse RX input 1: inverse RX input
5 WLS2	**Determine whether 9 bits data mode is available** 0: not available 1: available
4 LOOP	**LOOP** 0: for user normal use 1: control the uart into loop mode(can used for testing uart by itself)
1 TXEN	**UART Transmitter enable** 0: disable 1: enable
0 RXEN	**UART Receiver enable** 0: disable 1: enable

8.6.6 UART_FCR

Table 8-11 UART_FCR register

UART_FCR **FIFO Control register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																FIFOE
Type																RW
Reset																0

Bits	Description
0 FIFOE	**Enable FIFO** 0: Disable both RX and TX FIFOs 1: Enable both RX and TX FIFOs.

8.6.7 UART_EFR

Table 8-12 UART_EFR register

UART_EFR **Hardware flow enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									AUTO_CTS	AUTO_RTS						
Type									RW	RW						
Reset									0	0						

Note: AUTOCTS=1 represents enabling the Hardware flow function of pin CTS_n, so if AUTOCTS=1, **user must make the n_CTS pin be tied to a fixed level, such as the other MCU's or device's pin**. If AUTOCTS=0, user needn't to care about the CTS_n pin.

Bits	Description
7 AUTO_CTS	**Enable hardware transmission flow control** 0: Disable 1: Enable

Bits	Description
6 AUTO_RTS	**Enable hardware reception flow control** 0: Disable 1: Enable

8.6.8 UART_IER

Table 8-13 UART_IER register

UART_IER **Interrupt Enable Register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								ETXDF	EDCTS	EOEBI	ENE	EFE	EPE	ETC	ETXE	ERXNE
Type								RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset								0	0	0	0	0	0	0	0	0

Bits	Description
8 ETXDF	**Transmission register or transmit FIFO full interrupt enable** 0: disable 1: enable
7 EDCTS	**CTS_n changing interrupt enable** 0: disable 1: enable
6 EOEBI	**Interrupt enable of overflow error or break error** 0: disable 1: enable
5 ENE	**Interrupt enable of noise error** 0: disable 1: enable
4 EFE	**Interrupt enable of frame error** 0: disable 1: enable
3 EPE	**Interrupt enable of parity error** 0: disable 1: enable

Bits	Description
2 ETC	**Interrupt enable of transmitting completed** 0: disable 1: enable
1 ETXE	**Interrupt enable of transmitting data empty** 0: disable 1: enable **Note: fifoe=1 represents fifo empty;** **fifoe=0 represent data register empty.**
0 ERXNE	**Interrupt enable of receiving data not empty** 0: disable 1: enable **Note: fifoe=1 represents fifo not empty;** **fifoe=0 represent data register not empty.**

8.6.9 UART_LSR0

Table 8-14 UART_LSR0 register

UART_LSR0 **Line Status Register 0** **Reset: 0x00000020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								TXDF	NE	TC	THRE	BI	FE	PE	OE	DR
Type								R	W1C	R	R	W1C	W1C	W1C	W1C	R
Reset								0	0	0	1	0	0	0	0	0

Note

SUB NE/PE/FE error just aims at the current byte data. Meanwhile, OE/BI will exist until it is cleared.

Bits	Description
8 TXDF	**Transmission data register or transmit FIFO full flag** 0: transmission data register(fifoe=0) or transmission FIFO (fifoe=1) not full 1: transmission data register (fifoe=0) or transmission FIFO (fifoe=1) full

Bits	Description
	Note: This flag reflects the data and transmission status
7 NE	**Noise error flag** 0**:** No noise error**.** 1: Noise error has been occurred. **Note: write 1 to clear this flag to 0.**
6 TC	**The flag of Transmitting finished** 0: TX FIFO(fifoe=1) or TX register(fifoe=0) is not empty, or transmitter has not finished the data shifting. 1: TX FIFO(fifoe=1) or TX register(fifoe=0) is empty and transmitter has finished the data shifting. **Note: The default value at power-on is 0, and TXC will only work after TXEN is 1. Write data to TX FIFO(fifoe=1)/TX register(fifoe=0) to clear this flag to 0. For LIN function, set SBRK bit also can clear this flag to 0.**
5 THRE	**The empty flag of TX holding register or TX FIFO** 0: Reset whenever the contents of the TX FIFO are not empty, or whenever TX holding register is not empty (FIFOs are disabled). 1: Set whenever the contents of the TX FIFO are empty, or whenever TX holding register is empty (FIFOs are disabled). **Note: write data to TX FIFO(fifoe=1)/TX register(fifoe=0) to clear this flag to 0.**
4 BI	**Break error flag** 0: No break error 1: Break error has been occurred. If FIFOs are disabled, this bit is set whenever the SIN is held in the 0 state for more than one transmission time (START bit + DATA bits + PARITY + STOP bit). When a break occurs, only one zero character is loaded into FIFO or TX holding register. **Note: write 1 to clear this flag to 0.**
3 FE	**Framing error flag** 0: No framing error. 1: Framing error has been occurred. This bit will be set if the received data do not have a valid STOP bit. **Note: write 1 to clear this flag to 0.**
2 PE	**Parity error flag** 0: No parity error.

Bits	Description
	1: Parity error has been occurred. This bit will be set if the received data do not have a valid parity bit. **Note: write 1 to clear this flag to 0.**
1 OE	**Overflow error flag** 0: No RX overflow error. 1: RX overflow error has been occurred. If FIFOs are disabled, this bit will be set if the RX buffer is not read by the CPU before the new data from the RX shift register overwrites the previous contents. If FIFOs are enabled, an overflow error occurs when RX FIFO is full and the RX shift register becomes full. OE is set as soon as this happens. The character in the shift register is then overwritten, but not transferred to FIFO. **Note: write 1 to clear this flag to 0.**
0 DR	**Data ready flag** 0: Data not ready 1: Data ready. Set by the RX buffer becoming full or RX FIFO not empty (at least one byte being transferred into the FIFO). **Note: Read register UART_RBR/THR, or read all RX FIFO to clear this flag to 0.**

8.6.10 UART_LSR1

Table 8-15 UART_LSR1 register

UART_LSR1 **Line Status Register 1** **Reset: 0x000000E0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									RTS	CTS	UART_IDLE	BRKWAK	DCTS	FBRK	SYNERR	IDLE
Type									R	R	R	R	W1C	W1C	W1C	W1C
Reset									1	1	1	0	0	0	0	0

Bits	Description
7 RTS	**Hardware flow status – RTS** 0: Under hardware flow control function, it represents RX FIFO or RX register is already full. This signal can inform other device not transmitting data to the MCU. 1: Under hardware flow control function, it represents RX FIFO or RX register is not full.

Bits	Description
	Note: RTS is opposite to the signal of PIN RTS_n. It is not an interrupt source.
6 CTS	**Hardware flow status – CTS** 0: Under hardware flow control function, it represents RX FIFO or RX register of other device is already full. This signal can inform the MCU not transmitting the next data. 1: Under hardware flow control function, it represents RX FIFO or RX register of other device is not full. **Note: CTS is opposite to the signal of PIN CTS_n. It is not an interrupt source**
5 UART_IDLE	**UART_IDLE** 0: UART is at operation. 1: UART is not at operation, namely, transmitter and receiver are not working or has finished the data transmission or reception.
4 BRKWAK	**LIN BREAK wakeup flag** 0: break has not been received (exceed 29 bits) 1: break has been received (exceed 29 bits)
3 DCTS	**The flag of pin CTS_n signal changing** 0: no change. 1: represents CTS_n pin signal changing from 1 to 0 or 0 to 1. **Note: write 1 to clear this flag to 0.**
2 FBRK	**The flag of LIN BREAK occurred.** 0: has not detected the break field in LIN frame just in LIN function. 1: has detected the break field in LIN frame just in LIN function **Note: write 1 to clear this flag to 0.**
1 SYNERR	**LIN Sync field error flag** 0: there exists no error 1: there exists error **Note: write 1 to clear this flag to 0.**
0 IDLE	**IDLE flag** 0: idle line has not been detected 1: idle line detected Receiver has received the data followed by a high level maintaining at least one byte data time. IDLE status flag will be work after the bus IDLE detection (ILEN) is enabled **Note: write 1 to clear this flag to 0.**

8.6.11 UART_SMP_CNT

Table 8-16 UART_SMP_CNT register

UART_SMP_CNT **Sample counter register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															SMP_CNT	
Type															RW	
Reset															0	

Bits	Description
1: 0 SMP_CNT	**UART sample counter** 00: Based on 16*baud_pulse, baud_rate = system clock frequency/16/{DLH, DLL} 01: Based on 8*baud_pulse, baud_rate = system clock frequency/8/{DLH, DLL} 10: Based on 4*baud_pulse, baud_rate = system clock frequency/4/{DLH, DLL} 11: Based on sampe_count * baud_pulse, baud_rate = system clock frequency/16 /{DLM, DLL}

8.6.12 UART_GUARD

Table 8-17 UART_GUARD register

UART_GUARD **Guard time register** **Reset: 0x0000000F**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												GUARD_EN	GUARD_CNT			
Type												RW	RW			
Reset												0	F			

Note

Adding the guard time can contribute to eliminate the accumulated error every byte, so it is significant to improve the accuracy of the baud rate by using the fraction divisor with the guard time.

Bits	Description
4 GUARD_EN	**Guard interval time added enabling signal** 0: disable 1: enable
3: 0 GUARD_CNT	**Guard interval count value** 0~15: 0 ~ 15 bits time

8.6.13 UART_SLEEP_EN

Table 8-18 UART_SLEEP_EN register

UART_SLEEP_EN **Sleep enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																SLEEP_EN
Type																RW
Reset																0

Bits	Description
0 SLEEP_EN	**Sleep function enable** 0: Does not deal with sleep mode indication signal 1: Activate hardware flow control according to software initial settings when the chip enters the sleep mode. Release the hardware flow when the chip wakes up.

8.6.14 UART_DMA_EN

Table 8-19 UART_DMA_EN register

UART_DMA_EN **DMA enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															TX_DMA_EN	RX_DMA_EN
Type															RW	RW
Reset															0	0

Bits	Description
1 TX_DMA_EN	**TX_DMA mechanism enabling signal** 0: Does not use DMA in TX 1: Use DMA in TX. When this register is enabled.
0 RX_DMA_EN	**RX_DMA mechanism enabling signal** 0: Does not use DMA in RX 1: Use DMA in RX. When this register is enabled.

8.6.15 UART_DIV_FRAC

Table 8-20 UART_DIV_FRAC register

UART_DIV_FRAC **Fractional Divider Address** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												DIV_FRAC				
Type												RW				
Reset												0				

Bits	Description
4: 0 DIV_FRAC	**Fractional divisor** If actual divisor is 135.65, then DIV_FRAC is 0.65*32 = [20.8]=21 , and the DIV_L=135 .

8.6.16 UART_RS485CR

Table 8-21 UART_RS485CR register

UART_RS485CR **RS485 control register** **RESET:0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									RS485EN		INVPOL	DLYEN				
Type									RW		RW	RW				
Reset									0		0	0				

Note

1. Delay ahead of transmitting is corresponding to the UARTn_CNTR. Delay after the transmitting can use the UARTn_GUARD.

2. RS485 function use PIN RTS_n as a transmitting or receiving direction control PIN.

Bits	Description
7 RS485EN	0: disable rs485 mode 1: enable rs485 mode
5 INVPOL	0: don't inverse the polarity of rts_n 1: inverse the polarity of rts_n
4 DLYEN	A DELAY is inserted between the RS485 switch to output state and the actual START bit. The specific DELAY time is determined by the register UART_CNTR. That is, the delay1 corresponding to Figure 8-7. 0: disable delay 1: enable delay

8.6.17 UART_CNTR

Table 8-22 UART_CNTR register

UART_ CNTR **RS485 time delay register** **RESET:0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CNTR							
Type									RW							
Reset									0							

Bits	Description
7: 0 CNTR	**COUNTER** 0 to 255 bits time for time delay in RS485 mode.

8.6.18 UART_IDLE

Table 8-23 UART_IDLE register

UART_IDLE **Idle interrupt enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ILEN			IDLEIE				
Type									RW			RW				
Reset									0			0				

Bits	Description
	Bus idle detect enable
7 ILEN	0: disabled 1: enabled
	IDLE interrupt enable
4 IDLEIE	0: disabled 1: enabled

8.6.19 UART_LINCR

Table 8-24 UART_LINCR register

UART_LINCR **LIN control register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									LINEN	LBRKIE	LBRKDL	SDBRK	LABAUDEN	SYNERRIE	BRKWAKIE	
Type									RW	RW	RW	RW	RW	RW	RW	
Reset									0	0	0	0	0	0	0	

Bits	Description
7 LINEN	**LIN Mode enable** 0: disable 1: enable

Bits	Description
6 LBRKIE	**LIN Break character detect interrupt enable** 0: disable 1: enable
5 LBRKDL	**LIN Mode break detect length** 0: 10 bits 1: 11bits
4 SDBRK	**LIN transmit Break enable** 0: disable 1: enable transmit break, the Break length is determined by the register BRKLGH. **Note:** set by software and cleared by MCU internal hardware during the stop bit of the break.
3 LABAUDEN	**LABAUDEN** 0:0X55 not used to auto baud rate detection 1:0x55 used to auto baud rate detection
2 SYNERRIE	**SYNERRIE** 0: Disable sync byte error interrupt 1: Enable sync byte error interrupt
1 BRKWAKIE	**BRKWAKIE** 0: Disable break wakeup interrupt 1: Enable break wakeup interrupt

8.6.20 UART_BRKLGH

Table 8-25 UART_BRKLGH register

UART_BRKLH **LIN sync break length register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													BRKLGH			
Type													RW			
Reset													0			

Bits	Description

3: 0 BRKLGH	**BRKLGH** 0000: 13bits sync break length 0001: 14bits sync break length …… 1111: 28bits sync break length

9 ADC

9.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 12 external channels and 2 internal channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The analog monitor feature allows the application to monitor whether the input voltage exceeds the set voltage range.

9.2 ADC features

- 12-bit resolution.
- Channel input voltage range: AVSS < Vin < AVDD.
- Maximum conversion rate: 1Msps.
- 14 channels with sample time configurable for each: 12 external channels, 2 internal channels (Bandgap, T-sensor).
- Conversion sequence classified as regular group and injected group.
  - Regular group: configurable maximum 12 channels.
  - Injected group: configurable maximum 4 channels.
- 8 operation modes (called mode x for convenience, x=1 to 8):
  - Single regular group channel single conversion (mode1).
  - Single regular group channel continuous conversion (mode2).
  - Multiple regular group channels single scan with injected trigger (mode3 scan injected).
  - Multiple regular group channels single scan with discontinuous injected trigger (mode3 discontinuous injected).
  - Multiple regular group channels single scan with automatically injected (mode4).
  - Multiple regular group channels continuous scan with injected trigger (mode5 scan injected).
  - Multiple regular group channels continuous scan with injected trigger (mode5 discontinuous injected).
  - Multiple regular group channels continuous scan with automatically injected (mode6).
  - Multiple regular group channels under discontinuous conversion mode (mode7).
  - Multiple injected group channels under discontinuous conversion mode(mode8).
- Internal software trigger or external hardware trigger conversion.
- Analog monitor function:

– Monitor single or all channels voltage by configuration.

– Monitor whether the channel voltage is less than low threshold or more than high threshold.

- Interrupts:
  – End of conversion (EOC) for regular or injected group.
  – End of injected group conversion (IEOC).
  – Analog Monitor event (AMO).
- DMA request generation during regular channel conversion.

9.3 ADC functional description

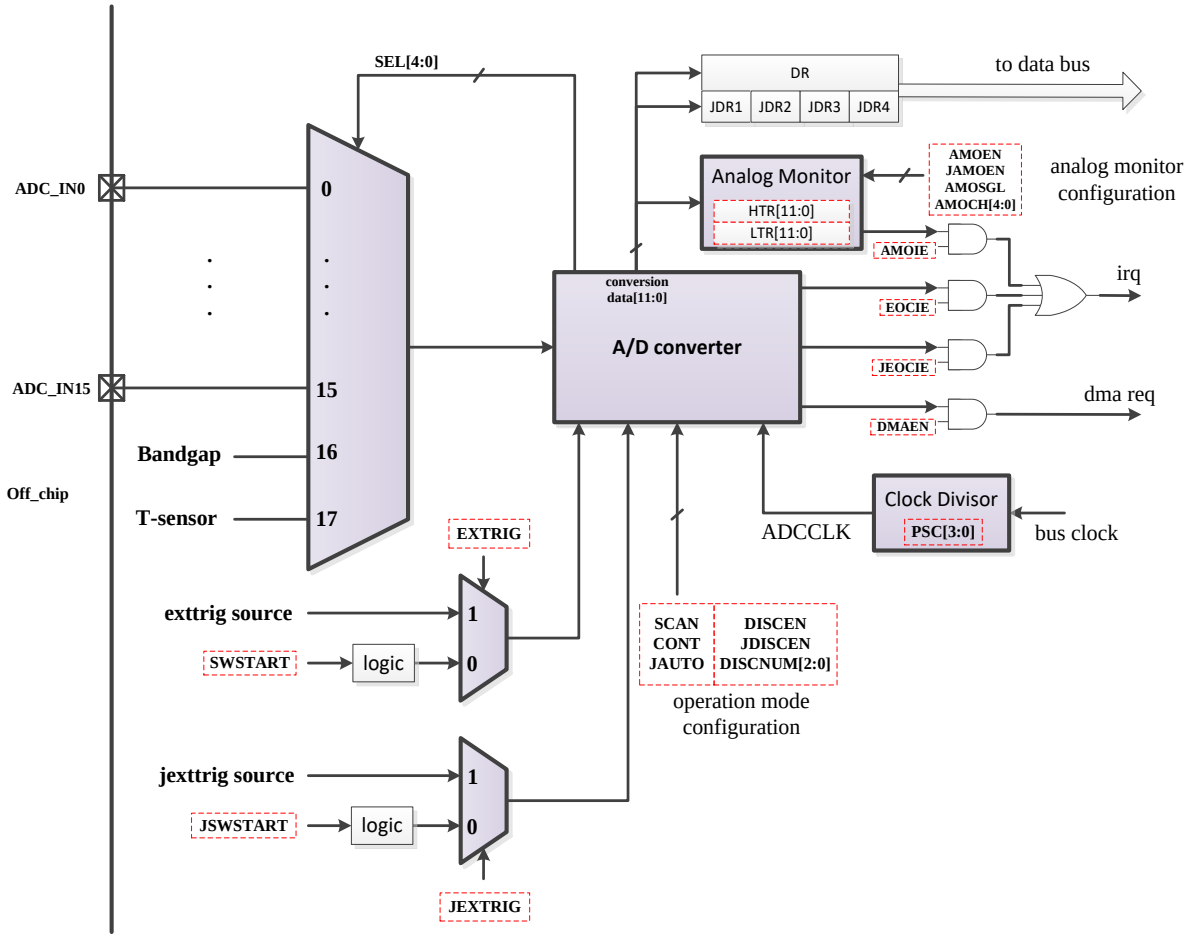

Figure 9-1 ADC block diagram

9.4 Function Description

ADC is composed of converter unit, input channel selector, clock divisor, and analog monitor, etc. As illustrated in Figure 9-1 , A/D converter unit operates at ADC clock (ADCCLK) and the other circuit parts operate at bus clock.

A typical operation flow is introduced in the following paragraph.

Firstly, ADC should be powered on, ADC can be triggered to start by the internal SWSTART or external trigger source, which is derived from the CTU module. After trigger, the ADC converter unit starts to work and sends out the selecting signal to input channel selector to select the desired channel one by one based on the regular or injected group channels sequence. After one channel has finished the conversion, the conversion result is stored into the RDR or IDRx based on which group the current conversion channel belongs to, and generate corresponding EOC or IEOC flag bit. Meanwhile, the analog monitor starts to work and the related statuses flag appear if the corresponding event occurs. Until now, a single channel conversion flow goes to the end. To point out, there exist some differences for different operation mode and detailed information will be illustrated later.

9.4.1 ADC power on sequence

Before starting all the functions, ADC should be powered on at first. Then a valid trigger can start the ADC to operate based on the configured mode. The power on sequence is illustrated as follows.

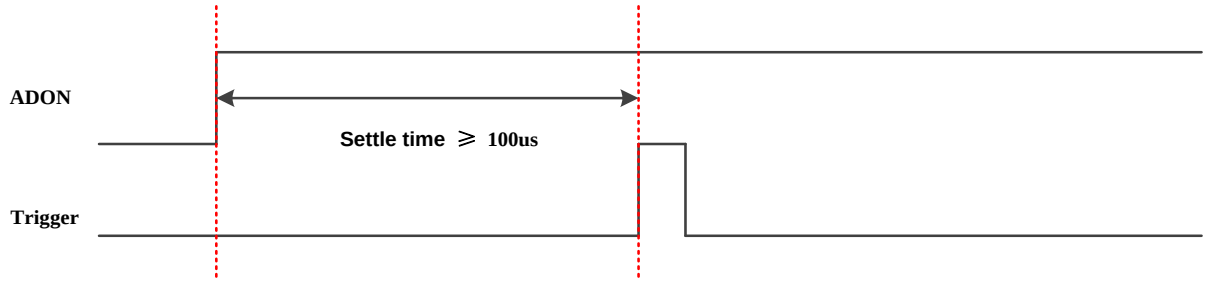

Figure 9-2 ADC power on sequence

As illustrated in the Figure 9-2, bit ADC_CTRL1[ADON] is set to 1 to control the power on process. After the ADON is set to 1, the settle time for A/D Converter unit power on is not less than 100us.

9.4.2 ADC operation modes

Different modes can be used flexibly based on the practical application. After power on and a valid trigger, ADC operates at one of the following modes.

Table 9-1 ADC operation modes and its corresponding configuration

Operation mode	MODE_BITS	Trigger source	Conversion sequence
mode1	5'b0000x	regular trigger	single regular group channel single conversion
mode2	5'b0100x	regular trigger	single regular group channel continuous conversion
mode3 (injected group scan mode)	5'b10000 (INTERVAL=0)	regular trigger (+injected trigger)	multiple regular group channels (+injected group channels) single conversion
mode3 (injected group discontinuous mode)	5'b10000 (INTERVAL=1)	regular trigger (+injected trigger)	multiple regular group channels (+injected group channels) single conversion
mode4	5'b10001	regular trigger + auto injected trigger	multiple regular group channels + injected group channels single conversion
mode5 (injected group scan mode)	5'b11000 (INTERVAL=0)	regular trigger (+injected trigger)	multiple regular group channels (+injected group channels) continuous conversion
mode5 (injected group discontinuous mode)	5'b11000 (INTERVAL=1)	regular trigger (+injected trigger)	multiple regular group channels (+injected group channels) continuous conversion
mode6	5'b11001	regular trigger + auto injected trigger	multiple regular group channels + injected group channels continuous conversion
mode7	5'b1x10x	regular trigger	regular group subgroup scan mode conversion
mode8	5'b1x01x	injected trigger	injected group channels scan mode conversion

Note: MODE_BITS = {SCAN, CONT, DISCEN, IDISEN, IAUTO}.

Before describing the mode operation, it's necessary to introduce some terminologies, such as regular group, injected group. For ADC input channels, they are called ch0 to ch13, of which ch0 to ch11 are the external input channels, ch12 bandgap reference voltage channel, and ch13 temperature sensor channel.

Regular group is the input channels arranged to convert in order. Based on register ADC_RSQR0, ADC_RSQR1 and ADC_RSQR2, the regular group is composed of maximum 12 channels in sequence from RSQ0 to RSQ11.

For example, if the RSQ0 to RSQ11 are set to 9,8,12,1,5,4,7,3,13,2,0,0 respectively, a regular group is arranged in Figure 9-3.

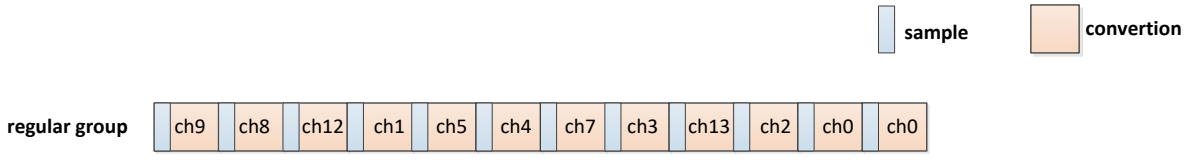

Figure 9-3 Regular group sequence

If the RSQL is 2(length=3), the last 3 channels will be invalid and not be converted. Valid regular group sequence is illustrated in Figure 9-4.

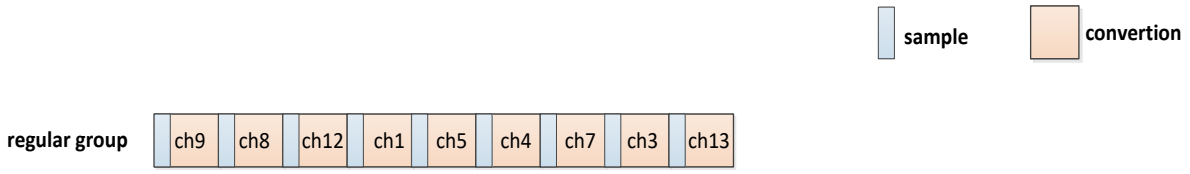

Figure 9-4 Valid regular group sequence

In the same way, injected group is the input channels arranged to convert in order. Based on register ADC_ISQRx, the injected group is composed of maximum 4 channels in sequence from ISQ0 to ISQ3.

For example, if the ISQ0 to ISQ3 are set to 12,7,13,2 respectively, an injected group is arranged as Figure 9-5.

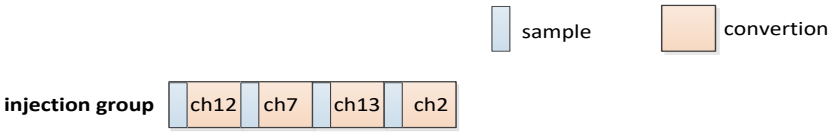

Figure 9-5 Injected group sequence

If the ISQL is 2(length=3), the last 1 channel will be invalid and not be converted. Valid injected group sequence is illustrated in Figure 9-6 .

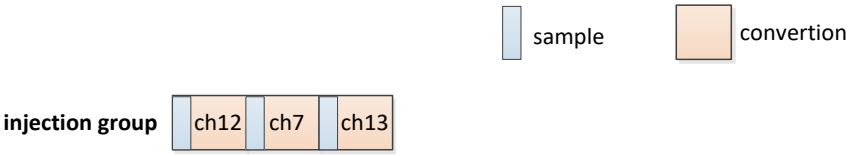

Figure 9-6 Valid injected group sequence

Obviously, regular group trigger and injected group trigger are the corresponding signals to start conversion of the regular and injected group sequence. The trigger is derived from the internal SWSTART or external trigger source illustrated in the ADC block diagram. And the regular trigger is invalid when ADC is in the process of regular group channel conversion. Based on the basic introduction, detailed information for each mode is described as follows.

9.4.2.1 Mode 1

This mode only converts first channel in regular group, ignore RSQL values. After the mode is configured in Table 9-1, a valid trigger can make the ADC work in this mode.

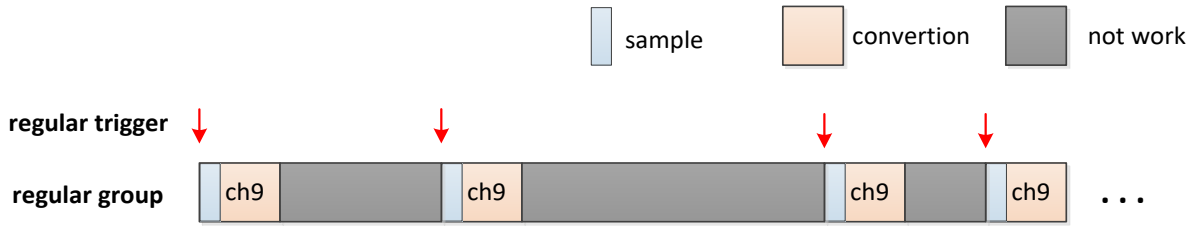

Figure 9-7 Mode 1 operation flow

As the Figure 9-7 shows, the first channel in regular group is converted once after a valid regular trigger. Then the ADC goes to idle until the next valid regular trigger for the next conversion.

9.4.2.2 Mode 2

This mode only converts first channel in regular group, ignore RSQL values. After the mode configuration as Table 9-1, a valid trigger can make the ADC work in this mode.

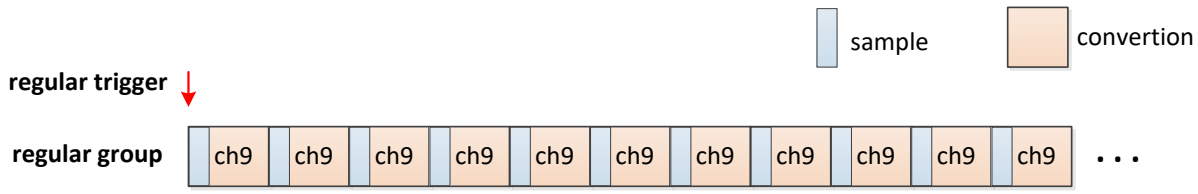

Figure 9-8 Mode 2 operation flow

As shown in Figure 9-8, the first channel in regular group is converted forever except power down, top reset or mode change after a valid regular trigger.

9.4.2.3 Mode 3

9.4.2.3.1 interval bit=0, injected group is the scan mode

This mode converts regular group and injected group. The valid regular and injected group channel number are decided by RSQL and ISQL respectively. With the mode configuration in Table 9-1, a valid trigger can make the ADC work in this mode. For example, RSQL is set to 6 (length=7), and ISQL is set to 2 (length=3). A typical operation shows in Figure 9-9. The initial regular trigger starts conversion of the first 7 channels in group. When ADC converts the channel 1 in regular group, an injected trigger switches the conversion to 3 injected group channels after the end of channel 1 conversion. After 3 injected group channels has been converted completely, conversion switches back to regular group channels automatically convert the channel 5. When finishing the valid regular channels conversion, ADC runs to idle until the next trigger.

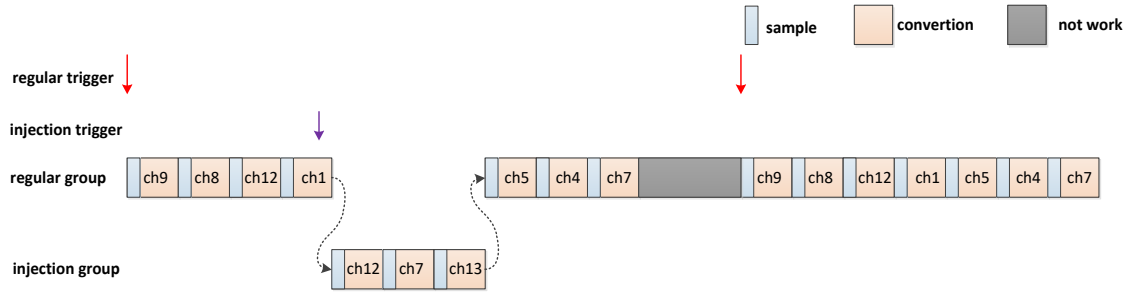

Figure 9-9 Mode 3 operation flow with injected trigger scan mode

Especially, if the injected trigger occurs when the ADC is idle, the ADC will finish the conversion of the valid injected group channels as the following Figure 9-10.

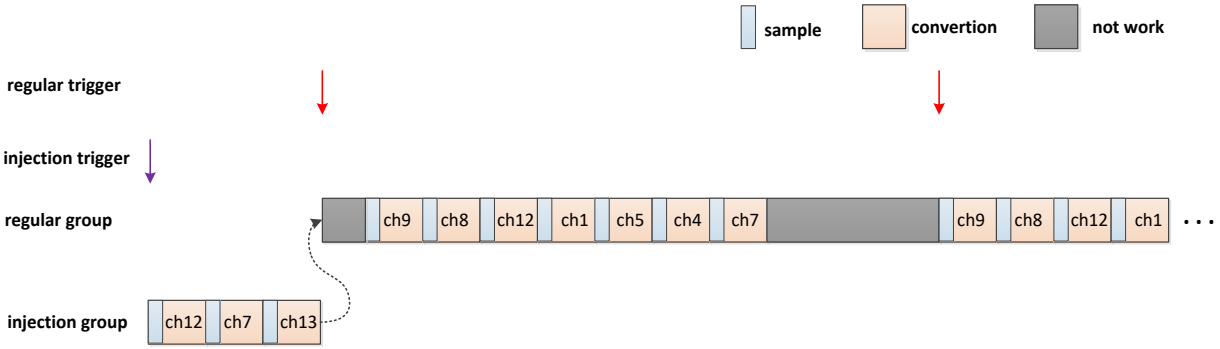

Figure 9-10 Mode 3 operation flow with injected trigger at ADC idle state

9.4.2.3.2 interval bit=1, injected group is the discontinuous mode

Compared to Figure 9-9, the generation of an injected trigger will only convert one channel of the injected group sequence, and the next time the injected trigger occurs, the next channel of the injected group sequence will be converted.

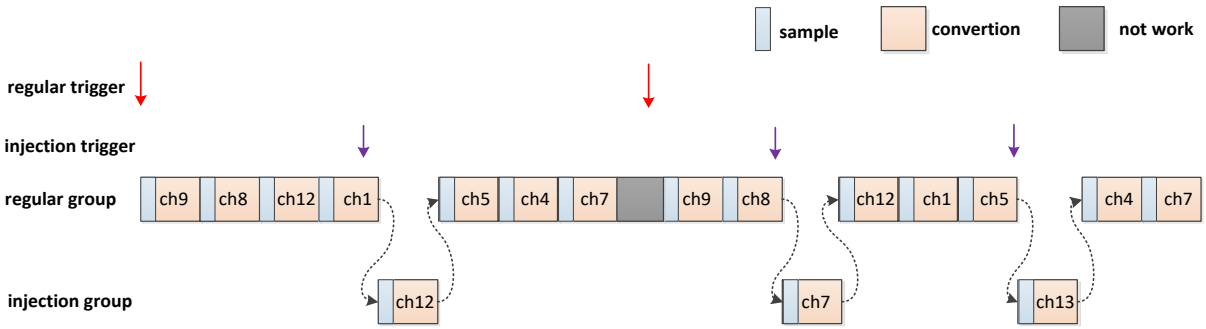

Figure 9-11 Mode 3 operation flow with injected trigger discontinuous mode

9.4.2.4 Mode 4

After this mode is triggered, the regular group is automatically convert first and then convert the injected group. The valid regular channels and injected group channels are decided by RSQL and ISQL respectively. With the mode configuration in Table 9-1, a valid trigger can make the ADC

work in this mode. For example, RSQL is set to 6 and ISQL is set to 2. A typical operation shows in Figure 9-12. A regular trigger starts conversion of the first 7 regular group channels followed by 3 injected group channels automatically. After the total 10 channels has been converted completely, ADC runs to idle state until the next valid regular trigger.

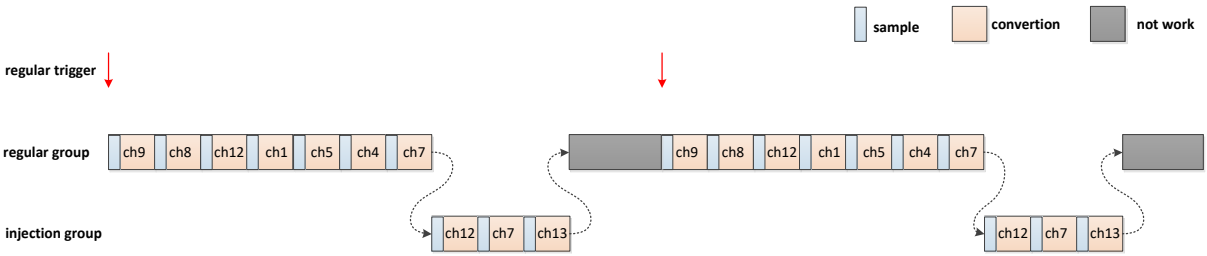

Figure 9-12 Mode 4 operation flow

9.4.2.5 Mode 5

9.4.2.5.1 interval bit=0, injected group is the scan mode

The difference with Mode 3 is that this mode is a continuous conversion. The valid regular channels and injected group channels are decided by SQL and ISQL respectively. Different from Mode 3, continuous conversion is done in this mode. With the mode configuration in Table 9-1, a valid trigger can make the ADC work in this mode. A key feature for this mode is that a single regular trigger can make the ADC work all the time except power down, top reset and mode change. For example, RSQL is set to 6 and ISQL is set to 2. A typical operation shows in Figure 9-13. The ADC works on regular group channels in order circularly after a regular trigger or works on the injected group channels if an injected trigger occurs.

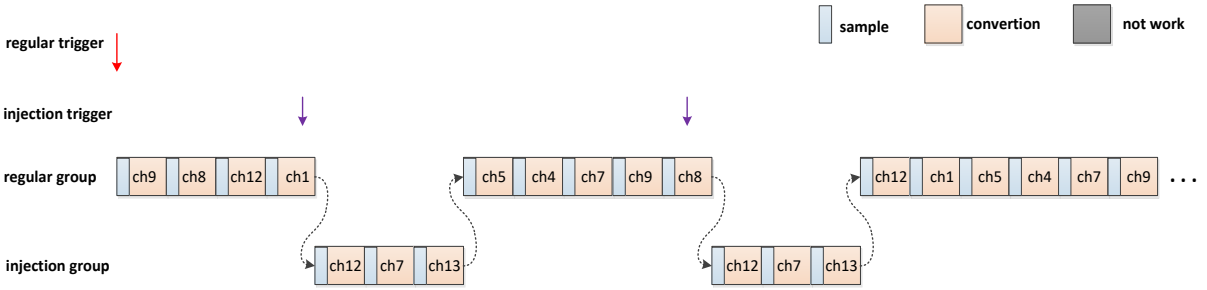

Figure 9-13 Mode 5 operation flow of injected group scan mode

Especially, if the injected trigger occurs when the ADC is idle, the ADC will finish the conversion of the valid injected group channels at first as the following Figure 9-14.

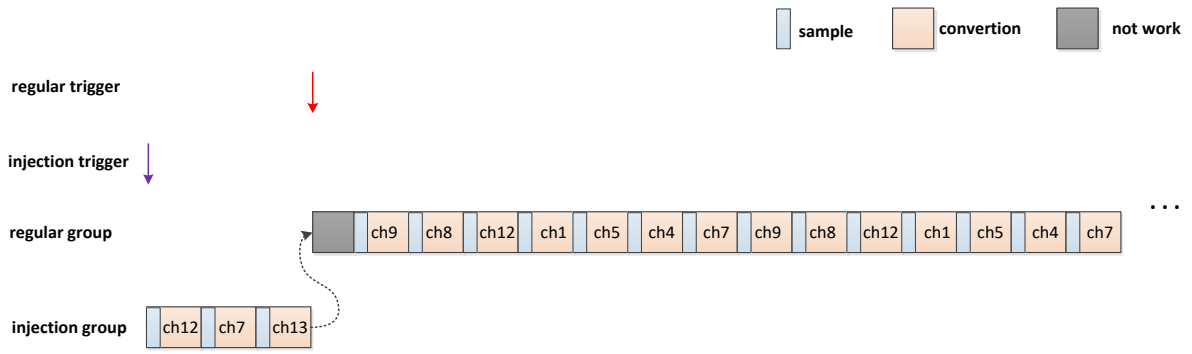

Figure 9-14 Mode 5 operation flow with injected trigger at ADC idle state

9.4.2.5.2 interval bit=1, infection group is the discontinuous mode

Compared to Figure 9-13, the generation of an injected trigger will only convert one channel of the injected group sequence, and the next time the injected trigger occurs, the next channel of the injected group sequence will be converted.

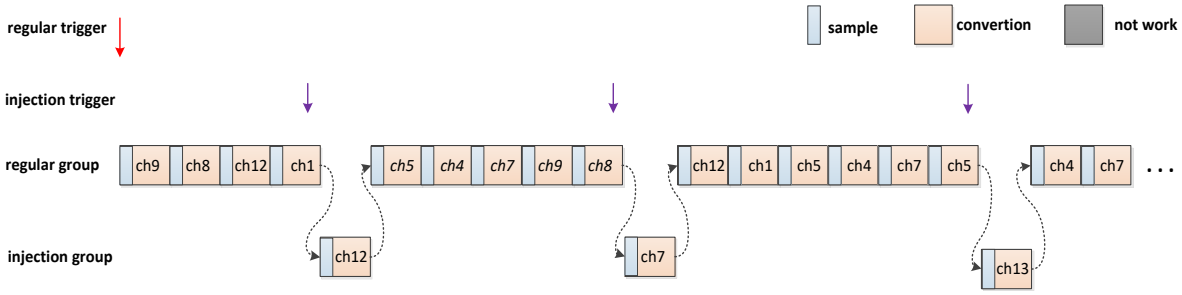

Figure 9-15 Mode 5 operation flow with injected trigger discontinuous mode

9.4.2.6 Mode 6

The difference from Mode 4 is that this mode is a continuous conversion. The valid regular channels and injected group channels are decided by RSQL and ISQL respectively. Different from Mode4, continuous conversion is done in this mode. With the mode configuration in Table 9-1, a valid regular trigger can make the ADC work in this mode. A key feature for this mode is that a single regular trigger can make the ADC work all the time except power down, top reset and mode change. For example, RSQL is set to 6 and ISQL is set to 2. An operation flow shows in Figure 9-16. The ADC works on regular group channels in order followed by injected group channels circularly after a regular trigger.

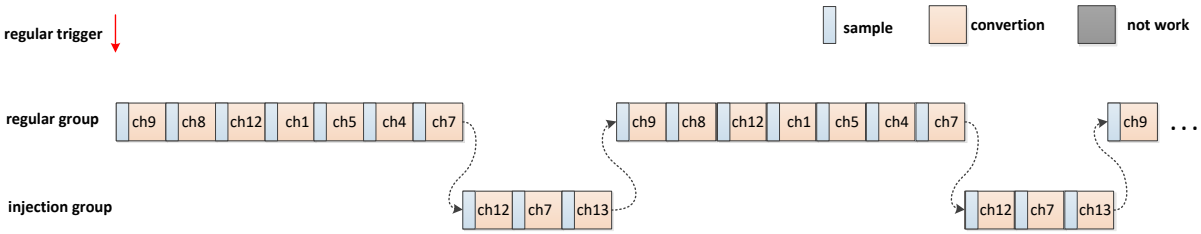

Figure 9-16 Mode 6 operation flow

9.4.2.7 Mode 7

This mode only converts regular group. The valid regular channels group channels are decided by RSQL. With the mode configuration in Table 9-1, ADC can operate in this mode. The valid regular channels are divided into several subgroups every DISCNUM channels.

For example, RSQL is set to 6 and DISCNUM is set to 1.

First regular trigger: ch9, ch8;

Second regular trigger: ch12, ch1;

Third regular trigger: ch5, ch4;

Fourth regular trigger: ch7, generates EOC flag;

Therefore, the practical conversion flow is illustrated in Figure 9-17.

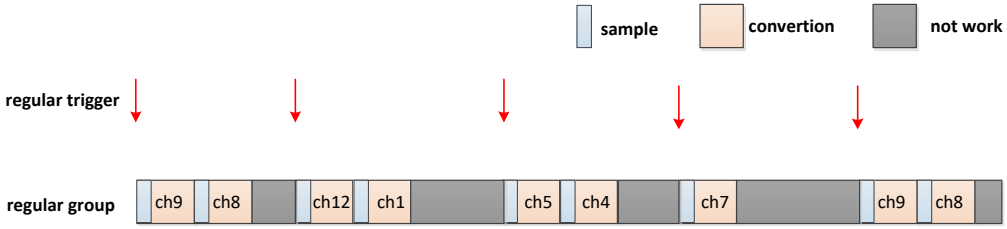

Figure 9-17 Mode 7 operation flow

9.4.2.8 Mode 8

This mode only converts injected group. The valid injected channels group channels are decided by ISQL. With the mode configuration in Table 9-1, ADC can operate in this mode. The valid injected channels are divided into several subgroups every one channel.

For example, ISQL is set to 2.

First regular trigger: ch12;

Second regular trigger: ch7;

Third regular trigger: ch13, generates IEOC flag;

Fourth regular trigger: ch12;

…

Therefore, the practical conversion flow is illustrated in Figure 9-18.

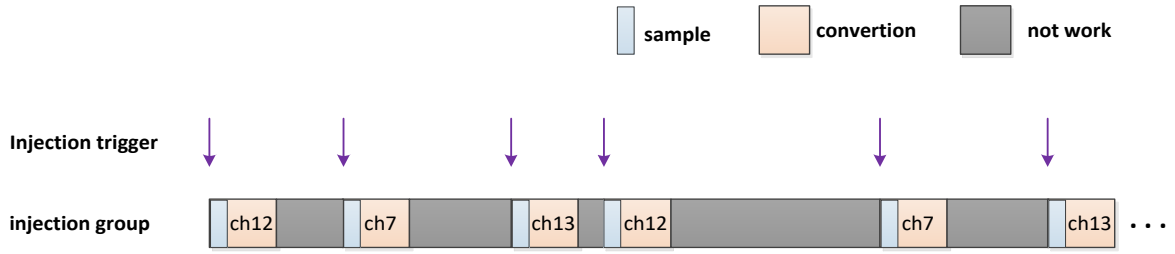

Figure 9-18 Mode 8 operation flow

9.4.3 Trigger mode

According to different triggering methods, the following 7 scenarios can be combined.

Table 9-2 Responsive behavior under different triggering methods

Trigger method	Responsive behavior
Trigger regular group	Regular group conversion
Trigger injected group	Injected group conversion
Regular trigger is generated during regular group conversion	The regular group continues to convert, and the second trigger event does not respond
Injected trigger is generated during regular group conversion	Wait for the current channel of the regular group to be converted before switching to the injected group After the injected group conversion, switch to the original regular group to continue convert (if the regular group sequence has not been converted).
Injected trigger is generated during injected group conversion	The injected group continues to convert, the second injected trigger event does not respond
Regular trigger is generated during injected group conversion	A regular event is generated during the injected conversion, the injected conversion will not be interrupted, but the regular sequence will be executed after the injected sequence ends up.
Regular trigger and injected trigger are generated at the same time	Convert regular group after injecting trigger responses

9.4.4 Analog monitor

Analog monitor supports level triggered monitoring event and edge triggered monitoring event mode. If the monitored channels voltage is more than the high threshold or less than the low threshold, the analog monitor will generate a monitor event. Analog monitor can be used to monitor none, single channel, multiple channels based on bits AMOEN, IAMOEN, AMOSGL and AMOCH.

The thresholds are configured through the AMOHR and AMOLR registers. The unit configured for each field (AMOHT/ AMOLT/ AMOHO/ AMOLO) of these two registers is LSB, which is the voltage corresponding to one ADC code. It should be noted that the ADC raw data is used for the threshold comparison. If the injected group offset is set, the original value data is used for comparison.

Table 9-3 Analog monitor configuration

Analog monitor channel	{AMOEN,IAMOEN,AMOSGL}	Configurable operation mode	Comment
None	3'b00x	All modes	-
All injected group channels	3'b010	mode3 ~ mode6, mode8	-
All regular group channels	3'b100	Except mode8	-
All channels	3'b110	All modes	-
Single injected group channel	3'b011	mode3 ~ mode6, mode8	Conversion sequence must contain the injected channel determined by AMDCH[4:0].
Single regular group channel	3'b101	mode1 ~ mode7	Conversion sequence must contain the regular channel determined by AMDCH[4:0].
Single regular or injected group channel	3'b111	All modes	Conversion sequence must contain the regular or injected channel determined by AMDCH[4:0].

9.4.4.1 Level trigger mode

Set AMOMODE=0, the analog monitor operates in level trigger mode.

If the monitored channels voltage is more than the high threshold AMOHT or less than the low threshold AMOLT, analog monitor set the AMO flag to 1, which is cleared by writing 0 to it. Meanwhile, the interrupt occurs if AMO flag is 1 and AMOIE is configured to 1.

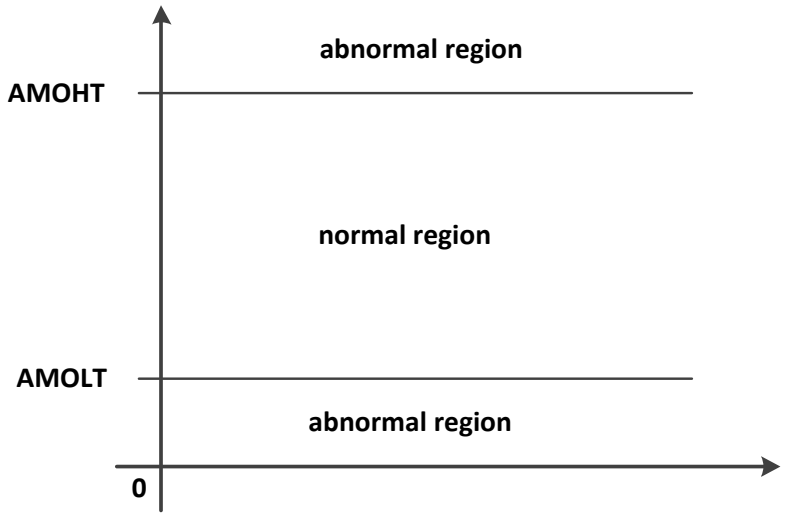

Figure 9-19 Analog monitor detecting region

9.4.4.2 Edge trigger mode

Set AMOMODE=1, the analog monitor operates in edge trigger mode.

If the monitored channels voltage goes from the normal area to the abnormal area (more than the high threshold or less than the low threshold), a monitor abnormal event is generated. analog monitor set the AAMO flag to 1. Meanwhile, the monitor interrupt occurs if AMOIE is configured to 1.

When the monitored channels voltage goes from an abnormal area to a normal area, a monitoring recovery event is generated. The analog monitor sets the NAMO flag to 1. If the AMOIE is configured to 1, a monitoring event interrupt is generated. Boundary value = [High threshold-high offset value, low threshold + low offset value], which means [AMOHT-AMOHO, AMOLT+AMOLO].

Note

The edge trigger mode can only be used when monitoring a single channel. Monitoring multiple channels cannot distinguish between abnormalities triggered by different channels and recovery interrupts.

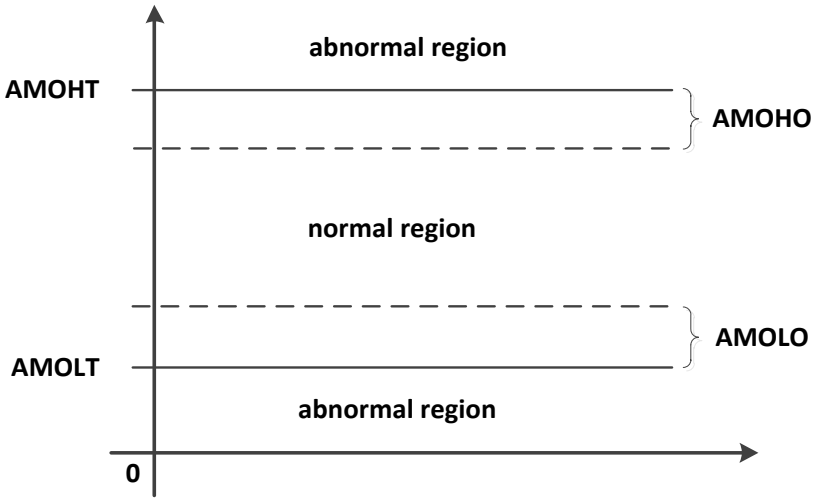

Figure 9-20 Monitor detecting region in edge trigger mode

9.4.5 Status flag

There are three flags to indicate the conversion status: EOC, IEOC, AMO (AAMO and NAMO are used in edge trigger mode). The EOC flag indicates the end of the conversion for both regular and injected group channels. The IEOC flag indicates whether all the injected group channels are converted completely. The AMO flag indicates whether the analog monitor event occurs. The analog monitor event is whether the current conversion result is more than high threshold or less than low threshold based on the configuration. The flag EOC and IEOC are generated at different time for different modes, but can be classified to three conditions. The flag AMO is generated at the same time for all the modes. The following descriptions are introduced with the assumption that ch5 is less than AMOLT, ch7 is more than the AMOHT, and the analog monitor is configured to check all the channels including regular group channels and injected group channels for example.

[Scenario 1] The flags EOC and IEOC is generated at the same time for mode 1 ~ mode 6. The flag AMO is generated at the same time for all the 8 modes. The detailed information about three flags is illustrated in Figure 9-21 based on mode 6.

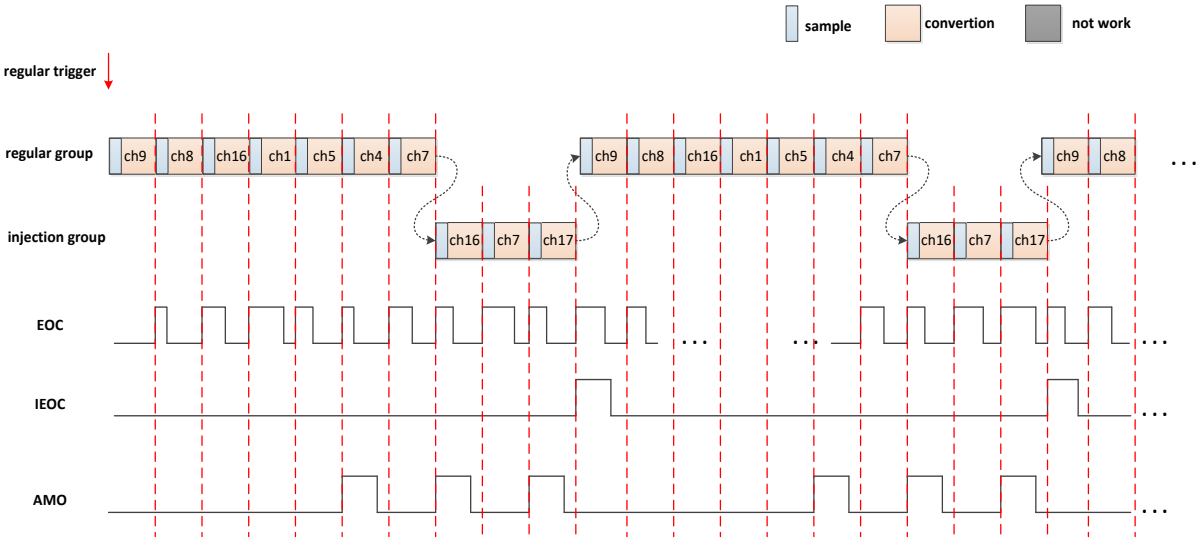

Figure 9-21 Three flags under condition 1

In Figure 9-21, EOC is set to 1 when a channel conversion is finished for both regular and injected group channels. And the EOC is cleared by writing 0 to it or reading the ADC_RDR register. For IEOC flag, it is set to 1 when all the valid injected group channels have been converted completely and cleared by writing 0 to IEOC bit. AMO flag is set to 1 when the channel voltage is out of the analog monitor normal region, such as ch5, ch7. To point out, the duration of the flag maintaining 1 is decided by the CPU response time, which is related with the current loading of the CPU at that moment.

[Scenario 2] The time when the flags are generated is different between mode7 and mode 1~ mode 6. The detailed information about three flags is illustrated in Figure 9-22.

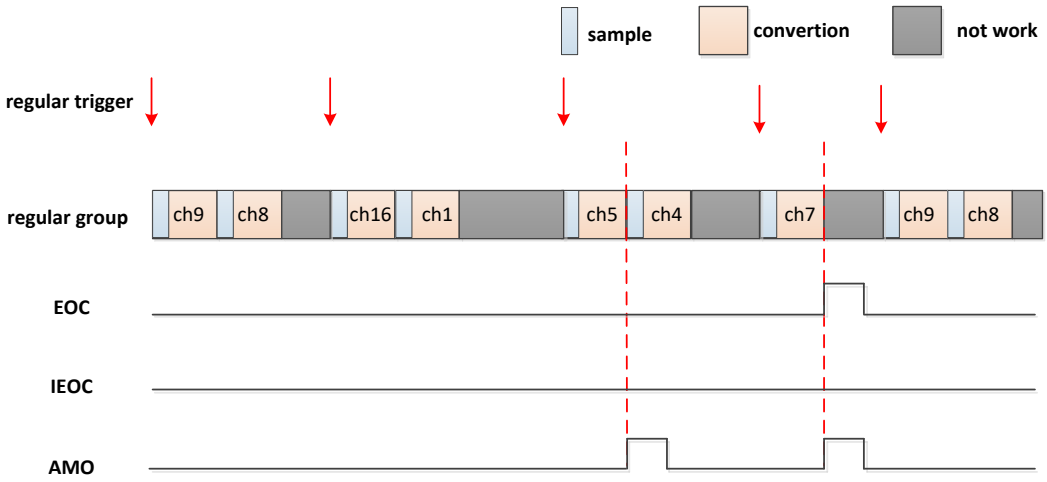

Figure 9-22 Three flags under condition 2

In Figure 9-22, EOC is set to 1 when the channel conversions are finished for regular group channels. And the EOC is cleared by writing 0 to it or reading the ADC_RDR register. For IEOC flag, it is still 0 all the time for this mode. AMO flag is set to 1 when the channel voltage is out of the analog monitor normal region, such as ch5, ch7. To point out, the duration of the flag

maintaining 1 is decided by the CPU response time, which is related with the current loading of the CPU at that moment.

[Scenario 3] The flags also show a different scenario under mode 8. It's easy to describe the flag information because ADC conversion just aims at injected group channels.

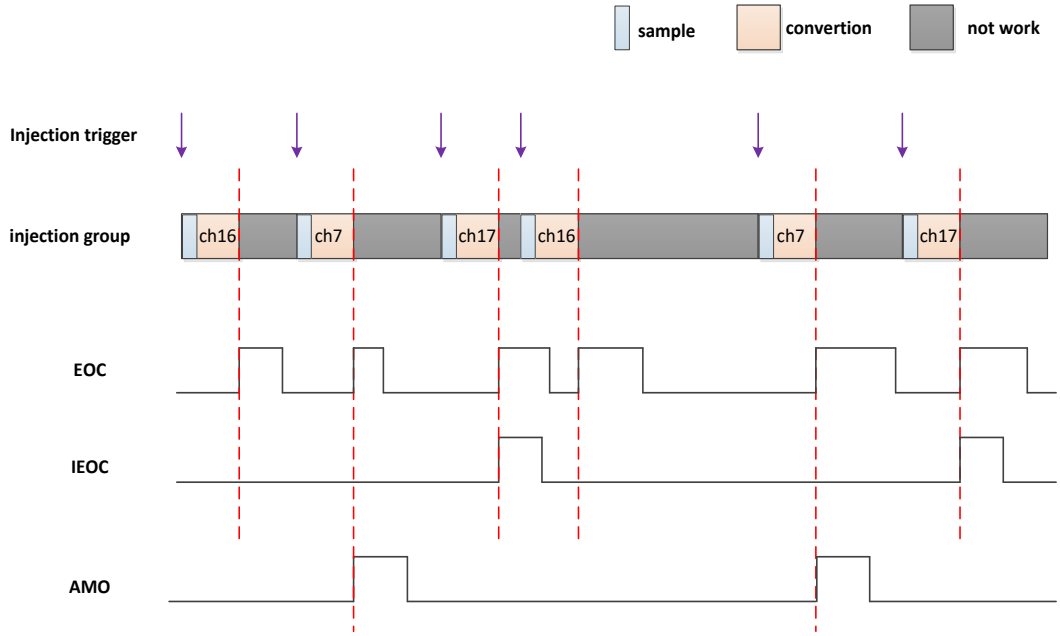

Figure 9-23 Three flags under condition 3

In Figure 9-23, EOC and IEOC are set to 1 when the channel conversions are finished for all the injected group channels.

9.4.6 Calibration

The calibration function can make the ADC conversion result more accurate and correct the errors caused by GE (Gain Error) and OE (Offset Error).

During chip production, machine testing is required, and the measured GE & OE values are stored in a specific area of the chip. When the calibration function is enabled, the data register gives the final result of using GE & OE for calibration.

9.4.7 Sampling conversion time

The ADC needs to use several ADC_CLK cycles to sample the input voltage, i.e. charging the ADC's internal circuitry to the level of the external input signal, and completing the sampling before the analog to digital conversion can take place. The number of sampling cycles can be changed by the SPT [2:0] bits in the ADC_SPT register. Each channel can be sampled at different times.

Total conversion time: **(SPT+ 12) * ADC cycles + 5 APB cycles**

For example, if APB=24MHz, ADCCLK=24MHz, SPT=7 ADCCLK, total conversion time= (7+12)/24+(5/24) = 1μs.

9.4.8 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (TA) of the device. The temperature sensor is internally connected to the ADC channel which is used to convert the sensor output voltage into a digital value.

The internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

Temperature using the following formula:

Temperature (°C) = {(V_{TEMP25} - V_{SENSE}) / Slope} + 25

V_{TEMP25}: V_{SENSE} value for 25°C

V_{SENSE}: V_{SENSE} value

Slope = Average slope for Temperature (mV/°C).

Refer to the Electrical characteristics section for the actual values of V_{TEMP25} and Slope.

9.4.9 DMA Request

Since converted regular channels value are stored in a unique data register, it is necessary to use DMA for conversion of more than one regular channel. This avoids the loss of data already stored in the ADC_RDR register. The DMA function is only for the regular group.

Only the end of conversion of a regular channel generates a DMA request, which allows the transfer of its converted data from the ADC_RDR register to the destination location selected by the user.

9.4.10 Low power mode

ADC has two power consumption modes available. One is the normal mode, the other is low power mode. The ADC clock operates at a lower frequency in low-power mode to reduce power consumption. ADC enters low-power mode when MCU runs into stop mode. The ADC analog monitor event can wake up the MCU from stop mode to normal mode.

The power mode switching process is shown in Figure 9-24 .

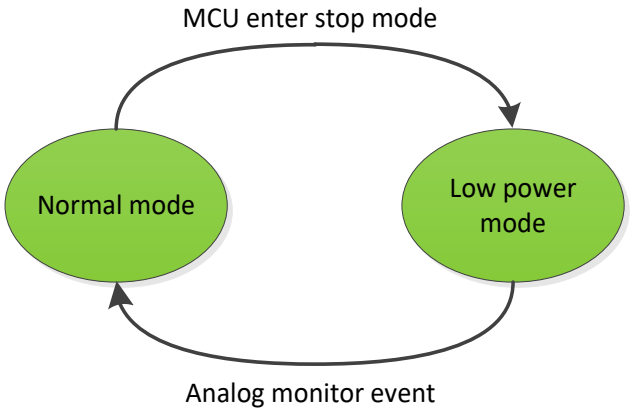

Figure 9-24 CPU power mode switching flow

9.5 Register definition

Table 9-4 ADC register mapping

ADC0 base address = 0x40003000

Address	Name	Width (in bit)	Description
ADCx base addr+0x0	ADC_STR	32	status register
ADCx base addr +0x4	ADC_CTRL0	32	control register 0
ADCx base addr +0x8	ADC_CTRL1	32	control register 1
ADCx base addr +0xC	ADC_SPT0	32	sample time selection register 0
ADCx base addr +0x10	ADC_SPT1	32	sample time selection register 1
ADCx base addr +0x14	ADC_IOFR0	32	injected group offset register 0(ADC_IOFR0)
ADCx base addr +0x18	ADC_IOFR1	32	injected group offset register 1(ADC_IOFR1)
ADCx base addr +0x1C	ADC_IOFR2	32	injected group offset register 2(ADC_IOFR2)
ADCx base addr +0x20	ADC_IOFR3	32	injected group offset register 3(ADC_IOFR3)
ADCx base addr +0x24	ADC_AMOHR	32	AMO high threshold register
ADCx base addr +0x28	ADC_AMOLR	32	AMO low threshold register
ADCx base addr +0x2C	ADC_RSQR0	32	regular group sequence configure register 0
ADCx base addr +0x30	ADC_RSQR1	32	regular group sequence configure register 1
ADCx base addr +0x34	ADC_RSQR2	32	regular group sequence configure register 2
ADCx base addr +0x38	ADC_ISQR	32	injected group sequence configure register
ADCx base addr +0x3C	ADC_IDR0	32	injected group data register 0(ADC_IDR0)
ADCx base addr +0x40	ADC_IDR1	32	injected group data register 1(ADC_IDR1)
ADCx base addr +0x44	ADC_IDR2	32	injected group data register 2(ADC_IDR2)
ADCx base addr +0x48	ADC_IDR3	32	injected group data register 3(ADC_IDR3)

Address	Name	Width (in bit)	Description
ADCx base addr +0x4C	ADC_RDR	32	regular group data register

9.5.1 ADC_STR

Table 9-5 ADC_STR register

ADC_STR **ADC status register** **Reset: 0x00000010**

Bit	31~7	6	5	4	3	2	1	0
Name		AAMO	NAMO	IDLE		IEOC	EOC	AMO
Type		W0C	W0C	RO		W0C	W0C	W0C
Reset		0	0	1		0	0	0

Bits	Description
6 AAMO	**Analog monitor event occurs (edge trigger mode)** 0: no analog monitor event 1: analog monitor event occurs, write 0 to clear
5 NAMO	**Analog monitor recovery event occurs (edge trigger mode)** 0: no recovery event 1: recovery event occurs, write 0 to clear.
4 IDLE	**ADC idle state flag** 0: ADC not idle 1: ADC idle
2 IEOC	**Injected group conversion completed flag** 0: injected group conversion not completed 1: injected group conversion completed, write 0 to clear
1 EOC	**Regular group conversion completed flag** 0: Regular group conversion not completed 1: Regular group conversion completed, write 0 or read ADC_RDR to clear
0 AMO	**Analog monitor event occurs (level trigger mode)** 0: no analog monitor event 1: analog monitor event occurs, write 0 to clear

9.5.2 ADC_CTRL0

Table 9-6 ADC_CTRL0 register

ADC_CTRL0 **ADC ctrl register 0** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SWSTART	ISWSTART						INTERVAL	AMOMODE	ALIGN	IEXTTRIG	EXTTRIG	DMAEN	AMOIE	IEOCIE	EOCIE
Type	RW	RW						RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0						0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SCAN	CONT	DISCEN	IDISCEN	IAUTO	DISCNUM[2: 0]			AMOEN	IAMOEN	AMOSGL	AMOCH[4: 0]				
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
31 SWSTART	**Software trigger for regular channels** write 1 to trigger; read as 0
30 ISWSTART	**Software trigger for inject channels** write 1 to trigger; read as 0
24 INTERVAL	**Interval mode (used only in Mode3/5)** 0: injected group is scan mode 1: inject group is interval mode
23 AMOMODE	**Analog monitor trigger mode** 0: level trigger mode 1: edge trigger mode
22 ALIGN	**Data alignment** 0: right alignment. 1: left alignment.
21 IEXTTRIG	**Inject group trig source select** 0: internal (software trig) 1: external
20 EXTTRIG	**Regular group trig source select**

Bits	Description
	0: internal (software trig) 1: external
19 DMAEN	**DMA function enable** 0: disable 1: enable
18: 16 AMOIE,IEOCIE,EOCIE	**Interrupt function enable** 0: disable 1: enable
15: 11 Modes control bits	**ADC operation mode** detailed configuration in Table 9-1.
10: 8 DISCNUM	**Discontinuous conversion length of channel** 0 ~ 7: decide the subgroup length under mode 7
7: 5 Analog monitor control bits	**Analog monitor function configuration** detailed configuration in Table 9-3.
4: 0 AMOCH	**Analog monitor detecting channel** Specify the monitored channel when analog monitor is configured to detect just single channel

9.5.3 ADC_CTRL1

Table 9-7 ADC_CTRL1 register

ADC_CTRL1 **ADC control register 1** **Reset: 0x0000F002**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSC[3: 0]														CALEN	ADON
Type	RW	RW	RW	RW											RW	RW
Reset	1	1	1	1											1	0

Bits	Description
15: 12 PSC	**ADC clock Prescaler** 0 ~ 15 : 1 ~ 16 divisor
1 CALEN	**Calibration enable** 0: disable calibration

Bits	Description
	1: enable calibration Notice that this function should be kept enable to assure the accuracy of ADC.
0 ADON	**ADC Power on** 0: ADC power down and reset the ADC (but the configure registers are not be reset). 1: ADC power on

9.5.4 ADC_SPT0

Table 9-8 ADC_SPT0 register

ADC_SPT0 **ADC Sample Time Register 0** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					SPT13[2: 0]			SPT12[2: 0]			SPT11[2: 0]			SPT10[2: 0]		
Type					RW											
Reset					0											

Bits	Description
11: 0 SPTx(x=10 ~ 13)	**Sample time selection for each channels** The numbering rule of SPTx is the same as ADC_RSQRx is: 0~11: External channels 12: Bandgap voltage 13: Temperature sensor voltage The meaning of SPTx code is as below: 000：9 ADCCLK 001：7 ADCCLK 010：15 ADCCLK 011：33 ADCCLK 100：64 ADCCLK 101：140 ADCCLK 110：215 ADCCLK 111：5 ADCCLK

9.5.5 ADC_SPT1

Table 9-9 ADC_SPT1 register

ADC_SPT1 **ADC Sample Time Register 1** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name			SPT9[2: 0]			SPT8[2: 0]			SPT7[2: 0]			SPT6[2: 0]			SPT5[2: 0]	
Type			RW													
Reset			0													
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		SPT4[2: 0]			SPT3[2: 0]			SPT2[2: 0]			SPT1[2: 0]			SPT0[2: 0]		
Type	RW	RW														
Reset	0	0														

Bits	Description
29: 0 SPTx (x=0 ~ 9)	**Sample time selection for each channels** The numbering rule of SPTx is the same as ADC_RSQRx is: 0~11: External channels 12: Bandgap voltage 13: Temperature sensor voltage The meaning of SPTx code is as below: 000：9 ADCCLK 001：7 ADCCLK 010：15 ADCCLK 011：33 ADCCLK 100：64 ADCCLK 101：140 ADCCLK 110：215 ADCCLK 111：5 ADCCLK

9.5.6 ADC_IOFRx

Table 9-10 ADC_IOFRx (x=0~3)register

ADC_IOFRx (x= 0 to 3) **ADC Injected group offset Register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					IOFR											
Type					RW											
Reset					0											

Bits	Description
11: 0 IOFR	**Injected group offset value** Injected group channels conversion values IDFR has been minus by Offset value defined in the register ADC_IOFR.

9.5.7 ADC_AMOHR

Table 9-11 ADC_AMOHR register

ADC_AMOHR **ADC AMO High threshold Register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name					AMOHO											
Type					RW											
Reset					0											
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					AMOHT											
Type					RW											
Reset					0											

Bits	Description
27: 16 AMOHO	**Recovery offset corresponds to the High threshold value for analog monitor** Define offset of the high threshold value .
11: 0 AMOHT	**High threshold value for analog monitor** Define the high threshold value .

9.5.8 ADC_AMOLR

Table 9-12 ADC_AMOLR register

ADC_AMOLR **ADC AMO High threshold Register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name					AMOLO											
Type					RW											
Reset					0											
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					AMOLT[11: 0]											
Type					RW											
Reset					0											

Bits	Description
27: 16 AMOLO	**Recovery offset corresponds to the Low threshold value for analog monitor** Define offset of the low threshold value
11: 0 AMOLT	**Low threshold value for analog monitor** Define the low threshold value .

9.5.9 ADC_RSQR0

Table 9-13 ADC_RSQR0 register

ADC_RSQR0 **ADC regular group sequence configure register 0** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									RSQL[3: 0]							
Type									RW							
Reset									0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																
Type																
Reset																

Bits	Description
23: 20 RSQL	**Length of the regular group** 0 ~ 11: Regular group length 1~12. **Note:** length must be less than the number of actual valid regular group sequence

9.5.10 ADC_RSQR1

Table 9-14 ADC_RSQR1 register

ADC_RSQR1 **ADC regular group sequence configure register 1** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name			RSQ11[4: 0]					RSQ10[4: 0]					RSQ9[4: 0]			
Type			RW													
Reset			0													
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		RSQ8[4: 0]					RSQ7[4: 0]					RSQ6[4: 0]				
Type		RW														
Reset		0														

Bits	Description
29: 0 RSQx(x=6~11)	**Channel selection for regular group** 0~11: External channels 12: Bandgap voltage 13: Temperature sensor voltage

9.5.11 ADC_RSQR2

Table 9-15 ADC_RSQR2 register

ADC_RSQR2 **ADC regular group sequence configure register 2** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name			RSQ5[4: 0]					RSQ4[4: 0]					RSQ3[4: 0]			
Type			RW													
Reset			0													
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		RSQ2[4: 0]					RSQ1[4: 0]					RSQ0[4: 0]				
Type		RW														
Reset		0														

Bits	Description
29: 0 RSQx(x=0~5)	**Channel selection for regular group** 0~11: External channels 12: Bandgap voltage 13: Temperature sensor voltage

9.5.12 ADC_ISQR

Table 9-16 ADC_ISQR register

ADC_ISQR **ADC injected group sequence configure register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name											ISQL[1: 0]		ISQ3[4: 0]			
Type											RW					
Reset											0					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		ISQ2[4: 0]					ISQ1[4: 0]					ISQ0[4: 0]				
Type		RW														
Reset		0														

Bits	Description
21: 20 ISQL	**Length of the injected group** 0 ~ 3: Injected group length. **Note:** The length must be less than the number of actual valid injected group sequence.
19: 0 ISQx(x=0~3)	**Channel selection for regular group** 0~11: External channel 12: Bandgap voltage 13: Temperature sensor voltage

9.5.13 ADC_IDRx

Table 9-17 ADC_IDRx(x=0~3) register

ADC_IDRx (x=0 ~ 3) **ADC Injected group data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IDR															
Type	RO															
Reset	0															

Bits	Description
15: 0 IDR	**Data registers for injected group** **Note: ADC_IDR has minuses the offset defined in the ADC_IOFR register.**

9.5.14 ADC_RDR

Table 9-18 ADC_RDR register

ADC_RDR **ADC Regular group data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RDR															
Type	RO															
Reset	0															

Bits	Description
15: 0 RDR	**Data register for regular group** **Note:** there is just one register for regular group. If ADC is operating at high speed and the CPU can't handle in time, so user must open the DMA function for ADC to avoid the data from being lost.

10 ACMP

10.1 Introduction

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages. The analog MUX provides a circuit for selecting an analog input signal from 8 channels: One channel provided by the 6-bit DAC, and others by external input.

The polling mode and hall output function are designed for motor application.

10.2 Features

- On-chip 6-bit resolution DAC with selectable reference voltage from VDD or internal bandgap.
- Configurable hysteresis, supporting 20/40 mV.
- Selectable interrupt type on rising edge, falling edge, or both rising or falling edges of comparator output.
- Up to 8 selectable comparator inputs (ADC_IN0~ADC_IN6 and internal DAC).
- Support Stop mode wakeup.
- Support polling mode.
- Support hall output.

10.3 Block diagram

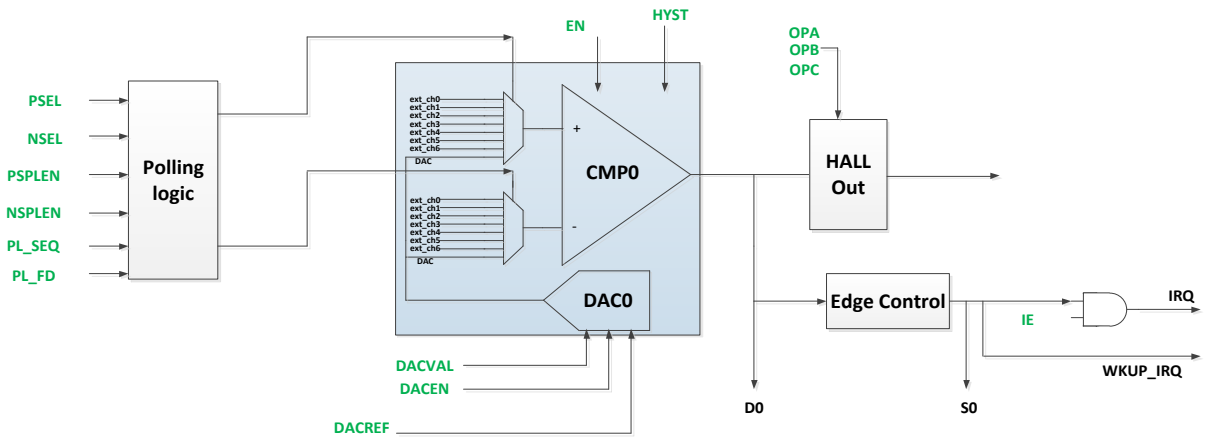

Figure 10-1 ACMP block diagram

10.4 Functional description

The ACMP module is functionally composed of two parts: digital-to-analog (DAC) and comparator (CMP).

The DAC includes a 64-level DAC (digital to analog converter) and control logic. DAC can select one of two reference inputs, Vdd or on-chip bandgap as the DAC input source, through DACREF. After the DAC is enabled, it converts the data set in DACVAL to a stepped analog output, which is used as ACMP input.

The ACMP can achieve the analog comparison between positive input and negative input, and then give out a digital output and interrupt. Both the positive input and negative input of analog comparator can be selected from the eight common inputs.

10.4.1 Normal mode

In normal mode, positive and negative inputs are compared with fixed selected channels, the comparison result appears as a digital output. Whenever a valid edge defined in the output occurs, the SR status bit is asserted. If IE is set, a CPU interrupt occurs.

The ACMP output is synchronized by the bus clock to generate the comparison result, so that CPU can read the comparison. The data register changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

10.4.2 Polling mode

In polling mode, ACMP can switch the input channel for comparator's positive or negative input by logic control dynamic switch. The switch sequence is defined in ACMP_CR4[PLSEQ], and the frequency is controlled by ACMP_FD[PLFD]. PSPLEN and NSPLEN is the enable bit of polling mode. PSPLEN and NSPLEN cannot be enabled simultaneously. Both PSPLEN and NSPLEN enabled will not trigger the polling mode. So software must ensure that one of the two fields above is enabled.

This page gives an example of polling mode. Set ACMP positive input polling, polling frequency is source_clk/100, external channel 1-4 and DAC outputs are used as polling channel, ACMP negative input selects external input 0, falling edge trigger interrupt.

Step 1: IE = 1'b1, MOD = 2'b00;

Step 2: DACEN = 1'b1, DACVAL=value;

Step 3: PSPLEN = 1'b1, NSPLEN = 1'b0;

Step 4: PLFD = 2'b01, PLSEQ = 8'b10011110, NSEL = 3'b000;

Step 5: EN = 1'b1.

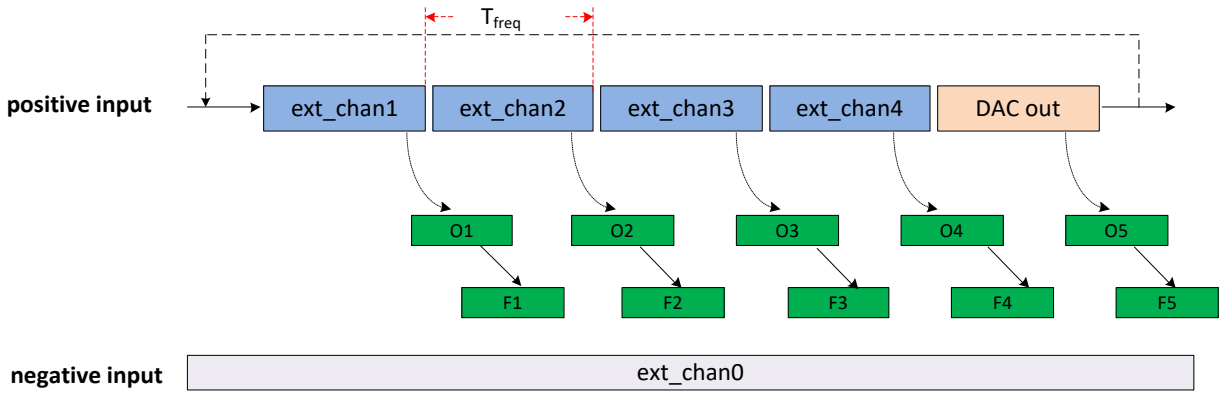

Figure 10-2 Operation flow in polling mode

10.4.3 HALL output in polling mode

ACMP has 3 hall outputs: Hall A Output, Hall B Output and Hall C Output. These signals are connected internally to PWDT. Hall output cooperates with polling function to obtain the position of the sensorless motor (The position of the rotor of the motor is detected by the BEMF). Every hall output can be selected one of the eight channels with polling mode.

For example, if polling mode PLSEQ = 8'b00001110, the polling sequence is external input 1- >external input 2 -> external input 3.

Set ACMP_OPA[OPA_SEL] = 3'b010, then Hall A Output is DR[O2].

10.4.4 Hysteresis

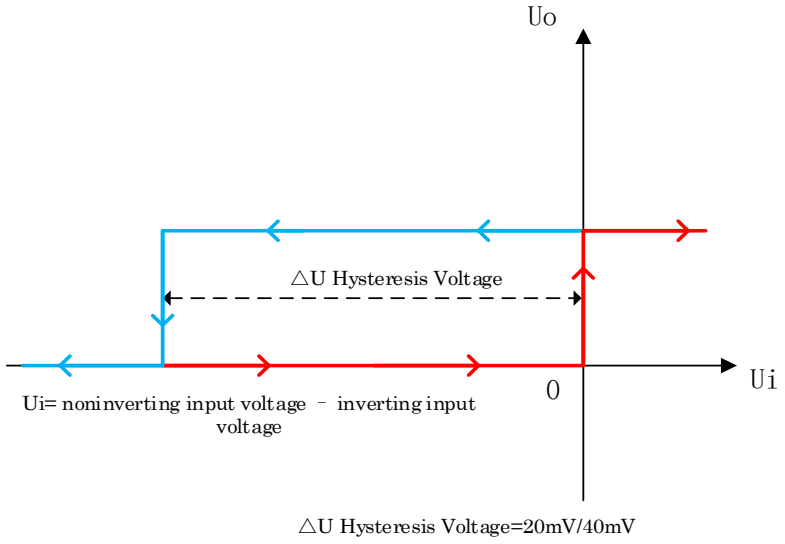

Figure 10-3 Hysteresis

10.4.5 Low power mode wakeup

In stop mode, a valid edge on ACMP output generates an asynchronous interrupt that can wake the MCU from low power mode. This interrupt can be cleared by writing 1 to WPF.

Note

The wake-up function only works in normal mode, and the polling mode does not work in low-power mode.

10.5 Register definition

Table 10-1 ACMP register mapping

ACMP0 base addr:0x40005000

Address	Name	Width (in bit)	Description
ACMPx base addr+0x0	ACMP_CR0	32	configuration register 0
ACMPx base addr +0x4	ACMP_CR1	32	configuration register 1
ACMPx base addr +0x8	ACMP_CR2	32	configuration register 2
ACMPx base addr +0xC	ACMP_CR3	32	configuration register 3
ACMPx base addr +0x10	ACMP_CR4	32	configuration register 4
ACMPx base addr +0x14	ACMP_DR	32	data output register
ACMPx base addr +0x18	ACMP_SR	32	status register
ACMPx base addr +0x1C	ACMP_FD	32	polling frequency divider register
ACMPx base addr +0x20	ACMP_OPA	32	hall A output set register
ACMPx base addr +0x24	ACMP_OPB	32	hall B output set register
ACMPx base addr +0x28	ACMP_OPC	32	hall C output set register
ACMPx base addr +0x2C	ACMP_DACSR	32	DAC reference select register
0x40008820	ACMP_ANACFG	32	Analog configuration register

10.5.1 ACMP_CR0

Table 10-2 ACMP_CR0 register

ACMP_CR0 **configuration register 0** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									EN			IE	OUTEN	OPE	MOD	
Type									RW			RW	RW	RW	RW	
Reset									0			0	0	0	0	

Bits	Description
7 EN	**ACMP Enable** 0: disable 1: enable
4 IE	**Interrupt enable** 0: disable 1: enable
3 OUTEN	**The comparison result output to the external PIN** 0: disable 1: enable
2 OPE	**hall output enable** 0: disable 1: enable
1: 0 MOD	**Interrupt trigger mode** 00: interrupt on output falling edge. 01: interrupt on output rising edge. 10: interrupt on output falling edge(00 and 10 are of the same configuration). 11: interrupt on output falling or rising edge.

10.5.2 ACMP_CR1

Table 10-3 ACMP_CR1 register

ACMP_CR1 **configuration register 1** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										PSEL				NSEL		
Type										RW				RW		
Reset										0				0		

Bits	Description
6: 4 PSEL	**Positive input select** 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: external input 5 110: external input 6 111: DAC output
2: 0 NSEL	**Negative input select** 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: external input 5 110: external input 6 111: DAC output

10.5.3 ACMP_CR2

Table 10-4 ACMP_CR2 register

ACMP_CR2 **configuration register 2** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DACEN		DACVAL					
Type									RW		RW					
Reset									0		0					

Bits	Description
7 DACEN	**DAC enable** 0: disable 1: enable
5: 0 DACVAL	**DAC output voltage**

10.5.4 ACMP_CR3

Table 10-5 ACMP_CR3 register

ACMP_CR3 **configuration register 3** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PSPLEN				NSPLEN			
Type									RW				RW			
Reset									0				0			

Bits	Description
7 PSPLEN	**Positive input polling mode enable** 0: disable 1: enable **Note:** PSPLEN and NSPLEN can't be enabled simultaneously. Both PSPLEN and NSPLEN enabled will not trigger the polling mode.
3 NSPLEN	**Negative input polling mode enable**

Bits	Description
	0: disable 1: enable **Note:** PSPLEN and NSPLEN can't be enabled simultaneously. Both PSPLEN and NSPLEN enabled will not trigger the polling mode.

10.5.5 ACMP_CR4

Table 10-6 ACMP_CR4 register

ACMP_CR4 **configuration register 4** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PLSEQ							
Type									RW							
Reset									0							

Bits	Description
7: 0 PLSEQ	**Polling channel sequence set** 0: disable corresponding channel 1: enable corresponding channel

10.5.6 ACMP_DR

Table 10-7 ACMP_DR register

ACMP_DR **data output register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								O	O7	O6	O5	O4	O3	O2	O1	O0
Type								RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset								0	0	0	0	0	0	0	0	0

Bits	Description
8 O	**Normal mode output**
7 O7	**Polling mode channel 7 output**

Bits	Description
6 O6	**Polling mode channel 6 output**
5 O5	**Polling mode channel 5 output**
4 O4	**Polling mode channel 4 output**
3 O3	**Polling mode channel 3 output**
2 O2	**Polling mode channel 2 output**
1 O1	**Polling mode channel 1 output**
0 O0	**Polling mode channel 0 output**

10.5.7 ACMP_SR

Table 10-8 ACMP_SR register

ACMP_SR **status register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							WPF	F	F7	F6	F5	F4	F3	F2	F1	F0
Type							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset							0	0	0	0	0	0	0	0	0	0

Bits	Description
9 WPF	**Low power mode wakeup interrupt flag** write 1 clear the flag
8 F	**Normal mode interrupt flag** write 1 clear the flag
7 F7	**Polling mode channel 7 interrupt flag** write 1 clear the flag
6 F6	**Polling mode channel 6 interrupt flag** write 1 clear the flag
5 F5	**Polling mode channel 5 interrupt flag** write 1 clear the flag
4	**Polling mode channel 4 interrupt flag**

Bits	Description
F4	
	write 1 clear the flag
3	**Polling mode channel 3 interrupt flag**
F3	
	write 1 clear the flag
2	**Polling mode channel 2 interrupt flag**
F2	
	write 1 clear the flag
1	**Polling mode channel 1 interrupt flag**
F1	
	write 1 clear the flag
0	**Polling mode channel 0 interrupt flag**
F0	
	write 1 clear the flag

10.5.8 ACMP_FD

Table 10-9 ACMP_FD register

ACMP_FD **polling frequency divider register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															PLFD	
Type															RW	
Reset															0	

Bits	Description
1: 0	**Polling mode frequency divider**
PLFD	
	This divider controls the switch frequency of polling channels 00: source_clk/256 01: source_clk/100 10: source_clk/70 11: source_clk/50

10.5.9 ACMP_OPA

Table 10-10 ACMP_OPA register

ACMP_OPA **Hall A output set register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														OPA_SEL		
Type														RW		
Reset														0		

Bits	Description
2: 0 OPA_SEL	**Hall A output set** 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3 100: polling channel 4 101: polling channel 5 110: polling channel 6 111: polling channel 7

10.5.10 ACMP_OPB

Table 10-11 ACMP_OPB register

ACMP_OPB **Hall B output set register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														OPB_SEL		
Type														RW		
Reset														0		

Bits	Description
2: 0 OPB_SEL	**Hall B output set** 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3

Bits	Description
	100: polling channel 4 101: polling channel 5 110: polling channel 6 111: polling channel 7

10.5.11 ACMP_OPC

Table 10-12 ACMP_OPC register

ACMP_OPC **Hall C output set register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														OPC_SEL		
Type														RW		
Reset														0		

Bits	Description
2: 0 OPC_SEL	**Hall C output set** 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3 100: polling channel 4 101: polling channel 5 110: polling channel 6 111: polling channel 7

10.5.12 ACMP_DACSR

Table 10-13 ACMP_DACSR register

ACMP_DACSR **DAC reference select register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																DACREF
Type																RW
Reset																0

Bits	Descriptions
0 DACREF	**DAC reference select** 0: The DAC selects bandgap as the reference source 1: The DAC selects Vdd as the reference source

10.5.13 ACMP_ANACFG

Table 10-14 ACMP_ANACFG register

ACMP_ANACFG **ANA configuration register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									HYST	LPFSEL						
Type									RW	RW						
Reset									0	11						
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																
Type																
Reset																

Bits	Description
23 HYST	**Analog comparator hysteresis voltage selection** 0: 20mV 1: 40mV
22: 21 LPFSEL	**LPF Select** **It is recommended to select 1MHz for better performance** 00: 200kHz 01: 500kHz 10: 750kHz 11: 1MHz

11 PWM

11.1 Introduction

The PWM module is a multi-functional timer that supports input capture, output compare, quadrature decoder mode and the generation of PWM signals to control electric motor and power management applications.

The device contains two PWM modules, each PWM module supports eight channels.

11.2 Features

The PWM features include:

- PWM source clock is selectable. Source clock can be the bus clock, or HSI clock.
- 16-bit Prescaler divide-by 1 to 65536.
- 16-bit counter.
  - It can be a free-running counter or a counter with initial and final value.
  - The counting can be up or up-down.
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode, center-aligned PWM mode.
- In Input Capture mode, the capture can occur on rising edges, falling edges or both edges.
- An input filter can be selected for some channels.
- In Output Compare mode, the output signal can be set, cleared, or toggled on match.
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal.
- The PWM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs.
- The dead-time insertion is available for each complementary pair.
- Generation of match triggers.
- Software control of PWM outputs.
- Output mask can set the channel to invalid state.
- Up to 3 fault inputs for global fault control.
- The polarity of each channel is configurable.
- The generation of an interrupt per channel.
- The generation of an interrupt when the counter overflows.

- The generation of an interrupt when the fault condition is detected.
- Synchronized loading of write buffered PWM registers.
- Write protection for critical registers.
- Dual edge capture for pulse and period width measurement.
- Supports Quadrature decoder (The input pins of phase A and B are mapped to CH0 and CH1 of each PWM module).
- Supports global time base.
- Supports PWM output waveform phase shift.

11.3 Block diagram

The PWM uses one input/output (I/O) pin per channel, CHn (PWM channel (n)), where n is the channel number (0 to 7).

The following figure shows the PWM structure. The central component of the PWM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

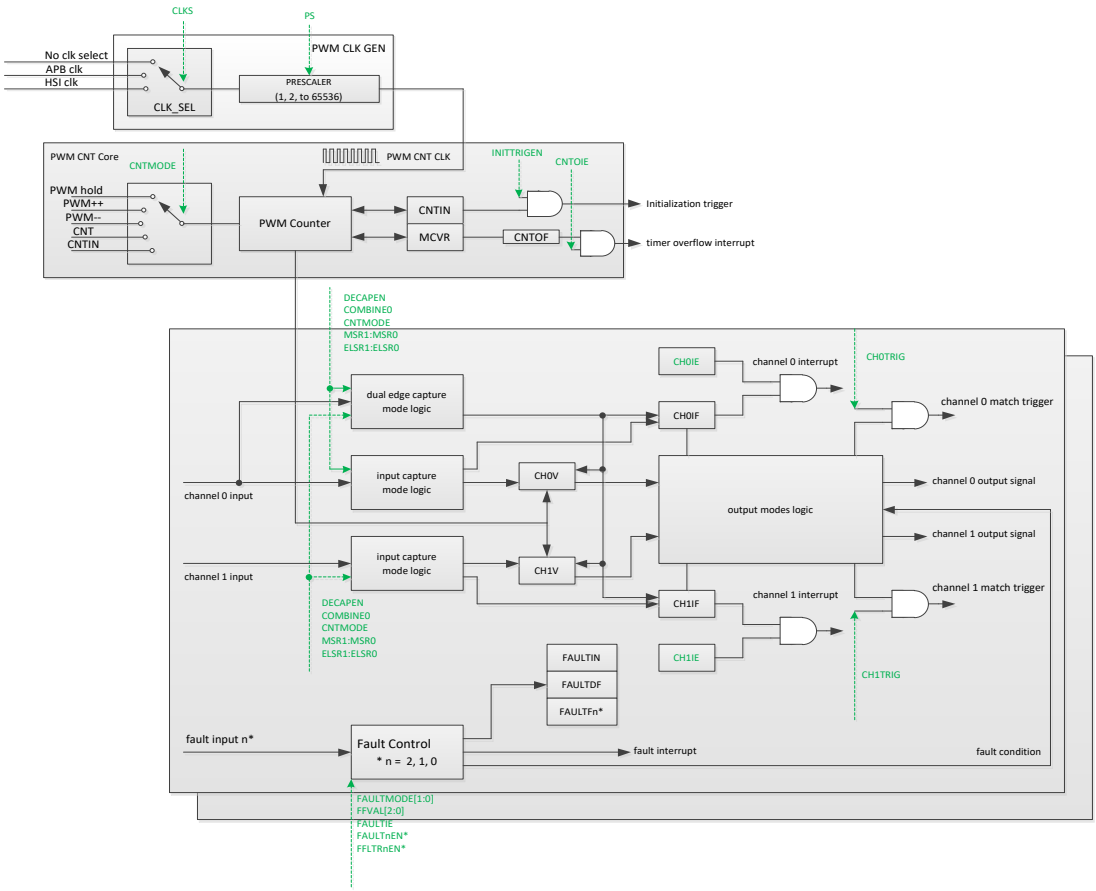

Figure 11-1 PWM block diagram

11.4 Functional description

11.4.1 Clock source

The CLKSRC[1:0] bits in the PWM_INIT register select one of two possible clock sources for the PWM counter or disable the PWM counter. After any MCU reset, CLKSRC[1:0] = 00, so no clock source is selected. Disabling the PWM counter by writing 00 to the CLKSRC[1:0] bits does not affect the PWM counter value or other registers.

The fixed frequency clock is an internal HSI clock source for the PWM counter that allows the selection of a clock other than the system clock or an external clock. This clock frequency equals to 8 MHz.

11.4.2 Counter

The PWM has a 16-bit counter that is used by the channels either for input or output modes. The PWM counter clock is the selected clock divided by the prescaler. The PWM counter has these modes of operation:

- Up counting.
- Up-down counting.
- Quadrature decoder mode.

11.4.2.1 Up counting

Up counting is selected when (QDIEN=0) and (CNTMODE = 0). CNTIN defines the starting value of the count and MCVR defines the final value of the count, see the following figure. The value of CNTIN is loaded into the PWM counter, and the counter increments until the value of MCVR is reached, at which point the counter is reloaded with the value of CNTIN. The PWM period when using up counting is (MCVR – CNTIN + 0x0001) × period of the PWM counter clock. The CNTOF bit is set when the PWM counter changes from MCVR to CNTIN.

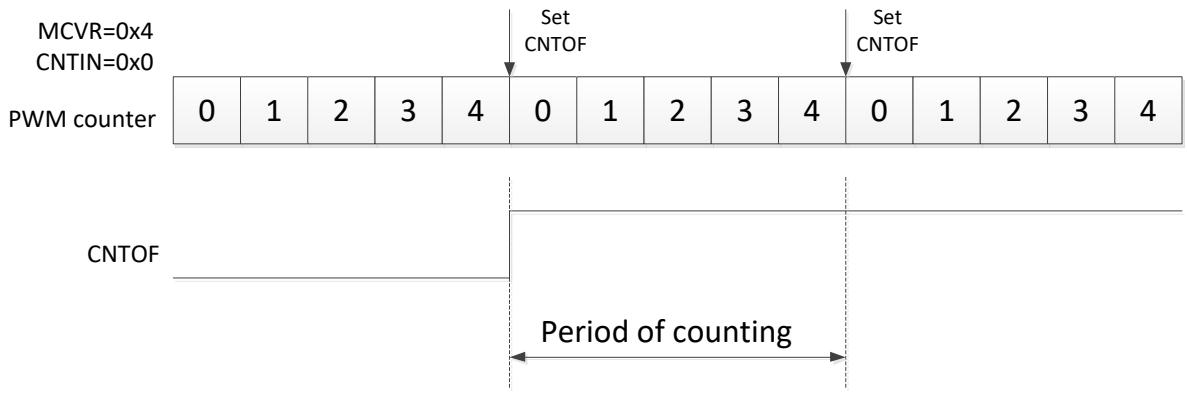

Figure 11-2 Up counting

11.4.2.2 Up-Down counting

Up-down counting is selected when (QDIEN = 0) and (CNTMODE = 1). CNTIN defines the starting value of the count and MCVR defines the final value of the count. The value of CNTIN is loaded into the PWM counter, and the counter increments until the value of MCVR is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The PWM period when using up-down counting is 2 × (MCVR – CNTIN) × period of the PWM counter clock. The CNTOF bit is set when the PWM counter changes from MCVR to (MCVR – 1). See the following figure.

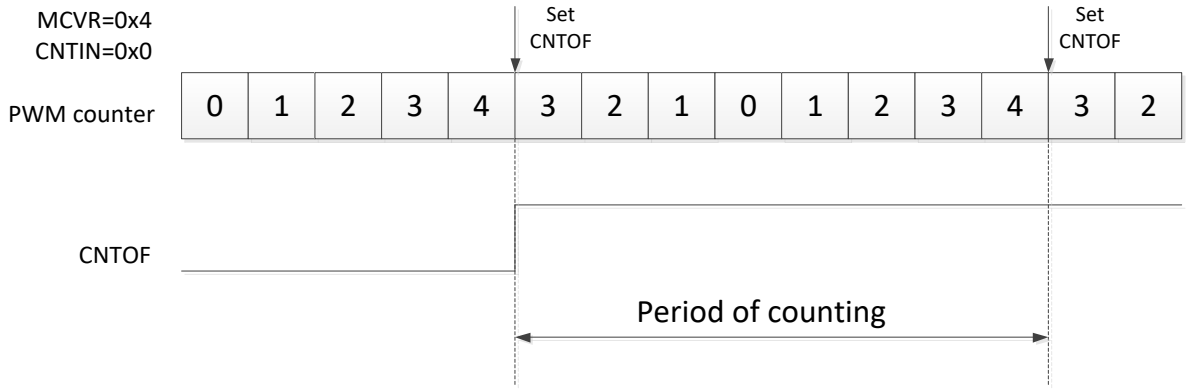

Figure 11-3 Up-Down counting

11.4.3 Operation mode

PWM can be configured for input capture, output compare, or edge-aligned PWM mode, center-aligned PWM mode, combination mode, Quadrature decoder mode. Please refer to the following table in detail.

Table 11-1 Operation mode configuration

DECAPEN	COMBINE	CNTMODE	MSR1: MSR0	ELSR1: ELSR0	Operation mode	Configuration
0	0	0	00	01	Input capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			01	01	Output compare	Toggle Output on match

DECAPEN	COMBINE	CNTMODE	MSR1: MSR0	ELSR1: ELSR0	Operation mode	Configuration
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-aligned PWM	Clear Output on match
				X1		Set Output on match
		1	XX	10	Center-aligned PWM	Clear Output on match-up
				X1		Set Output on match-up
	1	0	XX	10	Combine mode + Up counting	Set on channel (n) match, and clear on channel (n+1) match
				X1		Clear on channel (n) match, and set on channel (n +1) match
		1		10	Combine mode + Up-down counting	Set on channel (n) match, and clear on channel (n+1) match
				X1		Clear on channel (n) match, and set on channel (n +1) match
1	0	0	X0	01	Dual edge single capture	Rising Edge
				10		Falling Edge
				11		Rising or Falling Edge
			X1	01	Dual edge continuous capture	Rising Edge
				10		Falling Edge
				11		Rising or Falling Edge

DECAPEN	COMBINE	CNTMODE	MSR1: MSR0	ELSR1: ELSR0	Operation mode	Configuration
X	X	X	XX	XX	Quadrature decoder	QDIEN=1,enable Quadrature Decoder, which is prior to the other modes.

11.4.4 Input capture mode

When a selected edge occurs on the channel input, the current value of the PWM counter is captured into the PWM_CHnV register. At the same time, the CHnIF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. When a channel is configured for input capture, the PWMx_CHn pin is an edge-sensitive input. ELSnR1:ELSnR0 control bits determine which edge, falling or rising, triggers input capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling. Writes to the CHnV register is ignored in Input Capture mode.

The following figure shows the channel rising edge capture.

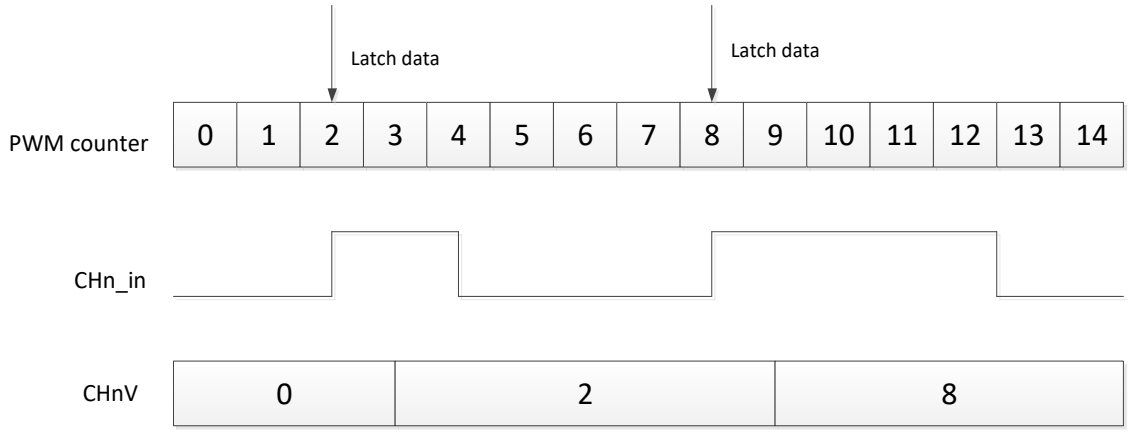

Figure 11-4 Input capture mode

For PWM Channel0 to Channel3, there exist additional capture filter to make input signal clean. The filter is 5-bit counter and can be configured through register CHnCAPFVAL(n = 0,1,2,3). When the CHnCAPFVAL[4:0] = 0, the filter function is disabled, if CHnCAPFVAL[4:0] ≠00000, the input signal will be delayed CHnCAPFVAL[4:0] × 4 system Clock, and then be transported to the input capture function.

The clock of the counter in the channel input filter is divided by 4 of the bus clock.

11.4.5 Output Compare mode

In Output Compare mode, the PWM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHnV register of an

output compare channel, the channel (n) output can be set, cleared, or toggled. When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs. The CHnIF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel(n) match (PWM counter = CHnV).

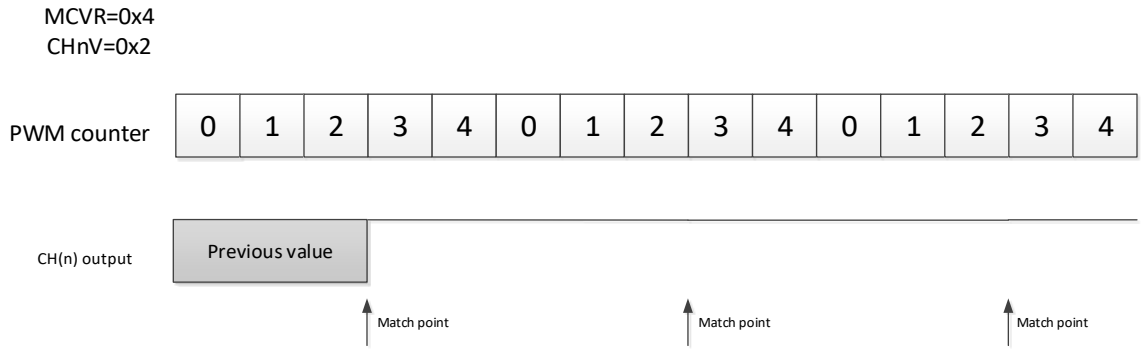

Figure 11-5 Match setting output compare mode

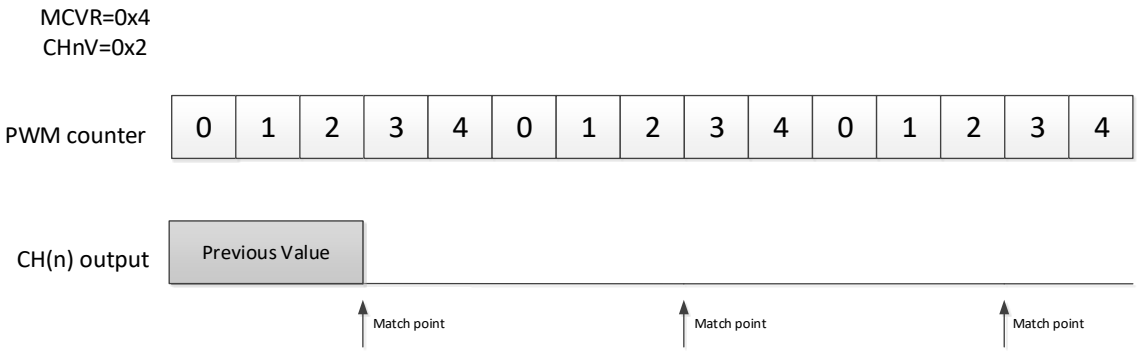

Figure 11-6 Match clear output compare mode

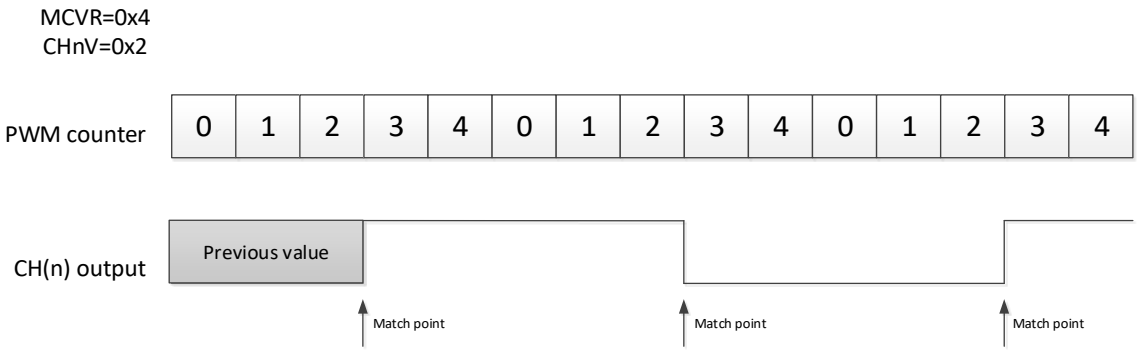

Figure 11-7 Match convert output compare mode

11.4.6 Edge-Aligned PWM (EPWM) mode

The EPWM period = (MCVR - CNTIN + 0x0001) × PWM timer clock cycle.

The pulse width (duty cycle) = (CHnV + 0x0001 - CNTIN) × PWM timer clock cycle.

ELSnR1: ELSnR0 = 1:0, the output Channel (n) output is forced high when the CNTIN value is loaded into the PWM counter, and forced low when channel (n) matches (PWM counter = CHnV).

ELSnR1: ELSnR0 = X:1, Channel (n) output is forced low when the CNTIN value is loaded into the PWM counter, and forced high when channel (n) matches (PWM counter = CHnV).

CHnIE = 1, When the channel (n) matches (PWM counter = CHnV), the CHnIF bit is set and the channel (n) interrupt is generated.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an PWM.

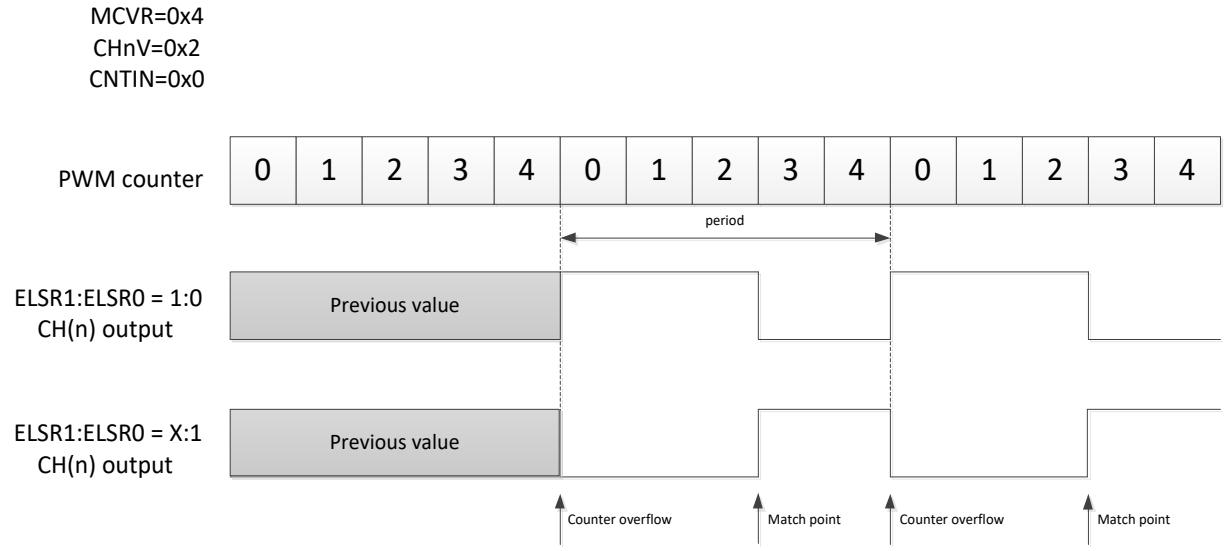

Figure 11-8 Edge-aligned PWM mode

ELSnR1: ELSnR0= 1:0, if (CHnV = 0x0000), the channel (n) output is 0% duty cycle; If (CHnV >= MCVR), the channel (n) output is 100% duty cycle. ELSnR1: ELSnR0=X: 1, the reverse happens.

11.4.7 Center-Aligned PWM(CPWM) mode

The CPWM period = 2 × (MCVR - CNTIN) × PWM timer clock cycle.

The CPWM pulse width (duty cycle) = 2 × (CHnV - CNTIN) × PWM timer clock cycle.

In CPWM mode, the PWM counter counts up until MCVR is reached, and then counts down until CNTIN is reached.

ELSnR1: ELSnR0 = 1:0, the Channel (n) output is forced to high level when it matches channel (n) while down counting (PWM counter = CHnV), and forced low when matching with channel (n) while counting up (PWM counter = CHnV).

ELSnR1: ELSnR0 = X:1, Channel (n) output is forced low when it matches channel (n) while down counting (PWM counter = CHnV), and forced high when matching with channel (n) while up counting (PWM counter = CHnV).

The CHnIF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (PWM counter = CHnV) when the PWM counting is down (at the begin of the pulse width) and when the PWM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

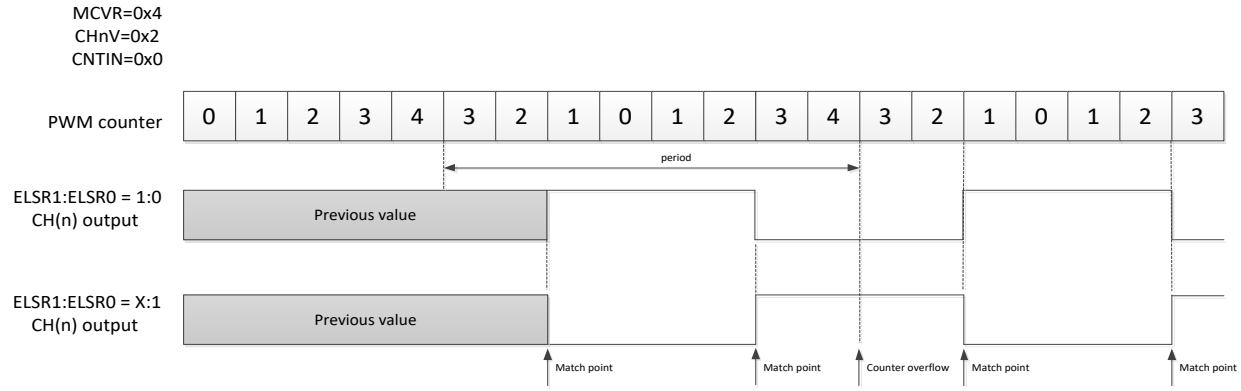

Figure 11-9 Center-aligned PWM mode

ELSnR1: ELSnR0= 1:0, if (CHnV = 0x0000), the channel (n) output is 0% duty cycle; If (CHnV >= MCVR), the channel (n) output is 100% duty cycle. ELSnR1: ELSnR0=X: 1, the reverse happens.

11.4.8 Combine mode

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

According to the counting methods, it can be divided into up counting combination mode and up-down counting combination mode.

If (ELSnR1:ELSnR0 = 1:0), then the channel (n) output is forced high at the beginning of the period (PWM counter = CNTIN) and at the channel (n+1) match (PWM counter = CH(n+1)V). It is forced low at the channel (n) match (PWM counter = CH(n)V).

If (ELSnR1:ELSnR0 = X:1), then the channel (n) output is forced low at the beginning of the period (PWM counter = CNTIN) and at the channel (n+1) match (PWM counter = CH(n+1)V). It is forced high at the channel (n) match (PWM counter = CH(n)V).

If CH(n)IE = 1, when the channel (n) matches (PWM counter = CH(n)V), the CH(n)IF bit is set and the channel (n) interrupt is generated.

If CH(n+1)IE = 1, when the channel (n+1) matches (PWM counter = CH(n+1)V), the CH(n+1)IF bit is set and the channel (n+1) interrupt is generated.

Note

ELS(n+1)R1 and ELS(n+1)R0 cannot be used to control the output of the channels (n) and (n+1) .

11.4.8.1 Up counting combine mode

Period = (MCVR - CNTIN + 0x0001) × PWM timer clock period.

Pulse width (duty cycle) = |CH(n+1)V - CH(n)V|× PWM timer clock period.

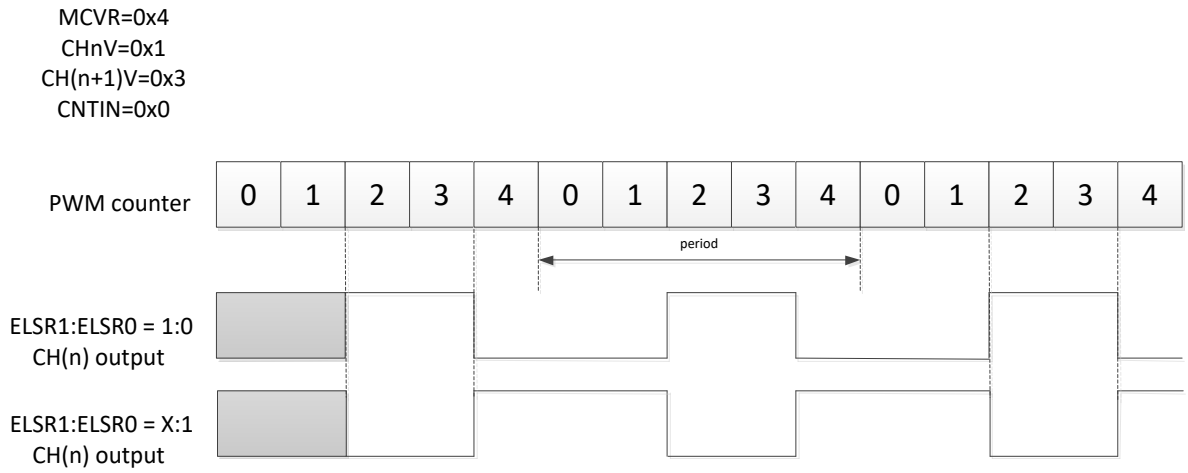

Figure 11-10 CH(n) output in up counting mode

The following figures show the PWM signal output waveforms under various conditions of the up counting combine mode.

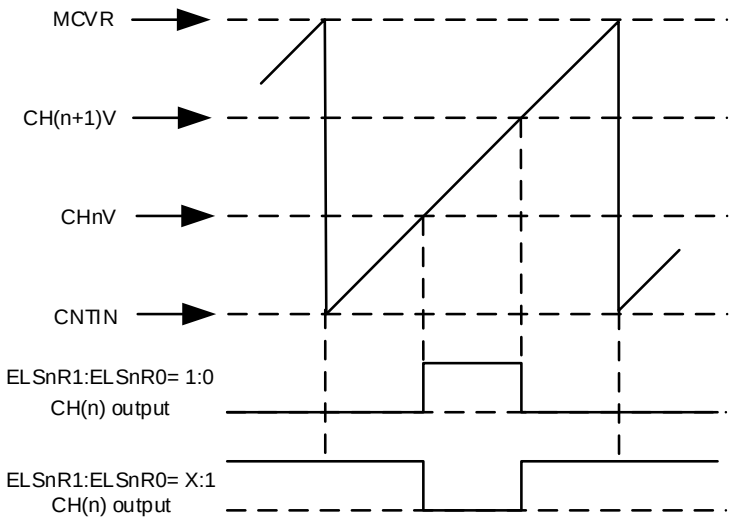

Figure 11-11 CH(n) output if (CNTIN < CHnV/CH(n+1)V < MCVR) &(CHnV < CH(n+1)V)

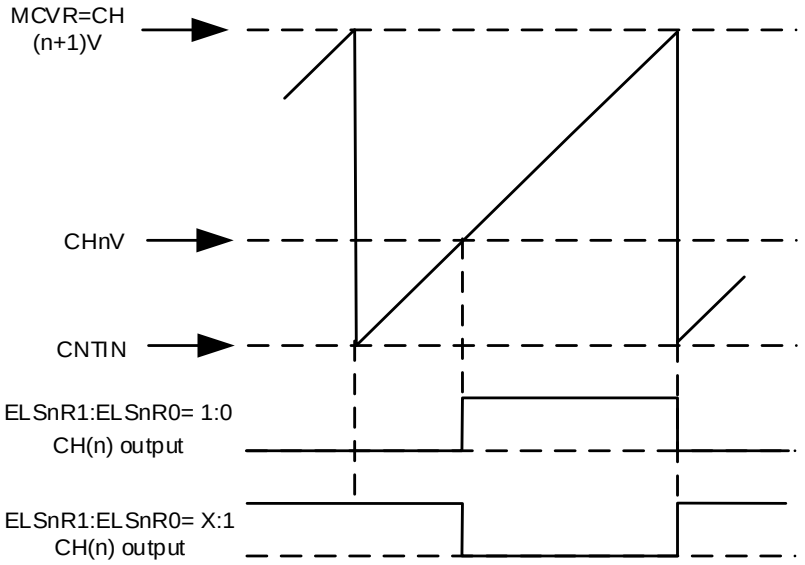

Figure 11-12 CH(n) output if (CNTIN < CHnV < MCVR)&(CH(n+1)V = MCVR)

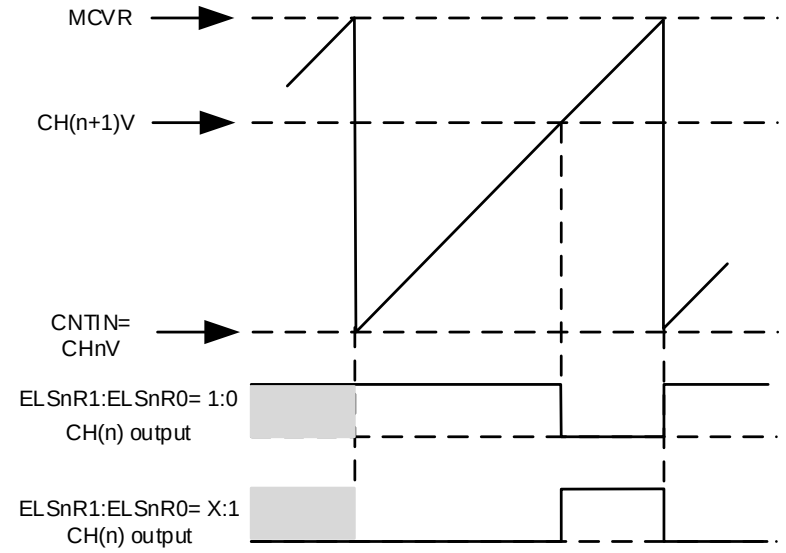

Figure 11-13 CH(n) output if (CHnV = CNTIN)&(CNTIN < CH(n+1)V < MCVR)

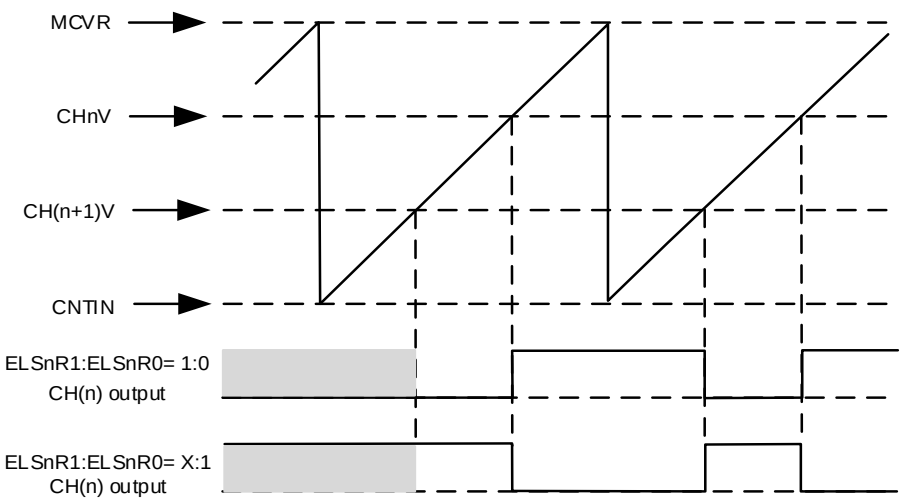

Figure 11-14 CH(n) output if (CNTIN < CHnV/ CH(n+1)V < MCVR) & (CHnV > CH(n+1)V)

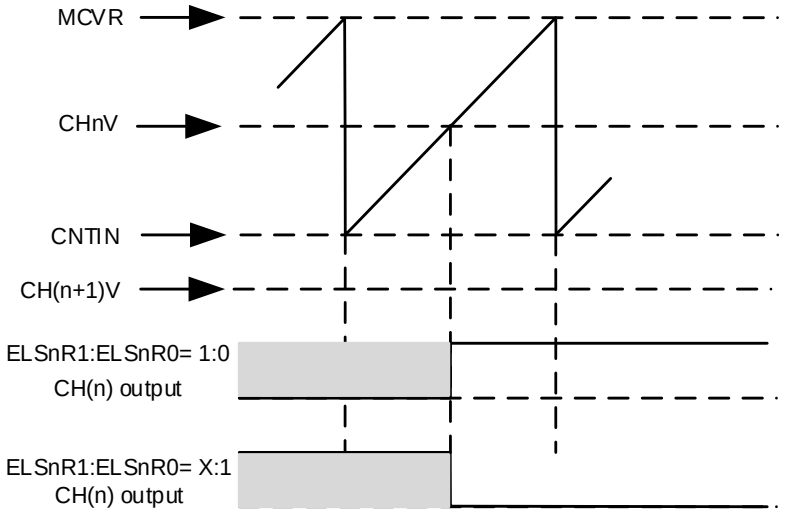

Figure 11-15 CH(n) output if (CH(n+1)V < CNTIN)&(CNTIN < CHnV < MCVR)

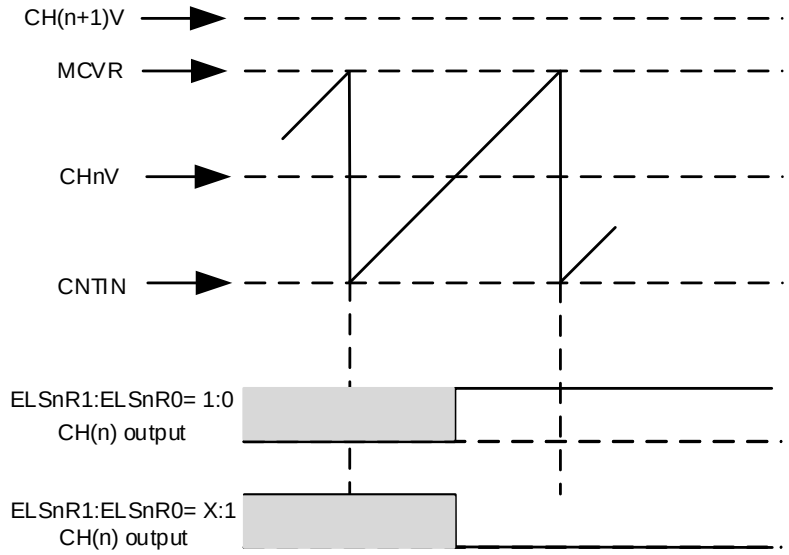

Figure 11-16 CH(n) output if (CH(n+1)V > MCVR) & (CNTIN < CHnV < MCVR)

11.4.8.2 Up-down counting combine mode

Period = 2 × (MCVR - CNTIN) × PWM timer clock cycle

During the up-down counting, the channel generates a match while up counting, and the down counting also generates a match.

In order to facilitate the control of the channel output, the matching effective point setting function is provided, and the setting of the matching effective point CHSCR[DIR] can be used in the process of up counting or down counting. The matching effective point depends on the counting direction, it is necessary to clearly define the range of the up counting and down counting interval, as is shown in the following chart.

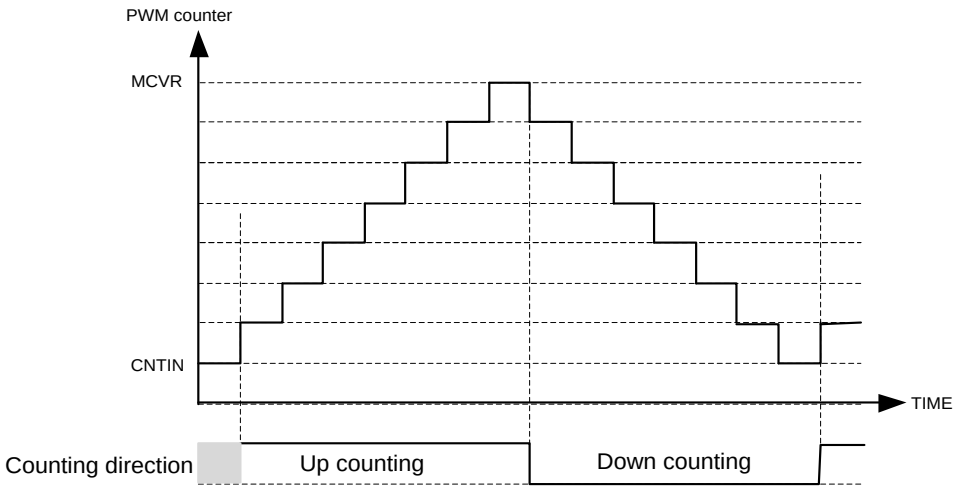

Figure 11-17 Up-down counting range define

Note

The CNTIN value in the first counting period is the upward counting interval, and the CNTIN value in the subsequent periods is the downward counting interval.

Two matching points can be combined into the following 4 situations:

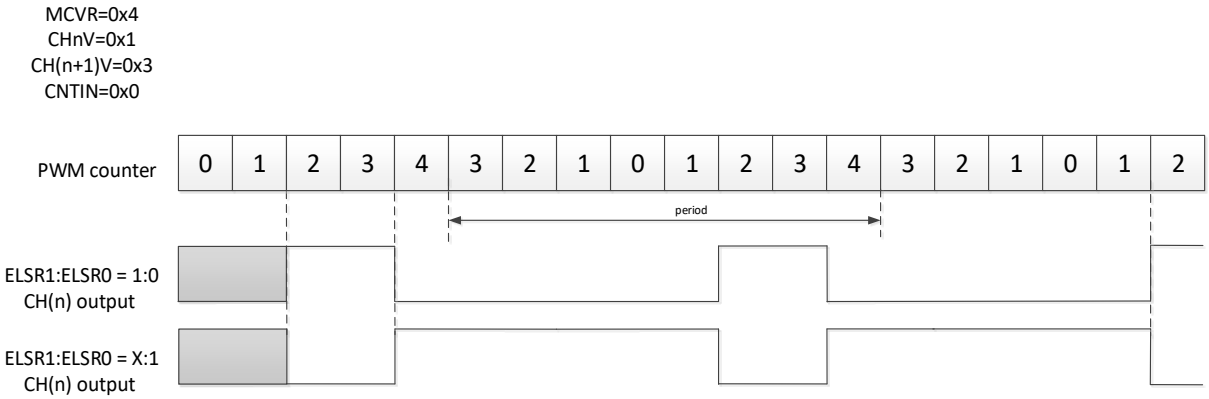

Figure 11-18 CHn matching point DIR=1(Up), CH(n+1) matching point DIR=1(Up)

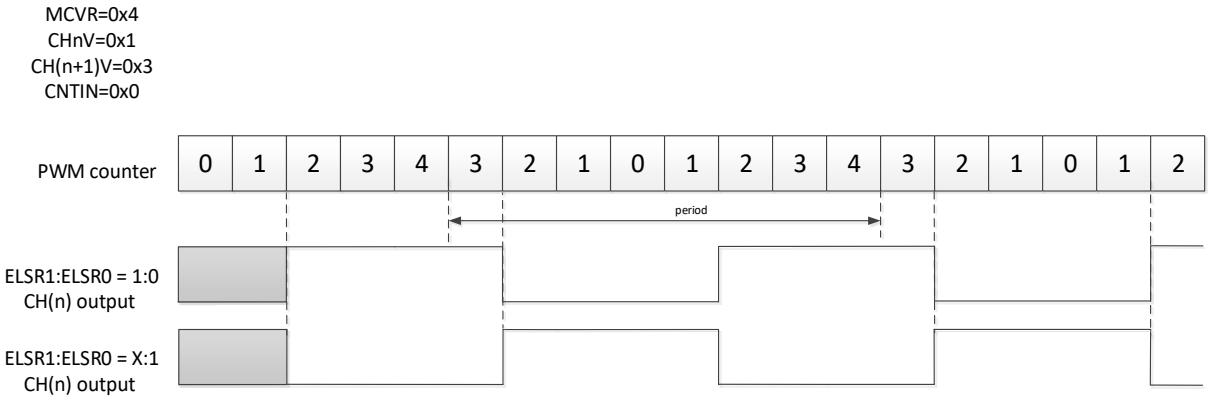

Figure 11-19 CHn matching point DIR=1(Up), CH(n+1) matching point DIR=0(Down)

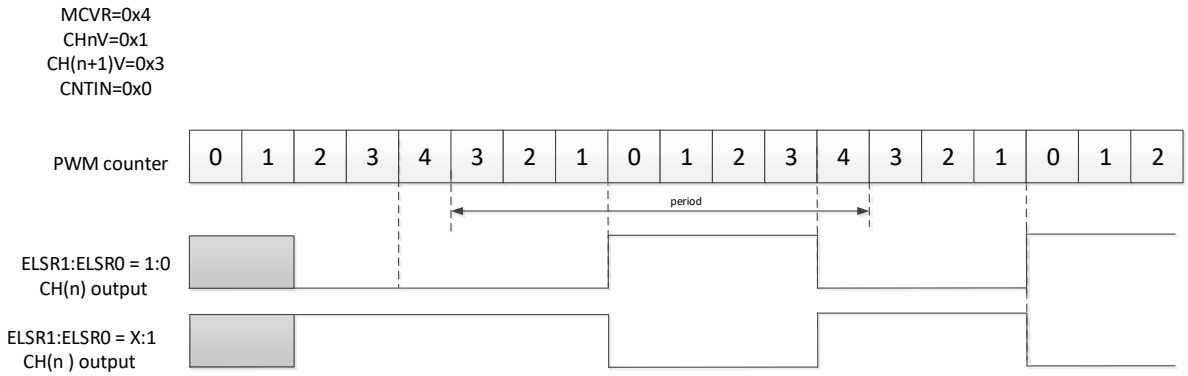

Figure 11-20 CHn matching point DIR=0(Down), CH(n+1) matching point DIR=1(Up)

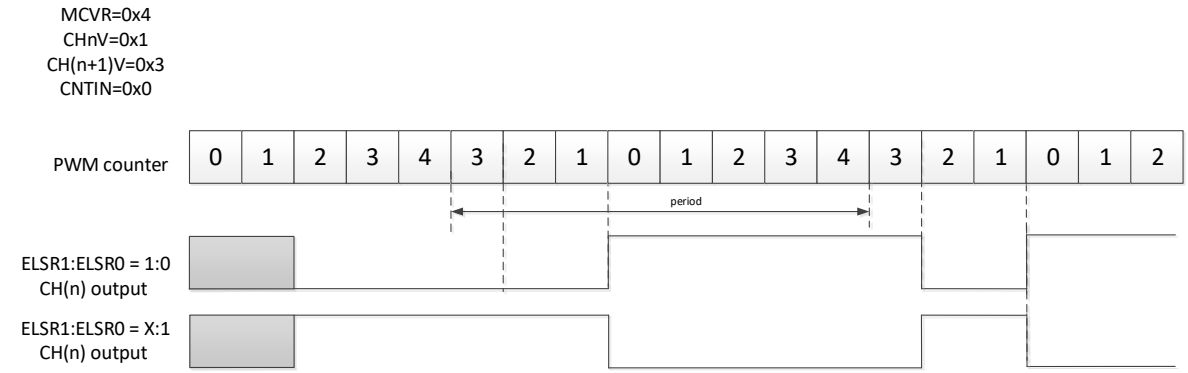

Figure 11-21 CHnV matching point DIR=0(Down), CH(n+1)V matching point DIR=0(Down)

11.4.8.3 Complementary mode

The Complementary mode is supported in the combine mode. By enabling PAIRnCOMPEN=1 in Complementary mode, the channel (n+1) output is the inverse of the channel (n)output.

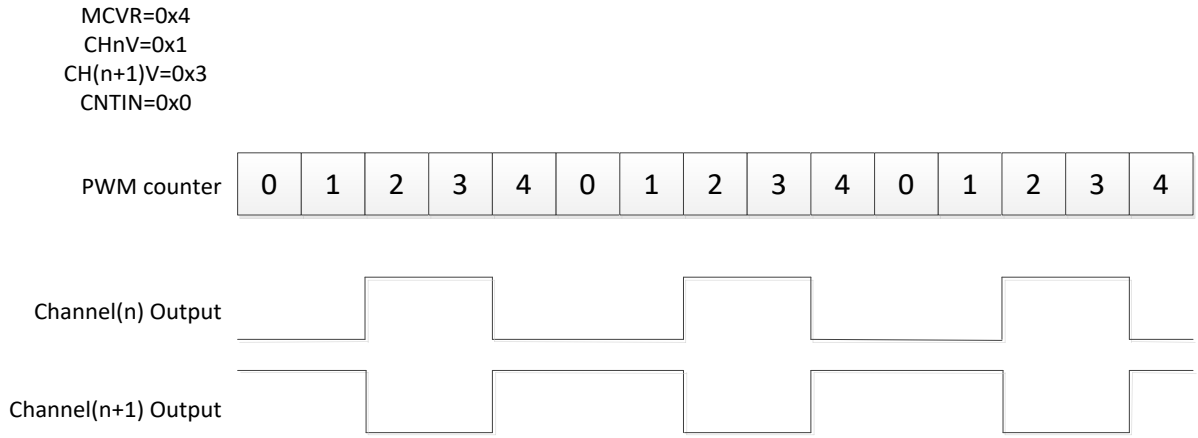

Figure 11-22 Output in complementary mode

11.4.8.4 Dead-time insertion

The dead-time insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero). PWM_DTSET register defines the dead-time delay that can be used for all PWM channels. The DTPSC[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the dead-time modulo, that is, the number of the dead-time prescaler clocks. The dead-time delay insertion ensures that no two complementary signals (channels (n)and (n+1)) drive the active state at the same time.

If CH(n)POL = 0, CH(n+1)POL = 0, and the dead-time is enabled, then when the channel (n) match (PWM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the dead-time delay when the channel (n) output is set. Similarly, when the channel (n+1) match (PWM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the dead-time delay when the channel (n+1) output is set.

If CH(n)POL = 1, CH(n+1)POL = 1, and the dead-time is enabled, then when the channel (n)match (PWM counter = CH(n)V) occurs, the channel (n) output remains at the high value until the end of the dead-time delay when the channel (n) output is cleared. Similarly, when the channel (n+1) match (PWM counter = CH(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the dead-time delay when the channel (n+1) output is cleared.

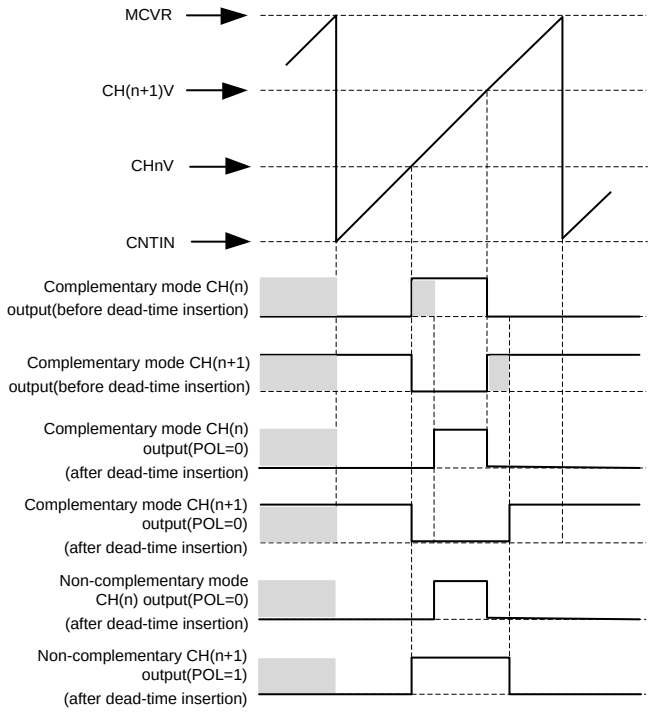

Figure 11-23 Dead-time insertion

11.4.8.5 Inverting

The Invert functionality swaps the signals between channel(n) and channel(n+1) outputs. The inverting operation is selected when PAIR(n)INVEN = 1, n=0,1,2,3.

11.4.8.6 Phase shift

In some cases, channel (n+1) matching will occur in the next PWM cycle. The phase shift occurs nex cycle when (CNTIN < CHnV < MCVR) & (CNTIN < CH(n+1)V < MCVR) & (CHnV > CH(n+1)V).

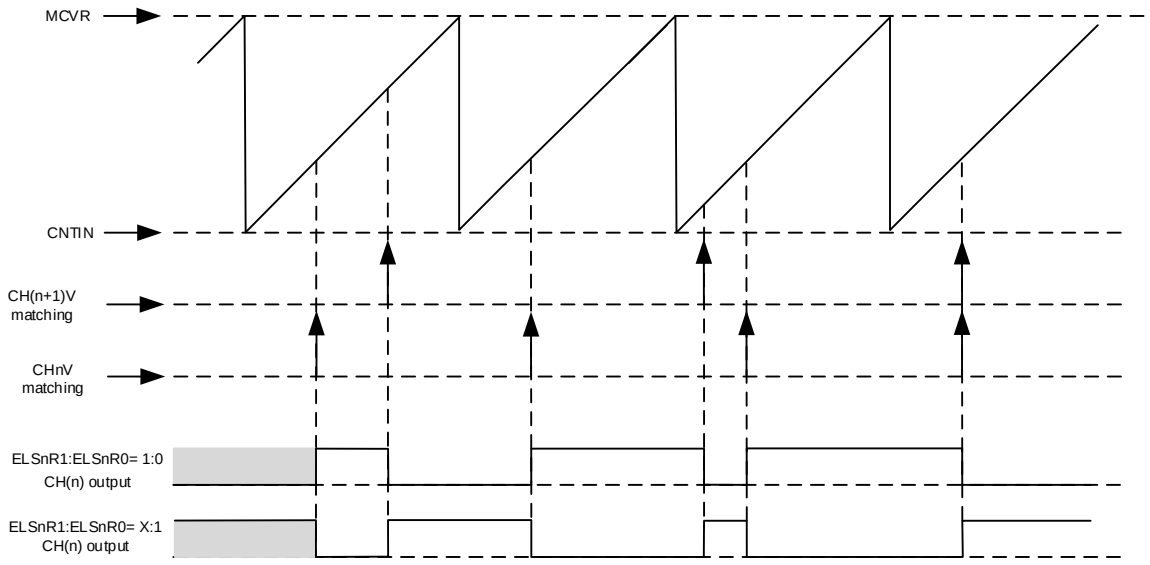

Figure 11-24 CH(n+1)V output in the next period matching

Based on the above features, when multiple channels of the PWM module output in combine mode, the phase shift between different channel pairs of the PWM module can be set. This behavior is useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other pair to help eliminate noise generation.

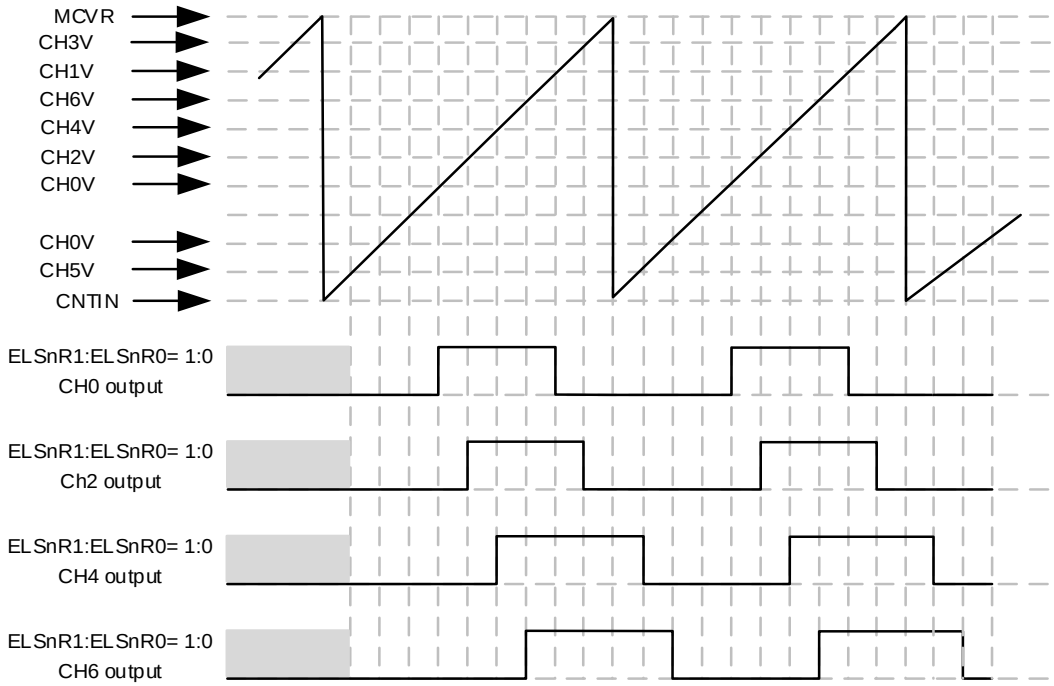

Figure 11-25 Multi-channel phase shift output waveform

11.4.9 Dual Edge Capture mode

The Dual edge capture mode is selected if DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n). In this mode, only channel (n) input is used and channel (n+1) input is ignored. The channel (n) filter can be active in this mode when n is 0 or 2. The ELS(n)R1:ELS(n)R0 bits select the edge that is captured by channel (n), and ELS(n+1)R1:ELS(n+1)R0 bits select the edge that is captured by channel (n+1). If both ELS(n)R1:ELS(n)R0 and ELS(n+1)R1:ELS(n+1)R0 bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)IF bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)IF = 1), then CH(n+1)IF bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1). The Dual edge capture does not support enabling channel (n) and channel (n+1) interrupts simultaneously.

The PWM_CH(n)V register stores the value of PWM counter when the selected edge by channel(n) is detected at channel (n) input. The PWM_CH(n+1)V register stores the value of PWM counter when the selected edge by channel (n+1) is detected at channel (n) input. In this mode, a coherency mechanism ensures coherent data when the PWM_CH(n)V and PWM_CH(n+1)V registers are read. The only requirement is that CH(n)V must be read before CH(n+1)V.

As shown in Figure 11-26, channel (n) selects rising edge capture and channel (n+1) selects falling edge capture for pulse width measurement.

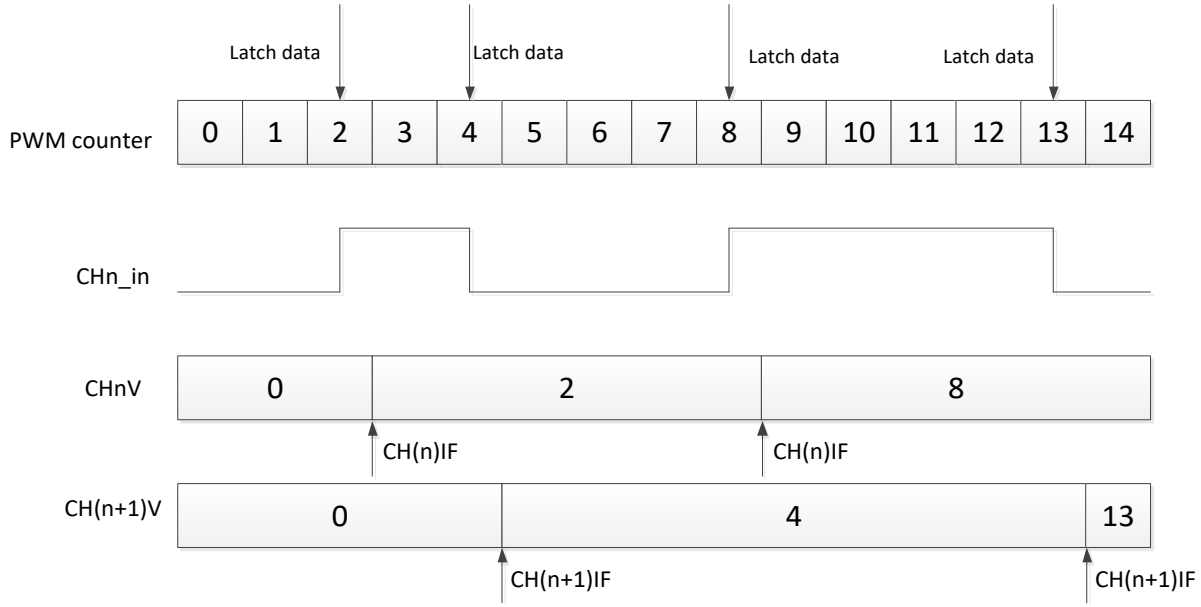

Figure 11-26 Dual edge capture mode

11.4.10 Quadrature decoder mode

The quadrature decoder mode is enabled if QDIEN = 1. The quadrature decoder mode uses the input signals phase A and B to control the PWM counter increment and decrement. Each one of input signals phase A and B has a filter that is equivalent to the channel input filter. The phase A input filter value is defined by CH0CAPFVAL[4:0] bits. The phase B input filter value is defined by CH1CAPFVAL[4:0] bits. The filter is disabled when its value is 0(the bit CH(n)CAPFVAL[4:0] is in the WM_CAPFILTER register).

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input. The QUADMODE selects the encoding mode used in the quadrature decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled, see the following figure. In this mode, the phase B input value indicates the counting direction, and the phase A input defines the counting rate. The PWM counter is updated when there is a rising edge at phase A input signal.

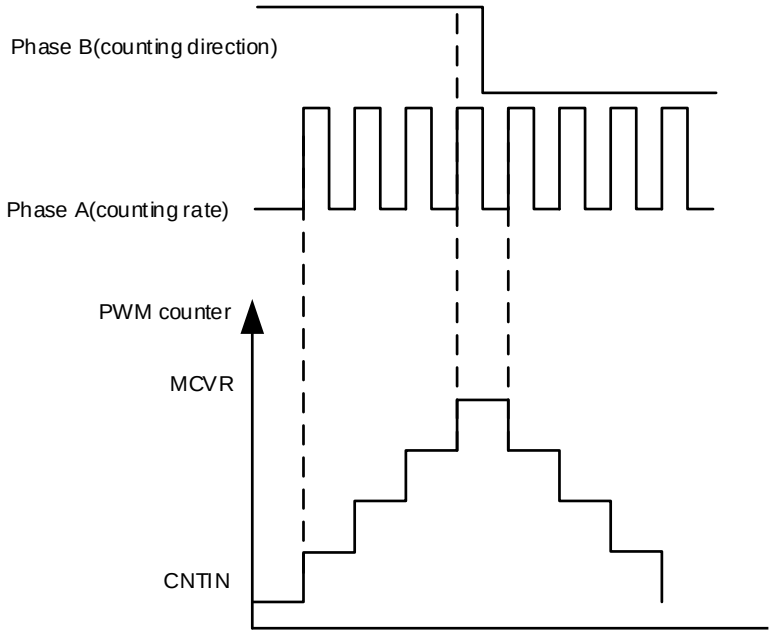

Figure 11-27 Quadrature Decoder—Count and Direction Encoding mode

If QUADMODE = 0, then the phase A and phase B Encoding mode is enabled, see the following figure. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The PWM counter is updated when there is an edge either at the phase A or phase B signals.

If PHAPOL = 0 and PHBPOL = 0, then the PWM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one.

and the PWM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

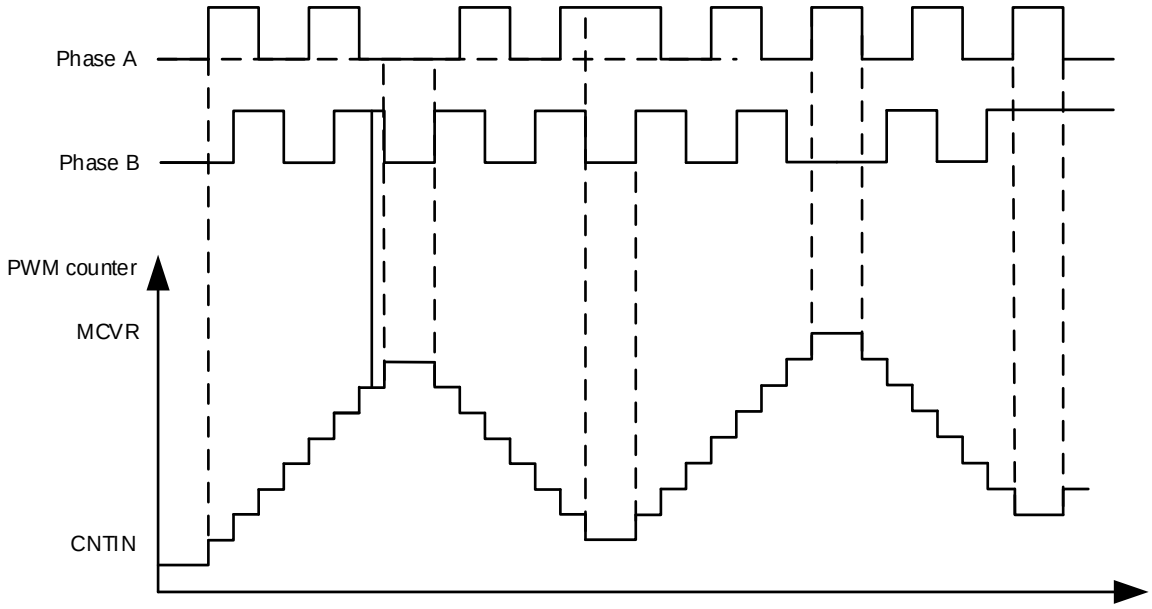

Figure 11-28 Quadrature Decoder—Phase A and Phase B Encoding Mode

The following figure shows the PWM counter overflow in up counting. In this case, when the PWM counter changes from MCVR to CNTIN, CNTOF and CNTOFDIR bits are set. CNTOF bit indicates the PWM counter overflow occurred. CNTOFDIR indicates the counting was up when the PWM counter overflow occurred.

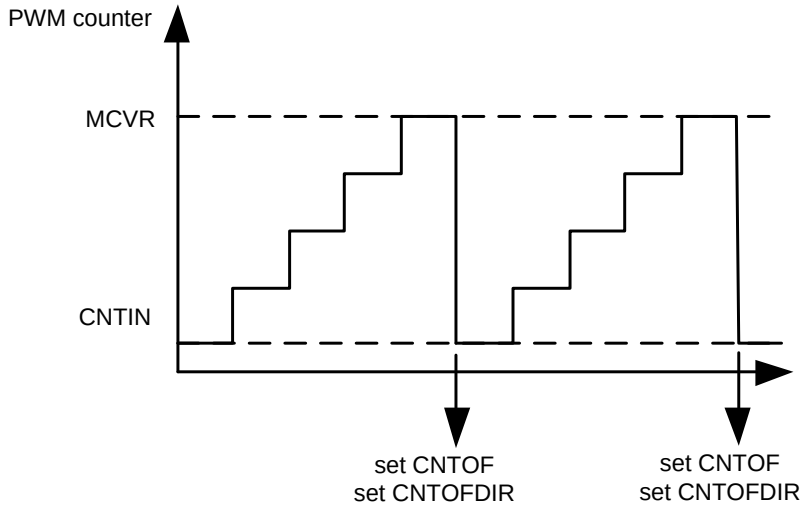

Figure 11-29 PWM Counter Overflow in Up Counting for Quadrature Decoder Mode

The following figure shows the PWM counter overflow in down counting. In this case, when the PWM counter changes from CNTIN to MCVR, CNTOF bit is set and CNTOFDIR bit is cleared. CNTOF bit indicates the PWM counter overflow occurred. CNTOFDIR indicates the counting was down when the PWM counter overflow occurred.

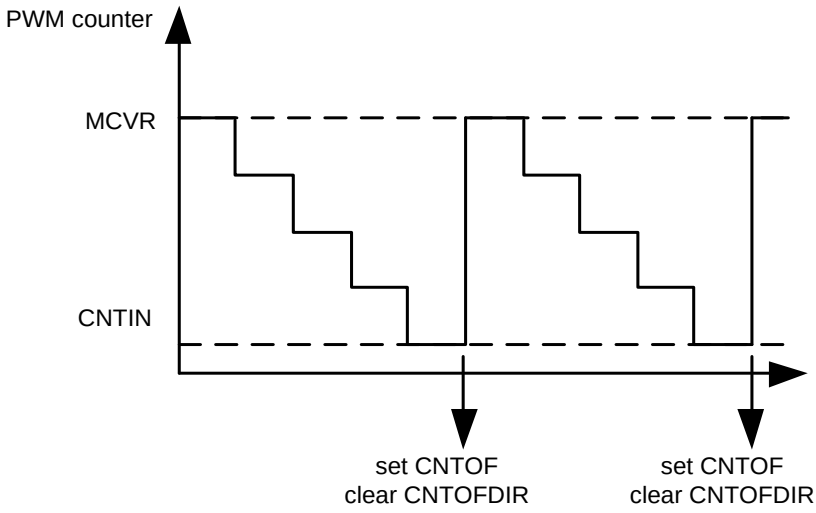

Figure 11-30 PWM Counter Overflow in Down Counting for Quadrature Decoder Mode

11.4.11 Write protection

In some application scenarios with high security level requirements (such as motor control), incorrectly modifying the register configuration can cause system abnormalities, which can easily lead to motor damage. The write protection function can provide write protection for certain key register bits. After completing the initial configuration, the user can activate the write protection function by setting PWM_FDSR[WPEN] = 1, to avoid misoperation of key register bits caused by human or other abnormalities. Set PWM_FUNCSEL[WPDIS] = 1 to disable the write protection function and write again. The bits that support the write protection are described in chapter 11.5 .

11.4.12 Initialization

The initialization forces the PWM_OUTINIT[CHnOIV] bit value to the channel (n) output when a one is written to the INIT bit. The initialization feature can only be used when the PWM counter is disabled. When the PWM counter starts to work, the INIT bit is automatically cleared, and the initialization output remains until the PWM channel starts to take over control.

11.4.13 Polarity control

The CHnPOL bit selects the channel(n) output polarity, when n=0,1,2,3,4 and 5:

- If CHPOLCR[CHnPOL] = 0, the channel(n) output polarity is high, so the logical one is the active state and logical zero is the inactive state.
- If CHPOLCR[CHnPOL] = 1, the channel(n) output polarity is low, so the logical zero is the active state and logical one is the inactive state.

11.4.14 Output mask

The output mask can be used to force the channel output to its respective invalid state through software. If CHnOMEN = 1, the channel(n) output is forced to the inactive state of the channel (PWM_CHOPOLCR[CHnPOL] bit value).

11.4.15 Software output control

The software output control forces the channel output according to software defined values in the PWM generation.

The CH(n)SWEN bit enables the software output control, and the CH(n)SWCV selects the value that is forced to this channel output. Software output control forces the following values on channs(n) and (n+1) when the COMP bit is zero. Independent mode, software output function can be set separately for each channel, software output function as a table in combine mode.

Table 11-2 Software output control behavior in combine mode

CH(n) SWEN	CH(n+1) SWEN	CH(n) SWCV	CH(n+1) SWCV	Channel(n) Output	Channel(n+1) Output
0	X	X	X	Software control disable	Software control not disable
1	X	0	0	0	0
1	X	0	1	0	1
1	X	1	0	1	0
1	X	1	1	1	0/1(Note2)

Note 1: X is either 0 or 1.

Note 2: If the PAIRnCOMPEN bit is 0, the output is 1. If the PAIRnCOMPEN bit is 1, the output is 0.

11.4.16 Initialization trigger

If INITTRIGEN = 1, PWM generates a trigger when the PWM counter is updated with the PWM_CNTIN register value in the following cases.

- The PWM counter is automatically updated with the PWM_CNTIN register value by the selected counting mode.
- When there is a write to CNT register.
- When there is the PWM counter synchronization.

11.4.17 Channel trigger output

If CHnTRIG = 1, PWM generates a trigger when the channel (n) match occurs (PWM counter = CH(n)V). The channel trigger output provides a trigger signal that is used for on-chip modules.

In the up-down combination mode, the channel value generates a match while up counting, and it also generates a match while down counting. In order to facilitate the control channel matching trigger, the matching effective point PWM_CHnSCR[DIR] can be set to take effect while up counting or down counting.

11.4.18 Fault control

PWM provides 3 fault input sources: one is from chip inside, two is from external pin input.

FERnEN bit enable fault input n, FFnEN bit enable fault input n filter.

The FFVAL bit selects the value of each fault input filter that has been enabled.

If the fault control and fault input n are enabled and effective edge at the fault input n signal is detected, a fault condition has occurred and the FAULTDFn bit is set. The FAULTDF bit is the logical OR of the FAULTDFn[2:0] bit.

If the fault control is enabled (FAULTMODE[1:0] ≠ 0:0), a fault condition has occurred and

(FAULTEN = 1), then outputs are forced to their safe values:

- Channel (n) output takes the value of CHnPOL
- Channel (n+1) takes the value of CH(n+1)POL

The fault interrupt is generated when (FAULTDF = 1) and (FAULTIE = 1).

Table 11-3 Fault Source and Number Table

Fault Input Number	Fault Source	Comment
FAULT0	ACMP0_OUT	Internal Fault Input
FAULT1	PWM_FLT0	External Pin Fault Input
FAULT2	PWM_FLT1	External Pin Fault Input

11.4.18.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTMODE[1:0] = 1:1), then the channels output disabled by fault control is again enabled when the fault input signal returns to zero and a new PWM cycle begins.

11.4.18.2 Manual fault clearing

If the manual fault clearing is selected (FAULTMODE[1:0]=0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTDF bit is cleared and a new PWM cycle begins.

11.4.18.3 Fault inputs polarity control

The FLTnPOL bit selects the fault input n polarity:

- If FLTnPOL = 0, the fault n input polarity is high, so the logical one at the fault input n indicates a fault.
- If FLTnPOL = 1, the fault n input polarity is low, so the logical zero at the fault input n indicates a fault.

11.4.19 Registers updated from write buffers

11.4.19.1 PWM_CNTIN register update buffer

Table 11-4 PWM_CNTIN register update buffer

Condition	Update opportunity
CLKSRC[1:0] = 0:0	When PWM_CNTIN register is written
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	At the next system clock after PWM_CNTIN was written
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	Refer to Chapter 11.4.20.6 PWM_CNTIN register synchronization

11.4.19.2 PWM_CH(n)V register update buffer

Table 11-5 PWM_CH(n)V register update buffer

Condition	Update opportunity
CLKSRC[1:0]=0:0	When PWM_CH(n)V register is written
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	• If the selected mode is EPWM, then register is updated after the PWM counter changes from MCVR to CNTIN. • If the selected mode is CPWM, then register is updated after the PWM counter changes from MCVR to (MCVR – 0x0001).
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	By the PWM_CH(n)V register synchronization Refer to Chapter 11.4.20.7 PWM_CH(n)V and PWM_CH(n+1)V register synchronization

11.4.19.3 PWM_MCVR register update buffer

Table 11-6 PWM_MCVR register update buffer

Condition	Update opportunity
CLKSRC[1:0]=0:0	When PWM_MCVR register is written
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	• If the selected mode is EPWM, then register is updated after the PWM counter changes from MCVR to CNTIN. • If the selected mode is CPWM, then register is updated after the PWM counter changes from MCVR to (MCVR – 0x0001).

CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	Refer to Chapter 11.4.20.4 PWM_MCVR register synchronization

11.4.20 PWM synchronization

The PWM synchronization provides an opportunity to update the PWM_MCVR, PWM_CNTIN, PWM_CHnV, PWM_OMCR, PWM_INVCR and PWM_CHOSWCR registers with their buffered value and force the PWM counter to the PWM_CNTIN register value.

11.4.20.1 Hardware trigger

Three hardware trigger signal inputs of the PWM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock.

The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs If (HWTRIGMODESEL = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.

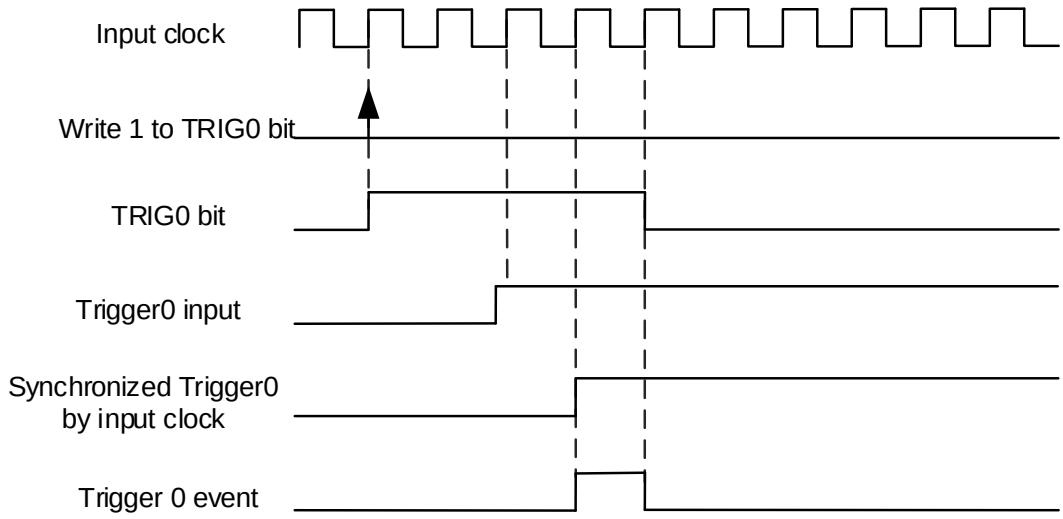

Figure 11-31 Hardware trigger event with HWTRIGMODE = 0

11.4.20.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time, the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 1, then the SWSYNC bit is also cleared by PWM according to the CNTVSWSYNC bit. If CNTVSWSYNC=0, then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred. If CNTVSWSYNC = 1, then SWSYNC bit is cleared when the software trigger event occurs.

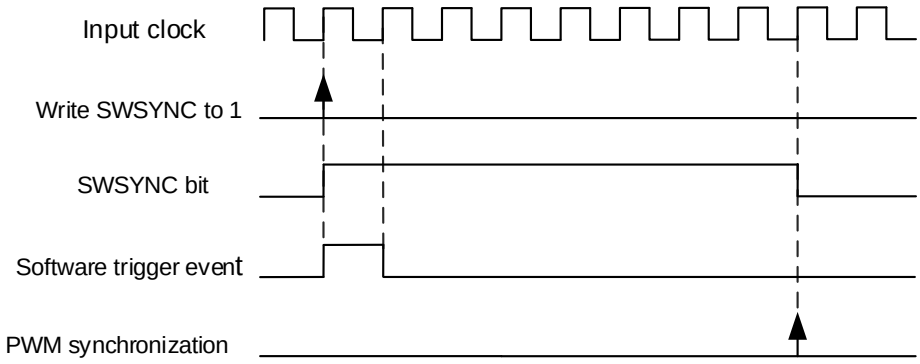

Figure 11-32 Software trigger event

11.4.20.3 Synchronization Points

In Up counting mode, the boundary period is defined as when the counter becomes its initial value (CNTIN). In Up-down counting, the boundary period is defined as when the counter changes from down counting to up counting and from up counting to down counting. The following figure shows the boundary period and synchronization points of the register.

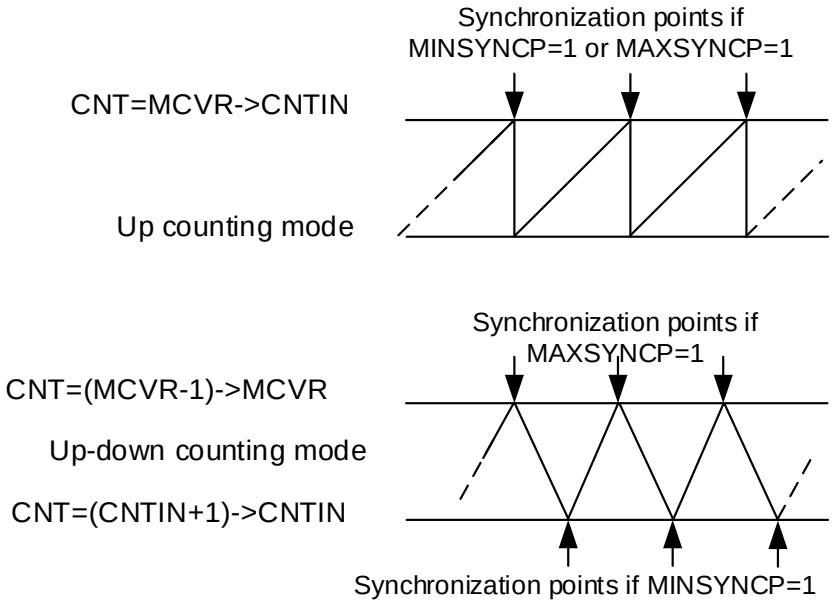

Figure 11-33 Synchronization points

In up counting mode, the MINSYNCP or MAXSYNCP is 1, then enable the synchronization point. In up-down counting mode, it is the MINSYNCP & MAXSYNCP to select the synchronization point. In the two counting modes, if both MINSYNCP and MAXSYNCP are not 1, the boundary period is not used as a synchronization point for register updates, even if the trigger signal is received, a synchronization event will not be generated (CNTVSWSYNC=0). For details, see the description of register synchronization in the following sections.

11.4.20.4 PWM_MCVR register synchronization

The PWM_MCVR register synchronization updates the PWM_MCVR register with its buffer value. This synchronization is enabled if (PWMSYNCEN = 1).

The flow chart of the PWM_MCVR register synchronization is shown below.

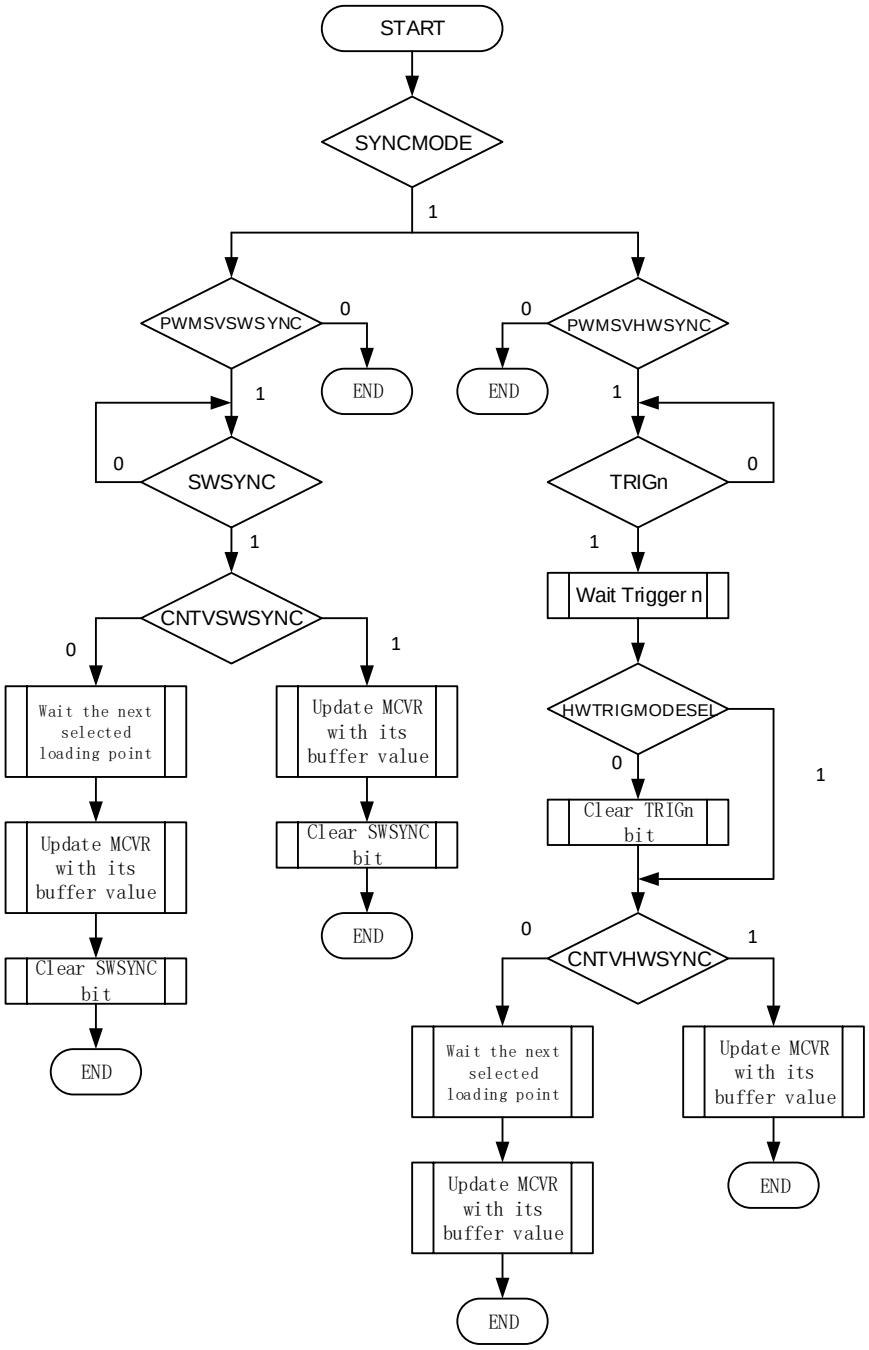

Figure 11-34 PWM_MCVR register synchronization flowchart

11.4.20.5 PWM_CNT register synchronization

The PWM_CNT register synchronization function can restart to generate PWM at a specific point in the PWM cycle. The channel output is forced to its initial value, and the PWM_CNT register is forced to the initial count value defined by the PWM_CNTIN register. The flow chart of the PWM_CNT register synchronization is give below.

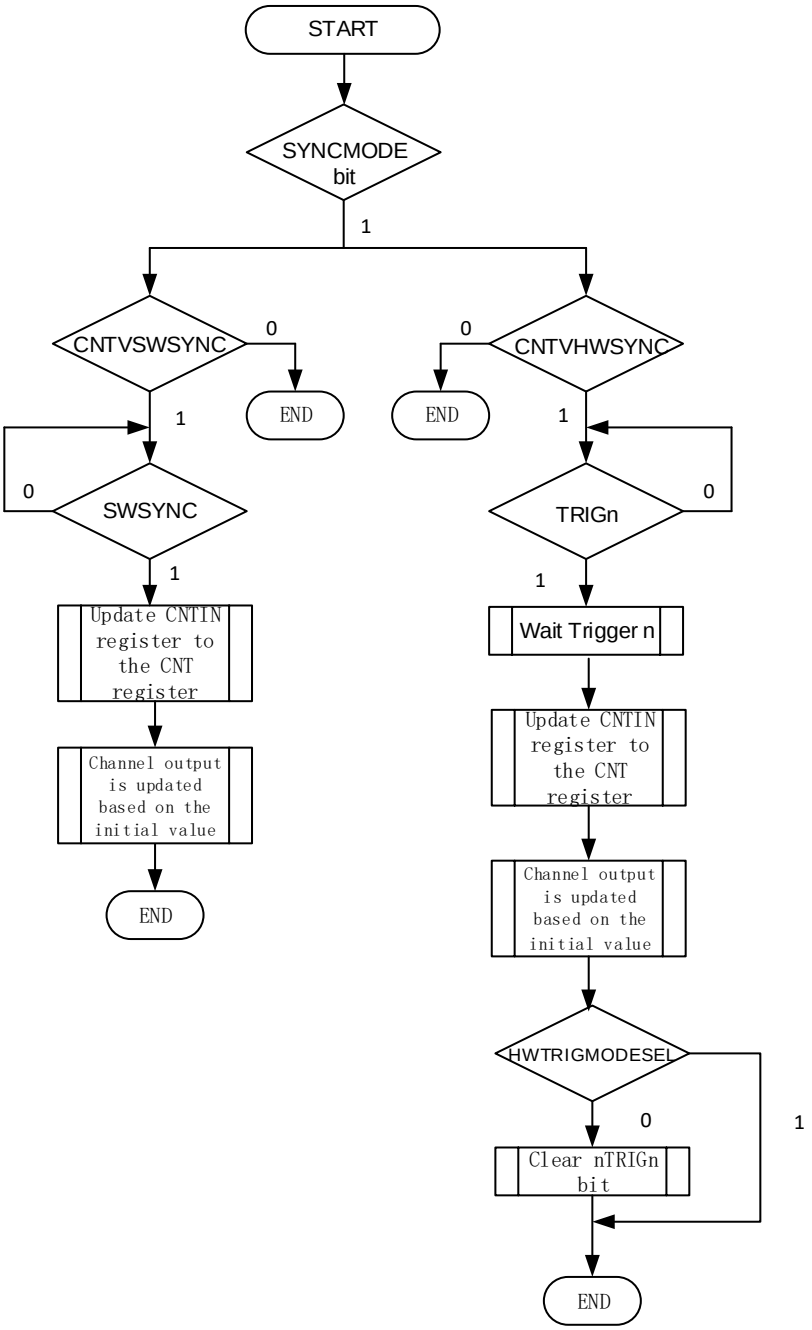

Figure 11-35 PWM_CNT register synchronization flow

11.4.20.6 PWM_CNTIN register synchronization

This synchronization is enabled if (PWMSYNCEN=1) , (SYNCMODE = 1) and (CNTINC = 1). The synchronization mechanism is the same as the PWM_MCVR register synchronization.

11.4.20.7 PWM_CH(n)V and PWM_CH(n+1)V register synchronization

This synchronization is enabled if (PWMSYNCEN = 1) and (PAIR(n)SYNCEN=1).The synchronization mechanism is the same as the PWM_MCVR register synchronization.

11.4.20.8 PWM_OMCR register synchronization

The PWM_OMCR register synchronization updates the PWM_OMCR register with its buffer value.

For the flow chart, refer to Figure 11-36.

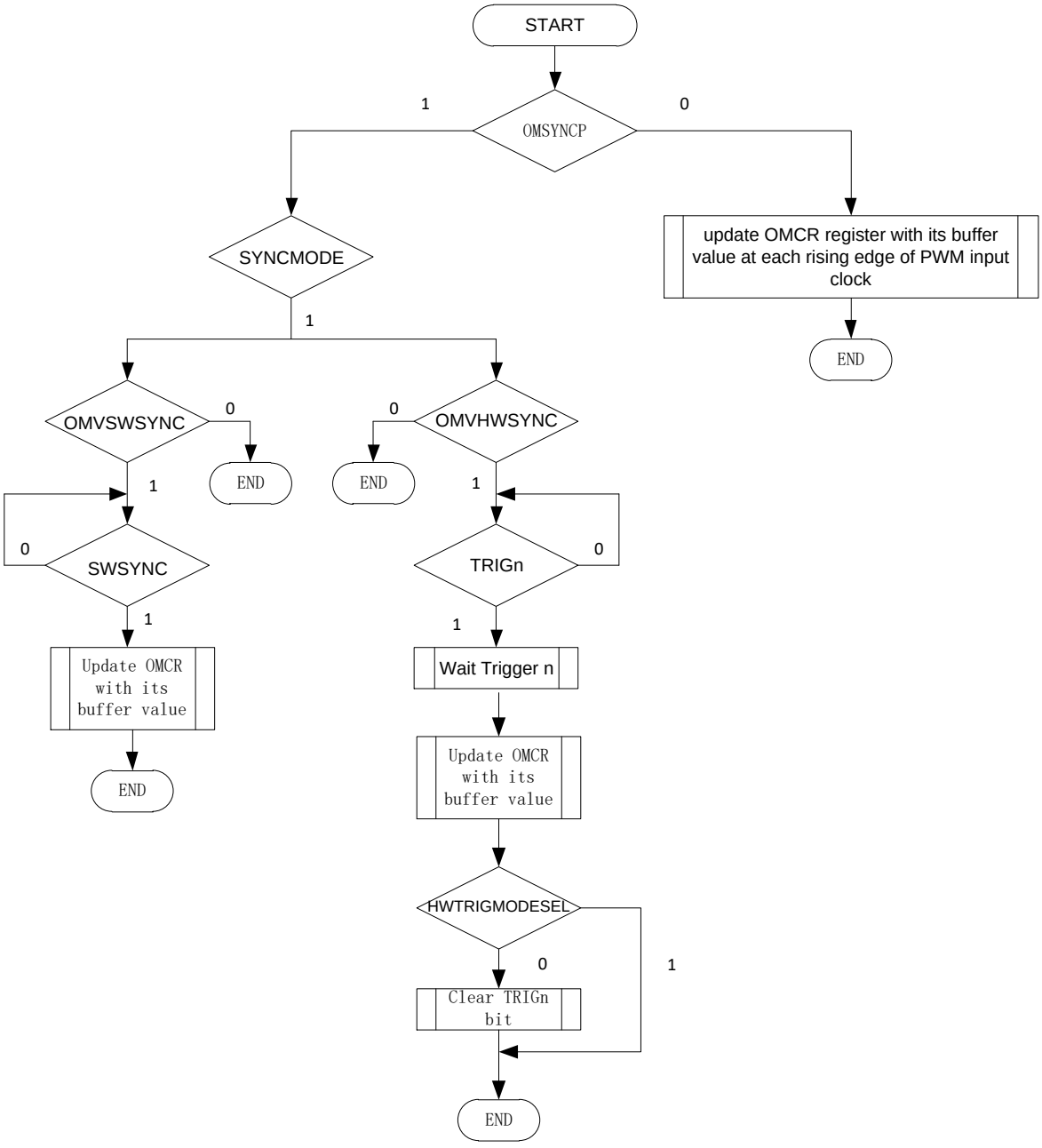

Figure 11-36 PWM_OMCR register synchronization flowchart

11.4.20.9 PWM_INVCR register synchronization

The PWM_INVCR register synchronization updates the PWM_INVCR register with its buffer value. For the flow chart, refer to Figure 11-37.

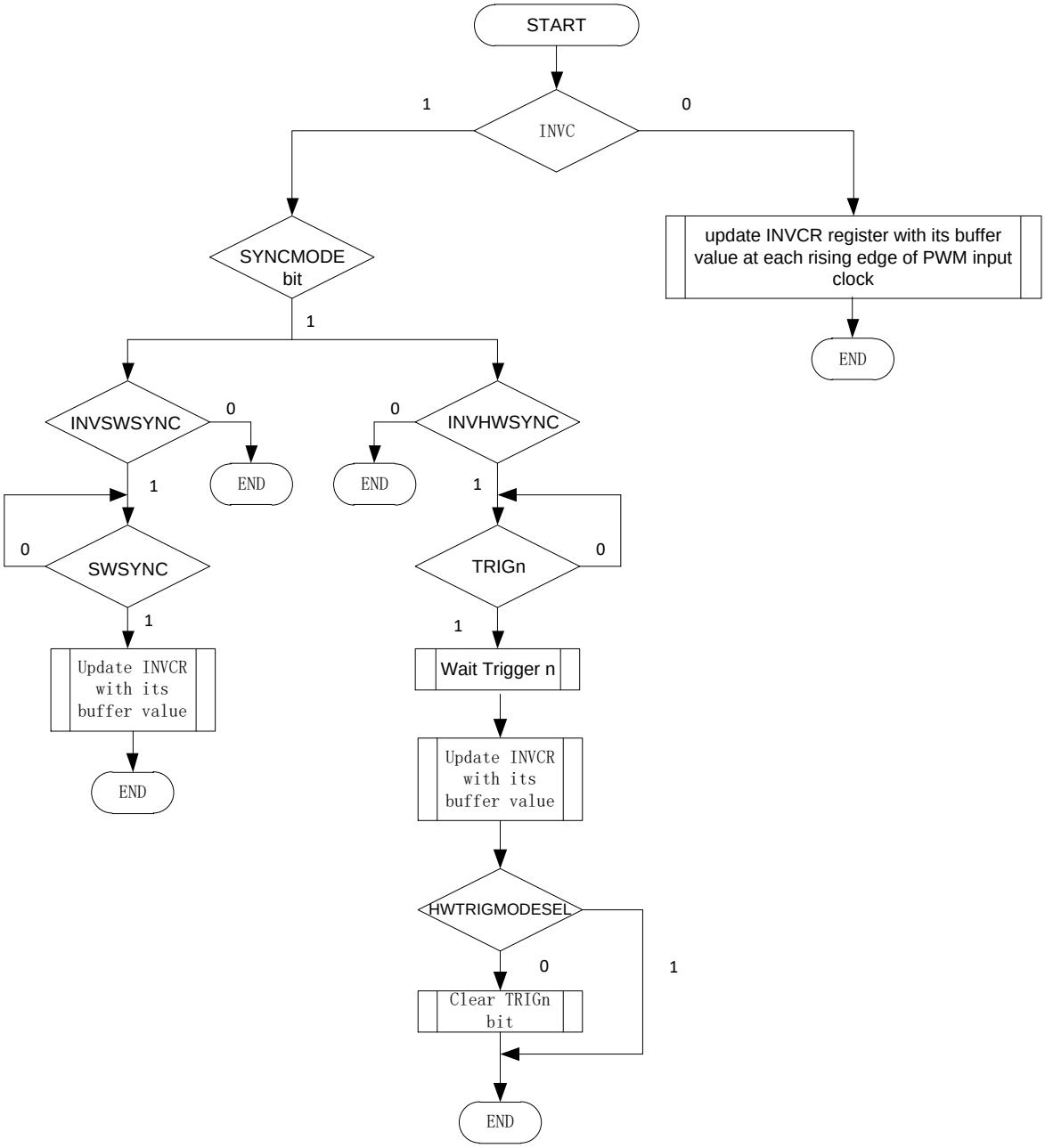

Figure 11-37 PWM_INVCR register synchronization flowchart

11.4.20.10 PWM_CHOSWCR register synchronization

The PWM_CHOSWCR register synchronization updates the PWM_CHOSWCR register with its buffer value. For the flow chart, refer to Figure 11-38.

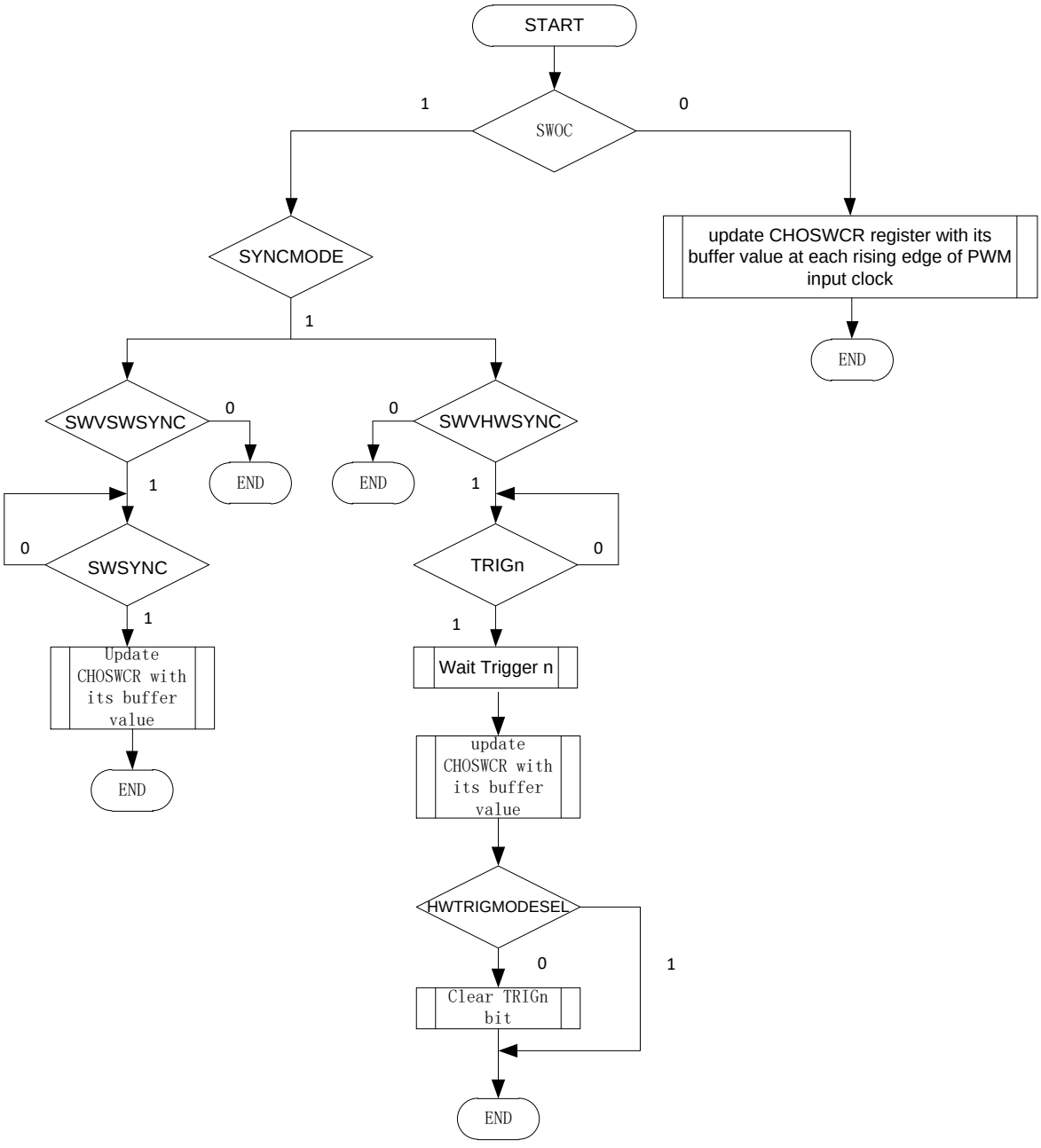

Figure 11-38 PWM_CHOSWCR register synchronization flowchart

11.4.20.11 PWM_CHOPOLCR register synchronization

The PWM_CHOPOLCR register synchronization updates the PWM_CHOPOLCR register with its buffer value. For the flow chart, refer to Figure 11-39.

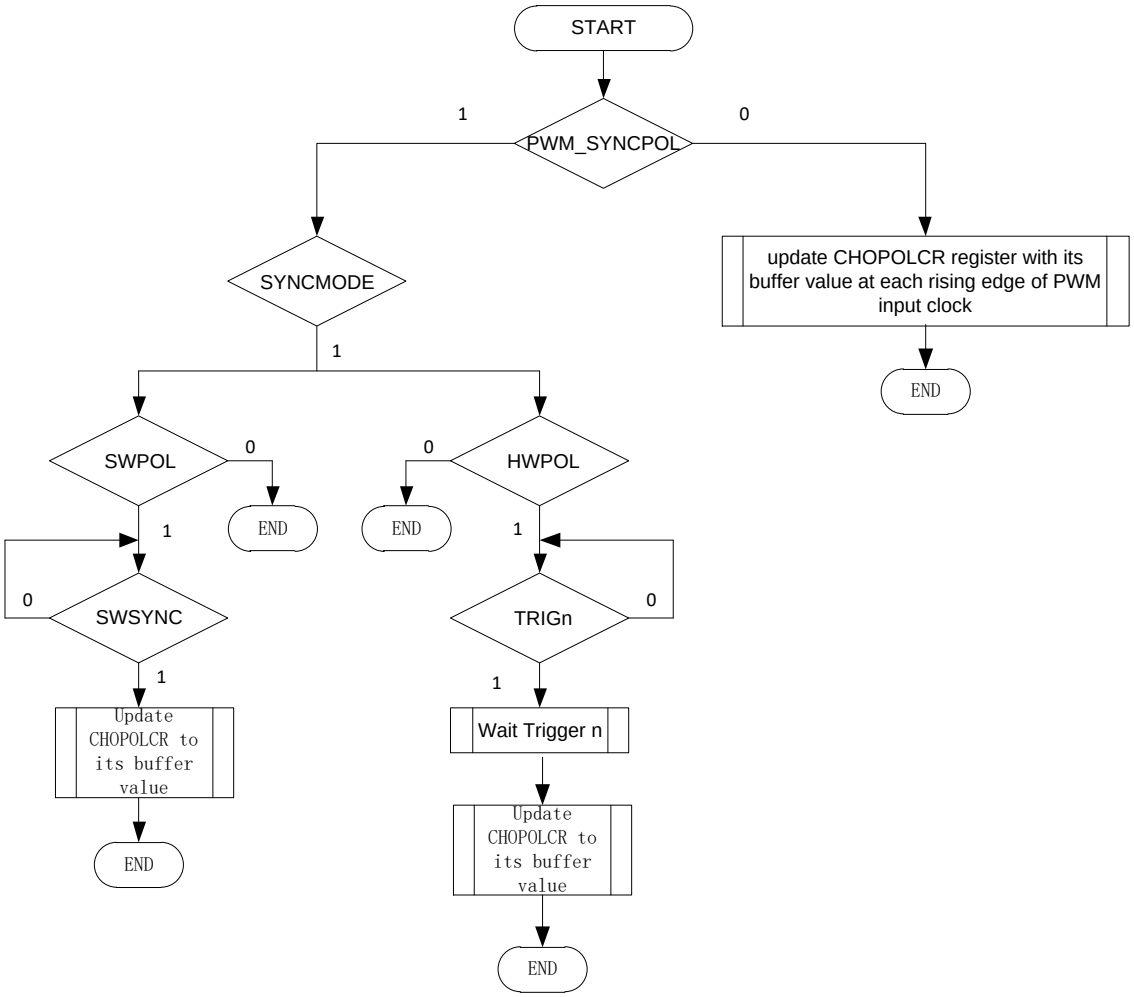

Figure 11-39 PWM_CHOPOLCR register synchronization flowchart

11.4.21 Features priority

The following figure shows the priority of the features used at the generation of CH(n) and CH(n+1) outputs signals.

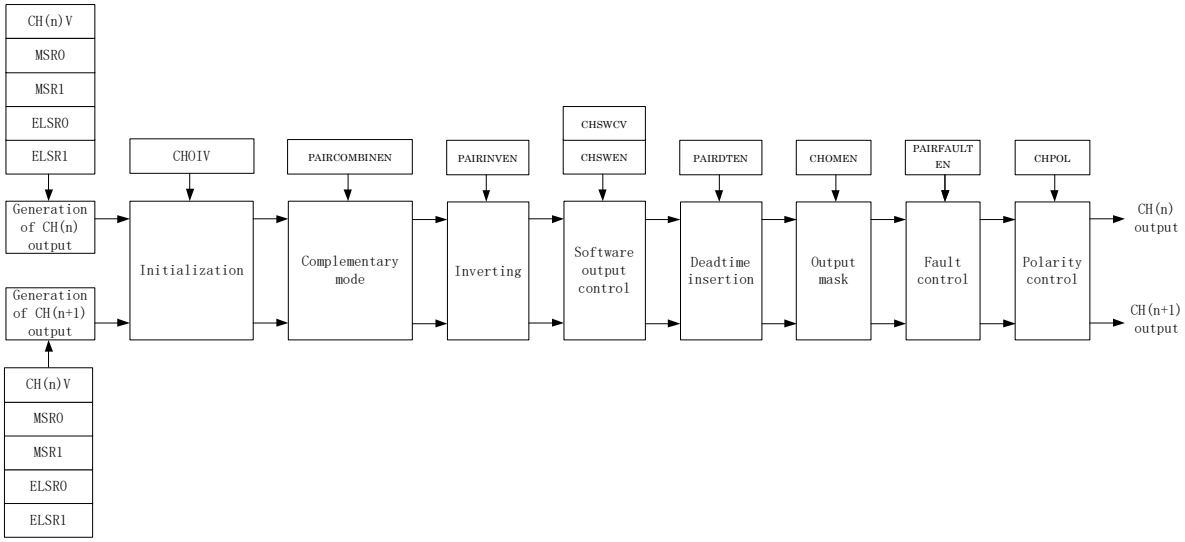

Figure 11-40 Features priority

11.4.22 Global time base (GTB)

The global time base (GTB) is a PWM function that allows the synchronization of multiple PWM modules on a chip.

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the PWM_CONF register.

To enable the GTB feature, follow these steps for each participating PWM module:

1. Stop the PWM counter: Write 00b to PWM_INIT[CLKSRC];
2. Write 1 to PWM_CONF[GTBEEN] and write 0 to PWM_CONF[GTBEOUT] at the same time;
3. Select the intended PWM counter clock source in PWM_INIT[CLKSRC];
4. Reset the PWM counter by writing any value to the CNT register;
5. Write 1 to PWM_CONF[GTBEOUT] in the PWM module used as the time base.

11.4.23 PWM interrupts

- The Count Overflow interrupt is generated when (CNTOIE=1) and (CNTOF=1).
- The Channel(n) interrupt is generated when (CHnIE=1) and (CHnIF = 1).
- The Fault interrupt is generated when (FAULTIE =1) and (FAULTDF = 1).

11.5 Register definition

Table 11-7 PWM register mapping

PWM0 base address = 0x40013000

PWM1 base address = 0x40014000

Address	Name	Width (in bit)	Description
PWMx base address +0x00	PWM_INIT	32	PWM Initialization Register
PWMx base address +0x04	PWM_CNT	32	PWM counter value register
PWMx base address +0x08	PWM_MCVR	32	Max Count Value Register
PWMx base address +0x0C	PWM_CHnSCR	32	Channel (0) Status And Control Register
PWMx base address +0x10	PWM_CHnV	32	Channel (0) Value
PWMx base address +0x14	PWM_CHnSCR	32	Channel (1) Status And Control Register
PWMx base address +0x18	PWM_CHnV	32	Channel (1) Value
PWMx base address +0x1C	PWM_CHnSCR	32	Channel (2) Status And Control Register
PWMx base address +0x20	PWM_CHnV	32	Channel (2) Value
PWMx base address +0x24	PWM_CHnSCR	32	Channel (3) Status And Control Register
PWMx base address +0x28	PWM_CHnV	32	Channel (3) Value
PWMx base address +0x2C	PWM_CHnSCR	32	Channel (4) Status And Control Register
PWMx base address +0x30	PWM_CHnV	32	Channel (4) Value
PWMx base address +0x34	PWM_CHnSCR	32	Channel (5) Status And Control Register
PWMx base address +0x38	PWM_CHnV	32	Channel (5) Value
PWMx base address +0x3C	PWM_CHnSCR	32	Channel (6) Status And Control Register
PWMx base address +0x40	PWM_CHnV	32	Channel (6) Value
PWMx base address +0x44	PWM_CHnSCR	32	Channel (7) Status And Control Register
PWMx base address +0x48	PWM_CHnV	32	Channel (7) Value
PWMx base address +0x4C	PWM_CNTIN	32	Counter Initial Value
PWMx base address +0x50	PWM_STR	32	Capture And Compare Status Register
PWMx base address +0x54	PWM_FUNCSEL	32	Features Mode Selection
PWMx base address +0x58	PWM_SYNC	32	Synchronization register
PWMx base address +0x5C	PWM_OUTINIT	32	Initial State For Channels Output
PWMx base address +0x60	PWM_OMCR	32	Output Mask Control Register
PWMx base address +0x64	PWM_MODESEL	32	Mode Selection Register

Address	Name	Width (in bit)	Description
PWMx base address +0x68	PWM_DTSET	32	Dead-time Setting Register
PWMx base address +0x6C	PWM_EXTTRIG	32	PWM External Trigger
PWMx base address +0x70	PWM_CHOPOLCR	32	Channel Output Polarity Control Register
PWMx base address +0x74	PWM_FDSR	32	Fault Detect Status Register
PWMx base address +0x78	PWM_CAPFILTER	32	Input Capture Mode Filter Control
PWMx base address +0x7C	PWM_FFAFER	32	Input Capture Mode Filter Control
PWMx base address +0x80	PWM_QDI	32	Quadrature Decoder Control And Status
PWMx base address +0x84	PWM_CONF	32	Configuration register
PWMx base address +0x88	PWM_FLTPOL	32	PWM Fault Input Polarity
PWMx base address +0x8C	PWM_SYNCONF	32	Synchronization Configuration
PWMx base address +0x90	PWM_INVCR	32	PWM Inverting Control
PWMx base address +0x94	PWM_CHOSWCR	32	Channel software output control register

Note: x=0,1 in the above table.

11.5.1 PWM_INIT

Table 11-8 PWM_INIT register

PWM_INIT **PWM Initialization Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									CLKPSC[15: 8]							
Type									RW							
Reset									0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CLKPSC[7: 0]								CNTOF	CNTOIE	CNTMODE	CLKSRC				
Type	RW								W0C	RW	RW	RW				
Reset	0								0							

Bits	Description
23: 8 CLKPSC	**PWM CLK prescaler** The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
7 CNTOF	**Timer Overflow Flag** 0: PWM counter has not overflowed. 1: PWM counter has overflowed.

Bits	Description
	Set by hardware when the PWM counter passes the value in the MCVR register. The CNTOF bit is cleared by reading the INIT register while CNTOF is set and then writing a 0 to CNTOF bit. Writing a 1 to CNTOF has no effect. If another PWM overflow occurs between the read and write operations, the write operation has no effect. Therefore, CNTOF remains set indicating an overflow has occurred. In this case, a CNTOF interrupt request is not lost due to the clearing sequence for a previous CNTOF.
6 CNTOIE	**Timer Overflow Interrupt Enable** 0: Disable CNTOF interrupts. Use software polling. 1: Enable CNTOF interrupts. An interrupt is generated when CNTOF equals one. Enables PWM overflow interrupts.
5 CNTMODE	**Count Mode Select** 0: PWM counter operates in Up Counting mode. 1: PWM counter operates in Up-Down Counting mode. Select count mode. This mode configures the PWM to operate in Up-Down Counting mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
4: 3 CLKSRC	**Clock Source Selection** 00: No clock selected. This in effect disables the PWM counter. 01: Bus clock 10: HSI clock 11: reserved Select one of the three PWM counter clock sources. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.2 PWM_CNT

Table 11-9 PWM_CNT register

PWM_CNT **PWM counter value** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COUNT															
Type	RW															
Reset	0															

Bits	Description
15: 0 COUNT	**PWM counter value.** The CNT register contains the PWM counter value. Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

11.5.3 PWM_MCVR

Table 11-10 PWM_MCVR register

PWM_MCVR **Max Count Value Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MCVR															
Type	RW															
Reset	0															

Bits	Description
15: 0 MCVR	**Max Count Value Register** The MCVR register contains the modulo value for the PWM counter. After the PWM counter reaches the MCVR value, the overflow flag (CNTOF) becomes set at the next clock, and the next value of PWM counter depends on the selected counting method. Writing to the MCVR register latches the value into a buffer. The MCVR register is updated with the value of its write buffer according to Registers updated from write buffers. Initialize the PWM counter, by writing to CNT, before writing to the MCVR register to avoid confusion about when the first counter overflow will occur.

11.5.4 PWM_CHnSCR

Table 11-11 PWM_CHnSCR register

PWM_CHnSCR **Channel (n) Status And Control Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CHIF	CHIE	MSR1	MSR0	ELSR1	ELSR0	DIR	
Type									W0C	RW	RW	RW	RW	RW	RW	
Reset									0	0	0	0	0	0	0	

Bits	Description
7 CHIF	**Channel Interrupt Flag** 0: No channel event has occurred. 1: A channel event has occurred. Set by hardware when an event occurs on the channel. CHIF is cleared by reading the CHSCR register while CHnIF is set and then clear the CHIF bit by writing a 0 to this bit. Writing a 1 to CHIF has no effect. If another event occurs between the read and write operations, the write operation has no effect. Therefore, CHIF remains set indicating an event has occurred. In this case, a CHIF interrupt request is not lost due to the write clear operation.
6 CHIE	**Channel Interrupt Enable** 0: Disable channel interrupts. 1: Enable channel interrupts. Enables channel interrupts.
5 MSR1	**Channel Mode Select Register 1** Used for further selections in the channel logic. Its functionality is dependent on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
4 MSR0	**Channel Mode Select Register 0** Used for further selections in the channel logic. Its functionality is dependent on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
3 ELSR1	**Edge or Level Select Register 1** The functionality of ELSR1 and ELSR0 depends on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
2 ELSR0	**Edge or Level Select Register 0** The functionality of ELSR1 and ELSR0 depends on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
1 DIR	**Match point direction** 0: Match points take effect during down counting. 1: Match points take effect during up counting. Only used in up-down counting combination mode, and used to select the direction of matching effective point

11.5.5 PWM_CHnV

Table 11-12 PWM_CHnV register

PWM_CHnV	Channel (n) value															Reset:00000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CHCVAL															
Type	RW															
Reset	0															

Bits	Description
15:0 CHCVAL	**Channel Count Value** These registers contain the captured PWM counter value for the input modes or the match value for the output modes. In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CHnV register is ignored. In output modes, writing to a CHnV register latches the value into a buffer. A CHnV register is updated with the value of its write buffer according to registers updated from write buffers.

11.5.6 PWM_CNTIN

Table 11-13 PWM_CNTIN register

PWM_CNTIN	Counter Initial Value															Reset:00000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CNTINIT															
Type	RW															
Reset	0															

Bits	Description
15: 0 CNTINIT	**Counter Initial Value** The Counter Initial Value register contains the initial value for the PWM counter. Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to registers updated from write buffers. When the PWM clock is initially selected, by writing a non-zero value to the CLKS bits, the PWM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the PWM clock, write the new value to the CNTIN register and then initialize the PWM counter by writing any value to the CNT register.

11.5.7 PWM_STR

Table 11-14 PWM_STR register

PWM_STR **Capture And Compare Status Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CH 7SF	CH 6SF	CH 5SF	CH 4SF	CH 3SF	CH 2SF	CH 1SF	CH 0SF
Type									W0C	W0C	W0C	W0C	W0C	W0C	W0C	W0C
Reset									0	0	0	0	0	0	0	0

Bits	Description
7 CH7SF	**Channel 7 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
6 CH6SF	**Channel 6 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
5 CH5SF	**Channel 5 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
4 CH4SF	**Channel 4 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
3 CH3SF	**Channel 3 Status Flag** 0: No channel event has occurred.

Bits	Description
	1: A channel event has occurred.
2 CH2SF	**Channel 2 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
1 CH1SF	**Channel 1 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.
0 CH0SF	**Channel 0 Status Flag** 0: No channel event has occurred. 1: A channel event has occurred.

11.5.8 PWM_FUNCSEL

Table 11-15 PWM_FUNCSEL register

PWM_FUNCSEL **Features Mode Selection** **Reset:00000004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FAULTIE	FAULTMODE			PWMSYNC	WPDIS	INIT	PWMSYNCEN
Type									RW	RW			RW	RW	RW	RW
Reset									0	0			0	1	0	0

Bits	Description
7 FAULTIE	**Fault Interrupt Enable** 0: Fault control interrupt is disabled. 1: Fault control interrupt is enabled. Enables the generation of an interrupt when a fault is detected by PWM and the PWM fault control is enabled.
6: 5 FAULTMODE	**Fault Control Mode** 00: Fault control is disabled for all channels. 01: Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing.

Bits	Description
	10: Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11: Fault control is enabled for all channels, and the selected mode is the automatic fault clearing. Defines the PWM fault control mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
3 PWMSYNC	**PWM Synchronization Mode** 0: No restrictions. Software and hardware triggers can be used by MCVR, CHnV, OMCR and PWM counter synchronization. 1: Software trigger can only be used by MCVR and CHnV synchronization, and hardware triggers can only be used by OMCR and PWM counter synchronization. Selects which triggers can be used by MCVR, CHnV, OMCR, and PWM counter synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is zero.
2 WPDIS	**Write Protection Disable** 0: Write protection is enabled. 1: Write protection is disabled. When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect.
1 INIT	**Initialize the Channels Output** When a 1 is written to INIT bit, the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
0 PWMSYNCEN	**PWM synchronization Enable** 0: disable PWM synchronization 1: All registers including the PWM-specific registers (second set of registers) are available for use with no restrictions. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.9 PWM_SYNC

Table 11-16 PWM_SYNC register

PWM_SYNC **Synchronization register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					PWM_SYNCPOL				SWSYNC	TRIG2	TRIG1	TRIG0	OMSYNCP		MAXSYNCP	MINSYNCP
Type					RW				RW	RW	RW	RW	RW		RW	RW
Reset					0				0	0	0	0	0		0	0

Bits	Description
11 PWM_SYNCPOL	**PWM_SYNCPOL** 0: POL register is updated with the value of its buffer in all rising edges of the system clock. 1: POL register is updated with the value of its buffer only by the PWM synchronization. Selects when the CHOPOLCR register is updated with the value of its buffer.
7 SWSYNC	**PWM Synchronization Software Trigger** 0: Software trigger is not selected. 1: Software trigger is selected. Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit.
6 TRIG2	**PWM Synchronization Hardware Trigger 2** 0: Trigger is disabled. 1: Trigger is enabled. Enables PWM channel 0 output to the PWM synchronization hardware trigger source. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal.
5 TRIG1	**PWM Synchronization Hardware Trigger 1** 0: Trigger is disabled. 1: Trigger is enabled.

Bits	Description
	Enables CTU control bit [PWMTRIG] output to the PWM synchronization hardware trigger source. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal.
4 TRIG0	**PWM Synchronization Hardware Trigger 0** 0: Trigger is disabled. 1: Trigger is enabled. Enables ACMP0 output to the PWM synchronization hardware trigger source. Hardware trigger 0 happens when a rising edge is detected at the trigger 0 input signal.
3 OMSYNCP	**Output Mask Synchronization** 0: OMCR register is updated with the value of its buffer in all rising edges of the system clock. 1: OMCR register is updated with the value of its buffer only by the PWM synchronization. Selects when the OMCR register is updated with the value of its buffer.
1 MAXSYNCP	**Maximum Loading Point Enable** 0: The maximum loading point is disabled. 1: The maximum loading point is enabled. Selects the maximum loading point to PWM synchronization. If MAXSYNCP is one, the selected loading point is when the PWM counter reaches its maximum value (MCVR register).
0 MINSYNCP	**Minimum Loading Point Enable** 0: The minimum loading point is disabled. 1: The minimum loading point is enabled. Selects the minimum loading point to PWM synchronization. If MINSYNCP is one, the selected loading point is when the PWM counter reaches its minimum value (CNTIN register).

11.5.10 PWM_OUTINIT

Table 11-17 PWM_OUTINIT register

PWM_OUTINIT **Initial State For Channels Output** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CH7OIV	CH6OIV	CH5OIV	CH4OIV	CH3OIV	CH2OIV	CH1OIV	CH0OIV
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bits	Description
7 CH7OIV	**Channel 7 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1. Selects the value that is forced into the channel output when the initialization occurs.
6 CH6OIV	**Channel 6 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
5 CH5OIV	**Channel 5 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
4 CH4OIV	**Channel 4 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
3 CH3OIV	**Channel 3 Output Initialization Value**

Bits	Description
	0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
2 CH2OIV	**Channel 2 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
1 CH1OIV	**Channel 1 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.
0 CH0OIV	**Channel 0 Output Initialization Value** 0: The initialization value is 0. 1: The initialization value is 1 Selects the value that is forced into the channel output when the initialization occurs.

11.5.11 PWM_OMCR

Table 11-18 PWM_OMCR register

PWM_OMCR **Output Mask Control Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CH7OMEN	CH6OMEN	CH5OMEN	CH4OMEN	CH3OMEN	CH2OMEN	CH1OMEN	CH0OMEN
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bits	Description
7 CH7OMEN	**Channel 7 Output Mask**

Bits	Description
	0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
6 CH6OMEN	**Channel 6 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
5 CH5OMEN	**Channel 5 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
4 CH4OMEN	**Channel 4 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
3 CH3OMEN	**Channel 3 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
2 CH2OMEN	**Channel 2 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
1 CH1OMEN	**Channel 1 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.
0 CH0OMEN	**Channel 0 Output Mask** 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state. Defines if the channel output is masked or unmasked.

11.5.12 PWM_MODESEL

Table 11-19 PWM_MODESEL register

PWM_MODESEL **PWM Function Mode Selection** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		PAIR3FAULTEN	PAIR3SYNCEN	PAIR3DTEN	PAIR3DECAP	PAIR3DECAPEN	PAIR3COMPEN	PAIR3COMBINEN		PAIR2FAULTEN	PAIR2SYNCEN	PAIR2DTEN	PAIR2DECAP	PAIR2DECAPEN	PAIR2COMPEN	PAIR2COMBINEN
Type		RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		PAIR1FAULTEN	PAIR1SYNCEN	PAIR1DTEN	PAIR1DECAP	PAIR1DECAPEN	PAIR1COMPEN	PAIR1COMBINEN		PAIR0FAULTEN	PAIR0SYNCEN	PAIR0DTEN	PAIR0DECAP	PAIR0DECAPEN	PAIR0COMPEN	PAIR0COMBINEN
Type		RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Bits	**Description**
30 PAIR3FAULTEN	**Fault Control Enable for n = 6** 0: The fault control in this pair of channels is disabled. 1: The fault control in this pair of channels is enabled. Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
29 PAIR3SYNCEN	**Synchronization Enable for n = 6** 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled. Enables PWM synchronization of registers CH(n)V and CH(n+1)V.
28 PAIR3DTEN	**Dead-time Enable for n = 6** 0: The dead-time insertion in this pair of channels is disabled. 1: The dead-time insertion in this pair of channels is enabled. Enables the dead-time insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

Bits	Description
27 PAIR3DECAP	**Dual Edge Capture Mode Captures for n = 6** 0: The dual edge captures are inactive. 1: The dual edge captures are active. Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.
26 PAIR3DECAPEN	**Dual Edge Capture Mode Enable for n = 6** 0: The Dual Edge Capture mode in this pair of channels is disabled. 1: The Dual Edge Capture mode in this pair of channels is enabled. Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1) R1:ELS(n+1)R0 bits in Dual Edge Capture mode. This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
25 PAIR3COMPEN	**Complement of Channel (n) for n = 6** 0: The channel (n+1) output is the same as the channel (n) output. 1: The channel (n+1) output is the complement of the channel (n) output. Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel(n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
24 PAIR3COMBINEN	**Combine Channels for n = 6** 0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined. Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
22 PAIR2FAULTEN	**Fault Control Enable for n = 4** 0: The fault control in this pair of channels is disabled. 1: The fault control in this pair of channels is enabled. Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
21 PAIR2SYNCEN	**Synchronization Enable for n = 4** 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled.

Bits	Description
	Enables PWM synchronization of registers CH(n)V and CH(n+1)V.
20 PAIR2DTEN	**Dead-time Enable for n = 4** 0: The dead-time insertion in this pair of channels is disabled. 1: The dead-time insertion in this pair of channels is enabled. Enables the dead-time insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
19 PAIR2DECAP	**Dual Edge Capture Mode Captures for n = 4** 0: The dual edge captures are inactive. 1: The dual edge captures are active. Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.
18 PAIR2DECAPEN	**Dual Edge Capture Mode Enable for n = 4** 0: The Dual Edge Capture mode in this pair of channels is disabled. 1 : The Dual Edge Capture mode in this pair of channels is enabled. Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1) R1:ELS(n+1)R0 bits in Dual Edge Capture mode. This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
17 PAIR2COMPEN	**Complement Of Channel (n) for n = 4** 0: The channel (n+1) output is the same as the channel (n) output. 1 : The channel (n+1) output is the complement of the channel (n) output. Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel(n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
16 PAIR2COMBINEN	**Combine Channels for n = 4** 0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined. Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
14 PAIR1FAULTEN	**Fault Control Enable for n = 2** 0: The fault control in this pair of channels is disabled.

Bits	Description
	1: The fault control in this pair of channels is enabled. Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
13 PAIR1SYNCEN	**Synchronization Enable for n = 2** 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled. Enables PWM synchronization of registers CH(n)V and CH(n+1)V.
12 PAIR1DTEN	**Dead-time Enable for n = 2** 0: The dead-time insertion in this pair of channels is disabled. 1: The dead-time insertion in this pair of channels is enabled. Enables the dead-time insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
11 PAIR1DECAP	**Dual Edge Capture Mode Captures for n = 2** 0: The dual edge captures are inactive. 1: The dual edge captures are active. Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when PWMEN2 = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.
10 PAIR1DECAPEN	**Dual Edge Capture Mode Enable for n = 2** .0: The Dual Edge Capture mode in this pair of channels is disabled. 1: The Dual Edge Capture mode in this pair of channels is enabled. Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1)R1:ELS(n+1)R0 bits in Dual Edge Capture mode This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
9 PAIR1COMPEN	**Complement of Channel (n) for n = 2** 0: The channel (n+1) output is the same as the channel (n) output. 1: The channel (n+1) output is the complement of the channel (n) output. Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel(n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
8 PAIR1COMBINEN	**Combine Channels for n = 2**

Bits	Description
	0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined. Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
6 PAIR0FAULTEN	**Fault Control Enable for n = 0** 0: The fault control in this pair of channels is disabled. 1: The fault control in this pair of channels is enabled. Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
5 PAIR0SYNCEN	**Synchronization Enable for n = 0** 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled. Enables PWM synchronization of registers CH(n)V and CH(n+1)V.
4 PAIR0DTEN	**Dead-time Enable for n = 0** 0: The dead-time insertion in this pair of channels is disabled. 1: The dead-time insertion in this pair of channels is enabled. Enables the dead-time insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
3 PAIR0DECAP	**Dual Edge Capture Mode Captures for n = 0** 0: The dual edge captures are inactive. 1: The dual edge captures are active. Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when PWMEN2 = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.
2 PAIR0DECAPEN	**Dual Edge Capture Mode Enable for n = 0** 0: The Dual Edge Capture mode in this pair of channels is disabled. 1: The Dual Edge Capture mode in this pair of channels is enabled. Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1)R1:ELS(n+1)R0 bits in Dual Edge Capture mode. This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
1 PAIR0COMPEN	**Complement of Channel (n) for n = 0**

Bits	Description
	0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output. Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0 PAIR0COMBINEN	**Combine Channels for n = 0** 0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined. Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.13 PWM_DTSET

Table 11-20 PWM_DTSET register

PWM_DTSET **Dead-time Insertion Control** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DTPSC		DTVAL					
Type									RW		RW					
Reset									0		0					

Bits	Description
7: 6 DTPSC	**Dead-time Prescaler Value** 0x: Divide the system clock by 1. 10: Divide the system clock by 4. 11: Divide the system clock by 16. Selects the division factor of the system clock. This prescaled clock is used by the dead-time counter. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
5: 0 DTVAL	**Dead-time Value** Selects the dead-time insertion value for the dead-time counter. The dead-time counter is clocked by a scaled version of the system clock. See the description of DTPS. Dead-time insert value = (DTPSC x DTVAL). DTVAL selects the number of dead-time counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted.

Bits	Description
	This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.14 PWM_EXTTRIG

Table 11-21 PWM_EXTTRIG register

PWM_EXTTRIG **PWM External Trigger** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							TRIGF	INITTRIGEN	CH7TRIG	CH6TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG	CH1TRIG	CH0TRIG
Type							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset							0	0	0	0	0	0	0	0	0	0

Bits	Description
9 TRIGF	**Channel Trigger Flag** 0: No channel trigger was generated. 1: A channel trigger was generated. Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF.
8 INITTRIGEN	**Initialization Trigger Enable** 0: The generation of initialization trigger is disabled. 1: The generation of initialization trigger is enabled. Enables the generation of the trigger when the PWM counter is equal to the CNTIN register.
7 CH7TRIG	**Channel 7 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
6	**Channel 6 Trigger Enable**

Bits	Description
CH6TRIG	0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
5 CH5TRIG	**Channel 5 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
4 CH4TRIG	**Channel 4 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
3 CH3TRIG	**Channel 3 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register.
2 CH2TRIG	**Channel 2 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
1 CH1TRIG	**Channel 1 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled. Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.
0 CH0TRIG	**Channel 0 Trigger Enable** 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.

Bits	Description
	Enables the generation of the channel trigger when the PWM counter is equal to the CHnV register.

11.5.15 PWM_CHOPOLCR

Table 11-22 PWM_CHOPOLCR register

PWM_CHOPOLCR **Channels Output Polarity Control Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CH7POL	CH6POL	CH5POL	CH4POL	CH3POL	CH2POL	CH1POL	CH0POL
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bits	Description
7 CH7POL	**Channel 7 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
6 CH6POL	**Channel 6 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
5 CH5POL	**Channel 5 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
4 CH4POL	**Channel 4 Polarity** 0: The channel polarity is active high.

Bits	Description
	1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
3 CH3POL	**Channel 3 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
2 CH2POL	**Channel 2 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
1 CH1POL	**Channel 1 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0 CH0POL	**Channel 0 Polarity** 0: The channel polarity is active high. 1: The channel polarity is active low. Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.16 PWM_FDSR

Table 11-23 PWM_FDSR register

PWM_FDSR **Fault Detect Status Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FAULTDF	WPEN	FAULTIN			FAULTDF2	FAULTDF1	FAULTDF0
Type									W0C	RW	RO			W0C	W0C	W0C
Reset									0	0	0			0	0	0

Bits	Description
7 FAULTDF	**Fault Detection Flag** 0: No fault condition was detected. 1 : A fault condition was detected. Represents the logic OR of the individual FAULTDF j bits where j = 2, 1, 0. Clear FAULTDF by reading the FDSR register while FAULTDF is set and then writing a 0 to FAULTDF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTDF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTDF remains set after the clearing sequence is completed for the earlier fault condition. FAULTDF is also cleared when FAULTDF j bits are cleared individually.
6 WPEN	**Write Protection Enable** 0: Write protection is disabled. Write protected bits can be written. 1: Write protection is enabled. Write protected bits cannot be written. The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect.
5 FAULTIN	**Fault Inputs** 0 : The logic OR of the enabled fault inputs is 0. 1 : The logic OR of the enabled fault inputs is 1. Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled.
2 FAULTDF2	**Fault Detection Flag 2**

Bits	Description
	0 : No fault condition was detected at the fault input. 1 : A fault condition was detected at the fault input. Clear FAULTDF2 by reading the FDSR register while FAULTDF2 is set and then writing a 0 to FAULTDF2. Writing a 1 to FAULTDF2 has no effect. FAULTDF2 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the clear operation is invalid and the FAULTDF2 remains set.
1 FAULTDF1	**Fault Detection Flag 1** 0 : No fault condition was detected at the fault input. 1: A fault condition was detected at the fault input. Clear FAULTDF1 by reading the FDSR register while FAULTDF1 is set and then writing a 0 to FAULTDF1. Writing a 1 to FAULTDF1 has no effect. FAULTDF1 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the clear operation is invalid and the FAULTDF1 remains set.
0 FAULTDF0	**Fault Detection Flag 0** 0 : No fault condition was detected at the fault input. 1 : A fault condition was detected at the fault input. Clear FAULTDF0 by reading the FDSR register while FAULTDF0 is set and then writing a 0 to FAULTDF0. Writing a 1 to FAULTDF0 has no effect. FAULTDF0 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the clear operation is invalid and the FAULTDF0 remains set.

11.5.17 PWM_CAPFILTER

Table 11-24 PWM_CAPFILTER register

PWM_CAPFILTER **Input Capture Filter Control** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													CH3CAPFVAL[4: 1]			
Type													RW			
Reset													0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3CAPFVAL[0: 0]	CH2CAPFVAL					CH1CAPFVAL					CH0CAPFVAL				
Type	RW	RW					RW					RW				

PWM_CAPFILTER	Input Capture Filter Control			Reset:00000000
Reset	0	0	0	0

Bits	Description
19: 15 CH3CAPFVAL	**Channel 3 Input Filter** Selects the filter value for the channel input. The filter is disabled when the value is zero.
14: 10 CH2CAPFVAL	**Channel 2 Input Filter** Selects the filter value for the channel input. The filter is disabled when the value is zero.
9: 5 CH1CAPFVAL	**Channel 1 Input Filter** Selects the filter value for the channel input. The filter is disabled when the value is zero.
4: 0 CH0CAPFVAL	**Channel 0 Input Filter** Selects the filter value for the channel input. The filter is disabled when the value is zero.

11.5.18 PWM_FFAFER

Table 11-25 PWM_FFAFER register

PWM_FFAFER	Fault Filter and Fault Enable Register															Reset:00000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FFVAL									FF2EN	FF1EN	FF0EN		FER2EN	FER1EN	FER0EN
Type	RW									RW	RW	RW		RW	RW	RW
Reset	0									0	0	0		0	0	0

Bits	Description
15: 8 FFVAL	**Fault Input Filter** Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero.
6 FF2EN	**Fault Input 2 Filter Enable** 0: Fault input filter is disabled. 1: Fault input filter is enabled.

Bits	Description
	Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
5 FF1EN	**Fault Input 1 Filter Enable** 0: Fault input filter is disabled. 1: Fault input filter is enabled. Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
4 FF0EN	**Fault Input 0 Filter Enable** 0: Fault input filter is disabled. 1: Fault input filter is enabled. Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
2 FER2EN	**Fault Input 2 Enable** 0: Fault input is disabled. 1: Fault input is enabled. Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
1 FER1EN	**Fault Input 1 Enable** 0: Fault input is disabled. 1: Fault input is enabled. Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0 FER0EN	**Fault Input 0 Enable** 0: Fault input is disabled. 1: Fault input is enabled. Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.19 PWM_QDI

Table 11-26 PWM_QDI register

PWM_QDI **Quadrature Decoder Interface Configuration Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																

PWM_QDI **Quadrature Decoder Interface Configuration Register** **Reset:00000000**

Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											PHAPOL	PHBPOL	QUADMODE	QUADIR	CNTOFDIR	QDIEN
Type											RW	RW	RW	RO	RO	RW
Reset											0	0	0	1	0	0

Bits	Description
5 PHAPOL	**Phase A Input Polarity** 0: Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal. 1: Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal. Selects the polarity for the quadrature decoder phase A input.
4 PHBPOL	**Phase B Input Polarity** 0: Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal. 1: Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal. Selects the polarity for the quadrature decoder phase B input.
3 QUADMODE	**Quadrature Decoder Mode** 0: Phase A and phase B encoding mode. 1: Count and direction encoding mode. Selects the encoding mode used in the Quadrature Decoder mode.
2 QUADIR	**PWM Counter Direction in Quadrature Decoder Mode** 0: Counting direction is decreasing (PWM counter decrement). 1: Counting direction is increasing (PWM counter increment). Indicates the counting direction.
1 CNTOFDIR	**Timer Overflow Direction in Quadrature Decoder Mode** 0: TOF bit was set on the bottom of counting. There was an PWM counter decrement and PWM counter changes from its minimum value (CNTIN register) to its maximum value (MCVR register). 1: TOF bit was set on the top of counting. There was an PWM counter increment and PWM counter changes from its maximum value (MCVR register) to its minimum value (CNTIN register).

Bits	Description
	Indicates if the TOF bit was set on the top or the bottom of counting.
0 QDIEN	**Quadrature Decoder Mode Enable** 0: Quadrature Decoder mode is disabled. 1: Quadrature Decoder mode is enabled. Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the PWM counter direction. The Quadrature Decoder mode has precedence over the other modes. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.20 PWM_CONF

Table 11-27 PWM_CONF register

PWM_CONF **Configuration register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EVENT7PSC		EVENT6PSC		EVENT5PSC		EVENT4PSC		EVENT3PSC		EVENT2PSC		EVENT1PSC		EVENT0PSC	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name						GTBEOUT	GTBEEN			CNTOFNUM						
Type						RW	RW			RW						
Reset						0	0			0						

Bits	Description
31: 30 EVENT7PSC	**Channel 7 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
29: 28 EVENT6PSC	**Channel 6 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
27: 26 EVENT5PSC	**Channel 5 input event prescaler**

Bits	Description
	00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
25: 24 EVENT4PSC	**Channel 4 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
23: 22 EVENT3PSC	**Channel 3 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
21: 20 EVENT2PSC	**Channel 2 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
19: 18 EVENT1PSC	**Channel 1 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
17: 16 EVENT0PSC	**Channel 0 input event prescaler** 00: 1 input event triggers a capture 01: 2 input event triggers a capture 10: 4 input event triggers a capture 11: 8 input event triggers a capture
10 GTBEOUT	**Global time base output** 0: disable GTB signal generation 1: enable GTB signal generation Enables the global time base signal generation of other PWM.
9 GTBEEN	**Enable global time base** 0: disable external global time base 1: enable external global time base
6: 0 CNTOFNUM	**CNTOF Frequency**

Bits	Description
	Selects the ratio between the number of counter overflows to the number of times the CNTOF bit is set. CNTOFNUM = 0: The CNTOF bit is set for each counter overflow. CNTOFNUM = 1: The CNTOF bit is set for the first counter overflow but not for the next overflow. CNTOFNUM = 2: The CNTOF bit is set for the first counter overflow but not for the next 2 overflows. CNTOFNUM = 3: The CNTOF bit is set for the first counter overflow but not for the next 3 overflows.

11.5.21 PWM_FLTPOL

Table 11-28 PWM_FLTPOL register

PWM_FLTPOL **PWM Fault Input Polarity** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														FLT2POL	FLT1POL	FLT0POL
Type														RW	RW	RW
Reset														0	0	0

Bits	Description
2 FLT2POL	**Fault Input 2 Polarity** 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault. Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
1 FLT1POL	**Fault Input 1 Polarity** 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault. Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0 FLT0POL	**Fault Input 0 Polarity** 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault.

Bits	Description
	Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

11.5.22 PWM_SYNCONF

Table 11-29 PWM_SYNCONF register

PWM_SYNCONF **Synchronization Configuration** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name										HWPOL	SWPOL	SWVHWSYNC	INVHWSYNC	OMVHWSYNC	PWMSVHWSYNC	CNTVHWSYNC
Type										RW	RW	RW	RW	RW	RW	RW
Reset										0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				SWVSWSYNC	INVSWSYNC	OMVSWSYNC	PWMSVSWSYNC	CNTVSWSYNC	SYNCMODE		SWOC	INVC		CNTINC		HWTRIGMODESEL
Type				RW	RW	RW	RW	RW	RW		RW	RW		RW		RW
Reset				0	0	0	0	0	0		0	0		0		0

Bits	Description
22 HWPOL	**Channel CHPOLCR synchronization is activated by a hardware trigger.** 0: A hardware trigger does not activate the CHPOLCR register synchronization. 1: A hardware trigger activates CHPOLCR register synchronization.
21 SWPOL	**Channel CHPOLCR synchronization is activated by the software trigger.** 0: The software trigger does not activate the CHPOLCR register synchronization. 1: The software trigger activates CHPOLCR register synchronization.
20 SWVHWSYNC	**Software output control synchronization is activated by a hardware trigger.**

Bits	Description
	0: A hardware trigger does not activate the CHOSWCR register synchronization. 1 : A hardware trigger activates the CHOSWCR register synchronization.
19 INVHWSYNC	**Inverting control synchronization is activated by a hardware trigger.** 0: A hardware trigger does not activate the INVCR register synchronization. 1: A hardware trigger activates the INVCR register synchronization.
18 OMVHWSYNC	**Output mask synchronization is activated by a hardware trigger.** 0 : A hardware trigger does not activate the OMCR register synchronization. 1: A hardware trigger activates the OMCR register synchronization.
17 PWMSVHWSYNC	**MCVR, CNTIN, and CHV registers synchronization is activated by a hardware trigger.** 0 : A hardware trigger does not activate MCVR, CNTIN, and CHV registers synchronization. 1: A hardware trigger activates MCVR, CNTIN, and CHV registers synchronization.
16 CNTVHWSYNC	**PWM counter synchronization is activated by a hardware trigger.** 0 : A hardware trigger does not activate the PWM counter synchronization. 1 : A hardware trigger activates the PWM counter synchronization.
12 SWVSWSYNC	**Software output control synchronization is activated by the software trigger.** 0: The software trigger does not activate the CHOSWCR register synchronization. 1: The software trigger activates the CHOSWCR register synchronization.
11 INVSWSYNC	**Inverting control synchronization is activated by the software trigger.** 0 : The software trigger does not activate the INVCR register synchronization. 1 : The software trigger activates the INVCR register synchronization.
10 OMVSWSYNC	**Output mask synchronization is activated by the software trigger.** 0 : The software trigger does not activate the OMCR register synchronization. 1: The software trigger activates the OMCR register synchronization.
9 PWMSVSWSYNC	**MCVR, CNTIN, and CHV registers synchronization is activated by the software trigger.** 0 : The software trigger does not activate MCVR, CNTIN, and CHV registers synchronization. 1: The software trigger activates MCVR, CNTIN, and CHV registers synchronization.
8 CNTVSWSYNC	**PWM counter synchronization is activated by the software trigger.**

Bits	Description
	0 : The software trigger does not activate the PWM counter synchronization. 1 : The software trigger activates the PWM counter synchronization.
7 SYNCMODE	**Synchronization Mode** 0: Legacy PWM synchronization is selected. 1 : Enhanced PWM synchronization is selected. Selects the PWM Synchronization mode.
5 SWOC	**CHOSWCR Register Synchronization** 0 : CHOSWCR register is updated with its buffer value at all rising edges of system clock. 1 : CHOSWCR register is updated with its buffer value by the PWM synchronization.
4 INVC	**INVCR Register Synchronization** 0: INVCR register is updated with its buffer value at all rising edges of system clock. 1: INVCR register is updated with its buffer value by the PWM synchronization.
2 CNTINC	**CNTIN Register Synchronization** 0 : CNTIN register is updated with its buffer value at all rising edges of system clock. 1: CNTIN register is updated with its buffer value by the PWM synchronization.
0 HWTRIGMODESEL	**Hardware Trigger Mode** 0: PWM clears the TRIG j bit when the hardware trigger j is detected, where j = 0, 1, 2. 1: PWM does not clear the TRIG j bit when the hardware trigger j is detected, where j = 0, 1, 2.

11.5.23 PWM_INVCR

Table 11-30 PWM_INVCR register

PWM_INVCR **PWM Inverting Control Register** **Reset:00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													PAIR3INVEN	PAIR2INVEN	PAIR1INVEN	PAIR0INVEN
Type													RW	RW	RW	RW
Reset													0	0	0	0

Bits	Description
3 PAIR3INVEN	**Pair Channels 3 Inverting Enable** 0: Inverting is disabled. 1: Inverting is enabled.
2 PAIR2INVEN	**Pair Channels 2 Inverting Enable** 0: Inverting is disabled. 1: Inverting is enabled.
1 PAIR1INVEN	**Pair Channels1 Inverting Enable** 0: Inverting is disabled. 1: Inverting is enabled.
0 PAIR0INVEN	**Pair Channels 0 Inverting Enable** 0: Inverting is disabled. 1: Inverting is enabled.

11.5.24 PWM_CHOSWCR

Table 11-31 PWM_CHOSWCR register

PWM_CHOSWCR **PWM Channel Software Output Control Register** **Reset:00000000**

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

PWM_CHOSWCR **PWM Channel Software Output Control Register** **Reset:00000000**

Name	CH7SWCV	CH6SWCV	CH5SWCV	CH4SWCV	CH3SWCV	CH2SWCV	CH1SWCV	CH0SWCV	CH7SWEN	CH6SWEN	CH5SWEN	CH4SWEN	CH3SWEN	CH2SWEN	CH1SWEN	CH0SWEN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
15 CH7SWCV	**Channel 7 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
14 CH6SWCV	**Channel 6 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
13 CH5SWCV	**Channel 5 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
12 CH4SWCV	**Channel 4 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
11 CH3SWCV	**Channel 3 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
10 CH2SWCV	**Channel 2 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
9 CH1SWCV	**Channel 1 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
8 CH0SWCV	**Channel 0 Software Output Control Value** 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
7 CH7SWEN	**Channel 7 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
6 CH6SWEN	**Channel 6 Software Output Control Enable**

Bits	Description
	0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
5 CH5SWEN	**Channel 5 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
4 CH4SWEN	**Channel 4 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
3 CH3SWEN	**Channel 3 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
2 CH2SWEN	**Channel 2 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
1 CH1SWEN	**Channel 1 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.
0 CH0SWEN	**Channel 0 Software Output Control Enable** 0: The channel output is not affected by software output control. 1 : The channel output is affected by software output control.

12 PWDT

12.1 Introduction

Pulse Width Detect Timer (PWDT) is used to measure the pulse width or as a 16-bit timer. The device contains two PWDT modules, each PWDT module supports 3 external channels and 1 internal channel input.

12.2 Features

- 4 selectable pulse input.
- Support 2 functions: pulse width measurement function and timer function.
  - For pulse width measurement function:
    - Programmable start trigger edge
    - 4 programmable measurement modes
    - Support 3 hall sensors signal input measurement
    - Support 3 inputs from analog comparator.
  - For timer function
- 16-bit counter used for pulse width measurement or timer function.
- Interrupts:
  - OVF: counter overflows
  - RDYF: pulse width measurement value update

12.3 Block diagram

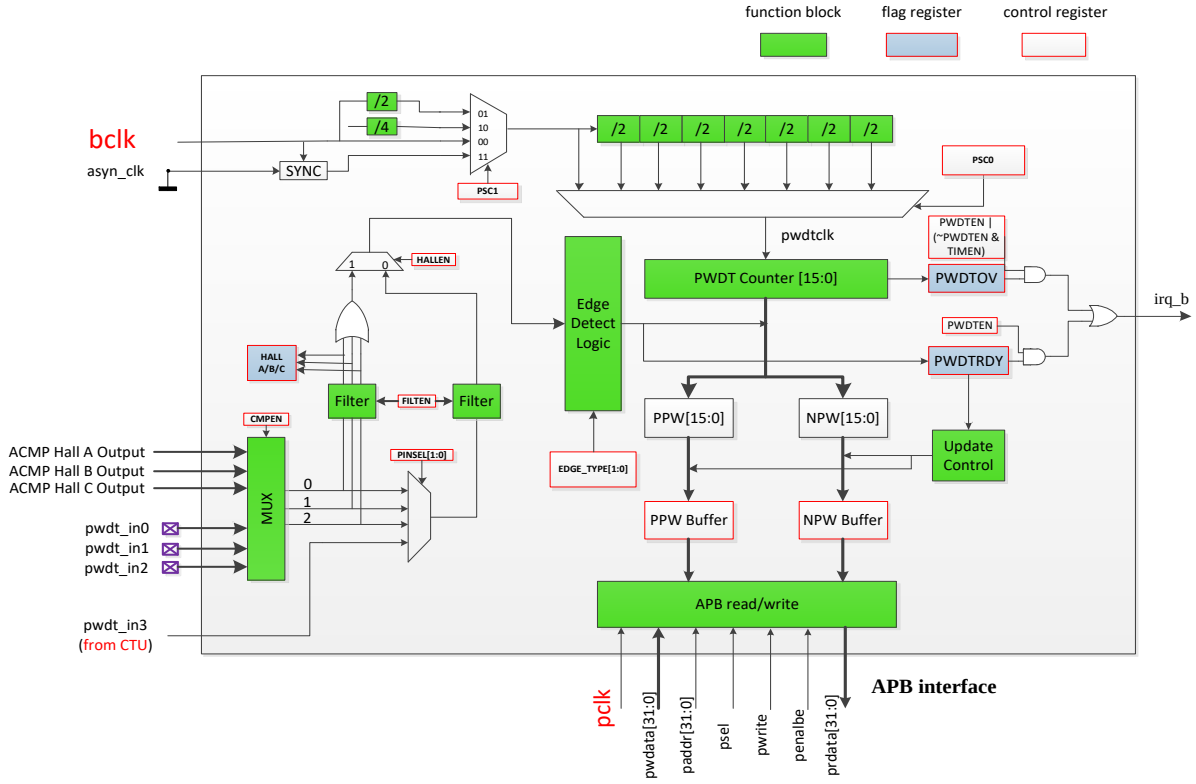

Figure 12-1 PWDT block diagram

12.4 Functional description

12.4.1 Pulse Width Measurement function

To get the pulse width measurement value, the PWDTC counter runs based on the pwdtclk described in block diagram after PWDTEN=1. And pwdtclk is deriving from the bus clock division by the PSC0[2:0] and PSC1[1:0], PSC1 is the pre-divider, PSC0 is the post-divider. In order to get more accurate measurement value, user had better take the smaller value of PSC for narrower pulse input and larger value of PSC for wider pulse input.

For pulse width measurement, user must clearly know the application scenarios, which mainly include two conditions: one is just single channel input for general measurement and the other is 3 channel inputs for hall measurement.

12.4.1.1 General measurement mode

For general measurement, user choose to measure specific channel input by setting PINSEL[1:0] and can choose one of the 4 measurement modes by setting EDGE[1:0] based on the practical applications, as illustrated in Figure 12-2.

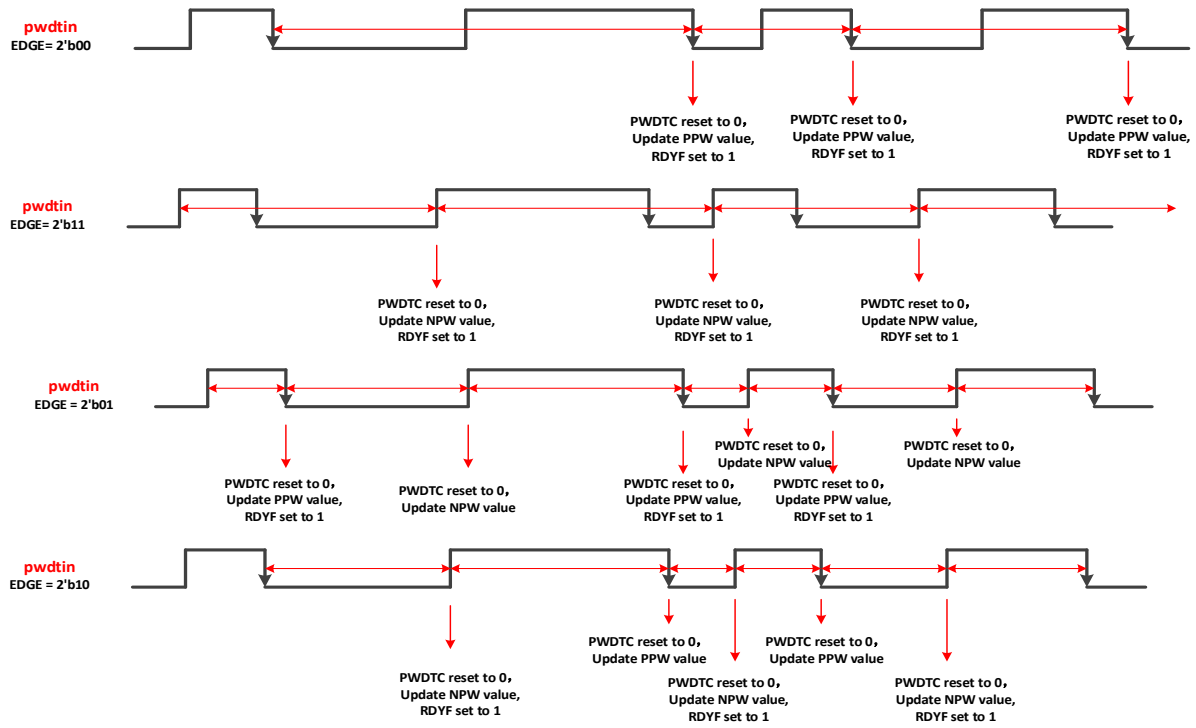

Figure 12-2 Four basic measurement modes(HALLEN=0)

12.4.1.2 Hall measurement mode

For hall measurement, the module measure the pulse input deriving from XOR of the 3 channel inputs and user should set the EDGE[1:0] = 2'b01. Further, the configuration of the internal 3 channel comparator inputs is similar to the hall measurement and just CMPEN should be configured to 1'b1 simultaneously.

For hall measurement or 3 internal comparator inputs, select this mode for motor speed calculation or commutation and is illustrated in Figure 12-3. Compared to the general measurement mode, the RDYF flag is set as 1 on both the rising edge and the falling edge.

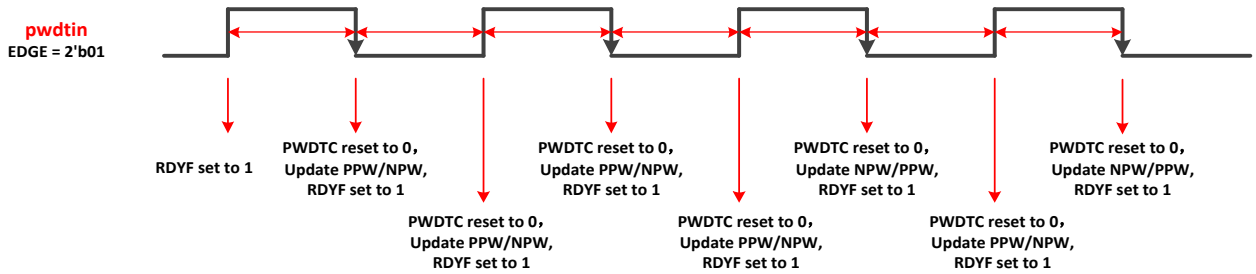

Figure 12-3 Hall measurement modes(HALLEN=1)

In the motors, installation of hall devices is used for detecting location of the rotor to commutate properly. And there usually two installations as Figure 12-4. One is 120 electrical degree interval and the other is 60 electrical degree interval.

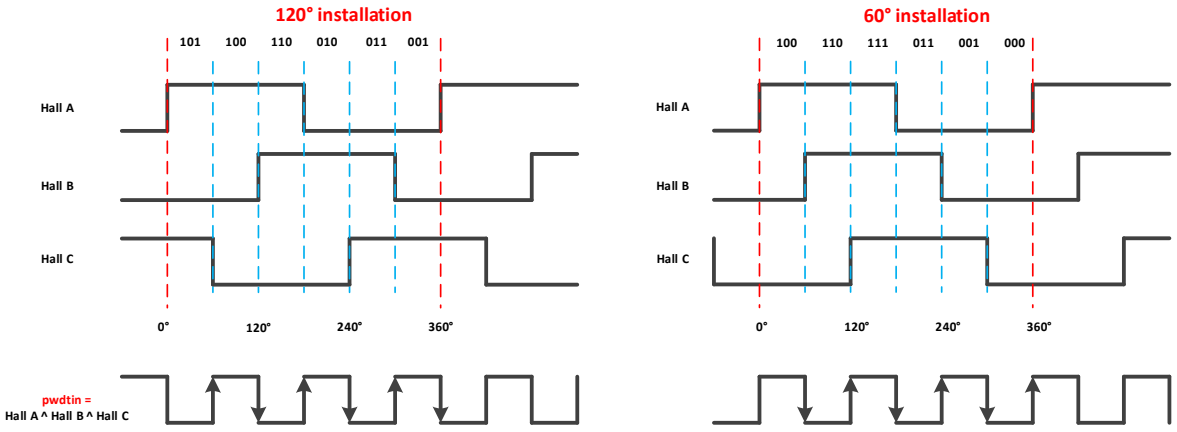

Figure 12-4 Two common installation ways

12.4.1.3 Input filter

Input filter is designed for filter the noise signal which high/low level is less than specific width described in the paragraph. The FILT_PSC[3:0] and FILTVAL[3:0] settings determine the maximum and minimum noise pulse width. The Figure 12-5 and Figure 12-6 introduce the noise width settings, judgement and filter. When user configures the FILTVAL=15 and FILT_PSC=2, the filter pulse width is 60 bclk and pulses less than 60 bclk are judged as noise pulse and will be filtered. The filterable pulse width is listed in Table 12-1.

Note: Enabling the filter function will cause signal delay (the delay time is the filter width). Therefore, if the PWDT module is running in the Hall measurement mode of the motor application, it may be necessary to consider the commutation time offset caused by the filter delay, and the user can make corresponding software compensation based on the filter delay.

Table 12-1 Filterable pulse width range

Item	Clock
Filterable pulse width range	2 bclk ~ 15*4096 bclk

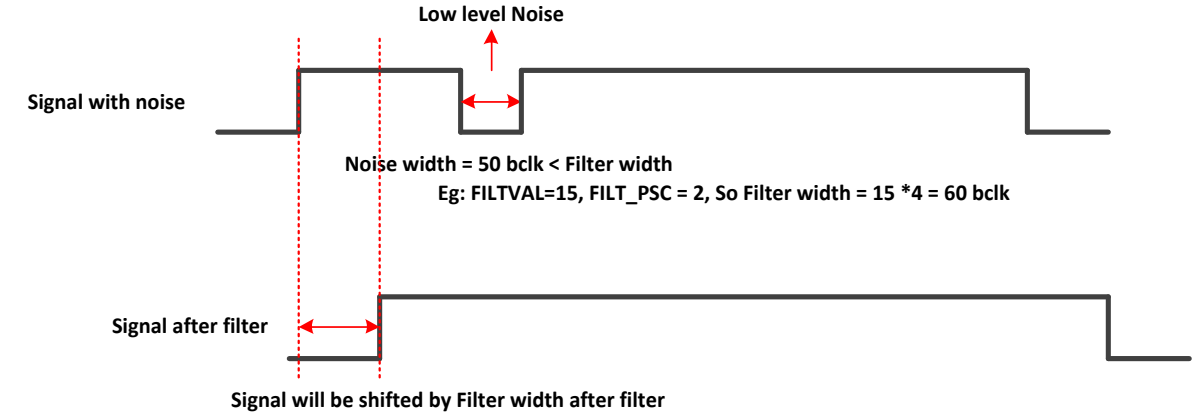

Figure 12-5 Example for low level noise and filter

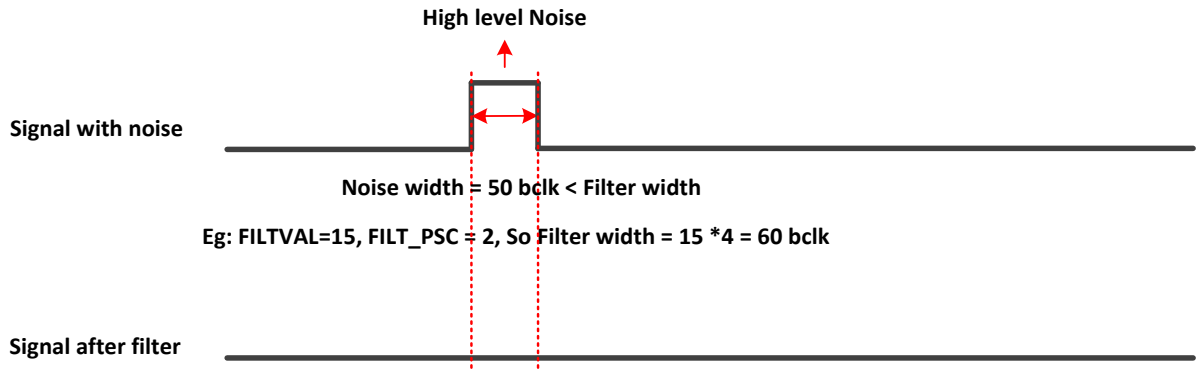

Figure 12-6 Example for high level noise and filter

12.4.1.4 Measurement error

User should understand the measurement accuracy for pulse width measurement to configure proper value of PSC to reach a more accurate measurement value. A basic principle is that more accurate measurement value can be gotten with smaller PSC. Obviously, more narrow input pulse, more relative measurement error. The Figure 12-7 describes the error when it operates for pulse width measurement function. In the Figure 12-7, the PWDTC counter and the pwdtclk divisor counter reset to 0 simultaneously when the pwdtin pulse changes from high to low or from low to high. And exactly here the counting error occurs which is deriving from the last counting value illustrated in Figure 12-7. The practical width value is less than measurement value by less than a pwdtclk period.

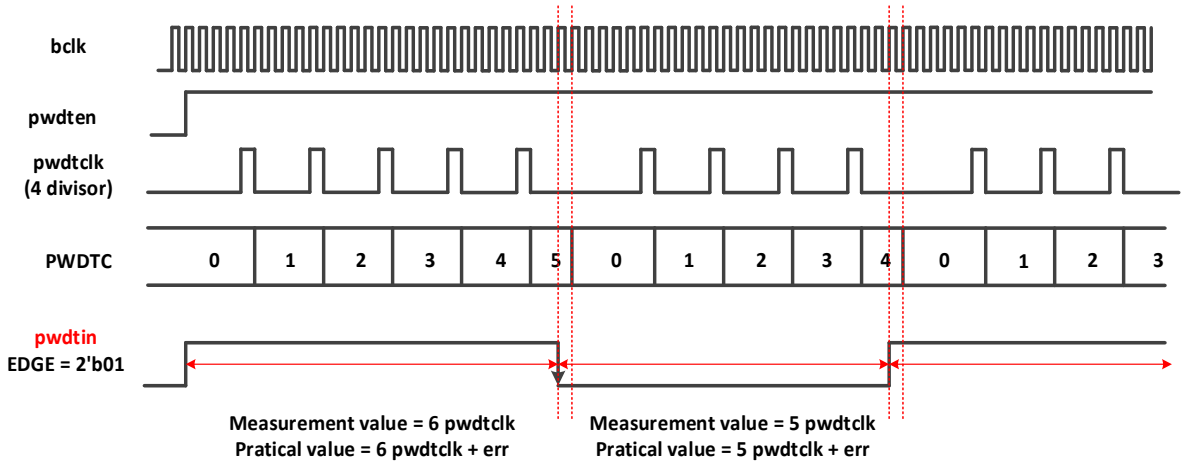

Figure 12-7 PWDTC counter and counting error

12.4.2 Timer function

For timer function, only OVF status is valid and occurs when PWDTC counter overflows. The counter load value TIMCNTVAL[15:0] can be modified all the time, but different timing of modifying the value leads to different operation in Figure 12-8 and Figure 12-9.

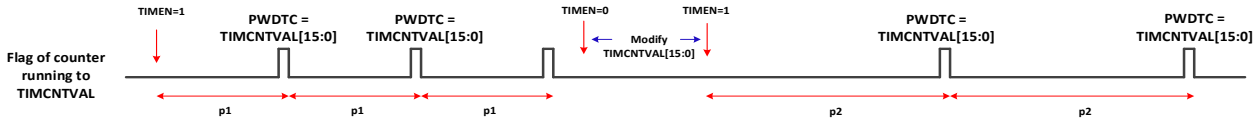

Figure 12-8 Modify the TIMCNTVAL during TIMEN=0

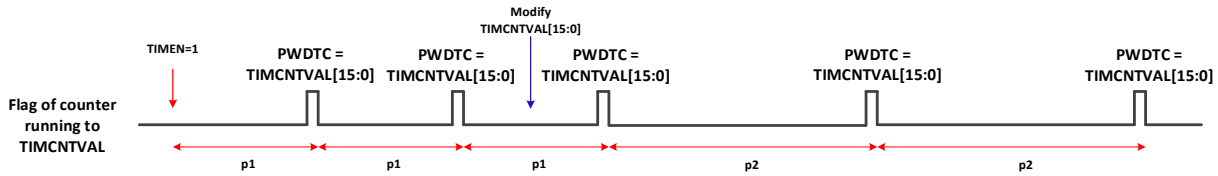

Figure 12-9 Modify the TIMCNTVAL during TIMEN=1

12.5 Program guide

12.5.1 Pulse Width Measurement function program guide

User must keep in mind that PWDTEN should be configured to 1 after all other controlling bits. Otherwise, abnormal condition may occur. Especially, HALLEN and CMPEN should be configured to 1 for the internal 3 comparator inputs.

12.5.2 Timer function program guide

Timer function is used easily with just configuration of the TIMLDVAL, PRESCALE and TIMEN and so on. And user should set TIMEN to 1 at last and must not set the PWDTEN to 1, because the pulse width measurement function is prior to timer function.

12.6 Register definition

Table 12-2 PWDT register mapping

PWDT0 Base address = 0x40017000
PWDT1 Base address = 0x40017800

Address	Name	Width (in bit)	Description
PWDTx base address +0x0	PWDT_INIT0	32	General control, status bits and positive pulse width contents
PWDTx base address +0x04	PWDT_NPW	32	Negative pulse width content and 16-bit free running counter
PWDTx base address +0x08	PWDT_INIT1	32	Hall function control and timer function control

Note: x=0,1 in the above table.

12.6.1 PWDT_INIT0

Table 12-3 PWDT_INIT0 register

PWDT_INIT0 **PWDT Initialization Register 0** **Reset:00000000**

Bit	31~16	15~14	13~12	11~10	9~7	6	5	4	3	2	1	0
Name	PPW	PSC1	PINSEL	EDGE	PSC0	PWDTEN	IE	PRDYIE	OVIE		RDYF	OVF
Type	RO	RW	RW	RW	RW	RW	RW	RW	RW		W0C	W0C
Reset	0	0	0	0	0	0	0	0	0		0	0

Bits	Description
31: 16 PPW	**Positive Pulse Width** Indicate the positive pulse width value.
15: 14 PSC1	**PWDT counter pre-prescaler** 00 ~ 11: respectively indicate 1/2/4/reserved.
13: 12 PINSEL	**Pin select** 00/01/10/11: respectively select pwdt_in0/ pwdt_in1/ pwdt_in2/ pwdt_in3. **Note:** internal pwdtin is from the CTU module internally.
11: 10 EDGE	**Selects the input edge trigger type** 00: first falling edge start, and all the subsequent falling edge trigger the pulse width to be captured. 01: first rising edge starts, and all the subsequent rising edge and falling edge trigger width to be captured. 10: first falling edge starts, and all the subsequent rising edge and falling edge trigger width to be captured. 11: first rising edge start, and all the subsequent rising edge trigger the pulse width to be captured. **Note:** details described in the subsequent diagram .
9: 7 PSC0	**PWDT counter post-prescaler** 000 ~ 111: indicates 1/2/4/8/…/128
6 PWDTEN	**PWDT pulse width measurement mode enable** 0: disable 1: enable **Note: PWDTEN (pwdt function) enable prior to TIMEN (timer function), so when timer function is going to be enabled, PWDTEN must be disabled .**

Bits	Description
5 IE	**PWDT interrupt enable** 0: disable 1: enable
4 PRDYIE	**PWDT pulse width data ready interrupt enable** 0: disable 1: enable
3 OVIE	**PWDT counter overflow interrupt enable** 0: disable 1: enable
1 RDYF	**PWDT pulse width data ready** 0: pwdt pulse width register is not up to date. 1: pwdt pulse width register has been updated, write 0 to clear.
0 OVF	**PWDT counter overflow** 0: no overflow 1: pwdt counter overflow, write 0 to clear.

12.6.2 PWDT_NPW

Table 12-4 PWDT_NPW register

PWDT_NPW **PWDT NPW count value** **Reset: 00000000**

Bit	31~16	15~0
Name	PWDTC	NPW
Type	RO	RO
Reset	0	0

Bits	Description
31: 16 PWDTC	**Pulse Width Counter** Used to count for pulse width measurement or for timer.
15: 0 NPW	**Negative Pulse Width Count Value** Indicate the negative pulse width value.

12.6.3 PWDT_INIT1

Table 12-5 PWDT_INIT1 register

PWDT_INIT1 **PWDT Initialization Register 1** **Reset:00000000**

Bit	31	30	29	28	27~12	11	10	9	8	7~4	3~0
Name		HALLA	HALLB	HALLC	TIMLDVAL	CMPEN	TIMEN	HALLEN	FILTEN	FILTPSC	FILTVAL
Type		RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0	0	0	0

Bits	Description
30: 28 **HALLA/B/C**	**HALLA/HALLB/HALLC status value** If 3 hall sensors installed spacing 60 electrical degree: 100 → 110 → 111 → 011 → 001 → 000 else 3 hall sensors installed spacing 120 electrical degree: 101 → 100 → 110 → 010 → 011 → 001
27: 12 **TIMLDVAL**	**Timer counter load value** Counter runs from 0x0000 to TIMLDVAL
11 **CMPEN**	**Comparator input enable** 0: enable the pwdt_in0 ~ pwdt_in2 derived from pad PWDT_IN0 ~ PWDT_IN2 externally. 1: enable pwdt_in0 ~ pwdt_in2 derived from acmp0_0 ~ acmp0_2 internally. **Note:** When CMPEN=1, pwdt_in0 ~ pwdt_in2 are derived from acmp0_0 ~ acmp0_2 internally. And then acmp0_0 ~ acmp0_2 signals will be measured by the behavior of HALL sensor if HALLEN=1, otherwise just one of them will be measured based on the PINSEL .
10 **TIMEN**	**Enable the timer function** 0: disable. 1: enable timer function. **Note: PWDTEN (pwdt function) enable prior to TIMEN (timer function), so when timer function is going to be enabled, PWDTEN must be disabled .**
9 **HALLEN**	**Enable hall sensor signal detect function** 0: disable the hall sensor signal detect function. 1: enable the hall sensor signal detect function.
8 **FILTEN**	**Enable the pwdt input filter function** 0: disable. 1: enable the filter function.

Bits	Description
7 ~ 4 **FILTPSC**	**Prescaler for filter** 1~ 12: respectively represent 2/4/8…/4096 divisor. 0, 13 ~ 15: not divide the filter clock.
3 ~ 0 **FILTVAL**	**Filter value** 0: disable filter 1 ~ 15 : for filter different width pulse.

13 TIMER

13.1 Introduction

The TIMER module is an array of timers that can be used to raise interrupts and triggers.

13.2 Features

- Ability of timers to generate interrupts.
- Ability of timers to generate trigger pulses.
- Independent timeout periods for each timer.
- Supports 4 * 32bit timer.
- Support link mode.

13.3 Block diagram

The following is the block diagram of the TIMER module.

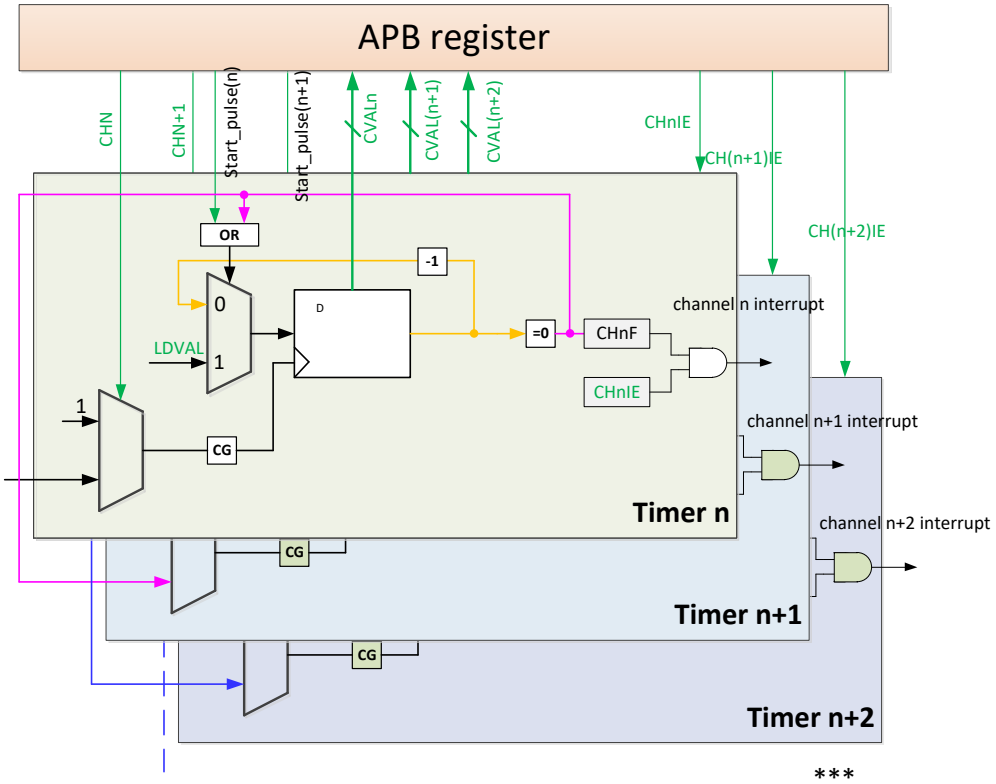

Figure 13-1 TIMER block diagram

13.4 Functional description

13.4.1 General mode

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their TIMER_LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

If desired, the current counter value of the timer can be read via the TIMER_CVAL registers. The counter period can be restarted, by first disabling, and then enabling the timer with TIMER_INIT[TEN].

13.4.2 Link mode

When a timer has link mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows linking some of the timers together to form a longer timer. The first timer (timer 0) cannot be linked to any other timer.

13.4.3 Interrupts

Timer interrupts can be enabled by setting TIE. TIF flag is set to 1 when a timeout occurs on the associated timer, and is cleared to 0 by writing a 1 to the corresponding TIF.

When using the link function, generally only the timer n interrupt is enabled, and the timer interrupts linked to timer n are all off.

13.5 Register definition

Table 13-1 TIMER register mapping

TIMER base address = 0x40011000
TIMER_CH0 base address =0x40011100
TIMER_CH1 base address =0x40011110
TIMER_CH2 base address =0x40011120
TIMER_CH3 base address =0x40011130

Address	Name	Width (in bit)	Description
TIMER base address+0x00	TIMER_MCR	32	TIMER MCR register
TIMER_CHx base address +0x00	TIMER_LDVAL	32	TIMER INITVAL register
TIMER_CHx base address +0x04	TIMER_CVAL	32	TIMER CVAL register
TIMER_CHx base address +0x08	TIMER_INIT	32	TIMER INIT register
TIMER_CHx base address +0x0C	TIMER_TF	32	TIMER TF register

Note: In the above table, x=0~3.

13.5.1 TIMER_MCR

Table 13-2 TIMER_MCR register

TIMER_MCR **TIMER module controller register** **Reset: 0x00000002**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															MDIS	
Type															RW	
Reset															1	

Bits	Description
1 MDIS	**Module Disable - (TIMER section)** 0: Timers module is enabled. 1: Timers module is disabled Disables the timer module. This field must be enabled before any other setup is done. **Note:** MDIS can pause and restart counting of 4 timers at the same time

13.5.2 TIMER_LDVAL

Table 13-3 TIMER_LDVAL register

TIMER_LDVAL **TIMER LDVAL register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LDVAL[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	LDVAL[15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 LDVAL	**Load Value Register**

Bits	Description
	Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer. Instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

13.5.3 TIMER_CVAL

Table 13-4 TIMER_CVAL register

TIMER_CVAL **TIMER CVAL register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CVAL[31: 16]															
Type	RO															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CVAL[15: 0]															
Type	RO															
Reset	0															

Bits	Description
31: 0 CVAL	**Current Timer Value** The conversion formula from CVAL to time is as follows: the timing period(Unit: second) = (CVAL + 1) / timing clock frequency(Unit: HZ)

13.5.4 TIMER_INIT

Table 13-5 TIMER_INIT register

TIMER_INIT **TIMER INIT register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														LINKEN	TIE	TEN
Type														RW	RW	RW
Reset														0	0	0

Bits	Description
2 LINKEN	**Link Mode** 0: Timer is not linked.

Bits	Description
	1: Timer is linked to previous timer. For example, for Timer 2, if this field is set, Timer 2 is linked to Timer 1. When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be linked.
1 TIE	**Timer Interrupt Enable** 0: Interrupt requests from Timer n are disabled. 1: Interrupt will be requested whenever TIF is set. When an interrupt is pending, or, TIF is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF must be cleared first.
0 TEN	**Timer Enable** 0: Timer n is disabled. 1: Timer n is enabled. Enables or disables the timer n.

13.5.5 TIMER_TF

Table 13-6 TIMER_TF register

TIMER_TF **TIMER TF register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																TIF
Type																W1C
Reset																0

Bits	Description
0 TIF	Timer timeout flag 0: Timeout has not yet occurred. 1: Timeout has occurred. Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. When TIE = 1, TIF causes an interrupt request.

14 CTU

14.1 Introduction

The CTU module defines module-to-module interconnections and signal transmission between different modules on the chip.

14.2 Features

- ACMP0 output capture.
- UART TX modulation.
- UART RX capture.
- UART RX filter.
- RTC capture.
- ADC trigger.
- PWM software synchronization.

14.3 Block diagram

The following is the block diagram of CTU.

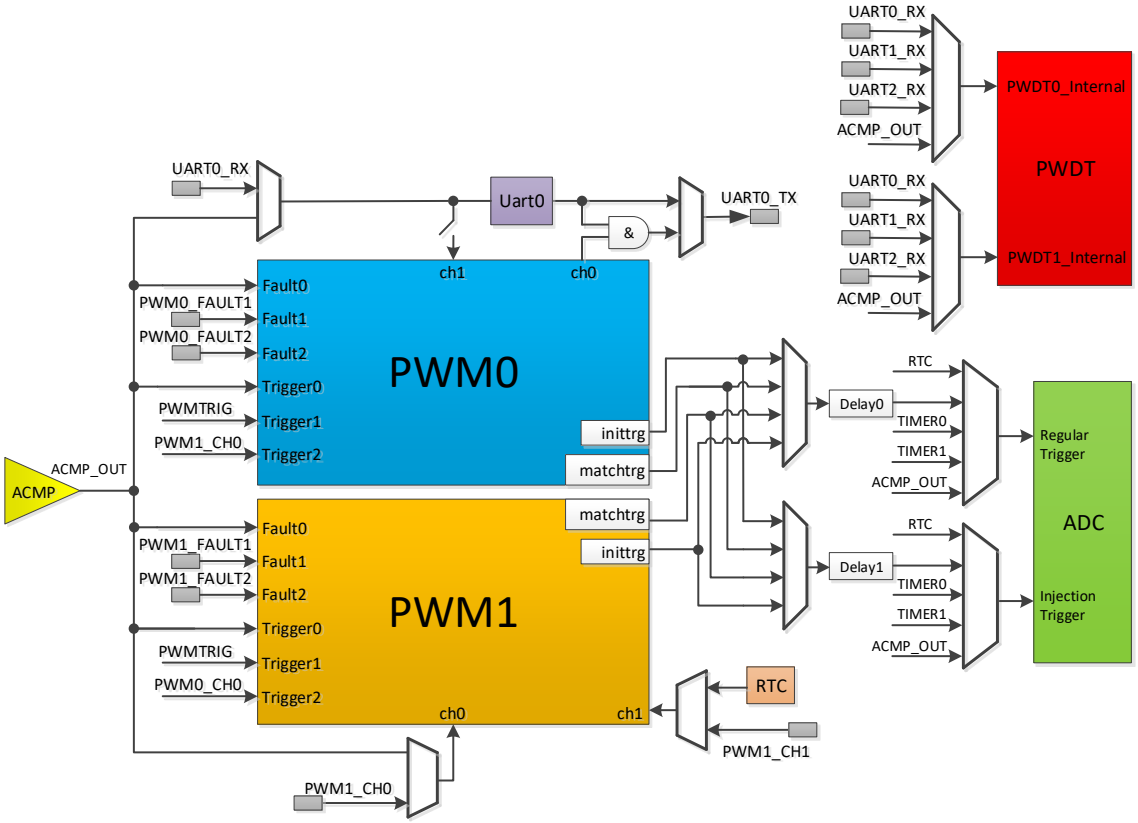

Figure 14-1 CTU block diagram

14.4 Function description

14.4.1 ACMP output capture

The CTU_CONFIG0[ACIC] bit enables the output of ACMP0 to connect to the PWM1_CH0, the PWM1_CH0 pin is released to other multi-functions. When setting the CTU_ CONFIG0 [RXDFE] to 1, the ACMP0 outputs can be selected to connect to the receiver channel of UART0. ACMP0 output is also connected to PWDT input (for BLDC using), or can be used as PWM trigger/fault input and ADC hardware trigger.

14.4.2 UART0_TX modulation

UART0_TX can be modulated by PWM0_CH0 output. When CTU_CONFIG0[TXDME] is set, the UART0_TX is gated by PWM0_CH0 output through an AND gate, and then mapped to UART0_TX pinout. When this field is clear, the UART0_TX is directly mapped on the pinout. To enable IR modulation function, both PWM0_CH0 and UART must be active. By setting the period and the duty cycle of the PWM, each data transmitted via UART0_TX from UART0 is modulated by the PWM0_CH0 output, and the PWM0_CH0 pin is released to other shared functions regardless of the configuration of PWM0 pin reassignment.

14.4.3 UART0_RX capture

When CTU_CONFIG0[RXDCE] is set, the UART0_RX pin is connected to both UART0 and PWM0_CH1, and the PWM0_CH1 pin is released to other multi functions. When this field is clear, the UART0_RX pin is connected to UART0 only.

14.4.4 UART0_RX filter

When CTU_CONFIG0[RXDFE] is set, the ACMP0 output can be connected to the receive channel of UART0. To enable UART0_RX filter function, both UART0 and ACMP must be active. If this function is active, the UART0 external UART0_RX pin is released to other multi-functions regardless of the configuration of UART0 pin reassignment. When UART0_RX capture function is active, the ACMP0 output is injected to PWM0_CH1 as well.

14.4.5 RTC capture

RTC overflow can be captured by PWM1_CH1 by setting CTU_CONFIG0[RTCC] bit. When this bit is set, the RTC overflow is connected to PWM1_CH1 for capture, the PWM1_CH1 pin is released to other multi-functions.

14.4.6 ADC hardware trigger

ADC module can be initialized to a conversion via a hardware trigger. The available ADC hardware regular trigger sources by setting CTU_CONFIG0[ADHWT0], ADC hardware injected trigger sources by setting CTU_CONFIG1[ADHWT1].

When ADC hardware trigger selects the output of PWM triggers, an 8-bit delay block will be enabled. This logic delays any trigger from PWM with an 8-bit counter whose value is specified by CTU_CONFIG0[DELAY]. The reference clock to this module is the bus clock with selectable predivider specified by CTU_CONFIG0 [PSC].

14.4.7 PWM software synchronization

FTM2 contains three synchronization input triggers, one of which is a software trigger by writing 1 to CTU_CONFIG0[PWMTRIG]. Writing 0 to this field takes no effect. This field is always read 0.

14.5 Register definition

Table 14-1 CTU register mapping

CTU base address = 0x40016000

Address	Name	Width (in bit)	Description
CTU base address +0x00	CTU_CONFIG0	32	CTU Configuration register 0
CTU base address +0x04	CTU_CONFIG1	32	CTU Configuration register 1

14.5.1 CTU_CONFIG0

Table 14-2 CTU_CONFIG0 register

CTU_CONFIG0 **CTU Configuration register 0** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DELAY0								DLYACT0	ADHWT0				PSC		
Type	RW								RO	RW				RW		
Reset	0								0	0				0		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TXDME	PWMTRIG		RXDCE	ACIC	RTCC	RXDFE									
Type	RW	RW		RW	RW	RW	RW									
Reset	0	0		0	0	0	0									

Bits	Description
31: 24 DELAY0	**Trigger Delay** Specifies the delay from PWM initial or match trigger to ADC hardware trigger. The 8-bit modulo value allows the delay from 0 to 255 upon the PSC settings. This is a one-shot counter that starts ticking when the trigger arrives and stops ticking when the counter value reaches the modulo value that is defined.
23 DLYACT0	**Trigger DELAY0 Active** 0: The delay is inactive. 1: The delay is active. This read-only field specifies the status if the PWM initial or match delay is active. This field is set when an PWM trigger arrives and the delay counter is ticking. Otherwise, this field will be clear.
22: 20	**ADC Regular Group Hardware Trigger Source**

Bits	Description
ADHWT0	000: RTC overflow as the ADC hardware trigger 001: PWM0 init trigger with 8-bit programmable counter delay 010: PWM0 match trigger with 8-bit programmable counter delay 011: PWM1 init trigger with 8-bit programmable counter delay 100: PWM1 match trigger with 8-bit programmable counter delay 101: TIMER channel0 overflow as the ADC hardware trigger 110 : TIMER channel1 overflow as the ADC hardware trigger 111 : ACMP0 out as the ADC hardware trigger. Selects the ADC hardware trigger source. All trigger sources start ADC conversion on rising-edge.
18: 16 PSC	**BUS Clock Output select** 000: Bus divided by 1 001: Bus divided by 2 010: Bus divided by 4 011: Bus divided by 8 100: Bus divided by 16 101: Bus divided by 32 110: Bus divided by 64 111: Bus divided by 128 Enables bus clock output via an optional prescaler.
15 TXDME	**UART0_TX Modulation Select** 0: UART0_TX output is connected to pinout directly. 1: UART0_TX output is modulated by PWM0 channel 0 before mapped to pinout. Enables the UART0_TX output modulated by PWM0 channel 0.
14 PWMTRIG	**PWM Synchronization Select** 0: No synchronization triggered. 1: Generates a PWM synchronization trigger to the PWM modules. Generates a PWM synchronization trigger to the PWM module if 1 is written to this field. Note that when set PWMTRIG = 1 to generate a trigger behavior, the PWMTRIG bit should be write to 0 manually.
12 RXDCE	**UART0_RX Capture Select** 0: UART0_RX input signal is connected to the UART0 module only. 1: UART0_RX input signal is connected to the UART0 module and PWM0 channel 1. Enables the UART1_RX to be captured by PWM0 channel 1.
11 ACIC	**Analog Comparator to Input Capture Enable** 0: ACMP0 output is not connected to PWM1 input channel 0. 1: ACMP0 output is connected to PWM1 input channel 0.

Bits	Description
	Connects the output of ACMP0 to PWM1 input channel 0.
10 RTCC	**Real-Time Counter Capture** 0: RTC overflow is not connected to PWM1 input channel 1. 1: RTC overflow is connected to PWM1 input channel 1. Allows the Real-time Counter (RTC) overflow to be captured by PWM1 channel 1.
9 RXDFE	**UART0 RxD Filter Select** 00: RXD input signal is connected to UART0 module directly. 01: RXD input signal is filtered by ACMP0, then injected to UART0. Enables the UART0 RxD input to be filtered by ACMP0. When this function is enabled, any signal tagged with ACMP inputs can be regarded UART1.

14.5.2 CTU_CONFIG1

Table 14-3 CTU_CONFIG1 register

CTU_CONFIG1 **CTU Config1 register** **Reset:0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		DELAY1								DLYACT1						
Type		RW								RW						
Reset		0								0						
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				PWDT1IN3S		PWDT0IN3S		ADHWT1								
Type				RW		RW		RW								
Reset				0		0		0								

Bits	Description
30:23 DELAY1	**Injected Trigger Delay** Specifies the delay from PWM initial or match trigger to ADC hardware trigger. The 8-bit modulo value allows the delay from 0 to 255 upon the PSC clock settings. This is a one-shot counter that starts ticking when the trigger arrives and stops ticking when the counter value reaches the modulo value that is defined.
22 DLYACT1	**Trigger DELAY1 Active** 0: The delay is inactive. 1: The delay is active. This read-only field specifies the status if the PWM initial or match delay is active. This field is set when an PWM trigger arrives and the delay counter is ticking. Otherwise, this field will be clear.
12:11	**PWDT1 IN3 input select**

Bits	Description
PWDT1IN3S	
	00: UART0 RX connects to the pwdt_in3 channel 01: UART1 RX connects to the pwdt_in3 channel 10: UART2 RX connects to the pwdt_in3 channel 11: ACMP0 OUT connects to the pwdt_in3 channel Selects PWDT1_IN3 input signal.
10:9 PWDT0IN3S	**PWDT0 IN3 input select** 00: UART0 RX connects to the pwdt_in3 channel 01: UART1 RX connects to the pwdt_in3 channel 10: UART2 RX connects to the pwdt_in3 channel 11: ACMP0 OUT connects to the pwdt_in3 channel Selects PWDT0_IN3 input signal.
8:6 ADHWT1	**ADC Injected Group Hardware Trigger Source** 000: RTC overflow as the ADC hardware trigger 001 : PWM0 init trigger with 8-bit programmable counter delay 010: PWM0 match trigger with 8-bit programmable counter delay 011: PWM1 init trigger with 8-bit programmable counter delay 100: PWM1 match trigger with 8-bit programmable counter delay 101: TIMER channel0 overflow as the ADC hardware trigger 110: TIMER channel1 overflow as the ADC hardware trigger 111: ACMP0 out as the ADC hardware trigger. Selects the ADC hardware trigger source. All trigger sources start ADC conversion on rising-edge.

15 CRC

15.1 Introduction

The cyclic redundancy check (CRC) module generates 16-/32-bit CRC code for error detection.

15.2 Features

- 16-bit or 32-bit check mode.
- Programmable polynomial.
- Transpose input data bitwise or bytewise.
- Transpose output data (the CRC result) bitwise or bytewise.
- Inverse the result bitwise.

15.3 Block diagram

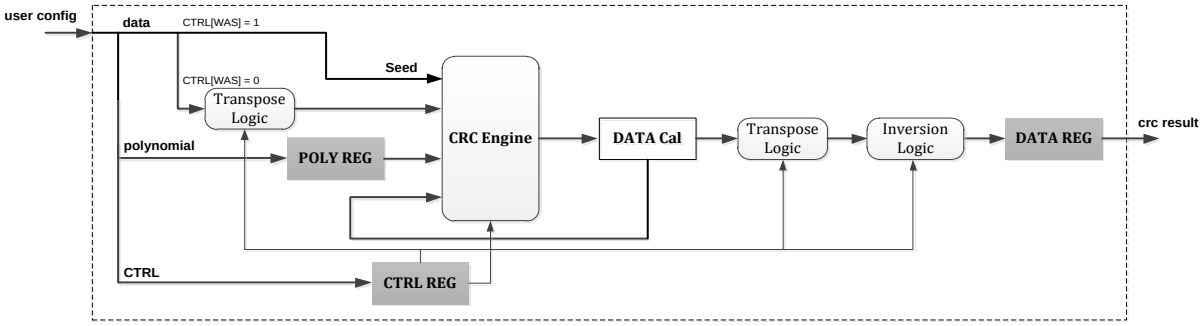

Figure 15-1 CRC block diagram

15.4 Functional description

15.4.1 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on the fly while being read or written.

After the input data transposition is enabled, the written data will be transposed first, and then CRC calculation is done. After the result transposition is enabled, the read data is the data after the CRC check result is transposed. After the result inversion function is enabled, the read data is the result of transposing and inverting the CRC check result.

15.4.2 Transpose type

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CRC_CTRL[TOTW] or CRC_CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CRC_CTRL[TOTW] or CRC_CTRL[TOTR] is 00.

   No transposition occurs.

2. CRC_CTRL[TOTW] or CRC_CTRL[TOTR] is 01.

   Bits in a byte are transposed, while bytes are not transposed. reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}.

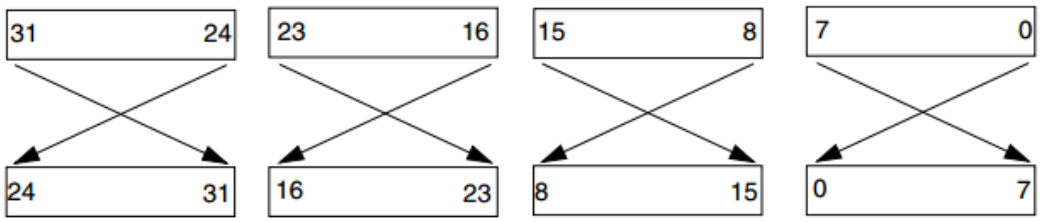

Figure 15-2 [TOTW] / [TOTR] is 01

3. CRC_CTRL[TOTW] or CRC_CTRL[TOTR] is 10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = { reg[0:7], reg[8:15], reg[16:23], reg[24:31]}.

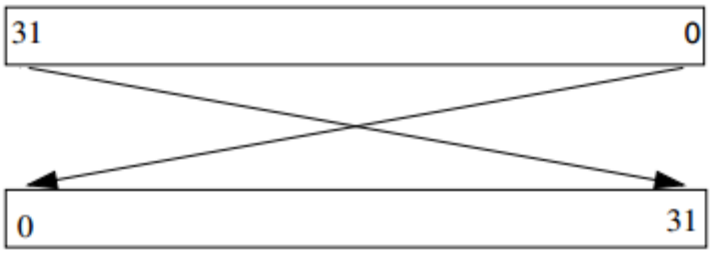

Figure 15-3 [TOTW] / [TOTR] is 10

4. CRC_CTRL[TOTW] or CRC_CTRL[TOTR] is 11.

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}.

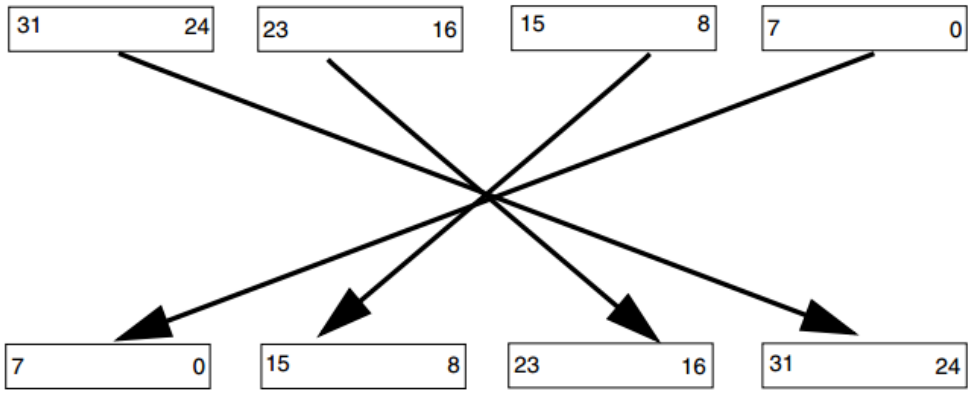

Figure 15-4 [TOTW] / [TOTR] is 11

Note

When writing one byte data to the CRC_DATA register, byte transpose does not occur;

When writing two bytes data to the CRC_DATA register, the two bytes will be transposed;

When writing four bytes data to the CRC_DATA register, the four bytes will be transposed.

15.4.3 CRC result complement

When CRC_CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CRC_CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

15.5 Application note

15.5.1 CRC initialization

Before operating the CRC_POLY and CRC_DATA registers, first configure the CRC_CTRL control register, select the CRC mode CRC16 / 32 to be used, [TCRC] select the CRC mode, [TOTW] enable write data transposition, [TOTR] enable read data transposition, [FXOR] enable read data inversion, [WAS] select whether to write seed or check data to the CRC_DATA register.

15.5.2 CRC Polynomial Configuration

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

15.5.3 CRC check

The CRC_DATA Data register contains the value of the seed, data, and checksum.

When CRC_CTRL[WAS] is set, any write to the data register is regarded as the seed value.

When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value. When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

15.5.4 CRC program guide

15.5.4.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 16-bit polynomial to the CRC_POLY[LOW] field. The CRC_POLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[15:0]. CRC_DATA[31:16] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[31:0]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[15:0].
8. When all values have been written, read the final CRC result from CRC_DATA[15:0].

15.5.4.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_POLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[31:0].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[31:0]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[31:0].
8. When all values have been written, read the final CRC result from CRC_DATA[31:0]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

15.6 Register description

Table 15-1 CRC register mapping

CRC base address: 0x20081000

Address	Name	Width (in bit)	Description
CRC base address + 0x00	CRC_DATA	32	CRC Data register
CRC base address + 0x04	CRC_POLY	32	CRC polynomial register
CRC base address + 0x08	CRC_CTRL	32	CRC control register

15.6.1 CRC_DATA

Table 15-2 CRC_DATA register

CRC_DATA **CRC Data register** **Reset:0xFFFFFFFF**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DATA[31: 24]								DATA[23: 16]							
Type	RW								RW							
Reset	0xFF								0xFF							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA[15: 8]								DATA[7: 0]							
Type	RW								RW							
Reset	0xFF								0xFF							

Bits	Description
31: 24 DATA	**CRC DATA Byte3** In 16-bit CRC mode (CRC_CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CRC_CTRL[TCRC] is 1), values written to this field are part of the seed value when CRC_CTRL[WAS] is 1. When CRC_CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23: 16 DATA	**CRC DATA Byte2** In 16-bit CRC mode (CRC_CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CRC_CTRL[TCRC] is 1), values written to this field are part of the seed value when CRC_CTRL[WAS] is 1. When CRC_CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
15: 8 DATA	**CRC DATA Byte1** When CRC_CTRL[WAS] is 1, values written to this field are part of the seed value. When CRC_CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
7: 0 DATA	**CRC DATA Byte0** When CRC_CTRL[WAS] is 1, values written to this field are part of the seed value. When CRC_CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

15.6.2 CRC_POLY

Table 15-3 CRC_POLY register

CRC_POLY **CRC polynomial register** **Reset: 0x00001021**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	POLY[31: 16]															
Type	RW															
Reset	0x0000															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	POLY[15: 0]															
Type	RW															
Reset	0x1021															

Bits	Description
31: 16 POLY	**Polynominal High Half-word** The field is writable and readable in 32-bit CRC mode (CRC_CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CRC_CTRL[TCRC] is 0).
15: 0 POLY	**Polynominal Low Half-word**

Bits	Description
	Writable and readable in both 32-bit and 16-bit CRC modes

15.6.3 CRC_CTRL

Table 15-4 CRC_CTRL register

CRC_CTRL **CRC control register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									TOTW		TOTR			FXOR	WAS	TCRC
Type									RW		RW			RW	RW	RW
Reset									0		0			0	0	0

Bits	Description
7: 6 TOTW	**Type of Transpose for Writes** 00: No transposition. 01: Bits in bytes are transposed; bytes are not transposed. 10: Both bits in bytes and bytes are transposed. 11: Only bytes are transposed; no bits in a byte are transposed. Defines the transpose configuration of the data written to the CRC data register.
5: 4 TOTR	**Type of Transpose for Read** 00 : No transposition. 01: Bits in bytes are transposed; bytes are not transposed. 10: Both bits in bytes and bytes are transposed. 11: Only bytes are transposed; no bits in a byte are transposed. Identifies the transpose configuration of the value read from the CRC Data register.
2 FXOR	**Complement Read of CRC Data Register** 0 : No XOR on reading. 1 : Invert or complement the read value of the CRC Data register. Some CRC protocols require the final checksum to be XOR with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data.
1 WAS	**CRC_DATA Register Writing type** Defines the type when the CRC_DATA register is writing

Bits	Description
	0 : Writes to the CRC data register are data values. 1 : Writes to the CRC data register are seed values.
0 TCRC	**Type of CRC** 0: CRC16 1: CRC32 Defines CRC check type.

16 GPIO

16.1 Introduction

The general-purpose input and output (GPIO) module is accessible via APB BUS and also accessible via AHB BUS for maximum performance. The GPIO registers support APB 32-bit accesses, and AHB byte accesses.

When the pin is configured for the GPIO function, the GPIO_CR register control the direction. The GPIO_ODR register control output data of each pin. Also the GPIO output level is controlled by setting/resetting the GPIO_BSRR register, and setting GPIO_BRR register.

When the pins are configured for input functions, the GPIO input data register shows the high and low levels on each pin (1 for high level, 0 for low level).

Also, The MCU I/O pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals sharing the same I/O pin. Each I/O pin has a multiplexer with alternate function that can be configured through the GPIO_PINMUX registers.

When the function of a certain peripheral/module needs to be transferred from the current IO to another IO, in addition to the new IO that needs to connect the multiplexing function to the peripheral/module, the multiplexing configuration of the original IO also needs to be closed, otherwise it will cause Peripherals/modules work abnormally on new IO pins.

16.2 Features

GPIO pins support the following modes:

- Up to 42 I/Os under control.
- Output states: push-pull or open drain (be related to I2C).
- Output data from output data register (GPIO_ODR) or peripheral (alternate function output).
- Driving capability selection for each I/O.
- Input states: floating, pull-up/down, analog (be related to ADC PAD).
- Input data to input data register (GPIO_IDR) or peripheral (alternate function input).
- Bit set and reset register (GPIO_BSRR) for bitwise write access to GPIO_ODR.
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions.
- Configurable rising or falling edge interrupt.
- Low power mode wakes up interrupt.

16.3 Block diagram

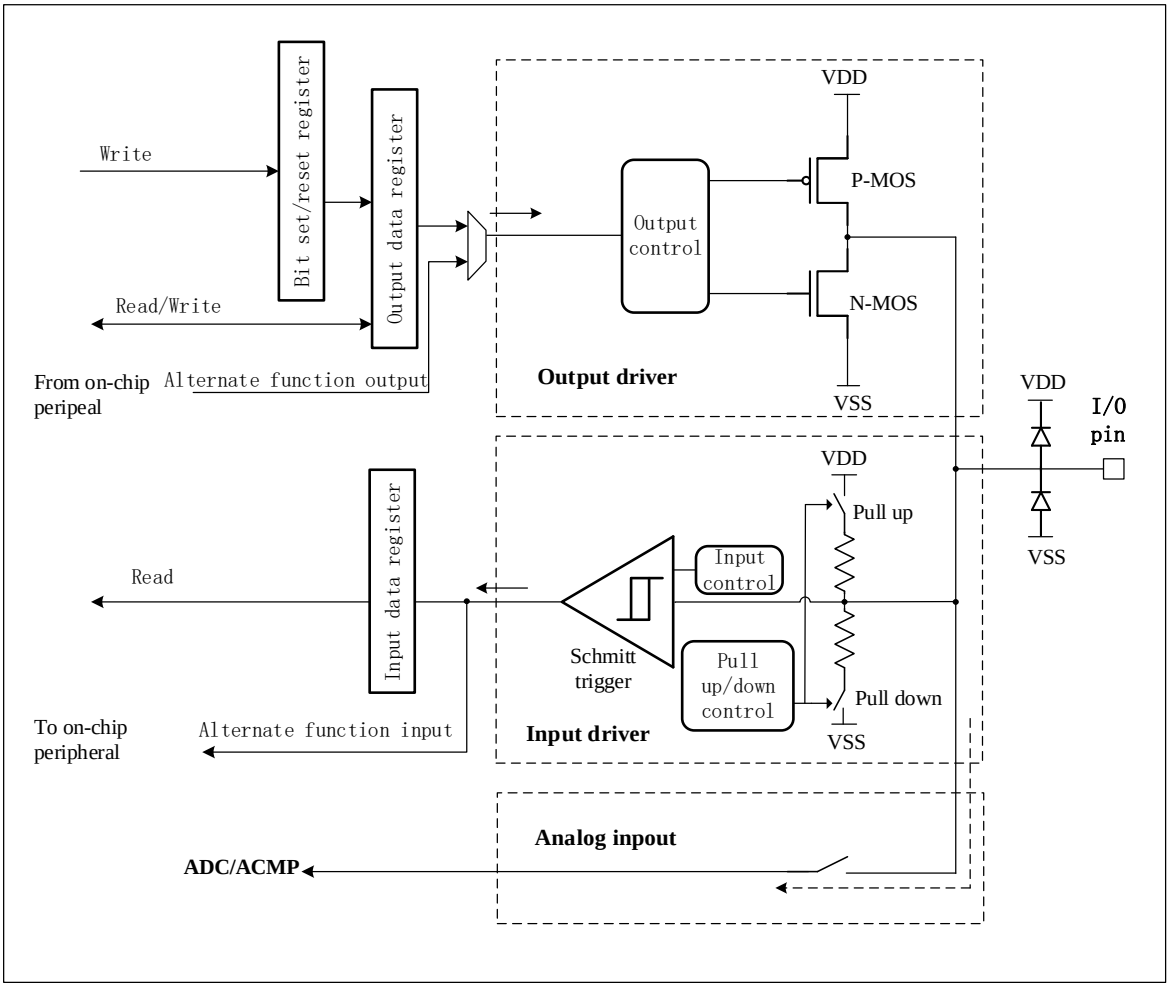

Figure 16-1 GPIO block diagram

16.4 Functional description

16.4.1 External interrupt

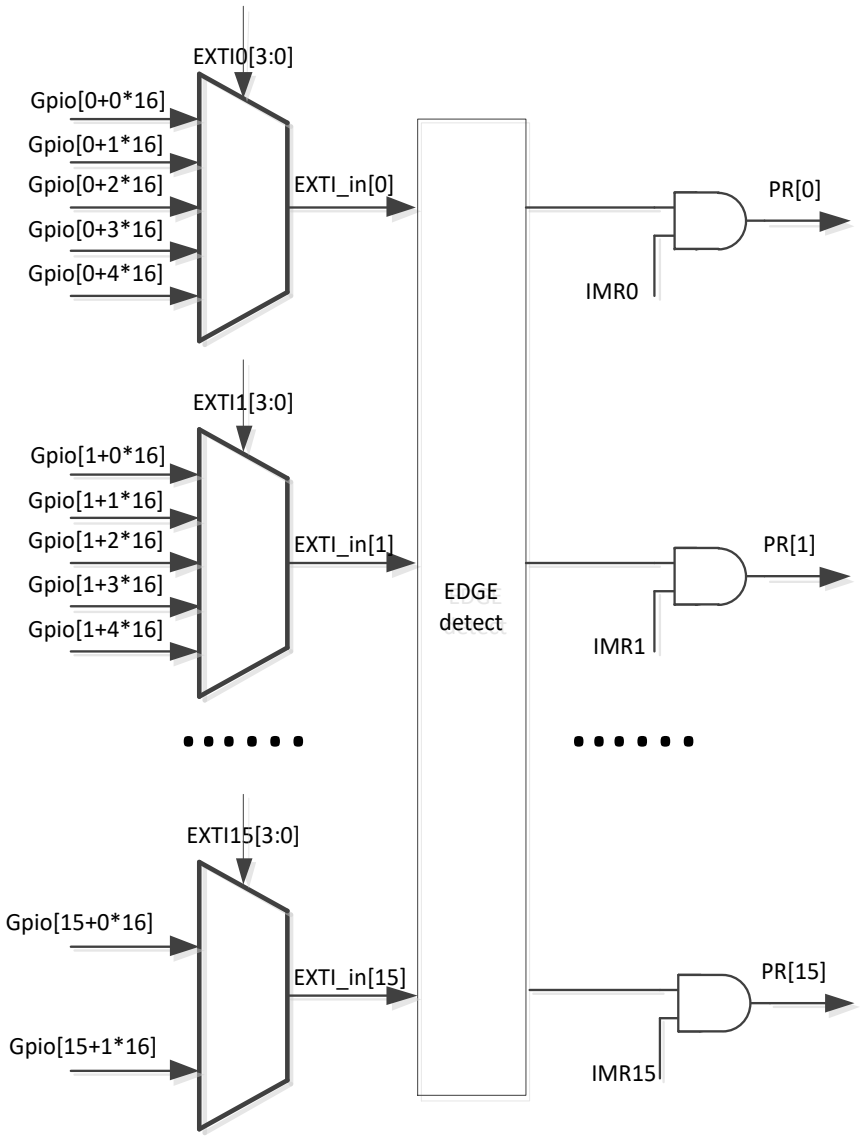

Figure 16-2 GPIO external interrupt

GPIO is divided into groups, and every 16 IOs form a group. Taking GPIO[y+x*16] as an example, 'y' represents the y-th PIN in a certain group of IOs, and x*16 represents the x-th group of GPIOs. For example, GPIO[1+0*16] represents GPIOA_PIN_1, GPIO[15+1*16] represents GPIOB_PIN_15.

The Correspondence relationship between external interrupt line and interrupt vector:

- When m≤2, EXTI_In[m] corresponds to the interrupt ventor EXTIm_IRQn
- When 3≤m≤8, EXTI_In[m] corresponds to the interrupt ventor EXTI3_8_IRQn
- When 9≤m≤15, EXTI_in[m] corresponds to the interrupt ventor EXTI9_15_IRQn

The corresponding relationship between GPIO external interrupts and ISR is shown below.

Table 16-1 Corresponding relationship between GPIO external interrupt and ISR

GPIO pin	Interrupt flag	ISR
PA0~PC0	EXTI0	EXTI0_IRQHandler
PA1~PC1	EXTI1	EXTI1_IRQHandler
PA2~PC2	EXTI2	EXTI2_IRQHandler
PA3~PC3	EXTI3	EXTI3_8_IRQHandler
PA4~PC4	EXTI4	
PA5~PC5	EXTI5	
PA6~PC6	EXTI6	
PA7~PC7	EXTI7	
PA8~PC8	EXTI8	
PA9~PC9	EXTI9	EXTI9_15_IRQHandler
PA10~PB10	EXTI10	
PA11~PB11	EXTI11	
PA12~PB12	EXTI12	
PA13~PB13	EXTI13	
PA14~PB14	EXTI14	
PA15~PB15	EXTI15	

16.4.2 Multi-Function

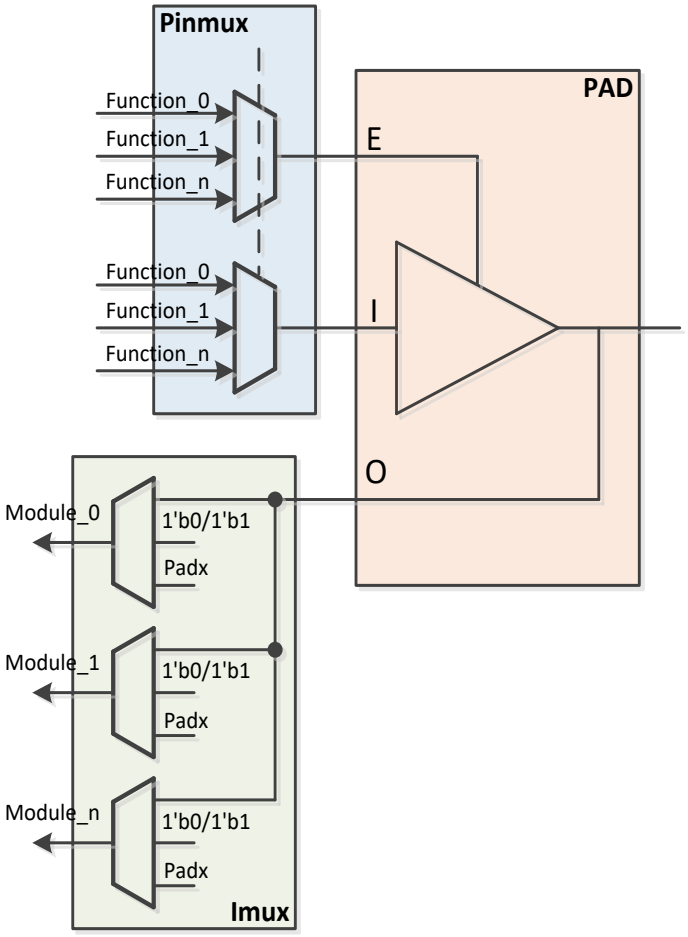

Figure 16-3 GPIO multi-function

Each IO have multi-functions, if we want to begin a peripheral communication, we should configure GPIO multi-function first. The corresponding multi-function of each GPIO is as below.

Table 16-2 GPIO multi-function

48 LQFP	32 HVQFN	20 TSSOP	Pin name	Function0	Function1	Function2	Function3	PINMUX	GPIO (NUM)
1			PB11	GPIO	PWM0_CH3	GPIO	SPI1_MOSI	PMUX2[23:21]	27
2			PB12	GPIO	PWM0_CH2	GPIO	SPI1_SCK	PMUX2[26:24]	28
3	1		PB0	GPIO	CAN_TX	PWM0_CH7	SPI1_MISO	PMUX1[20:18]	16
4	2		PB1	GPIO	CAN_RX	PWM0_CH6	SPI1_NSS	PMUX1[23:21]	17
5	3	4	VDD1	VDD1					
6			VDDA	VDDA					
7	4	5	VSS1	VSS1					
8	5	6	PA12	GPIO	I2C0_SCL	OSC_OUT[1]	PWM0_FLT0	PMUX1[8:6]	12
9	6	7	PA15	GPIO	I2C0_SDA	OSC_IN[1]	PWDT0_IN0	PMUX1[17:15]	15

48 LQFP	32 HVQFN	20 TSSOP	Pin name	Function0	Function1	Function2	Function3	PINMUX	GPIO (NUM)
10	7	8	PA0	GPIO	PWM0_CH1	UART0_RTS	I2C0_SCL	PMUX0[2:0]	0
11	8	9	PA1	GPIO	PWM0_CH0	UART0_CTS	I2C0_SDA	PMUX0[5:3]	1
12			PB13	GPIO	PWM0_CH7	GPIO	I2C1_SCL	PMUX2[29:27]	29
13	9		PB3	GPIO	PWM0_CH6	PWM1_CH7	SPI0_MOSI	PMUX1[29:27]	19
14	10	10	PA2	GPIO	PWM0_CH5	ADC_IN8	SPI0_MISO	PMUX0[8:6]	2
15	11	11	PA3	GPIO	PWM0_CH4	ADC_IN7	SPI0_SCK	PMUX0[11:9]	3
16	12	12	PA4	GPIO	PWM0_CH3	ADC_IN6/ ACMP_IN6	UART1_TX	PMUX0[14:12]	4
17	13	13	PA5	GPIO	PWM0_CH2	ADC_IN5/ ACMP_IN5	UART1_RX	PMUX0[17:15]	5
18	14	14	PA6	GPIO	BOOT[1]	GPIO	GPIO	PMUX0[20:18]	6
19			PB14	GPIO	PWM0_CH1	GPIO	SPI1_MOSI	PMUX3[2:0]	30
20			PB15	GPIO	PWM1_FLT0	ADC_IN11	SPI1_SCK	PMUX3[5:3]	31
21			PC0	GPIO	PWM1_CH3	ADC_IN10	SPI1_MISO	PMUX3[8:6]	32
22			PC1	GPIO	PWM1_CH2	ADC_IN9	SPI1_NSS	PMUX3[11:9]	33
23	15		PB4	GPIO	PWM1_CH1	ADC_IN8	SPI0_MISO	PMUX2[2:0]	20
24	16		PB5	GPIO	PWM1_CH0	ADC_IN7	SPI0_SCK	PMUX2[5:3]	21
25	17	15	PA7	GPIO	UART0_TX	ADC_IN4/ ACMP_IN4	SPI0_MOSI	PMUX0[23:21]	7
26	18	16	PA8	GPIO	UART0_RX	ADC_IN3/ ACMP_IN3	SPI0_NSS	PMUX0[26:24]	8
27			PC2	GPIO	UART1_TX	PWM0_FLT1	UART0_TX	PMUX3[14:12]	34
28			PC3	GPIO	UART1_RX	PWM1_FLT1	UART0_RX	PMUX3[17:15]	35
29	19	17	PA9	GPIO	PWM0_FLT0	ADC_IN2/ ACMP_IN2	RTC_CLKIN	PMUX0[29:27]	9
30	20		VSS2	VSS2					
31	21	18	VDD2	VDD2					
32			PC4	GPIO	PWM0_CH1	GPIO	I2C1_SDA	PMUX3[20:18]	36
33	22		PB6	GPIO	PWM1_CH6	PWM1_FLT0	CAN_STDBY	PMUX2[8:6]	22
34			PC5	GPIO	GPIO	PWDT0_IN1	SPI0_NSS	PMUX3[23:21]	37
35	23		PB7	GPIO	PWM1_CH3	ACMP_IN3	I2C0_SCL	PMUX2[11:9]	23
36	24		PB8	GPIO	PWM1_CH2	PWDT0_IN2	I2C0_SDA	PMUX2[14:12]	24
37	25	19	PA10	GPIO	PWM0_CH7	ADC_IN1/ ACMP_IN1	PWDT0_IN2	PMUX1[2:0]	10
38	26	20	PA11	GPIO	PWM0_CH6	ADC_IN0/ ACMP_IN0	PWDT0_IN1	PMUX1[5:3]	11
39			PC6	GPIO	UART1_TX	GPIO	PWDT1_IN2	PMUX3[26:24]	38
40			PC7	GPIO	UART1_RX	GPIO	PWDT1_IN1	PMUX3[29:27]	39
41			PC8	GPIO	PWM1_CH7	CAN_STDBY	PWDT1_IN0	PMUX4[2:0]	40
42			PC9	GPIO	PWM1_CH6	GPIO	ACMP_OUT	PMUX4[5:3]	41
43	27		PB9	GPIO	PWM1_CH5	I2C1_SCL	UART2_TX	PMUX2[17:15]	25

48 LQFP	32 HVQFN	20 TSSOP	Pin name	Function0	Function1	Function2	Function3	PINMUX	GPIO (NUM)
44	28		PB10	GPIO	PWM1_CH4	I2C1_SDA	UART2_RX	PMUX2[20:18]	26
45	29		PB2	GPIO	NMI_B[1]	PWM0_FLT0	PWDT0_IN0	PMUX1[26:24]	18
46	30	1	PA13	GPIO	SWD_CLK[1]		RTC_CLKOUT	PMUX1[11:9]	13
47	31	2	RESET_B	RESET_B					
48	32	3	PA14	GPIO	SWD_DIO[1]	ACMP_OUT	PWM1_CH0	PMUX1[14:12]	14

Note

1. **All the pins are default as function0 on the first time power on except some dedicated pins (marked as 1 in the above table). In the GPIO control register, except GPIO_PINMUX, others respectively represent PA, PB, PC, such as GPIO_CR, GPIO_IDR, etc., x=0,1,2, the lower 16 bits of each register respectively represent Px0~Px15.**
2. **Input/output configuration example: set PC1 as output mode, GPIO_CR[1] = 1.**
3. **Multi-function configuration (48LQFP as example): if we want to configure PB11 as PWM0_CH3, we should set PMUX2[23:21] to 1.**
4. **The suffix _B of NMI_B, RESET_B represents low level effective.**
5. **32 HVQFN package does not support I2C1 and SPI1, that is, PB0 and PB1 has no multi-function 3, PB9 and PB10 has no multi-function 2.**
6. **PA12 is configured as OSC_OUT function, switching to other multi-function is not supported. PA15 is configured as OSC_IN function, switching to other multi-function is not supported.**

16.5 Application note

16.5.1 External interrupt

The external interrupt/event controller consists of up to 16 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (event or interrupt) and the corresponding trigger event (rising edge, falling edge or both). Each line can also be masked independently. A pending register GPIO_PR maintains the status line of the interrupt requests.

16.5.2 Multi-Function

To optimize functionality in small packages, pins have several functions available via signal multiplexing. Pin_mux excel illustrates which of this device's signals are multiplexed on which external pin. The GPIO_PINMUX Register control which signal is present on the external pin. Refer to that register to find the detailed control operation of a specific multiplexed pin.

For details, please refer to the Multiplexing Function Selection Register GPIO_PINMUX and the chapter 16.4.2 Multi-Function.

16.5.3 Open-drain output

GPIO PIN is not separately configured as an open-drain output register, but it can be configured through a combination of registers to achieve the function of open-drain output.

Analog open-drain output high (external pull-up resistor is required):

1. Configure the corresponding BIT of the GPIO_CR register of the PIN to 0, input mode;
2. Configure the corresponding BIT of the GPIO_ODR register to 0;
3. Configure the corresponding BIT of the GPIO_PU and GPIO_PD register to 0, no pull-up and pull-down.

Analog open-drain output low:

1. Configure the corresponding BIT of the GPIO_CR register of the PIN to 1, output mode;
2. Configure the corresponding BIT of the GPIO_ODR register to 0;
3. Configure the corresponding BIT of the GPIO_PU and GPIO_PD register to 0, no pull-up and pull-down.

16.5.4 APB/AHB access

The GPIO module can be accessed through the APB bus or AHB to achieve the highest pin performance. Therefore, the registers of the GPIO module can be accessed through the APB address/AHB address, the specific addresses are listed in the following table.

Table 16-3 GPIO APB/AHB address

GPIO module	APB address	AHB address
GPIOA first address	0x40001000	0x20080000
GPIOB first address	0x40001030	0x20080030
GPIOC first address	0x40001060	0x20080060

Note

The offset of the GPIO register for the first address, whether it is APB or AHB, the offset address is the same. For example, The offset of the GPIO_IDR register is 0x04.Therefore, the address of the GPIO_IDR register accessed through APB is 0x40001004, and the address accessed through AHB is 0x20080004

16.5.5 GPIO function

During and just after reset, the GPIO functions are active and the I/O ports are configured in input mode, except RST and Arm debug interface. Before entering Stop mode, if the I/O is set to input state without pull-up or pull-down, you need to set a high or low stable level externally to avoid power leakage caused by unstable I/O level.

User can program GPIO_PINMUX Register to change I/O ports from other function to GPIO function.

When the pin is configured as output, the value written to the output data register is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the N-MOS is activated when 0 is output).The input data register (GPIO_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIO_PU and GPIO_PD register.

16.5.6 Programming guide

Firstly, after reset, all I/O ports are in GPIO input mode, except RST and Arm debug interface.

Secondly, Software can program GPIO_PINMUX to map/remap I/O function.

As I/O in GPIO function, SW can program external interrupt. When MCU in low power mode, external interrupt uses internal 32K LPOSC.

16.6 Register description

Table 16-4 GPIO register map

GPIOA base address: 0x20080000
GPIOB base address: 0x20080030
GPIOC base address: 0x20080060

Address	Name	Width (in bit)	Description
GPIOx base address + 0x00	GPIO_CR	32	Port configuration
GPIOx base address + 0x04	GPIO_IDR	32	Port input data
GPIOx base address + 0x08	GPIO_ODR	32	Port output data
GPIOx base address + 0x0C	GPIO_BSRR	32	Port set/reset
GPIOx base address + 0x10	GPIO_BRR	32	Port reset
GPIOx base address + 0x18	GPIO_PD	32	Pull-down enable
GPIOx base address + 0x1C	GPIO_PU	32	Pull-up enable
GPIOx base address + 0x20	GPIO_E4_E2	32	Driving Capability selection
GPIOA base address + 0x140 GPIOA base address + 0x144 GPIOA base address + 0x148 GPIOA base address + 0x14C GPIOA base address + 0x150	GPIO_PINMUX	32	Multi-function selection: total 5 registers
GPIOA base address + 0x160	GPIO_PR	32	External interrupt flag Pending
GPIOA base address + 0x164	GPIO_IMR	32	Interrupt mask
GPIOA base address + 0x168	GPIO_RTSR	32	Rising edge trigger event configuration
GPIOA base address + 0x16C	GPIO_FTSR	32	Falling edge trigger event configuration
GPIOA base address + 0x170 GPIOA base address + 0x174 GPIOA base address + 0x178 GPIOA base address + 0x17C	GPIO_EXTICR	32	external interrupt: total 4 registers

Note: in the above table, x=A, B, C.

16.6.1 GPIO_CR

Table 16-5 GPIO_CR register

GPIO_CR **Port configuration register** **Reset: 0x00000000**

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

Name	MODE(16*x+15)	MODE(16*x+14)	MODE(16*x+13)	MODE(16*x+12)	MODE(16*x+11)	MODE(16*x+10)	MODE(16*x+9)	MODE(16*x+8)	MODE(16*x+7)	MODE(16*x+6)	MODE(16*x+5)	MODE(16*x+4)	MODE(16*x+3)	MODE(16*x+2)	MODE(16*x+1)	MODE(16*x+0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[15: 0] MODE	**Mode(y):** Port y configuration bits These bits are written by software to configure the I/O direction mode. 0: Input (reset state) 1: output mode

16.6.2 GPIO_IDR

Table 16-6 GPIO_IDR register

GPIO_IDR **Port input data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IDR(16*x+15)	IDR(16*x+14)	IDR(16*x+13)	IDR(16*x+12)	IDR(16*x+11)	IDR(16*x+10)	IDR(16*x+9)	IDR(16*x+8)	IDR(16*x+7)	IDR(16*x+6)	IDR(16*x+5)	IDR(16*x+4)	IDR(16*x+3)	IDR(16*x+2)	IDR(16*x+1)	IDR(16*x+0)
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[15: 0] IDR	**IDR(y):** Port y input data These bits are read-only. They contain the input value of the corresponding I/O port.

16.6.3 GPIO_ODR

Table 16-7 GPIO_ODR register

GPIO_ODR **Port output data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ODR(16*x+15)	ODR(16*x+14)	ODR(16*x+13)	ODR(16*x+12)	ODR(16*x+11)	ODR(16*x+10)	ODR(16*x+9)	ODR(16*x+8)	ODR(16*x+7)	ODR(16*x+6)	ODR(16*x+5)	ODR(16*x+4)	ODR(16*x+3)	ODR(16*x+2)	ODR(16*x+1)	ODR(16*x+0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[15: 0] ODR	**ODR(y):** Port(y) output data These bits can be read and written by software. **Note:** For PIN set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A, B, C) and GPIOx_BRR register (x = A, B, C).

16.6.4 GPIO_BSRR

Table 16-8 GPIO_BSRR register

GPIO_BSRR **Port set/reset register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
Type	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BS(16*x+15)	BS(16*x+14)	BS(16*x+13)	BS(16*x+12)	BS(16*x+11)	BS(16*x+10)	BS(16*x+9)	BS(16*x+8)	BS(16*x+7)	BS(16*x+6)	BS(16*x+5)	BS(16*x+4)	BS(16*x+3)	BS(16*x+2)	BS(16*x+1)	BS(16*x+0)
Type	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[31:16] BR	**BR(y): Port(y) Reset bit y** 0: No action on the corresponding ODRy bit

Bits	Description
	1: Reset the corresponding ODRy bit These bits are write-only. A read to these bits returns the value 0x0000. This register only supports the APB access, the AHB access is not supported.
[15: 0] BS	**BS(y): Port(y) set bit y** 0: No action on the corresponding ODRy bit 1: Sets the corresponding ODRy bit These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000. **Note:** if BS and BR both configured, BR has higher priority.

16.6.5 GPIO_BRR

Table 16-9 GPIO_BRR register

GPIO_BRR **Port reset register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
Type	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[15: 0] BR	**BR(y):** Port(y) Reset bit y 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit These bits are write-only. A read to these bits returns the value 0x0000

16.6.6 GPIO_PD

Table 16-10 GPIO_PD register

GPIO_PD **Pull-down enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PD(16*x+15)	PD(16*x+14)	PD(16*x+13)	PD(16*x+12)	PD(16*x+11)	PD(16*x+10)	PD(16*x+9)	PD(16*x+8)	PD(16*x+7)	PD(16*x+6)	PD(16*x+5)	PD(16*x+4)	PD(16*x+3)	PD(16*x+2)	PD(16*x+1)	PD(16*x+0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0]	**PD (y):** Pull-down enable
[1]	
[2]	0: disable pull-down
.....	1: enable pull-down (typical value: 75 KΩ)
PD	
	These bits can be read and written by software.
	Note: Pull-up and Pull-down are not supported enable at the same time

16.6.7 GPIO_PU

Table 16-11 GPIO_PU register

GPIO_PU **Pull-down enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PU(16*x+15)	PU(16*x+14)	PU(16*x+13)	PU(16*x+12)	PU(16*x+11)	PU(16*x+10)	PU(16*x+9)	PU(16*x+8)	PU(16*x+7)	PU(16*x+6)	PU(16*x+5)	PU(16*x+4)	PU(16*x+3)	PU(16*x+2)	PU(16*x+1)	PU(16*x+0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0]	**PU (y):** Pull-up enable
[1]	
[2]	0: disable pull-up
.....	1: enable pull-up (typical value: 75 KΩ)
PU	
	These bits can be read and written by software.

Bits	Description
	Note: The default value of GPIOC PU register is 0x00000400, because bit10 is used for pin RESET_b, which is pulled by default. Pull-up and Pull-down are not supported enable at the same time.

16.6.8 GPIO_E4_E2

Table 16-12 GPIO_E4_E2 register

GPIO_E4_E2 **Driving Capability selection register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	E4_E2(16*x+15)		E4_E2(16*x+14)		E4_E2(16*x+13)		E4_E2(16*x+12)		E4_E2(16*x+11)		E4_E2(16*x+10)		E4_E2(16*x+9)		E4_E2(16*x+8)	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	E4_E2(16*x+7)		E4_E2(16*x+6)		E4_E2(16*x+5)		E4_E2(16*x+4)		E4_E2(16*x+3)		E4_E2(16*x+2)		E4_E2(16*x+1)		E4_E2(16*x+0)	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	

Bits	Description
[1: 0] [3: 2] [5: 4] …… E4_E2	**E4_E2 (y): Driving Capability selection** 00: 4mA 01: 8mA 10: 12mA 11: 16mA These bits can be read and written by software.

16.6.9 GPIO_PINMUX

Table 16-13 GPIO_PINMUX register

GPIO_PINMUX **Multi-function selection register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name			PINMUXx[29: 27]			PINMUXx[26: 24]			PINMUXx[23: 21]			PINMUXx[20: 18]			PINMUXx[17: 15]	
Type			RW			RW			RW			RW			RW	
Reset			0			0			0			0			0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		PINMUXx[14: 12]			PINMUXx[11: 9]			PINMUXx[8: 6]			PINMUXx[5: 3]			PINMUXx[2: 0]		
Type		RW			RW			RW			RW			RW		
Reset		0			0			0			0			0		

Bits	Description
[2: 0] [5: 3] [8: 6] …… PINMUXx	**PINMUXx (y):** Multi-function selection 000: function 0 001: function 1 010: function 2 011: function 3 These bits are written by software to configure alternate function I/Os Note: GPIO_PINMUX0 register default value is 0x00040000; GPIO_PINMUX1 register default value is 0x01011280; GPIO_PINMUX2 register default value is 0x00000000; GPIO_PINMUX3 register default value is 0x00000000; GPIO_PINMUX4 register default value is 0x00000000.

16.6.10 GPIO_PR

Table 16-14 GPIO_PR register

GPIO_PR **External interrupt flag Pending** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PR(15)	PR(14)	PR(13)	PR(12)	PR(11)	PR(10)	PR(9)	PR(8)	PR(7)	PR(6)	PR(5)	PR(4)	PR(3)	PR(2)	PR(1)	PR(0)
Type	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0] [1] [2] …… PR	**PR (y): External interrupt flag Pending bit** 0: No trigger request occurred 1: The selected trigger request occurred This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a '1' to the bit.

16.6.11 GPIO_IMR

Table 16-15 GPIO_IMR register

GPIO_IMR **Interrupt mask register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IMR(15)	IMR(14)	IMR(13)	IMR(12)	IMR(11)	IMR(10)	IMR(9)	IMR(8)	IMR(7)	IMR(6)	IMR(5)	IMR(4)	IMR(3)	IMR(2)	IMR(1)	IMR(0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0] [1] [2] …… IMR	**IMR (y): Interrupt mask y** 0: Interrupt request from Line y is masked 1: Interrupt request from Line y is not masked

16.6.12 GPIO_RTSR

Table 16-16 GPIO_RTSR register

GPIO_RTSR **Rising edge trigger event configuration** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTSR(15)	RTSR(14)	RTSR(13)	RTSR(12)	RTSR(11)	RTSR(10)	RTSR(9)	RTSR(8)	RTSR(7)	RTSR(6)	RTSR(5)	RTSR(4)	RTSR(3)	RTSR(2)	RTSR(1)	RTSR(0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0] [1] [2] …… RTSR	**RTSR (y): Rising edge trigger event configuration bit of line y** 0: Rising edge trigger disabled (for Event and Interrupt) for input line y 1: Rising edge trigger enabled (for Event and Interrupt) for input line y

16.6.13 GPIO_FTSR

Table 16-17 GPIO_FTSR register

GPIO_FTSR **Falling edge trigger event configuration register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FTSR(15)	FTSR(14)	FTSR(13)	FTSR(12)	FTSR(11)	FTSR(10)	FTSR(9)	FTSR(8)	FTSR(7)	FTSR(6)	FTSR(5)	FTSR(4)	FTSR(3)	FTSR(2)	FTSR(1)	FTSR(0)
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Description
[0] [1] [2] …… FTSR	**FTSR (y): Falling edge trigger event configuration bit of line y** 0: Falling edge trigger disabled (for Event and Interrupt) for input line y 1: Falling edge trigger enabled (for Event and Interrupt) for input line y

16.6.14 GPIO_EXTICR

Table 16-18 GPIO_EXTICR register

GPIO_EXTICR **external interrupt register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTI(4*x+3)				EXTI(4*x+2)				EXTI(4*x+1)				EXTI(4*x+0)			
Type	RW				RW				RW				RW			
Reset	0				0				0				0			

Bits	Description
[3: 0] [7: 4] [11: 8] …… EXTI	**EXTI (y): EXTI y configuration** 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin These bits are written by software to select the source input for the EXTI(y) external interrupt.

17 I2C

17.1 Introduction

I2C(Inter-IC) is a two-wire serial interface. The two signals are SCL and SDA. SCL is a clock signal that is driven by the master. SDA is bi-directional data signal that can be driven by either the master or the slave. This generic interface supports the master and slave role both, and conforms to the I2C specification.

17.2 Features

- Supports master and slave mode operation.
- Supports I2C standard mode 100kHz, fast mode 400kHz and fast mode plus mode 1MHz.
- Supports 7-bit address range.
- Slave 10-bit address extension.
- Supports slave stretch.
- Slave supports low power mode wakeup.
- Slave supports monitor function.
- Supports multi-master arbitration.
- Master switch to slave mode automatically while arbitration lost.
- Programmable input glitch filter.
- Bus START/STOP detection.
- Software-controlled acknowledge bit.
- Supports TX and RX DMA operation.
- Address match interrupt.
- Interrupt-driven byte-by-byte data transfer.
- No ack interrupt.
- Transmit/Receive overflow interrupt.

17.3 Block diagram

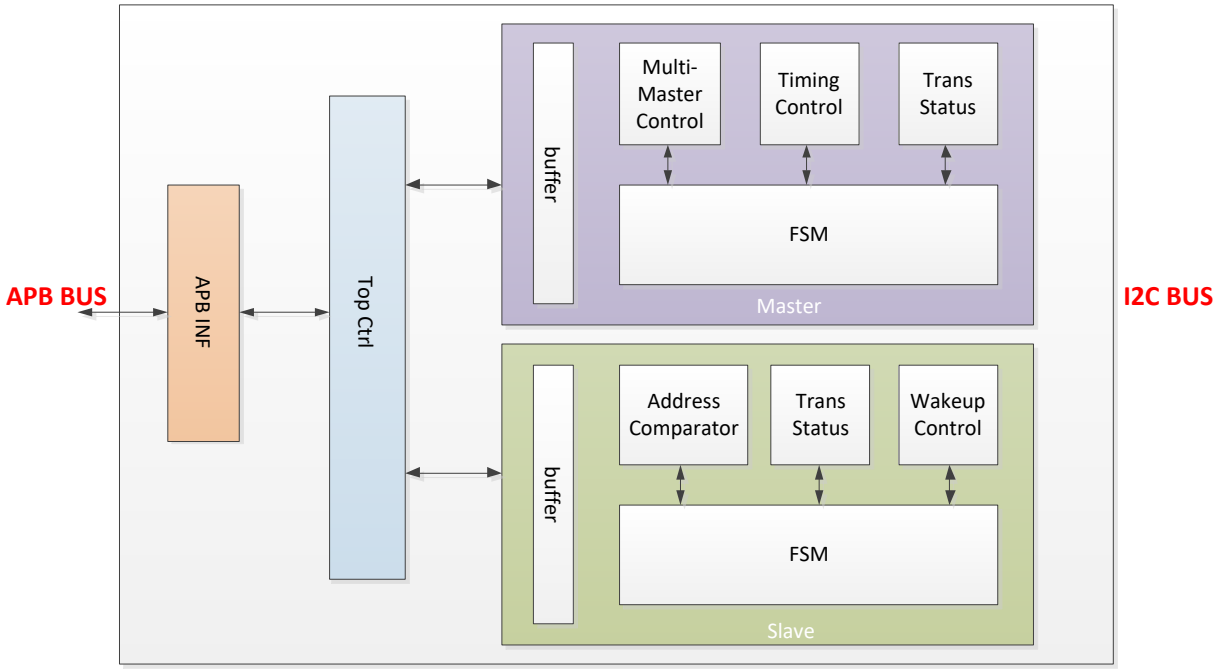

Figure 17-1 I2C block diagram

17.3.1 I2C signal

All transactions begin with a START (S) and terminated by a STOP (P). A high to low transition on the SDA line while SCL is high defines a START condition. A low to high transition on the SDA line while SCL is high defines a STOP condition (see Figure 17-2).

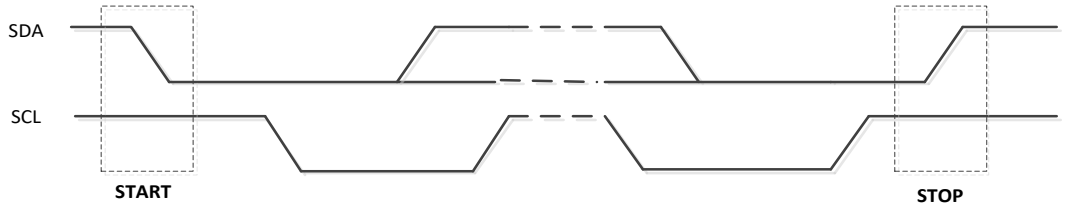

Figure 17-2 START and STOP conditions

Each frame of data consists of 9 bits, 8 bits of data (MSB first) and 1 bit of response signal, and the transmission times are not limited. (see Figure 17-3).

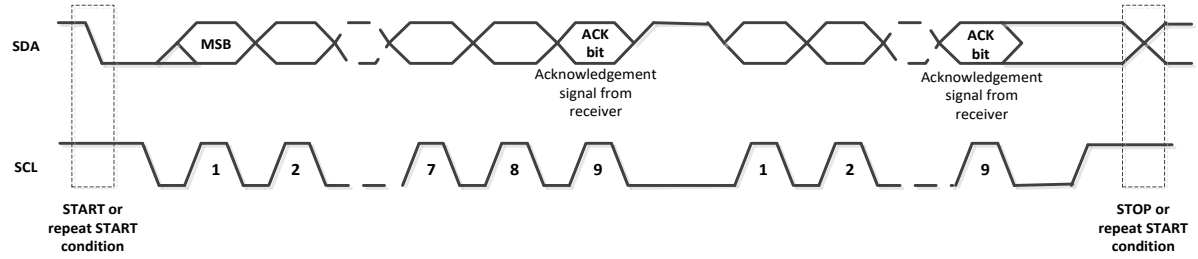

Figure 17-3 Data transmission format

The first data after the start signal is the address byte, and the data that continues to be transmitted after the address byte is the data byte.

In the 7-bit address mode, the I2C protocol stipulates that the high 7 bits of the address byte are the address of the slave, the lowest bit is 0, which means to write to the slave, and the highest bit is 1, which means to read from the slave.

In the 10-bit address mode, the address is composed of two bytes.

When writing to the slave, the I2C protocol stipulates that: the first byte is sent to 11110xx0, the high five bits are fixed to 11110, Bit2, Bit1 is the high two bits in the 10-bit address, Bit0 is the direction bit, and the value is 0, indicating that the direction is write; The second byte is the lower 8 bits of the 10-bit address.

When reading from the slave, I2C stipulates that the address to be written (two bytes) should be sent first, and then the address to be read (the address to be read only needs to send one byte).

The specific process is as follows: for the address to be written first, the first byte is sent to 11110xx0, the high five bits are fixed to 11110, Bit2 and Bit1 are the high two bits of the 10-bit address, Bit0 is the direction bit, and the value is 0, indicating that the direction bit is written, and the second byte is the low eight bits of the 10-bit address; For the address to be written again, 11110xx1 is sent. Bit2 and Bit1 are the upper two bits of the 10-bit address. Bit0 is the direction bit, and the value is 1, indicating that the direction bit is read.

17.3.2 Baud rate

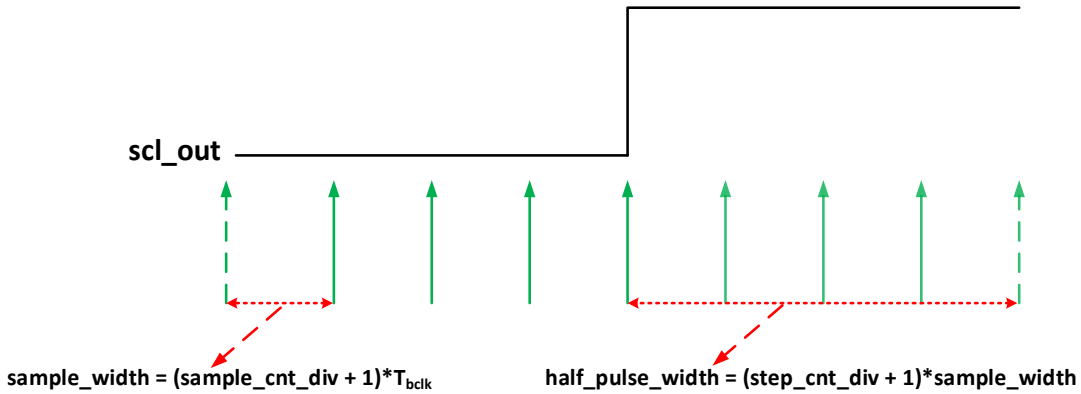

Figure 17-4 Baud rate generation

Baud rate: $f_{SCL}=f_{bclk}/(((SAMPLE\_CNT\_DIV+1) * (STEP\_CNT\_DIV+1))*2)$
f_{bclk} is the APB bus clock frequency

17.3.3 Data flow

For transmitter, data is written to an 8-bit TX buffer, then load to TX shift refer to baud rate control logic. The output data is most significant bit first. The transmitter sample the acknowledge bit every 9th SCL per byte.

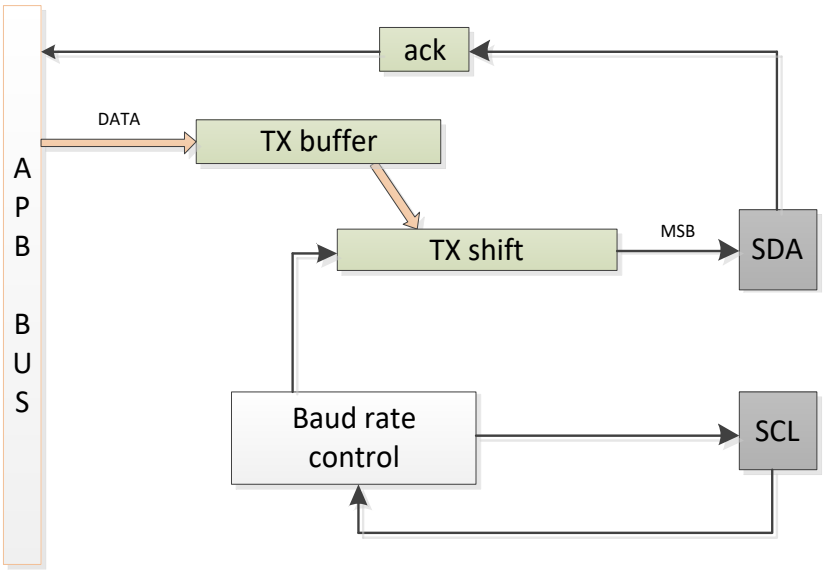

Figure 17-5 Data flow of transmitter

For receiver, RX shift sample and shift in the SDA line input data. Data received store into an 8-bit RX buffer after every 8th SCL per byte with the most significant bit first. The acknowledge bit is put on the SDA line at 9th SCL pulse.

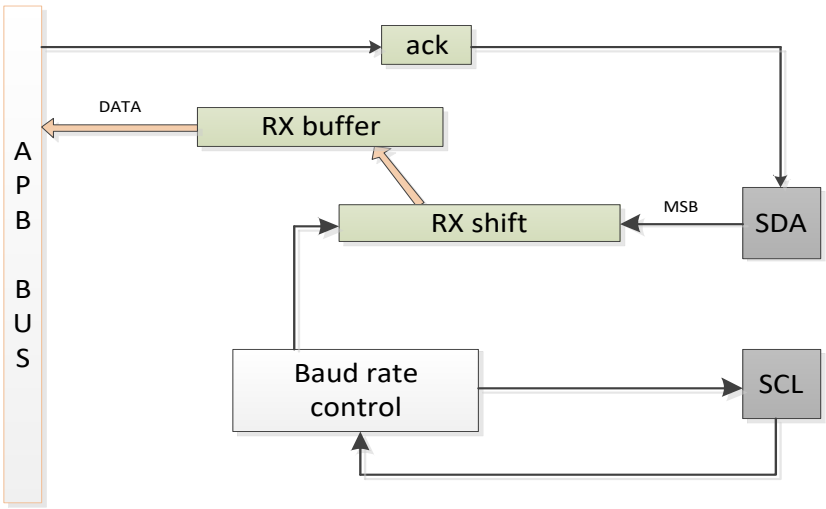

Figure 17-6 Data flow of Receiver

17.4 Functional description

The I2C module supports standard, fast, and high-speed communication modes. I2C peripherals can only be used as a master or slave mode at the same time. Once configured as slave mode, it will remain in slave mode unless the module is reset.

17.4.1 Master mode

In master mode, every time the master sends a byte of data, the BND flag is set to generate an interrupt request.

The master can send repeated start signals, and re-address the slave and write or read data without sending the STOP signal, as shown in Figure 17-7.

The host supports clock synchronization. When multi-master/slave communication or slave stretch, clock synchronization can synchronize with the clock of other master slave: when the output of SCL of the host is high, high-level time count will not be started, but the count will start after the SCL of the bus is pulled up by all the hosts. In this way, the clock synchronization between multiple hosts is realized. At this time, the actual SCL clock frequency will be slow, because the SCL pull-up is charged by the external pull-up resistance, and the frequency depends on the value of the external bus capacitance and pull-up resistance.

Support master arbitration. In the case of multi-master communication, the I2C module compares the data transmitted on the bus bit by bit.

When the master sends a 1 and the bus data is detected to be 0, the arbitration is lost, and the arbitration lost flag is set to generate an interrupt request and switch itself to the slave mode, because other masters may be addressing themselves at this time. The precondition of this function is that the clock synchronization function is turned on first.

In DMA mode, when transmitting or receiving, the DMA enable signal will mask the generation of Bnd interrupt. Specifically: during DMA transmission, the Bnd flag will not be set and Bnd interrupt will not be generated. When the last byte is sent, after the last byte is transmitted, the Bnd flag will be set and DMA will be turned off to generate Bnd interrupt.

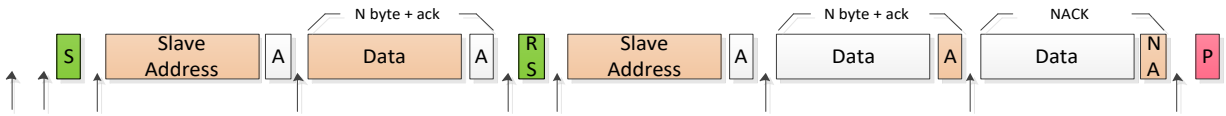

Figure 17-7 Master combined mode

17.4.2 Slave mode

In the slave mode, the slave supports four address matching modes: 7-bit address, 10-bit address, 7-bit range address, and broadcast address.

After the address of the slave is matched in the above four modes, the SAMF flag is set and the SRW flag is set to 0/1 according to the write/read signal of the master. Only after the address

matches, subsequently once the master sends each byte, the BND flag is set. If the master sends a NACK signal while reading, the BND flag is no longer set.

The slave has stretch function. For example, when the slave receives a byte of data but the software does not read it, if this function is enabled, the slave will actively pull down the SCL bus to prevent the master from sending the data clock signal of the next frame.

The slave has four status flags:

- **Transmit buffer empty flag(TXEF):** when the slave sending register is not loaded with sending data or one byte sending is completed, this flag is set and an interrupt request can be generated.
- **Transmit buffer overflow flag(TXUF):** when the slave has sent one byte of data, but has not written data of the next byte. And at this time, after the master has sent a byte of clock signal again, this flag is set and an interrupt request can be generated.
- **Receive buffer full flag(RXFF):** when the slave data register receives a byte of data, this flag is set and an interrupt request can be generated.
- **Receive buffer overview flag(RXOF):** After the slave receives a byte of data, if the software does not read the data in time, and the master sends a byte of clock signal again at this time, this flag is set and an interrupt request can be generated.

17.4.3 Interrupt request

I2C total has 10 interrupts.

Table 17-1 I2C Interrupt summary

Interrupt	Flag	Local enable	Global enable
One Byte Transfer End	BND		IICIE
Slave Address Match	SAMF		IICIE
Arbitration Lost	ARBLOST		IICIE
Bus START Detected	START	SSIE	IICIE
Bus STOP Detected	STOP	SSIE	IICIE
RX Buffer Overflow	RXOF	RXOFIE	IICIE
TX Buffer Overflow	TXUF	TXUFIE	IICIE
RX Buffer Full	RXFF	RXFIE	IICIE
TX Buffer Empty	TXEF	TXEIE	IICIE
Ack Get	RACK	NACKIE	IICIE

DMARXEN=1 or DMATXEN=1, BDN, RXFF, and TXEF will not generate interrupt even enable corresponding enable bit.

17.4.4 DMA operation

This I2C module supports byte-by-byte DMA TX/RX requests transmission. DMA requests are generated only for data transfer. DMA valid for data transfer phase only, and

START/STOP/RESTART signal triggered by software still. The DMA operation is different between transmitter and receiver.

17.4.4.1 Master transmitter

In master DMA transmission mode, after the address matches, DMA will automatically carry the data in memory to I2C data register. The ACK signal is detected after each byte of data is transmitted. When ACK, DMA request is generated to carry the next byte of data. When NACK is detected, DMA will not carry out subsequent transmission. If NACKIE = 1, interrupt request will be generated. After the transmission of the last byte is completed, the BND flag will be set. If DMA remains enabled, BND interrupt will not be generated. After DMA disable, BND interrupt can be generated again.

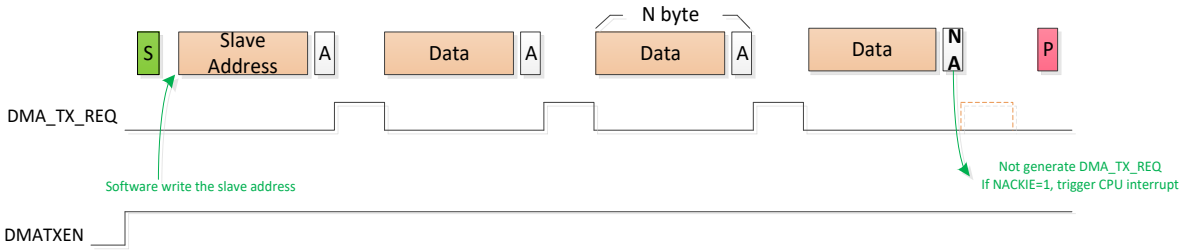

Figure 17-8 Master transmitter case 1

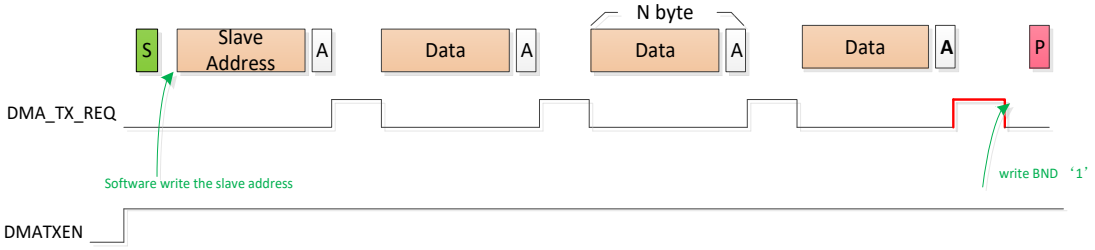

Figure 17-9 Master transmitter case 2

17.4.4.2 Master receiver

In the Master DMA receive mode, **after the address matches, the BND flag must be manually cleared**, and then the data transmission direction must be switched to read, and the idle read operation will be performed to generate a clock signal. After one byte is transferred, the BND flag will be set to request DMA to transfer data to memory. DMA will automatically return NACK at the last byte according to the set transfer length.

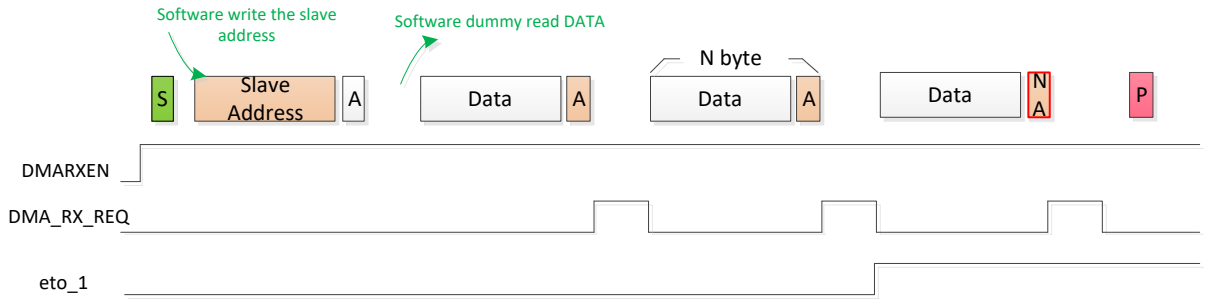

Figure 17-10 Master receiver

17.4.4.3 Slave transmitter

In the slave DMA transmission mode, if the address matches and it is a read signal, the DMA is triggered to carry the data to the data register, and detect the ACK signal of each subsequent byte. When receiving the ACK signal, DMA handling request will be generated. When receiving the NACK signal, DMA will stop handling. If NACKIE = 1, interrupt request will be generated. After the transmission of the last byte is completed, the BND flag is set. If DMA remains enabled, BND interrupt will not be generated. After DMA disable, BND interrupt request can be generated again.

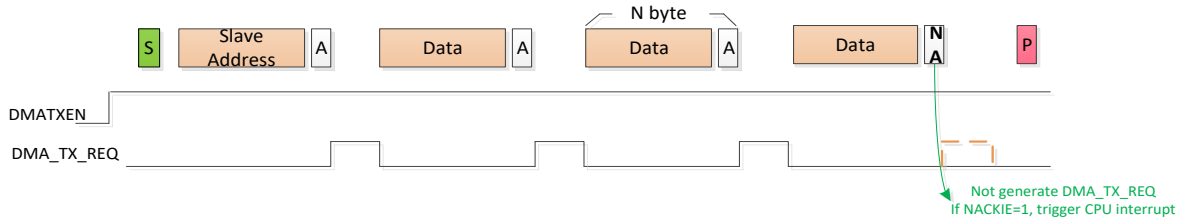

Figure 17-11 Slave transmitter

17.4.4.4 Slave receiver

In the Slave DMA receiving mode, **after the address matches, the address match flag must be manually cleared,** and then each time a byte of data is received, the BND flag generated will trigger the DMA to transfer the data to the memory. DMA will automatically return NACK at the last byte according to the set transfer length.

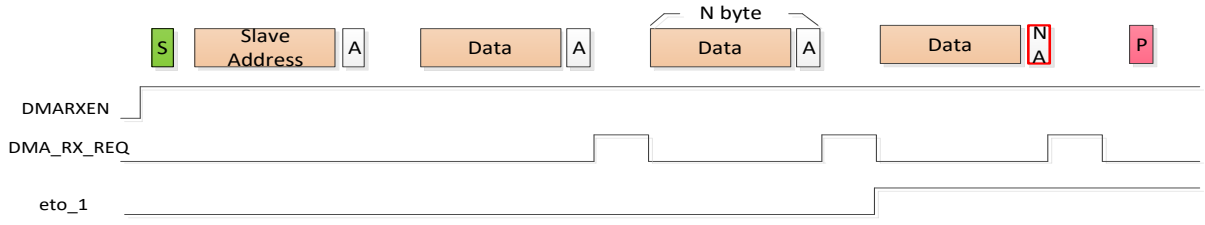

Figure 17-12 Slave receiver

17.4.5 Slave low power wakeup

If the wake-up function is enabled in the low-power mode of the MCU, when the address matches, an ACK low signal will be generated at the 9th SCL, and a wake-up signal will be generated to wake up the MCU. The data followed by address byte won't be ACKed, I2C need to be initialed or received a new start signal to handle data.

17.5 Application note

17.5.1 Data transmission

Write STARTSTOP [START] as '1' to send a START signal to the I2C bus. After the transmission, write STARTSTOP [STOP] as '1' to send a STOP signal to the I2C bus.

When the master sends, the TX control bit is set to 1. After writing the data register, the baud rate controller automatically starts to transmit data to the I2C bus. After the one-byte transmission is completed, the BND flag is set, indicating that data can be written again, write data or write 1 to the BND bit to clear the BND flag.

When the master receives, the TX control bit is set to 0, and the read operation of the data register will trigger the baud rate controller to automatically send a clock signal to the I2C bus, and the received data is loaded into the data register according to the RACK control response. The BND flag is set. Read the data register or write 1 to the BND bit to clear the BND flag.

When the slave sends, firstly write the data to be sent into the data register. After receiving the entire clock signal read by the master, the data is sent out, the TXEF flag is set, and the BND flag is set. The write operation to the data register can clear the TXEF and BND flags, or write 1 to the BND bit to clear the BND flag also.

When the slave receives, after the master sends one byte of entire data, the data is loaded into the slave data register, the RXFF flag is set, and the BND flag is set. Reading the data register can clear the RXFF and BND flags, or write 1 to the BND bit 1, the BND flag can also be cleared.

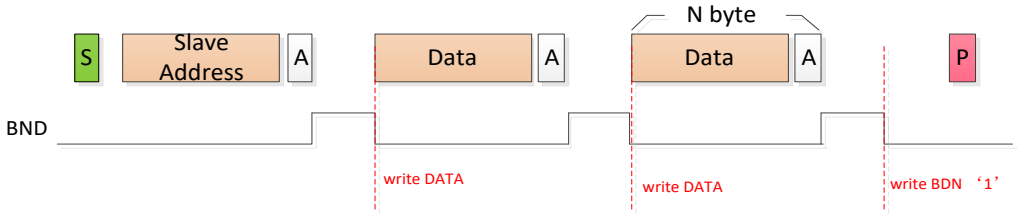

Figure 17-13 BND sequence of master write slave mode

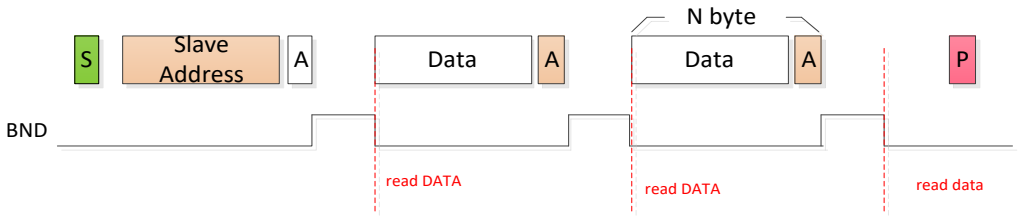

Figure 17-14 BND sequence of master read slave mode

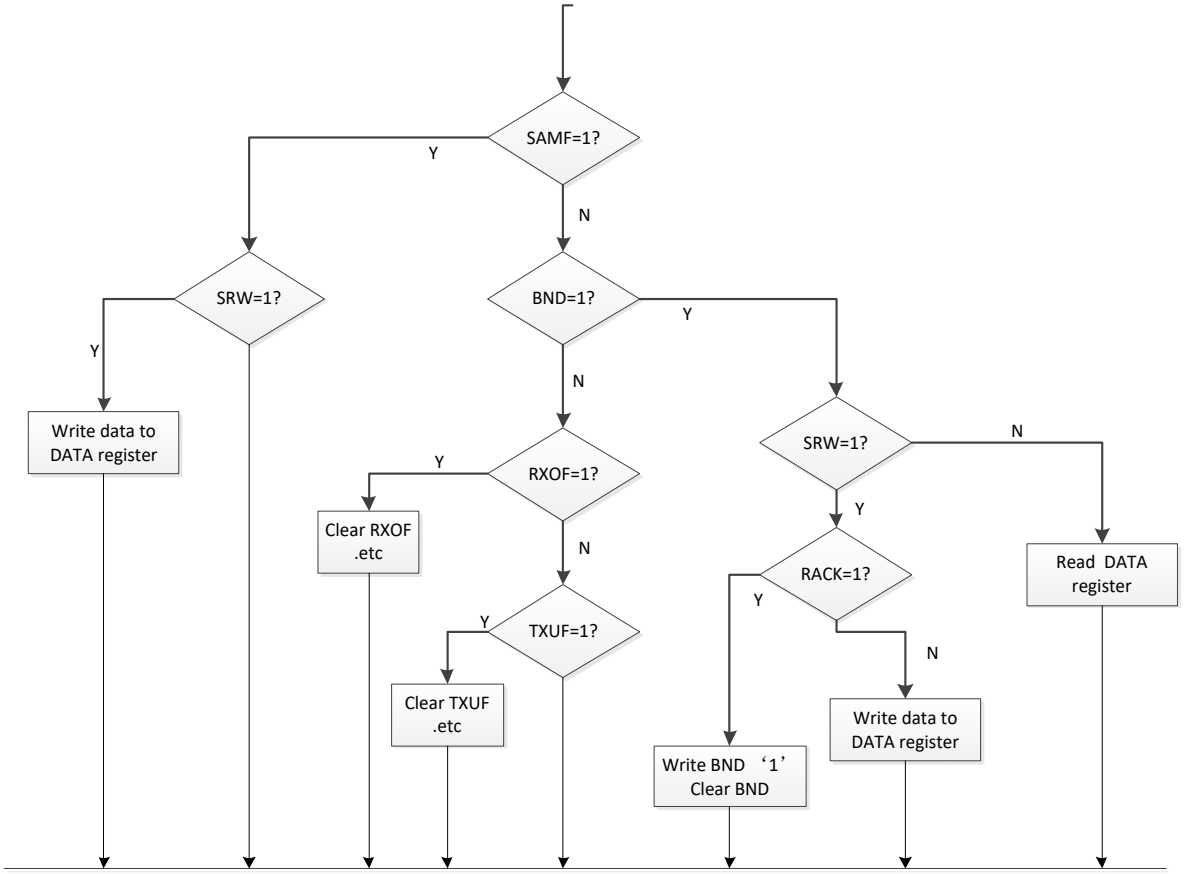

Figure 17-15 Typical I2C slave interrupt routine

17.5.2 ACK control

The I2C module controls the response signal through software.

When the master receives data, the value of the TACK bit determines the response value on the 9th clock signal after the next byte is read.

When the slave receives data, the value of the TACK bit determines the response value on the 9th clock signal after the next byte is read.

17.6 Register description

Table 17-2 I2C register mapping

I2C1 base address:0x4000e000
I2C2 base address: 0x4000f000

Address	Name	Width (in bit)	Description
I2Cx base address + 0x00	I2C_ADDR0	32	Address register 0
I2Cx base address + 0x04	I2C_ADDR1	32	Address register 1
I2Cx base address + 0x08	I2C_SAMPLE_CNT	32	Baud rate configuration register 0
I2Cx base address + 0x0C	I2C_STEP_CNT	32	Baud rate configuration register 1
I2Cx base address + 0x10	I2C_CTRL0	32	Control register 0
I2Cx base address + 0x14	I2C_CTRL1	32	Control register 1
I2Cx base address + 0x18	I2C_CTRL2	32	Control register 2
I2Cx base address + 0x1C	I2C_CTRL3	32	Control register 3
I2Cx base address + 0x20	I2C_STATUS0	32	Status register 0
I2Cx base address + 0x24	I2C_STATUS1	32	Status register 1
I2Cx base address + 0x28	I2C_DGLCFG	32	Deglitch configuration register
I2Cx base address + 0x2C	I2C_DATA	32	Data port register
I2Cx base address + 0x30	I2C_STARTSTOP	32	Master START STOP signal control register

Note: in the above table, x=1~2.

17.6.1 I2C_ADDR0

Table 17-3 I2C_ADDR0 register

I2C_ADDR0 **Address register 0** **Reset: 0x000000FE**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		AD [6: 0]							
Type		RW							
Reset		1							

Bits	Description
7: 1 AD[6: 0]	**Specifies the 7Bit address when I2C is slave mode** For the 7-bit address scheme and the lower seven bits in the 10-bit address scheme.

17.6.2 I2C_ADDR1

Table 17-4 I2C_ADDR1 register

I2C_ADDR1 **Address register 1** **Reset: 0x000007F7**

Bit	31: 13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		RMEN		RAD								AD[9: 7]		
Type		RW		RW								RW		
Reset		0		1								1		

Bits	Description
12 RMEN	**Enable range address in slave mode** 1: enable 0: disable Slave range address enable bit
10: 4 RAD	**Range address in slave mode** 7-bit slave range address value, when the range address is enabled, if the address received by the slave is larger than AD[6:0] and less than or equal to RAD[6:0], salve will get matched.
2: 0 AD [9: 7]	**Specifies the 10-bit address when I2Cx is slave mode** Contains the upper threes bits of the address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set.

17.6.3 I2C_SAMPLE_CNT

Table 17-5 I2C_SAMPLE_CNT register

I2C_SAMPLE_CNT **Baud Rate Config Register 0** **Reset: 0x00000004**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		SAMPLE_CNT							
Type		RW							
Reset		0							

Bits	Description
7: 0 SAMPLE_CNT	**This adjusts the width of each sample.** sample width = (SAMPLE_CNT +1) *T_{bclk}

17.6.4 I2C_STEP_CNT

Table 17-6 I2C_SAMPLE_CNT register

I2C_STEP_CNT **Baud Rate Configuration Register 1** **Reset: 0x00000004**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		STEP_CNT							
Type		RW							
Reset		0	0	0	0	0	1	0	0

Bits	Description
7: 0 STEP_CNT	**Specifies the number of samples per half pulse width, i.e. each high or low pulse.** half_baudrate width= (STEP_CNT +1) * sample width **Note:** minimum of STEP_CNT is 3 propose.

17.6.5 I2C_CTRL0

Table 17-7 I2C_CRTL0 register

I2C_CTRL0 **CTRL0** **Reset: 0x00000000**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		IICEN	IICIE	MSTR	TX	TACK	WUEN		
Type		RW	RW	RW	RW	RW	RW		
Reset		0	0	0	0	0	0		

Bits	Description
7 IICEN	**I2C module enable** 1: enable 0: disable
6 IICIE	**I2C interrupt enable** 1: enable the IIC module interrupt 0: disable
5 MSTR	**I2C operation mode select** 1: master 0: slave **Note:** if ARB_LOST is set, this bit auto clear.
4 TX	**Transmission direction select**

Bits	Description
	1: transmit(TX) 0: receive(RX) It is mainly used for reading in host mode: TX = 0, reading data register will trigger reading clock; TX = 1, reading data register will not trigger reading clock.
3 TACK	**Acknowledge control** 1: NACK will sent to the bus on the following(next) receiving byte 0: ACK will sent to the bus on the following(next) receiving byte Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers.
2 WUEN	**Wakeup function enable** 1: enable the wakeup function in low power mode. 0: disable the wakeup function. The I2C slave mode can wake the MCU from low power mode which no peripheral bus running when slave address matching occurs. **Note:** when a 7-bit address, 2 byte of 10-bit address(if ADEXT=1), or general call address(GCAEN=1) match occurs when I2C module is in slave mode, the I2C module will generates a request to wake up MCU from low power mode.

17.6.6 I2C_CTRL1

Table 17-8 I2C_CRTL1 register

I2C_CTRL1 **CTRL1** **Reset: 0x00000000**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		GCAEN	ADEXT		SYNCEN	ARBEN			STREN
Type		RW	RW		RW	RW			RW
Reset		0	0		0	0			0

Bits	Description
7 GCAEN	**Slave General Call Enable** 1: enable 0: disable
6 ADEXT	**Slave Address Extension** 1: 10-bit address scheme 0: 7-bit address scheme
4 SYNCEN	**Master SCL Sync Enable**

Bits	Description
	1: enable 0: disable Enables the SCL synchronization function of master
3 ARBEN	**Master Arbitration Enable** 1: enable 0: disable Enables the master's arbitration function. **Note:** If I2C used in multi-master system, multi-master function must set both SYNC_EN and ARB_EN. If I2C used in single master system, and slave have SCL stretch function, then can set SYNC_EN only.
0 STREN	**Slave SCL stretch enable** 1: enable 0: disable Enable this bit, slave hardware will stretch SCL low after 9th SCL falling per byte. **Note:** after SAMF=1, if SRW=1, RXFF=1 will cause SCL stretch; otherwise SRW=0, TXEF=1 will cause SCL stretch; so when enable STR_EN and slave is transmitter(TX), after 9th SCL falling per byte (include address byte) slave will stretch SCL, until write DATA register; similarly, when slave is receiver(RX), after 9th SCL falling per byte (without address byte) , slave will stretch SCL, until read DATA register. In slave mode, when enable GCA_EN or MNTEN, slave cannot stretch SCL.

17.6.7 I2C_CTRL2

Table 17-9 I2C_CRTL2 register

I2C_CTRL2 **CTRL2** **Reset: 0x00000000**

Bit	31: 8	7	6	5	4	3:2	1	0
Name		RXOFIE	TXUFIE	RXFIE	TXEMIE		NACKIE	MNTEN
Type		RW	RW	RW	RW		RW	RW
Reset		0	0	0	0		0	0

Bits	Description
7 RXOFIE	**Slave RX Buffer Overflow Error Interrupt Enable** 1: enable 0: disable
6 TXUFIE	**Slave TX Buffer Overflow Error Interrupt Enable** 1: enable 0: disable
5	**Slave RX Buffer Full Interrupt Enable**

Bits	Description
RXFIE	
	1: enable
	0: disable
4 TXEMIE	**Slave TX Buffer Empty Interrupt Enable**
	1: enable
	0: disable
1 NACKIE	**NACK Get Interrupt Enable**
	1: enable
	0: disable
0 MNTEN	**Slave Monitor Function Enable**
	1: enable
	0: disable

17.6.8 I2C_CTRL3

Table 17-10 I2C_CRTL3 register

I2C_CTRL3 **CTRL3** **Reset: 0x00000000**

Bit	31: 2	1	0
Name		DMA RXEN	DMA TXEN
Type		RW	RW
Reset		0	0

Bits	Description
1 DMARXEN	**DMA RX Enable**
	1: enable
	0: disable
0 DMATXEN	**DMA TX Enable**
	1: enable
	0: disable

17.6.9 I2C_STATUS0

Table 17-11 I2C_STATUS0 register

I2C_STATUS0 **STATUS0** **Reset: 0x00000008**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		BND	SAMF	BUSY	ARBLOST	READY	SRW		RACK
Type		W1C	W1C	R	W1C	R	R		W1C
Reset		0	0	0	0	**1**	0		0

Bits	Description
7 BND	**Byte End Flag** 1: one byte transfer finish (include ACK bit, total 9 SCL) 0: transfer in progress, one byte transfer not finish After reset, BND is '0'. BND is set only during data transmission period which between START and STOP signal on the bus. BND will set after per 9th SCL falling edge. In master mode, when sending data, the software writes data register to clear this bit and send data; When reading data, one byte transmission is completed, the software will clear this bit when reading data register, and send out the next byte data clock. In slave mode, SAMF is set after address matching. At this time, Bnd flag will not be set. Bnd is set every time data transmission after address matching. Specifically: after address matching, the master writes data and the slave sets Bnd. When the slave takes the initiative to NACK, if the master continues to write data, the slave does not set Bnd; The master reads data and the slave sets Bnd. If NACK is received from the master, the master continues to read data and the slave does not set Bnd. In master / slave DMA transmit or receive mode, Bnd interrupt is masked. When the last byte is sent, DMA transmission is completed, but I2C will continue to transmit the last byte. You can poll Bnd to determine whether the last byte is sent or not, or disable DMA and wait for Bnd interrupt to be generated. **Note:** when I2C is slave and MNTEN=1, when BND is '1', then read DATA register will clear this bit. write '1' can clear this bit too.
6 SAMF	**Slave Address Match Flag** 1: address match 0: address not match **Note:** this bit set by one of the following conditions. a. 7-bit address, address match b. 10-bit address, both 1st byte and 2nd byte match. And set after 2nd byte match. c. Universal broadcast address matching. d. general call match

Bits	Description
	Note: write '1' clear this bit.
5 BUSY	**Bus Busy** 1: bus is busy. 0: bus is idle. Indicates the status of the bus, and valid for both slave and master mode. This bit is set when a START signal is detected and cleared when a STOP signal is detected on the bus by hardware.
4 ARBLOST	**Arbitration Lost Flag** 1: arbitration lost 0: not lost **Note:** Write '1' clear. When arbitration lost , I2C master will switch to slave mode, and MSTR is cleared by hardware.
3 READY	**Internal Hardware Core Is Ready for New Command** 1: internal hardware is ready for software's new command 0: internal hardware is not ready This bit indicates the internal hardware states, and only valid for master mode. **Note:** this bit is valid for master, and maybe used when master generate a START/STOP signal. When I2C module is master mode, write START_STOP[0] will generate a START signal on the bus, then DGL_CFG[4] is set, and this case must wait READY bit is 1, software can write DATA register to send address byte. Similar with START signal, write START_STOP[1] will generate a STOP signal on the bus, then DGL_CFG[6] is set. This case must wait READY bit is 1, software can write START_STOP[0] , initial a next transmission.
2 SRW	**Slave Read/Write Direction** 1: slave is transmitter(TX), master read from slave 0: slave is receiver(RX), master write to slave
0 RACK	**Acknowledge Received** 1: No acknowledge signal detected 0: Acknowledge signal was received after one byte of data This field valid for transmitter(TX). **Note:** if NACKIE=1, RACK=1 will set interrupt, write 1 clear.

17.6.10 I2C_STATUS1

Table 17-12 I2C_STATUS1 register

I2C_STATUS1 **STATUS1** **Reset: 0x00000081**

Bit	31: 4	3	2	1	0
Name		RXOF	TXUF	RXFF	TXEF
Type		W1C	W1C	W1C	W1C
Reset		0	0	0	1

Bits	Description
3 RXOF	**Slave RX Buffer Overflow Flag** 1: RX buffer overflow 0: Not overflow **Note:** When RX buffer overflow, new received data does not store to RX buffer. Write '1' clear.
2 TXUF	**Slave TX Buffer Overflow Flag** 1: TX buffer overflow 0: No overflow **Note:** When TX buffer overflow, send the last DATA again. Write '1' clear.
1 RXFF	**Slave RX Buffer Full Flag** 1: RX buffer full 0: Not full **Note:** read DATA register will clear this bit.
0 TXEF	**Slave TX Buffer Empty Flag** 1: TX buffer empty 0: Not empty **Note:** write DATA register will clear this bit.

17.6.11 I2C_DGLCFG

Table 17-13 I2C_DGLCFG register

I2C_DGLCFG **DGLCFG** **Reset: 0x00000000**

Bit	31: 8	7	6	5	4	3	2	1	0
Name		DGLEN	STOPF	SSIE	STARTF	DGL_CNT			
Type		RW	W1C	RW	W1C	RW			
Reset		0	0	0	0	0			

Bits	Description
7 DGLEN	**Deglitch Filter Enable** 1: enable 0: disable
6 STOPF	**Bus STOP Flag** 1: STOP detected on I2C bus 0: No STOP detected on I2C bus Hardware set this bit when the STOP signal is detected on the I2C bus. **Note:** write "1" clear this bit.
5 SSIE	**Bus STOP Or START Interrupt Enable** 1: enable STRAT or STOP detection interrupt 0: disable **Note:** clear bus START or STOP interrupt: In the interrupt service routine, write '1 ' clear STARTF or STOPF flag , then interrupt clear auto.
4 STARTF	**Bus START Flag** 1: START detected on I2C bus 0: No START detected Hardware set this bit when the START signal is detected on the I2C bus. **Note:** write '1' clear.
3: 0 DGL_CNT	**Deglitch Counter** 0h: No filter/bypass 1-Fh: Filter glitches up to width of 1-15 clock cycles. Controls the width of the glitch, in terms of I2C module clock cycles, that the filter must absorb. For any glitch whose size is less or equal to this width setting, the filter does not allow the glitch to pass.

17.6.12 I2C_DATA

Table 17-14 I2C_DATA register

I2C_DATA **DATA** **Reset: 0x000001FF**

Bit	31: 9	8	7	6	5	4	3	2	1	0
Name		MAK	DATA							
Type		R	RW							
Reset		1	1							

Bits	Description
8 MAK	**Slave Monitor Function ACK Bit** For slave monitor, this field is the ACK bit from I2C bus. **Note:** MAK= '1', NACK MAK= '0', ACK For monitor, DATA[7:0] is the data transfer on I2C bus, and DATA[8] is the ACK bit.
7: 0 DATA	**Data** For master transmit(TX) mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. For master receive(RX) mode, reading this register initiates a receiving sequence of the next byte. **Note**: when making the transition out of master receive mode, switch the I2C mode before reading the DATA register to prevent an inadvertent initiation of a master receive data transfer.

17.6.13 I2C_STARTSTOP

Table 17-15 I2C_STARTSTOP register

I2C_STARTSTOP **Master START STOP signal control register** **Reset: 0x00000000**

Bit	31: 2	1	0
Name		STOP	START
Type		RW	RW
Reset		0	0

Bits	Description
1 STOP	**Master STOP Trig** write "1", master will send STOP signal. read this bit always return "0"
0 START	**Master START Trig** write "1", master will send START(RESTART) signal. read this bit always return "0".

18 SPI

18.1 Introduction

The SPI (Serial Peripheral Interface) is a bit-serial, synchronous serial, and full duplex protocol. This SPI module is a 4-wire interface that includes master and slave both.

Figure 18-1 gives an example of the connection between SPI master and SPI slave.

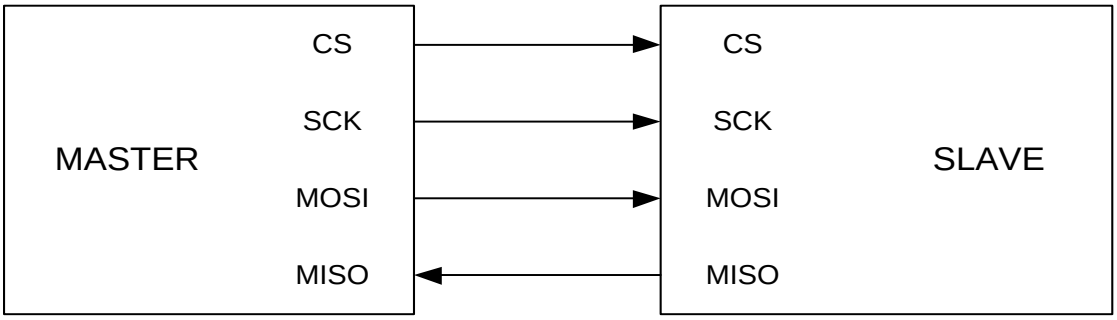

Figure 18-1 SPI system connection

18.2 Features

- Master mode or slave mode operation.
  - As master, the highest baud rate is $f_{bus}/2$ Hz (f_{bus} is APB bus clock)
  - As slave, the highest baud rate is 12 MHz
- Full-duplex.
- Master programmable baud rate.
- Serial clock phase and polarity options.
- Configurable continuous or discontinuous CS (slave select) output.
- Mode error flag with CPU interrupt capability.
- Selectable MSB-first or LSB-first shifting.
- Configurable CS setup time, hold time and idle time.
- Configurable SCK high and low period.
- 4-16 bits transfer frame format selection.
- DMA mode.
- TX buffer underflow and RX buffer overflow flag with interrupt capability.
- Slave support low power mode wake up function.

18.3 Block diagram

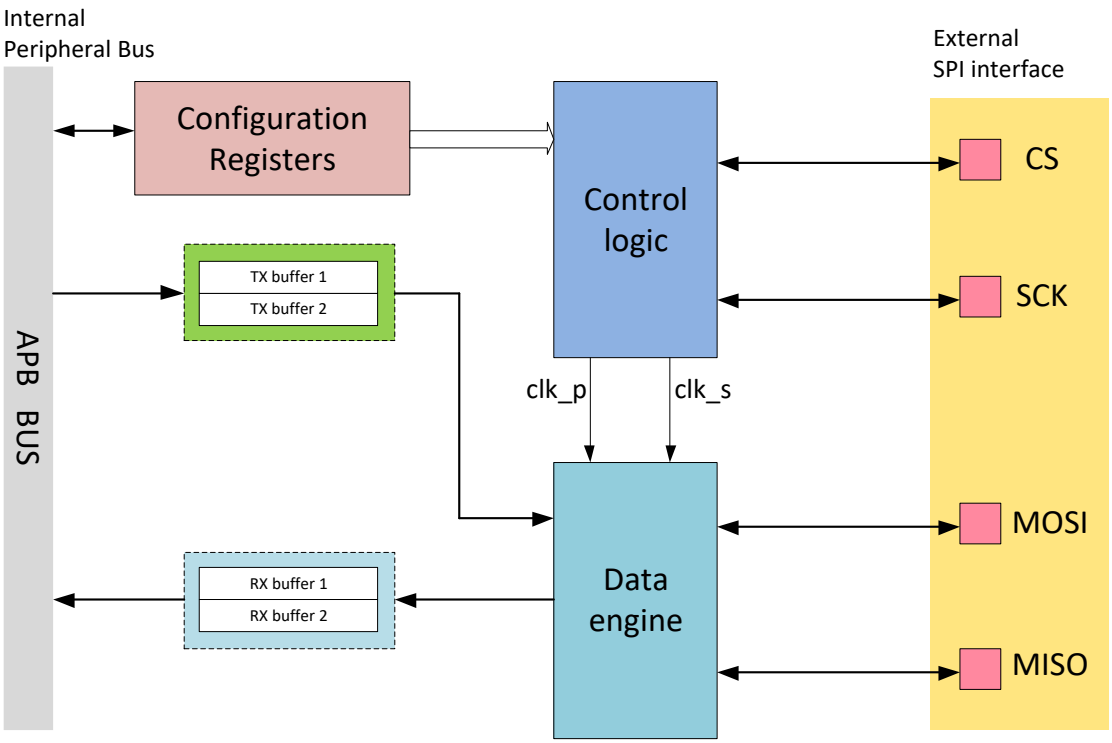

Figure 18-2 SPI block diagram

18.4 Functional description

18.4.1 Data flow & Algorithm

For master mode, data is written to a 16-bit TX buffer, then loaded to shift register by baud rate control unit. The output data is most significant first or least significant first controlled by TMSBF. After the numbers SCK periods specified by FRMSIZE, shift register shift in one byte data from MISO pin. Data received is stored into a 16-bit RX buffer.

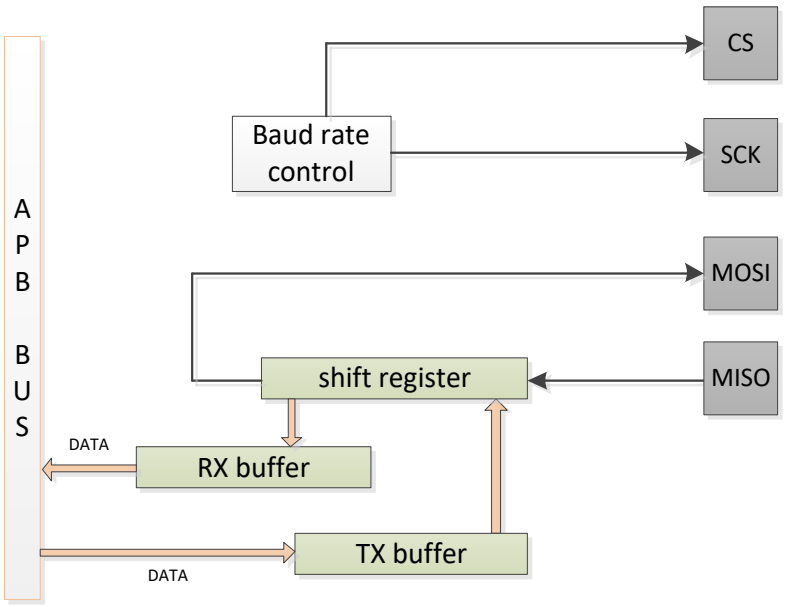

Figure 18-3 Data flow of master

For slave mode, data flow is similar to master mode. But the CS pin is the slave select input, and SCK is the SPI clock input from the master. Before a data transmission occurs, the CS pin of the slave SPI must be low. MOSI is the slave data input pin, and MISO is the data output pin.

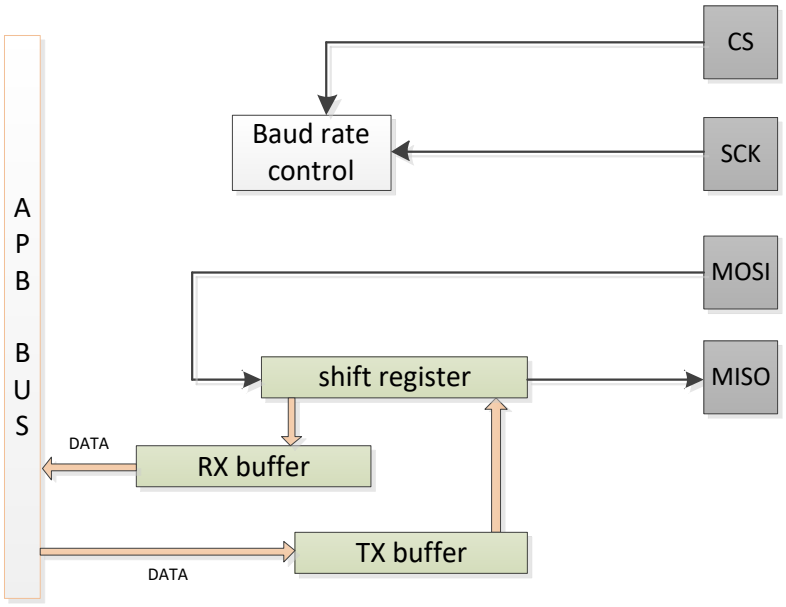

Figure 18-4 Data flow of slave

18.4.2 Input & Output timing

18.4.2.1 CPHA = 0 transfer format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after CS has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the shift register.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

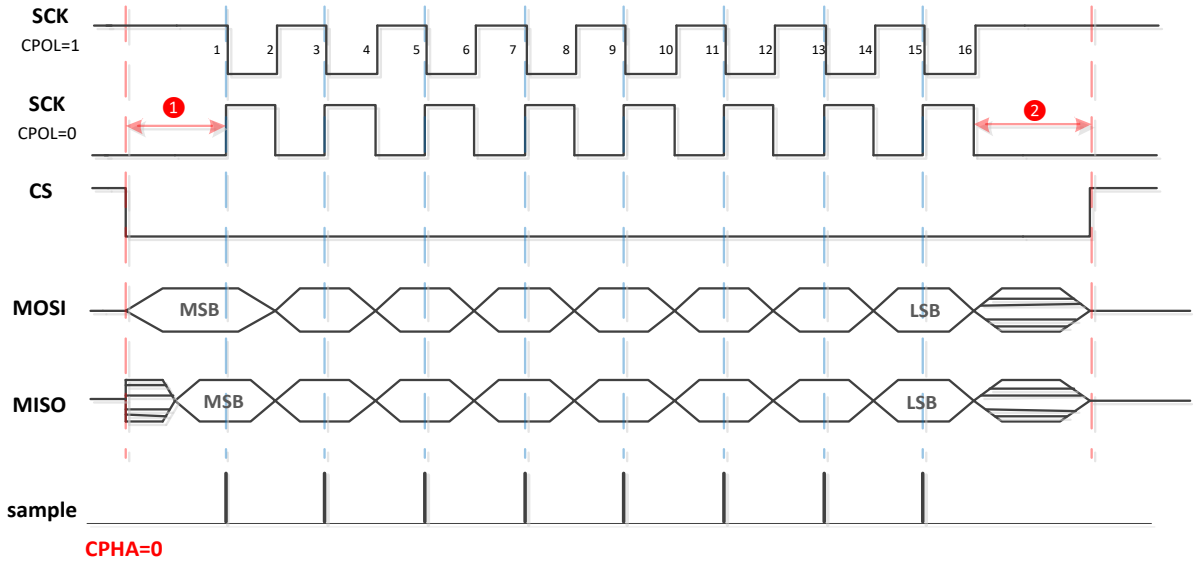

Figure 18-5 CPHA=0 transmission format

18.4.2.2 CPHA = 1 transfer format

Some peripherals require the first SCK edge before the first data bit becomes available at the data output pin, the second edge clocks data into the system.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master. A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the shift register. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

This process continues for a total 16 edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

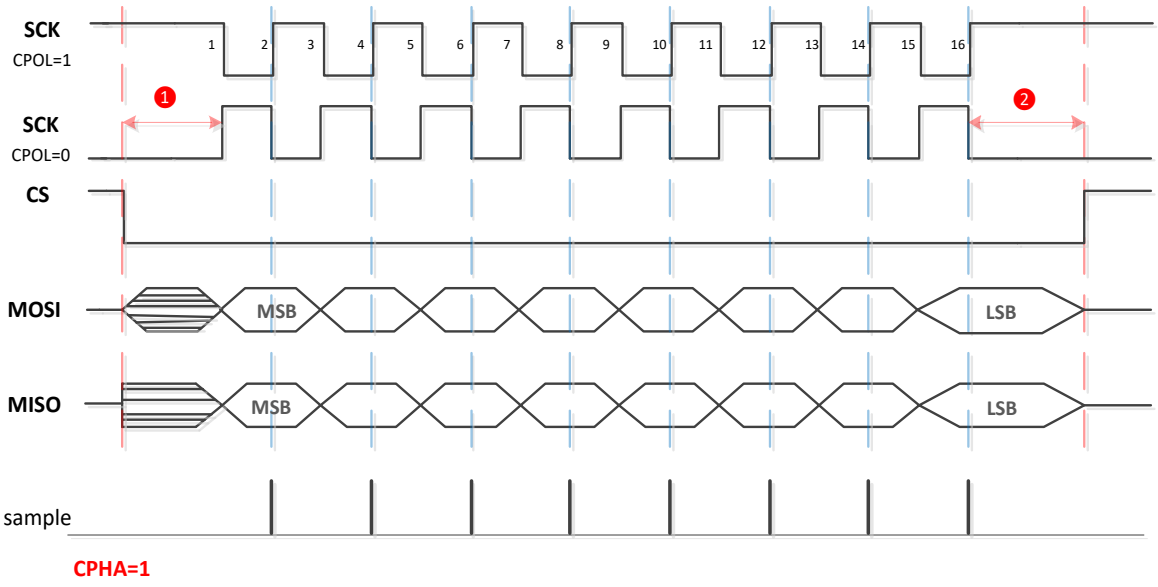

Figure 18-6 CPHA=1 transmission format

18.4.3 Master SCK output timing set

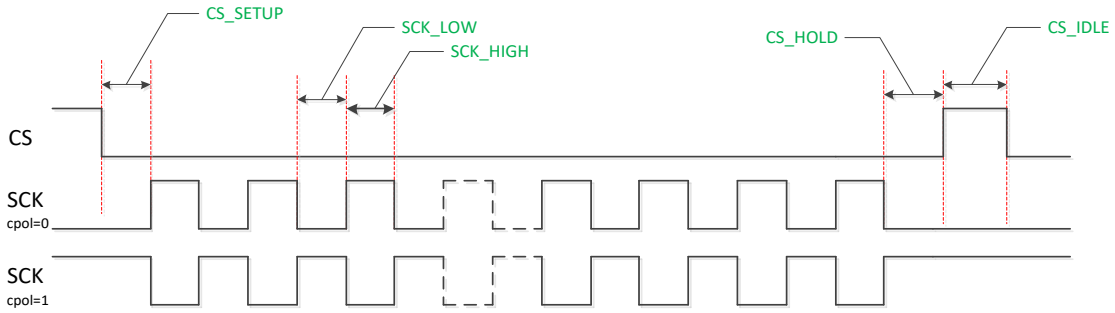

Figure 18-7 Baud rate generation

Baud rate: $f_{SCL}=f_{bclk}/(SCK\_LOW+1+SCK\_HIGH+1)$, f_{bclk} is the APB bus clock frequency.

18.4.4 Master mode fault detect

MODEF is set if the CS pin has been driven to low before SPI master initializes a transmission. Then master mode fault detect function is valid only when MSTR=1, MODFEN=1, CSOE=1.

The SCK is different from normal case if enable master mode fault detect function with CPOL=0. The SCK is high at IDLE state, and changes to low only if no mode fault error occurs. The red part

in Figure 18-8 is the time period for the master to detect the mode fault. Before CS is changed to low, SCK has been changed to low, so SPI is in communication mode of CPOL = 0. SCK keeps high after the transmission is finished.

The SCK is the same with normal case if enable master mode fault detect function with CPOL=1.

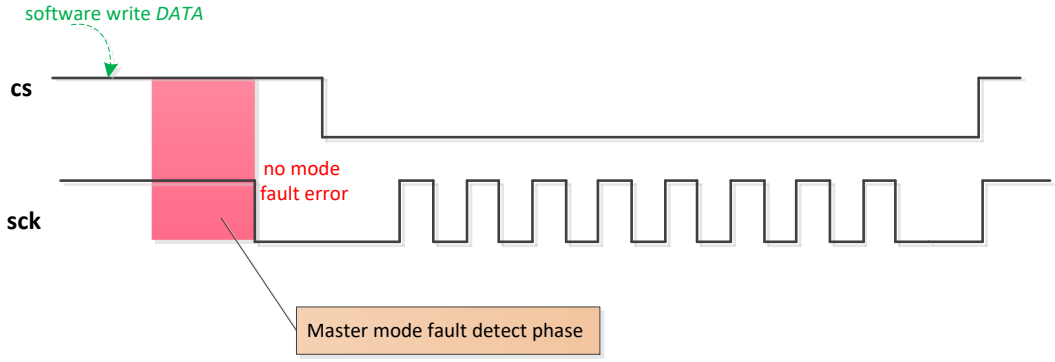

Figure 18-8 SCK output timing with mode fault detect enable

This mode fault detect function may can't detect the mode fault by master 1 with some special cases. If master 2 drive CS low during (1) period in Figure 18-9, master 1 will not set MODEF. Only master 2 drive CS low before (1) period that master 1 can set MODEF.

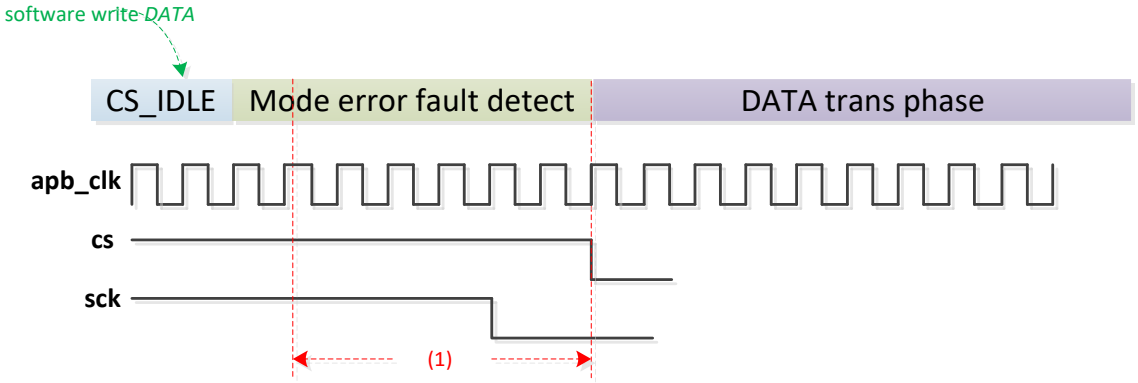

Figure 18-9 Limitation of mode fault detect

18.4.5 Slave low power wakeup

SPI slave in stop mode can generate an asynchronous interrupt to wake the CPU from the low power mode when receives a transmission. To ensure the low power wakeup function of the SPI module is correct, system must follow some regulations. Before CPU enters low power mode, system must confirm that SPI module is in IDLE state. Software can check SPI_STATUS[8] IDLEF bit state. For master mode, tx buffers are empty, rx buffers are empty, and internal hardware is idle, IDLEF can be '1'. For slave mode, tx buffers are empty, rx buffers are empty, and CS is deassert (CS is high), IDLEF can be '1'. SPI module can't ensure data valid or wakeup function correct if CPU enter low power mode when SPI module is busy.

The slave generates asynchronous wakeup interrupt only when all of the following conditions apply:

a. SPI module is in slave mode.

b. SPI slave is in IDLE states.

c. WUEN bit is '1'.

d. The one byte wakeup sequence transmission end.

CS high to low initializes the wakeup phase, and slave generates the asynchronous wakeup request after the numbers SCK cycle that FRMSIZE specified. SPI slave can receive the data of wakeup phase byte. The RXFF flag will be set after chip wakeup finish (clock recover), and read data register will return the data master sent in wakeup phase byte. Only one byte can be transmitted during wakeup phase. During the wakeup phase, a continuous transmission from a master would destroy the data received, and may lead to RXFF flag can't be set rightly.

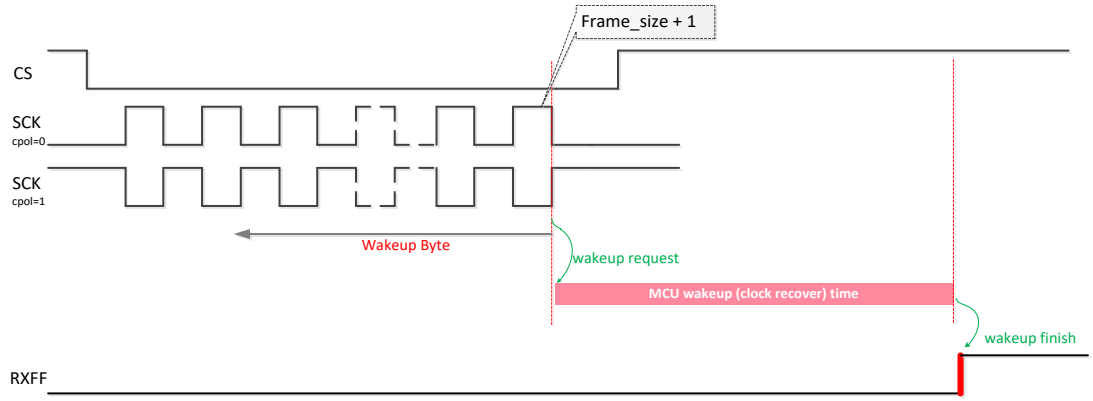

Figure 18-10 Wakeup sequence

18.4.6 Interrupt

SPI totally has 5 interrupts.

Table 18-1 Interrupt summary

Flag	Local enable
TXEF(TX Buffer Empty Flag)	TXEIE
RXFF(RX Buffer Full Flag)	RXFIE
TXUF(TX Underflow Flag)	TXUIE
RXOF(RX Overflow Flag)	RXOIE
MODFF(Mode Fault Error Flag)	MODFIE

18.5 Application note

18.5.1 Master CS continuous mode

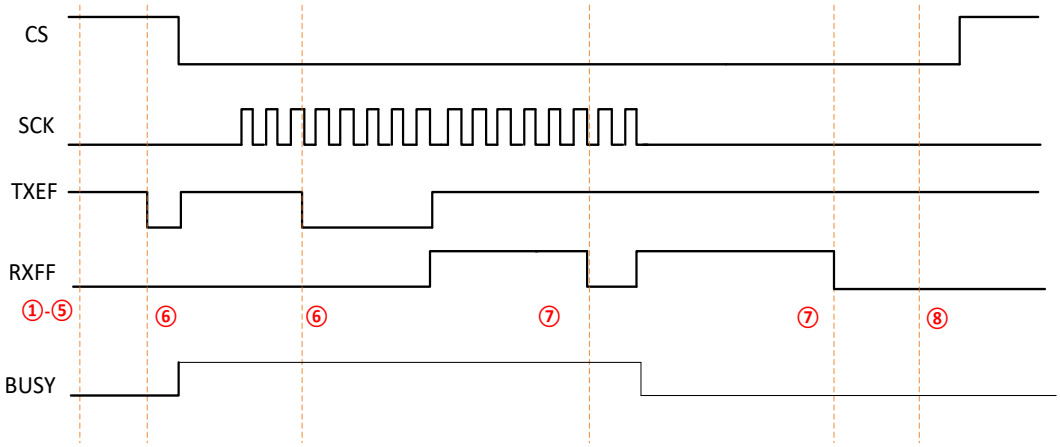

Figure 18-11 CS continuous mode

CS continuous output

1. Configure SPI_CFG0: CS_SETUP, CS_HOLD,SCK_LOW,SCK_HIGH.
2. Configure SPI_CFG1: CS_IDLE
3. Configure FRMSIZE, CPHA, CPOL, RMSBF, TMSBF, etc
4. Configure CSOE, CONT_CS, MSTR
5. SPIEN=1
6. TXEF=1, write data to DATA
7. RXFF=1, read data from DATA
8. Write CSRLS'1', release CS, then hardware goes to idle.

   That is, when CSOE = 1, CONT_CS = 1, CS is automatically pulled low by hardware.

   However, after the data is transmitted, it is needed for user software to write CSRLS'1' to pull CS high.

18.5.2 Master CS discontinuous output

1. Configure SPI_CFG0: CS_SETUP,CS_HOLD,SCK_LOW,SCK_HIGH.
2. Configure SPI_CFG1: CS_IDLE.
3. Configure FRMSIZE, CPHA, CPOL, RMSBF, TMSBF.etc.

4. Configure CSOE, CONT_CS, MSTR.
5. SPIEN=1
6. TXEF=1, write data to DATA.
7. RXFF=1, read data from DATA.

CS change to low or high by hardware when CS discontinuous mode. So software does not care the CSRLS.

18.5.3 Slave mode

1. Configure FRMSIZE, CPHA, CPOL, RMSBF, TMSBF, etc.
2. Configure MSTR.
3. SPIEN=1.
4. TXEF=1, write data to DATA.
5. RXFF=1, read data from DATA.

18.5.4 DMA mode

TXEF=1 will generate the DMA TX request.

RXFF=1 will generate the DMA RX request.

1. Init DMA.
2. Init SPI module, and enable DMATXEN, DMARXEN.
3. Wait DMA finish.
4. Other options similar to CS continuous or discontinuous mode.

18.6 Register definition

Table 18-2 SPI register mapping

SPI0 base address: 0x4000c000
SPI1 base address: 0x4000d000

Address	Name	Width (in bit)	Description
SPIx base address +0x00	SPI_CFG0	32	SPI configuration register 0
SPIx base address +0x04	SPI_CFG1	32	SPI configuration register 1
SPIx base address +0x08	SPI_CMD	32	SPI command register
SPIx base address +0x0c	SPI_STATUS	32	SPI status register
SPIx base address +0x10	SPI_DATA	32	SPI data register
SPIx base address +0x14	SPI_CFG2	32	SPI configuration register 2

Note: in the above table, x=0~1.

18.6.1 SPI_CFG0

Table 18-3 SPI_CFG0 register

SPI_CFG0 **SPI configuration register 0** **Reset: 0x05050505**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CS_SETUP								CS_HOLD							
Type	RW								RW							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SCK_LOW								SCK_HIGH							
Type	RW								RW							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1

Bits	Description
31: 24 CS_SETUP	**CS Setup Time Configuration** The chip select setup time = (CS_SETUP +1)*CLK_PERIOD, where CLK_PERIOD is the cycle time of the clock the SPI engine adopts.
23: 16 CS_HOLD	**CS Hold Time Configuration** The chip select hold time = (CS_HOLD +1)*CLK_PERIOD.
15:8 SCK_LOW	**SCK Low Time Configuration** The SCK clock low time = (SCK_LOW +1) * CLK_PERIOD. **Note: When CPOL is 0, this bit configures the time of SCK_LOW. When CPOL is 1, this bit configures the time of SCK_HIGH.**
7: 0 SCK_HIGH	**SCK High Time Configuration**

The SCK clock high time = (SCK_HIGH +1) * CLK_PERIOD.

Note: When CPOL is 0, this bit configures the time of SCK_HIGH. When CPOL is 1, this bit configures the time of SCK_LOW.

18.6.2 SPI_CFG1

Table 18-4 SPI_CFG1 register

SPI_CFG1 **SPI configuration register 1** **Reset: 0x027C0005**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		WKUEN		CONT_CS		MODFEN	CSOE		FRMSIZE				RMSBF	MSBF	CPHA	CPOL
Type		RW		RW		RW	RW		RW				RW	RW	RW	RW
Reset		0		0		0	1		0	1	1	1	1	1	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMARXEN	DMATXEN	MODFIE	MSTR	RXOIE	TXUIE	RXFIE	TXEIE	CS_IDLE							
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bits	Description
30 WKUEN	**Slave Wakeup Function Enable** 1: slave wake up function enable 0: slave wake up function disable
28 CONT_CS	**CS Continuous Output Enable** 0: CS output non-continuous 1: CS output continuous
26 MODFEN	**Master Mode Fault Detect Enable** 0: disable the master mode fault detect function 1: enable the master mode fault detect function
25 CSOE	**CS Hardware Output Enable** 0: disable the CS hardware output 1: enable the CS hardware output
23: 20 FRMSIZE	**Frame Size** 0000: 4bit 0001: 4bit 0010: 4bit 0011: 4bit ```

Bits	Description
	1110: 15bit 1111: 16bit
19 RMSBF	**RX MSB First** 0: the first bit of shifter shift in is the LSB of the input data 1: the first bit of shifter shift in is the MSB of the input data
18 MSBF	**TX MSB First** 1: TX MSB first (MSB first shift out) 0: TX LSB first (LSB first shift out)
17 CPHA	**Clock Phase** 1: the first SCK transition edge is the data capture edge 0: the first SCK transition edge is the data shift out edge
16 CPOL	**Clock Polarity** 0: SCK is 0 when idle 1: SCK is 1 when idle
15 DMARXEN	**DMA RX Request Enable** 0: disable the DMA RX request 1: enable the DMA RX request
14 DMATXEN	**DMA TX channel Enable** 0: disable the DMA RX request 1: enable the DMA RX request
13 MODFIE	**Mode Fault Interrupt Enable** 0: disable 1: enable
12 MSTR	**Master or Slave Mode Selection** 0: slave mode 1: master mode
11 RXOIE	**RX Buffer Overflow Interrupt Enable** 0: disable 1: enable
10 TXUIE	**TX Buffer Underflow Interrupt Enable** 0: disable 1: enable
9 RXFIE	**RX Buffer Full Interrupt Enable** 0: disable 1: enable Note: RX buffers Not Empty will generate an interrupt.

Bits	Description
8 TXEIE	**TX Buffer Empty Interrupt Enable** 0: disable 1: enable Note: TX buffers Not Full will generate an interrupt.
7: 0 CS_IDLE	**CS Idle Time** CS IDLE time = (CS_IDLEA_COUNT+1)*CLK_PERIOD

18.6.3 SPI_CMD

Table 18-5 SPI_CMD register

SPI_CMD **SPI command register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										RO TRIG	CS RLS	SW RST				SPI EN
Type										RW	RW	RW				RW
Reset										0	0	0				0

Bits	Description
6 ROTRIG	**Master RX only mode trig** **Note:** write this bit '1' will trigger a sequence of read, while ROEN=1 in CFG2. Read this bit will return '0' always.
5 CSRLS	**CS Release** 1: release CS 0: no effect Software write this bit '1', then CS will go to high. Read this bit always return "0". This bit valid for CS continuous output(CONT_CS=1, CSOE=1).
4 SWRST	**Software reset** 1: reset 0: no effect **Note:** this reset only reset master engine/buffer/flag logic, slave buffer/flag logic. CFG0/CFG1/CFG2/CMD control bits will not reset.

Bits	Description
0 SPIEN	**SPI Enable** 1: enable 0: disable

18.6.4 SPI_STATUS

Table 18-6 SPI_STATUS register

SPI_STATUS **SPI status register** **Reset: 0x00000101**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								IDLEF	MEBY			MODEF	RXOF	TXUF	RXFF	TXEF
Type								R	R			R	W1C	W1C	R	R
Reset								1	0			0	0	0	0	1

Bits	Description
8 IDLEF	**SPI IDLE Flag** 1: SPI module hardware IDLE 0: SPI module hardware not IDLE **Note:** For master, TX buffer empty, RX buffer empty, and internal hardware idle, this bit can be '1'. For slave, TX buffer empty, RX buffer empty, and CS deassert(not be selected) , this bit can be '1'.
7 MEBY	**SPI Master Engine Busy Flag** 1: SPI master hardware's one byte transmission not finish 0: SPI master hardware's one byte transmission finish
4 MODEF	**Mode Fault Error Flag** 1: master mode error detected 0: no multi-master error detected When SPI configured as a master, it will detect the CS line state before drive CS low. If CS have been low, indicating another master have initial a transmission, MODEF flag will be set. **Note:** Write "SWRST=1" clear this bit, SPI module has to be reset by software.

Bits	Description
3 RXOF	**RX Buffer Overflow Flag** 1: RX buffer overflow 0: no overflow There are two FIFOs in the receive buffer. If the RX buffer data is not read out by user in time after receiving two bytes, the next received data will come and generate RX overflow. **Note:** Write "1" clear this bit. When an overflow occurs, it needs to be cleared in time to avoid affecting the RXFF flag and DMA reception. If overflow, then only 1 byte valid data in DATA register can read out.
2 TXUF	**TX Buffer Underflow Flag** 1: TX buffer underflow 0: not underflow In slave mode, if no data is written to the TX data register, the master will start to communicate and TX down flow will occur. **Note**: write "1" clear this bit. When an overflow occurs, it needs to be cleared in time.
1 RXFF	**RX Buffer Full Flag** 1: full 0: not full **Note:** As long as rx buffers have valid data received (1 byte or 2 bytes), this bit will be '1'. So this bit '1' does not indicate that 2 bytes data received in rx buffers. Read DATA register will clear this bit hardware auto. This bit named RX Buffers Not Empty is more reasonable maybe.
0 TXEF	**TX Buffer Empty Flag** 1: empty 0: not empty **Note:** As long as tx buffers not full (2 bytes data), this bit will be '1'. Write DATA register will clear this bit hardware auto. This bit named TX buffers Not Full is more reasonable maybe.

18.6.5 SPI_DATA

Table 18-7 SPI_DATA register

SPI_DATA **SPI data register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA															
Type	RW															
Reset	0															

Bits	Description
15:0 DATA	**SPI Data Port Register** read: read will return DATA received write: write the DATA to be send

18.6.6 SPI_CFG2

Table 18-8 SPI_CFG2 register

SPI_CFG2 **SPI configuration register 2** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													ROEN	TOEN	MNDC	
Type													RW	RW	RW	
Reset													0	0	0	

Bits	Description
3 ROEN	**RX Only Mode Enable** 1: enable 0: disable **Note:** RX only mode, TXEF will keep 0. Not trigger RXEF IRQ even TXEIE=1.

Bits	Description
2 TOEN	**TX Only Mode Enable** 1: enable 0: disable **Note:** TX only mode, RXFF not set after one byte transfer finished. Not trigger RXFF IRQ even RXFIE=1.
1 MNOV	**Master No Overflow Mode** 1: enable 0: disable If rxbuff is full, then write to txbuff will not trigger new send.

19 DMA

19.1 Introduction

The Direct Memory Access controller (DMA) is used to speed up memory transmission between peripherals and memory or memory to memory. Without CPU operations, DMA can move data fast and improve microcontroller's performance.

There are 4 dedicated channels for different kinds peripherals, such as UART, I2C, SPI, ADC, etc. It has a round-robin arbiter for handling priorities between each channel.

19.2 Features

- 4 independently configurable channels.
- Each of the 4 channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel (memory to memory). This configuration is done by software.
- Priorities between requests from channels of one DMA are software programmable (4 levels consisting of very high, high, medium, low) or in case of equality, it will depend on hardware (channel 0 has the highest priority and channel 3 has the lowest priority).
- Independent source and destination transfer size (byte, half word, word). Source/destination addresses must be aligned on the data size.
- Support for circular buffer management.
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) OR together in a single interrupt request for each channel.
- Memory-to-memory transfer.
- Peripheral-to-memory and memory-to-peripheral transfers.
- Access to SRAM, APB peripherals as source and destination.
- Programmable number of data to be transferred: up to 32767.

19.3 Block diagram

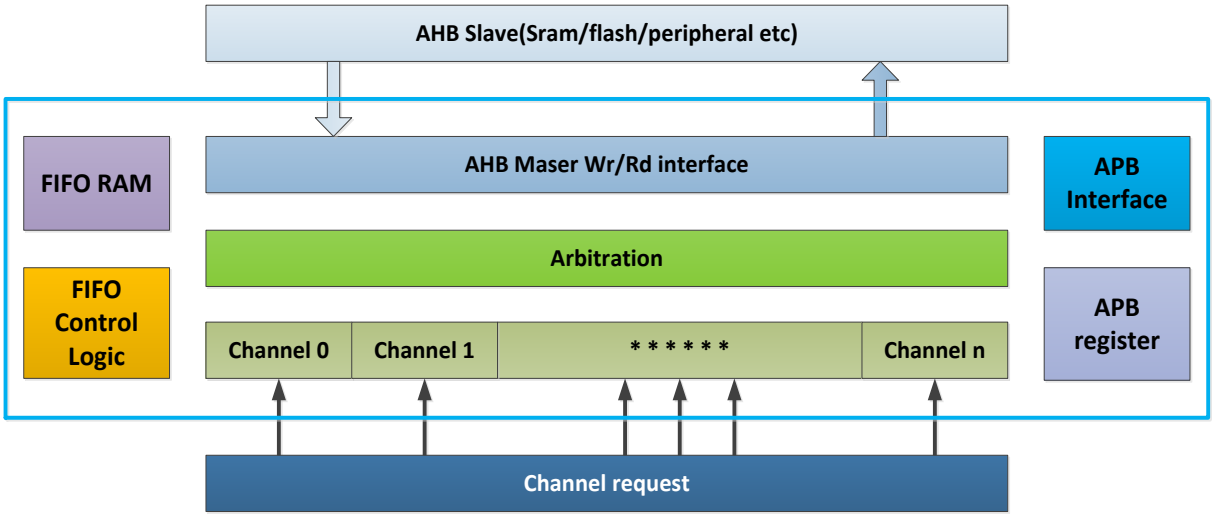

Figure 19-1 DMA block diagram

19.4 Function description

19.4.1 Mode of operations

DMA module doesn't support Stop mode or Standby mode. So user must close all DMA channels before enter Stop or Standby mode.

19.4.2 DMA request mapping

Table 19-1 DMA request mapping

Peripherals	DMA channel 0, 1, 2, 3 request
ADC	ADC
SPI	SPI0_TX/ SPI0_RX/ SPI1_TX/ SPI1_RX
UART	UART0_TX/ UART0_RX/ UART1_TX/ UART1_RX/ UART2_TX/ UART2_RX
I2C	I2C0_TX/ I2C0_RX/

Peripherals	DMA channel 0, 1, 2, 3 request
	I2C1_TX/ I2C1_RX

19.4.3 DMA arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CONFIG register. There are four levels:
  - Very high priority.
  - High priority.
  - Medium priority.
  - Low priority.
- When software priorities are equal, priority is determined based on the hardware channel number. For example, channel 0 has the highest priority, channel 3 has the lowest priority, and so on.

19.5 Application note

ALL registers except DMA_CHAN_ENABLE register should be programmed before user configures DMA_CHAN_ENABLE register. Because parameters take effect after CHAN_ENABLE's pose edge. And when CHAN_ENABLE is High, please don't modify parameters.

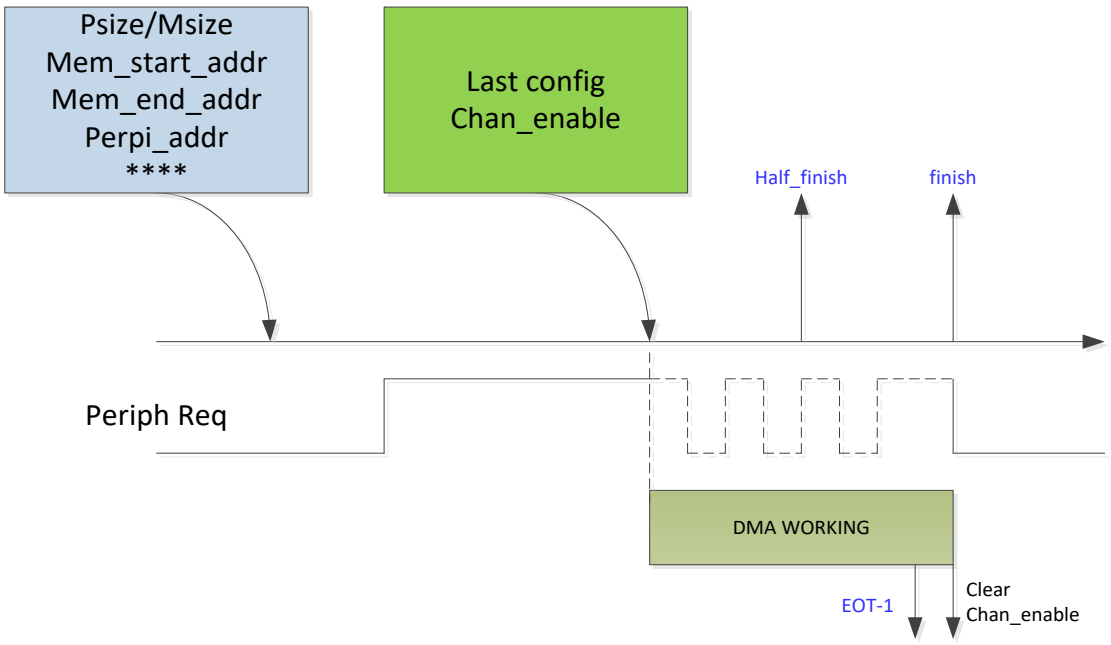

Figure 19-2 DMA configuration guide

19.5.1 Soft reset and hard reset

Soft reset and hard reset exist in DMA global control and each DMA engine. When soft reset is set, the engine will be reset after the current transaction is finished, and therefore the soft reset will not cause any bus hang. Conversely, when hard reset is set, the engine will be reset immediately, and therefore the bus may go down due to unfinished (and will never finish) transaction.

The mechanism of global soft reset is described as follows. When the software would like to re-start all engines or re-clear all engines in DMA, it can set the global soft resetter to 1. Then HW set WARM_RST back to 0 to finish the global soft reset.

The mechanism of global hard reset is described as follows. When the software would like to re-start all engines or re-clear all engines in DMA without waiting for any period of time, User can set the global HARD_RST to 1 and then set to 0 to complete the global hard reset. Note that this may break the bus protocol and result in system hang.

19.5.2 Pause and Resume

There are pause and resume functions available for DMA. The mechanism is as follows:

1. Start DMA. (Program necessary settings, then set CHAN_ENABLE = 1.)
2. Pause DMA. (Set STOP = 1.)
3. Resume DMA. (Set STOP = 0.)
4. Wait for DMA to finish. (CHAN_ENABLE will become 0, and interrupt flag will be set to 1.)

The software can repeat step 2 and 3 many times when DMA is running. DMA will not pause immediately and will wait for the last transaction to be finished.

19.5.3 Channel circular

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CHAN_CIRCULAR bit in the DMA_CONFIG register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

It should be noted that after read or write MEM_END_ADDR-0x01, the DMA master will return to MEM_START_ADDR to continue handling.

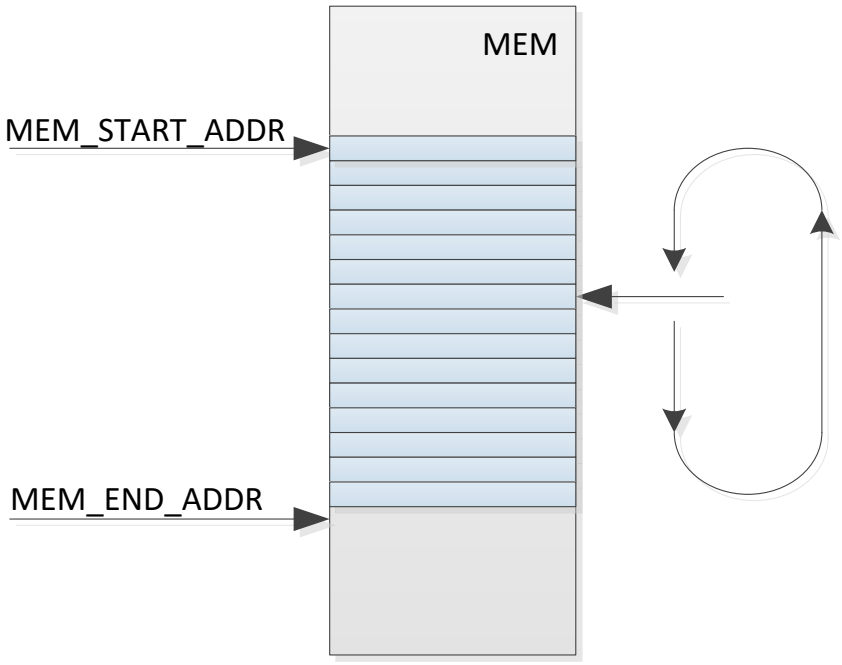

Figure 19-3 DMA channel circular

19.5.4 I2C using DMA

When the number of data transfers which has been programmed for the corresponding DMA stream is reached, the DMA controller sends an End of Transfer EOT-1 signal to the I2C interface. I2C hard engine has received the signal and decide whether to send ACK/NACK.

19.5.5 Programmable data width, data alignment

Table 19-2 Programmable data width & data alignment

MSIZE	PSIZE	Length	MEM_BYTE_MODE	Source address/data	Direction	Transfer operation	Destination address/data
32	8	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1.Read 03020100@0x00, then write 00@0x00; 2.Read 07060504@0x04, then write 04@0x01; 3. Read 0B0A0908@0x08, then write 08@0x02; 4.Read 0F0E0D0C@0x0C,then write 0C@0x03	0x00/00 0x01/04 0x02/08 0x03/0C
32	16	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1.Read 03020100@0x00, then write 0100@0x00; 2.Read 07060504@0x02, then write 0504@0x01; 3. Read 0B0A0908@0x08, then write 0908@0x04; 4.Read 0F0E0D0C@0x0C,then write 0D0C @0x06	0x00/0100 0x02/0504 0x04/0908 0x06/0D0C
32	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1.Read 03020100@0x00, then write 03020100@0x00; 2.Read 07060504@0x04, then write 07060504@0x04; 3. Read 0B0A0908@0x08, then write 0B0A0908@0x08; 4.Read 0F0E0D0C@0x0C,then write 0F0E0D0C@0x0C	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C
32	8	4	01	0x00/03020100 0x04/07060504	1(TX)	1. Read 03020100@0x00, split 03020100 into 0302 and 0100, then write 00@0x00, 02@0x01 2. Read 07060504@0x04,split 07060504 into 0706 and 0504, then write 04@0x02,06@0x03	0x00/00 0x01/02 0x02/04 0x03/06
32	16	4	01	0x00/03020100 0x04/07060504	1(TX)	1. Read 03020100@0x00, split 03020100 into 0302 and 0100, then write 0100@0x00, 0302@0x02 2. Read 07060504@0x04,split 07060504 into 0706 and 0504, then write 0504@0x04,0706@0x06	0x00/0100 0x02/0302 0x04/0504 0x06/0706
32	32	4	01	0x00/03020100	1(TX)	1. Read 03020100@0x00,	0x00/00000100

MSIZE	PSIZE	Length	MEM_BYTE_MODE	Source address/data	Direction	Transfer operation	Destination address/data
				0x04/07060504		split 03020100 into 0302 and 0100, then write **0000**0100@0x00, **0000**0302@0x04 2. Read 07060504@0x04,split 07060504 into 0706 and 0504, then write **0000**0504@0x08,**0000**0706@0x0C	0x04/00000302 0x08/00000504 0x0C/00000706
32	8	4	11	0x00/03020100	1(TX)	1.Read 03020100@0x00, split 0302010 into 03,02,01,00(4byte) then write 00@0x00, 01@0x01,02@0x02,03@0x03	0x00/00 0x01/01 0x02/02 0x03/03
32	16	4	11	0x00/03020100	1(TX)	1.Read 03020100@0x00, split 0302010 into 03,02,01,00(4byte) then write **00**00@0x00, **00**01@0x02,**00**02@0x04,**00**03@0x06	0x00/0000 0x02/0001 0x04/0002 0x06/0003
32	32	4	11	0x00/03020100	1(TX)	1.Read 03020100@0x00, split 0302010 into 03,02,01,00(4byte) then write **000000**00@0x00, **000000**01@0x04,**000000**02@0x08, **000000**03@0x0C	0x00/00000000 0x04/00000001 0x08/00000002 0x0C/00000003
8	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1.Read 03020100@0x00, then write 00@0x00; 2.Read 07060504@0x04, then write 04@0x01; 3. Read 0B0A0908@0x08, then write 08@0x02; 4.Read 0F0E0D0C@0x0C,then write 0C@0x03	0x00/00 0x01/04 0x02/08 0x03/0C
16	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1.Read 03020100@0x00, then write 0100@0x00; 2.Read 07060504@0x04, then write 0504@0x02; 3. Read 0B0A0908@0x08, then write 0908@0x04; 4.Read 0F0E0D0C@0x0C,then write 0D0C @0x06	0x00/0100 0x02/0504 0x04/0908 0x06/0D0C
32	32	4	00	0x00/03020100	0(RX)	1.Read 03020100@0x00,	0x00/03020100

MSIZE	PSIZE	Length	MEM_BYTE_MODE	Source address/data	Direction	Transfer operation	Destination address/data
				0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C		then write 03020100@0x00; 2.Read 07060504@0x04, then write 07060504@0x04; 3. Read 0B0A0908@0x08, then write 0B0A0908@0x08; 4.Read 0F0E0D0C@0x0C,then write 0F0E0D0C@0x0C	0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C
32	32	4	01	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1.Read 03020100@0x00, save 0100 to BIT[15:0] of the DMA inner buffer, 0302 is discarded; Read 07060504@0x04, save 0504 to BIT[31:16] of the DMA inner buffer,0706 is discarded, the re-assembled 32-bit data is placed at 0x00. 2. Read 0B0A0908@0x08, save 0908 to BIT[15:0] of the DMA inner buffer, 0B0A is discarded; Read 0F0E0D0C@0x0C, save 0D0C to BIT[31:16] of the DMA inner buffer,0F0E is discarded, the re-assembled 32-bit data is placed at 0x04 .	0x00/05040100 0x04/0D0C0908
32	32	4	11	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1.Read 03020100@0x00, save 00 to BIT[7:0] of the DMA inner buffer; 2. Read 07060504@0x04, save 04 to BIT[15:8] of the DMA inner buffer; 3. Read 0B0A0908@0x08, save 08 to BIT[23:16] of the DMA inner buffer; 4. Read 0F0E0D0C@0x0C, save 0C to BIT[31:24] of the DMA inner buffer, the re-assembled 32-bit data is placed at 0x00 .	0x00/0c080400

Note:

1. When dir=1(TX), data is transferred from MEM to PERIPH;

   When dir=0(RX), data is transferred from PERIPH to MEM.

2. Some 0 in bold means that DMA is an additional supplement to match DEST SIZE and is not read from SRC.

3. When dir=1(TX), MSIZE is only 32, which Indicates that DMA constantly reads 32bit data from MEM.

   When dir=0(RX), PSIZE is only 32, which Indicates that DMA constantly reads 32bit data from PERIPH, even if the effective bit of the peripheral register is only 8bit.

4. While using MEM2MEM to transfer data, it is needed to set MSIZE and PSIZE to the same size.

19.5.6 Channel configuration procedure

1. Configure the MEM starting address of the DMA channel, the ending address is DMA_MEM_START_ADDR, DMA_MEM_END_ADDR;
2. Configure the PERIPH address of the DMA channel: DMA_PERIPH_ADDR;
3. Configure the interrupt enable of the DMA channel: DMA_INTEN, which can be configured as half finished, full finished and transfer error interrupt;
4. Configure the DMA_CONFIG register of the DMA channel: PERIPH_SEL, CHAN_PRIORITY, CHAN_CIRCULAR, CHAN_DIR, MEM2MEM, MEM_INCREMENT, PERIPH_INCREMENT, MEM_BYTE_MODE, MEM_SIZE, PERIPH_SIZE.

   ① Configuration: from MEM to PERIPH

   Configure CHAN_DIR=1(read from memory), MEM2MEM=0(transfer between non-memory and memory), MEM_INCREMENT=1(the MEM address is incremented), PERIPH_INCREMENT=0 (the PERIPH address is fixed), PERIPH_SEL is selected as UARTx_TX/ SPIx_TX/ I2Cx_TX.

   ② Configuration: from PERIPH to MEM

   Configure CHAN_DIR=0(read from PERIPH), MEM2MEM=0(transfer between non-memory and memory), MEM_INCREMENT=1(the MEM address is increased), PERIPH_INCREMENT=0(the PERIPH address is fixed), PERIPH_SEL is selected as UARTx_RX/ SPIx_RX/ I2Cx_RX/ ADC.

   ③ Configuration: from MEM to MEM

   Configure CHAN_DIR=1(read from MEM), MEM2MEM=1(transfer between MEM and MEM), MEM_INCREMENT=1 (the MEM address is increased), PERIPH_INCREMENT=1 (the PERIPH address is increased).

5. Configure the DMA_CHAN_LENGTH register of the DMA channel, this register does not refer to the total byte length, but the times the DMA channel is transferred.
6. Enable the DMA channel, that is, configure the DMA_CHAN_ENABLE register: CHAN_ENABLE=1.

19.6 Register definition

Table 19-3 DMA register mapping

DMA0 base address:0x40012000
DMA0_Channel0 base address:0x40012040
DMA0_Channel1 base address:0x40012080
DMA0_Channel2 base address:0x400120c0
DMA0_Channel3 base address:0x40012100

Address	Name	Width (in bit)	Description
DMA0 base address +0x00	DMA_TOP_RST	32	General DMA reset
DMA0_Channelx base address +0x00	DMA_STATUS	32	Status register
DMA0_Channelx base address +0x04	DMA_INTEN	32	Interrupt enable
DMA0_Channelx base address +0x08	DMA_RST	32	Channel reset
DMA0_Channelx base address +0x0C	DMA_STOP	32	DMA channel stop
DMA0_Channelx base address +0x10	DMA_CONFIG	32	DMA channel configuration
DMA0_Channelx base address +0x14	DMA_CHAN_LENGTH	32	DMA channel length
DMA0_Channelx base address +0x18	DMA_MEM_START_ADDR	32	Memory start address
DMA0_Channelx base address +0x1C	DMA_MEM_END_ADDR	32	Memory end address
DMA0_Channelx base address +0x20	DMA_PERIPH_ADDR	32	Channel peripheral address
DMA0_Channelx base address +0x24	DMA_CHAN_ENABLE	32	Channel enable
DMA0_Channelx base address +0x28	DMA_DATA_TRANS_NUM	32	Data transfer number
DMA0_Channelx base address +0x2C	DMA_INTER_FIFO_DATA_LEFT_NUM	32	data left in inter fifo

19.6.1 DMA_TOP_RST

Table 19-4 DMA_TOP_RST register

DMA_TOP_RST **General DMA reset register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															HARD_RST	WARM_RST
Type															RW	RW
Reset															0	0

Bits	Description
1 HARD_RST	**General DMA hard reset (reset regardless of the current transaction)** 0: Disable 1: Enable SW sets 'HARD_RST' to 1 then sets 'HARD_RST' back to 0, and the reset mechanism is finished. HARD_RST will restore all DMA Channel registers to their default values.
0 WARM_RST	**General DMA soft reset (reset after the current transaction)** 0: Disable 1: Enable SW sets 'WARM_RST' to 1 and HW clears 'WARM_RST' back to 0, and the reset mechanism is finished. WARM_RST will restore all DMA Channel registers to their default values (except the DMA_INTEN register).

19.6.2 DMA_STATUS

Table 19-5 DMA_STATUS register

DMA_STATUS **DMA status register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														TRANS_ERROR	HALF_FINISH	FINISH
Type														W0C	W0C	W0C
Reset														0	0	0

Bits	Description
2 TRANS_ERROR	**Transfer Error Flag** 0: Error not happened 1: Error happened When channel n is reading or writing, whether or not error flag occurred, write “0” to this bit, clear status.
1 HALF_FINISH	**Half Finish Flag** 0: Half of data not finished 1: Half of data finished When channel n is reading or writing, whether or not half Number of Data transferred. write “0” to this bit, clear status.
0 FINISH	**Finish Flag** 0: Data transfer not finished 1: Data transfer finished When channel n is reading or writing, whether or not DMA channel finished data transfer. write “0”to this bit, clear status.

19.6.3 DMA_INTEN

Table 19-6 DMA_INTEN register

DMA_INTEN **Interrupt enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														TRANS_ERROR INTERRUPT ENABLE	HALF_FINISH INTERRUPT ENABLE	FINISH INTERRUPT ENABLE
Type														RW	RW	RW
Reset														0	0	0

Bits	Description
2 TRANS_ERROR INTERRUPT ENABLE	**TRANS_ERROR interrupt enable** 0: interrupt disabled 1: interrupt enabled
1 HALF_FINISH INTERRUPT ENABLE	**HALF_FINISH interrupt enable** 0: interrupt disabled 1: interrupt enabled Only when DMA_CHAN_LENGTH is greater than or equal to 8, the half complete interrupt will be enabled.
0 FINISH INTERRUPT ENABLE	**FINISH interrupt enable** 0: interrupt disabled 1: interrupt enabled

19.6.4 DMA_RST

Table 19-7 DMA_RST register

DMA_RST **Channel reset register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														FLUSH	HARD_RST	WARM_RST
Type														RW	RW	RW
Reset														0	0	0

Bits	Description
2 FLUSH	**DMA channel flush** 0: Disable 1: Enable Setting FLUSH = 1 will stop DMA and allow DMA to flush its internal buffer residual data to the memory. After the flush is finished, DMA will set channel enable to 0 and stop DMA. SW sets FLUSH = 1 and waits for HW to be clear to 0.
1 HARD_RST	**DMA channel hard reset (reset regardless of the current transaction)** 0: Disable 1: Enable SW sets 'HARD_RST' to 1 then sets 'HARD_RST' back to 0, and the reset mechanism is finished.
0 WARM_RST	**DMA channel soft reset (reset after the current transaction)** 0: Disable 1: Enable SW sets 'warm' to 1, then HW sets 'WARM_RST' back to 0, and the reset mechanism is finished.

19.6.5 DMA_STOP

Table 19-8 DMA_STOP register

DMA_STOP **DMA channel stop register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																STOP
Type																RW
Reset																0

Bits	Description
0 STOP	**DMA channel stop (stop after the current transaction)** 0: Not stop channel transfer 1: STOP channel transfer SW sets ' STOP ' to 1 then after current transaction, transfer paused, SW sets ' STOP ' to 0 then resume data transfer. From STOP mode to continue DMA, Data will not be lost from MEM to MEM or from MEM to peripheral. Data may be lost from peripheral to MEM, depending on whether there are overflows that the peripheral receive data FIFO after STOP.

19.6.6 DMA_CONFIG

Table 19-9 DMA_CONFIG register

DMA_CONFIG **DMA channel configuration register** **Reset: 0x00000050**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													PERIPH_SEL			
Type													RW			
Reset													0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				MEM_BYTE_MODE		CHAN_DIR	CHAN_CIRCULAR	PERIPH_INCREMENT	MEM_INCREMENT	PERIPH_SIZE		MEM_SIZE	CHAN_PRIORITY			MEM2MEM
Type				RW		RW	RW	RW	RW	RW		RW	RW			RW
Reset				0		0	0	0	0	10		10	0			0

Bits	Description
19:16 PERIPH_SEL	**PERIPH SELECTION** 0000: UART0_TX 0001: UART0_RX 0010: UART1_TX 0011: UART1_RX 0100: UART2_TX 0101: UART2_RX 0110: SPI0_TX 0111: SPI0_RX 1000: SPI1_TX 1001: SPI1_RX 1010: I2C0_TX 1011: I2C0_RX 1100: I2C1_TX 1101: I2C1_RX 1110: ADC
12: 11 MEM_BYTE_MODE	**Indicate MEM word split transfer number** 00: 1 01: 2 10: 2 11: 4
10 CHAN_DIR	**Indicate Data transfer direction** 0: read from Peripheral 1: read from MEM
9 CHAN_CIRCULAR	**Channel circular mode** 0: Circular mode disable 1: Circular mode enable
8 PERIPH_INCREMENT	**PERIPHERAL ADDRESS INCREMENT MODE** 0: Peripheral address fixed 1: Peripheral address increment
7 MEM_INCREMENT	**MEM address increment mode** 0: MEM address fixed 1: MEM address increment

Bits	Description
6: 5 PERIPH_SIZE	**PERIPH data length** 00: 8bit 01:16bit 10:32bit 11: reserve
4: 3 MEM_SIZE	**MEM data length** 00: 8bit 01:16bit 10:32bit 11: reserve
2: 1 CHAN_PRIORITY	**Channel priority** 00: low 01: middle 10: high 11: very high
0 MEM2MEM	**MEM to MEM mode** 0: transfer between non MEM and MEM 1: transfer between MEM and MEM

19.6.7 DMA_CHAN_LENGTH

Table 19-10 DMA_CHAN_LENGTH register

DMA_CHAN_LENGTH **DMA channel length** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CHAN_LENGTH															
Type	RW															
Reset	0															

Bits	Description
15: 0 CHAN_LENGTH	**DMA Channel Transfer Number** 0~32767 [15] bit should be 0 **Note: When I2C transfers using DMA, it is needed to configure CHAN_LENGTH >= 2.**

19.6.8 DMA_MEM_START_ADDR

Table 19-11 DMA_MEM_START_ADDR register

DMA_MEM_START_ADDR **Memory start address register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MEM_START_ADDR[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEM_START_ADDR [15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 MEM_START_ADDR	**MEM_START_ADDR** Memory initial address.

19.6.9 DMA_MEM_END_ADDR

Table 19-12 DMA_MEM_END_ADDR register

DMA_MEM_END_ADDR **Memory end address register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MEM_END_ADDR[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEM_END_ADDR [15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 MEM_END_ADDR	**MEM_END_ADDR** After handling the data at MEM_END_ADDR-0x01 address, the DMA master returns to the MEM_START_ADDR address to continue the handling.

19.6.10 DMA_PERIPH_ADDR

Table 19-13 DMA_PERIPH_ADDR register

DMA_PERIPH_ADDR **Channel peripheral address** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PERIPH_ADDR[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PERIPH_ADDR[15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 PERIPH_ADDR	**PERIPH_ADDR** Peripheral address

19.6.11 DMA_CHAN_ENABLE

Table 19-14 DMA_CHAN_ENABLE register

DMA_CHAN_ENABLE **Channel enable register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																CHAN_ENABLE
Type																RW
Reset																0

Bits	Description
0 CHAN_ENABLE	**Channel enable** 0: disable channel 1: enable channel In non channel circular mode, SW set CHAN_ENABLE to 1, and when transfer finish, HW clear CHAN_ENABLE to 0. In channel circular mode, SW set CHAN_ENABLE to 1, when transfer finish, HW restart new transfer with the same configuration. If you need to disable the DMA channel, you must disable the DMA channel circular mode first .

19.6.12 DMA_DATA_TRANS_NUM

Table 19-15 DMA_DATA_TRANS_NUM register

DMA_DATA_TRANS_NUM **Data transfer number register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA_TRANS_NUM															
Type	R															
Reset	0															

Bits	Description
15: 0 DATA_TRANS_NUM	**DATA_TRANS_NUM** Indicates how many numbers has been transferred.

19.6.13 DMA_INTER_FIFO_DATA_LEFT_NUM

Table 19-16 DMA_INTER_FIFO_DATA_LEFT_NUM register

DMA_INTER_FIFO_DATA_LEFT_NUM **Data left in inter fifo register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											INTER_FIFO_DATA_LEFT_NUM						
Type											R					
Reset											0					

Bits	Description
5: 0 INTER_FIFO_DATA_LEFT_NUM	**INTER_FIFO_DATA_LEFT_NUM** Indicates how many byte data left in inter fifo. usually used it together with FLUSH. When Time out, and INTER_FIFO_DATA_LEFT_NUM not equal to 0,then SW can start flush process.

20 WDG

20.1 Introduction

The Watchdog Timer (WDG) module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDG module is not serviced (refreshed) within a certain period, it resets the MCU. This mechanism is often used in occasions with high security requirements.

20.2 Features

- Four clock source inputs.
- Programmable timeout period.
- Window mode option for the refresh mechanism.
- Optional timeout interrupt to allow post-processing diagnostics.

20.3 Block diagram

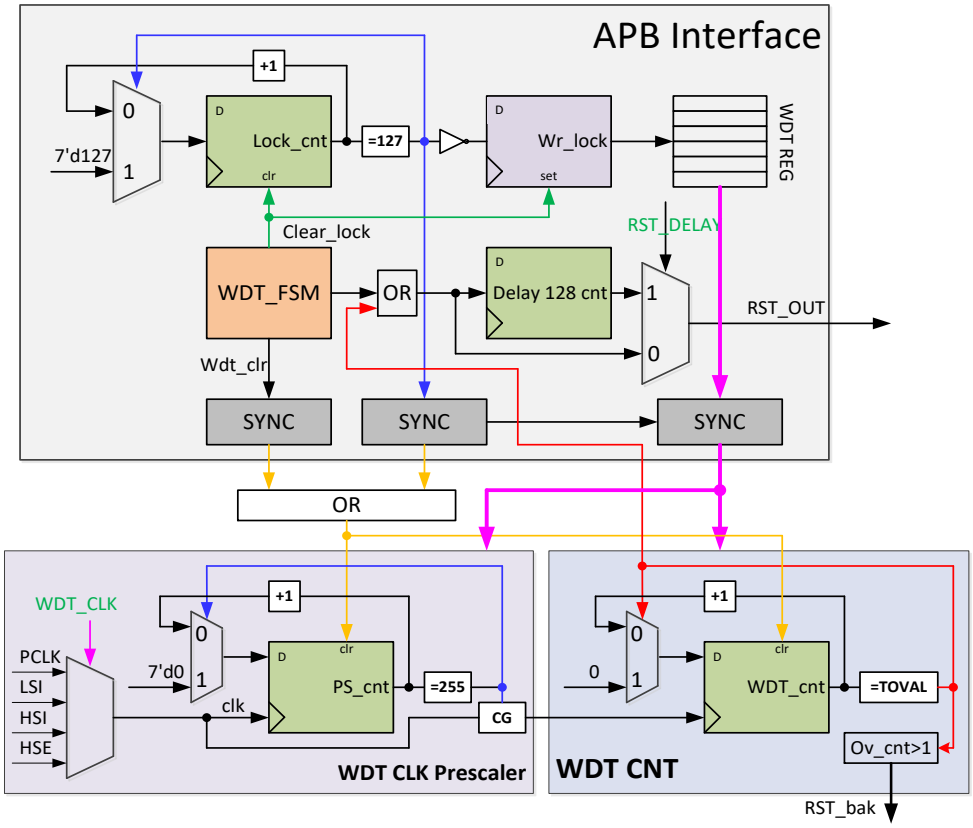

Figure 20-1 WDG block diagram

20.4 Functional description

20.4.1 Basic Watchdog

The watchdog has four clock sources: bus clock, internal 32 kHz RC oscillator, internal 8 MHz RC oscillator and external clock source. The watchdog timer uses a 32-bit programmable up counter and an optional fixed 256 clock prescaler.

After the watchdog is enabled, it starts counting. When the counting reaches the TOVAL value, a system reset will occur. Before the counting reaches the TOVAL value, refresh the watchdog to reset the counter and restart counting.

20.4.2 Window Watchdog

The watchdog has a window mode. In this mode, refreshing the watchdog before the counter reaches the window value WIN or not refreshing the watchdog before the calculator reaches the TOVAL value will cause the system to reset. When the counter is greater than the WIN value and less than TOVAL, refreshing the watchdog resets the counter and restarts counting.

20.4.3 Low-power behavior

In the Stop mode, the watchdog can keep running, but it needs to use the internal 32 kHz RC oscillator as the clock source, and the timeout at this time is twice as the normal. In Standby mode, the watchdog module will not work.

20.5 Application note

20.5.1 Configuring the Watchdog

The condition for configuring all registers of the watchdog is that the update bit WDG_CS0[UPDATE] is 1 and the watchdog is unlocked. After unlocking, any register of the watchdog can only be configured within 128 bus clocks, and then all registers are automatically locked, and for reconfiguration, it is needed to meet the above two conditions again.

When WDG_CS0[UPDATE] is 0, unlocking will cause the system reset.

Unlock sequence: write 0xE064D987 and 0x868A8478 to WDG_CNT register successively. If the written value is incorrect or the order is reversed, it will result in system reset.

20.5.2 Watchdog refresh mechanism

In basic watchdog or window watchdog mode, in order to ensure that the watchdog does not reset the system, it is needed for the software to refresh the watchdog within the specified time. After the watchdog is refreshed, the watchdog counter starts counting from 0 again, and the software needs to refresh the watchdog again. This mechanism makes the software must refresh the watchdog regularly, which reflects the normal operation of the program to a certain extent. When the program runs away unexpectedly, the watchdog will reset the system.

Refresh sequence: write 0x7908AD15 and 0x5AD5A879 to WDG_CNT register successively. If the written value is incorrect or the order is reversed, it will result in system reset.

20.5.3 Watchdog interrupt

The watchdog has an interrupt function. After the interrupt is enabled, if the watchdog counter times out, the watchdog will not reset the system immediately, but it will reset the system after a delay of 128 bus clocks. The watchdog delays forcing a reset for 128 bus clocks to allow the interrupt service routine (ISR) to perform tasks. The user software can perform simple program processing in the ISR, but it is not recommended to refresh the watchdog at this time.

20.6 Register definition

Table 20-1 WDG register mapping

WDG base address: 0x4000b000

Address	Name	Width (in bit)	Description
WDG base address +0x00	WDG_CS0	32	Watchdog Status CS0
WDG base address +0x04	WDG_CS1	32	Watchdog Status CS1
WDG base address +0x08	WDG_CNT	32	Watchdog Counter
WDG base address +0x0C	WDG_TOVAL	32	Watchdog Timeout Value
WDG base address +0x10	WDG_WIN	32	Watchdog Window

20.6.1 WDG_CS0

Table 20-2 WDG_CS0 register

WDG_CS0 **CS0** **Reset: 0x00000020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									EN	INT	UPDATE					
Type									RW	RW	RW					
Reset									0/1	0	1					

Bits	Description
7 EN	**Watchdog Enable** 0: Watchdog disabled. 1: Watchdog enabled. This write-once bit enables the watchdog counter to start counting. If the option byte is turned off, the enable bit is 0 at default. If the option byte is turned on, the enable bit is 1 at default.
6 INT	**Watchdog Interrupt** 0: Watchdog interrupts are disabled. Watchdog resets are not delayed. 1: Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks. This write-once bit configures the watchdog to generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), prior to forcing a reset. After the interrupt vector fetch, the reset occurs after a delay of 128 bus clocks.

Bits	Description
5 UPDATE	**Watchdog configuration update bit** 0: Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset. 1: Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence.

20.6.2 WDG_CS1

Table 20-3 WDG_CS1 register

WDG_CS1 **CS1** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									WIN	FLG		PRES			CLK	
Type									RW	W1C		RW			RW	
Reset									0	0		0			0	

Bits	Description
7 WIN	**Watchdog Window** 0: Window mode disabled. 1: Window mode enabled. This write-once bit enables window mode.
6 FLG	**Watchdog Interrupt Flag** 0: No interrupt occurred. 1: An interrupt occurred. This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it.
5 PRES	**Watchdog Prescalar** 0: 256 prescalar disabled. 1: 256 prescalar enabled. This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock.
1: 0 CLK	**Watchdog Clock** 00: Bus clock. 01: 32 kHz internal low-power RC oscillator (LPOSC).

Bits	Description
	10: 8 MHz internal RC oscillator (LFOSC). 11: External clock source(XOSC). Select the WDG counting clock source

20.6.3 WDG_CNT

Table 20-4 WDG_CNT register

WDG_CNT **Counter register** **Reset: 0x00000000**

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

Bits	Description
31: 16 CNT	**Watchdog Counter Register** The watchdog counter registers provide access to the value of the free running watchdog counter. Software can read the counter registers at any time. Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions: the refresh sequence and the unlock sequence.

20.6.4 WDG_TOVAL

Table 20-5 WDG_TOVAL register

WDG_TOVAL **Timeout value register** **Reset: 0x001F8000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TOVAL[31: 16]															
Type	RW															
Reset	0x1F															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TOVAL[15: 0]															
Type	RW															
Reset	0x8000															

Bits	Description
31: 0 TOVAL	**Watchdog Timeout Value Register** The watchdog counter is continuously compared with the timeout value. If the counter reaches the timeout value, the watchdog forces a reset. The counter length is equivalent to WDG_TOVAL+1, the default value is 0x1F8000.

20.6.5 WDG_WIN

Table 20-6 WDG_WIN register

WDG_WIN **Window register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	WIN[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WIN[15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 WIN	**Watchdog Window Register** When window mode is enabled (WDG_CS2[WIN] is set), WDG_WIN determines the earliest time that a refresh sequence is considered valid.

21 RTC

21.1 Introduction

Real time counter module (RTC), the main function is real-time counting. In standby low power consumption mode, RTC can keep running and wakeup the MCU.

21.2 Features

Features of the RTC module include:

- 32-bit up-counter.
- Programmable 20 bit prescaler.
- Count overflow to flip GPIO.

21.3 Block diagram

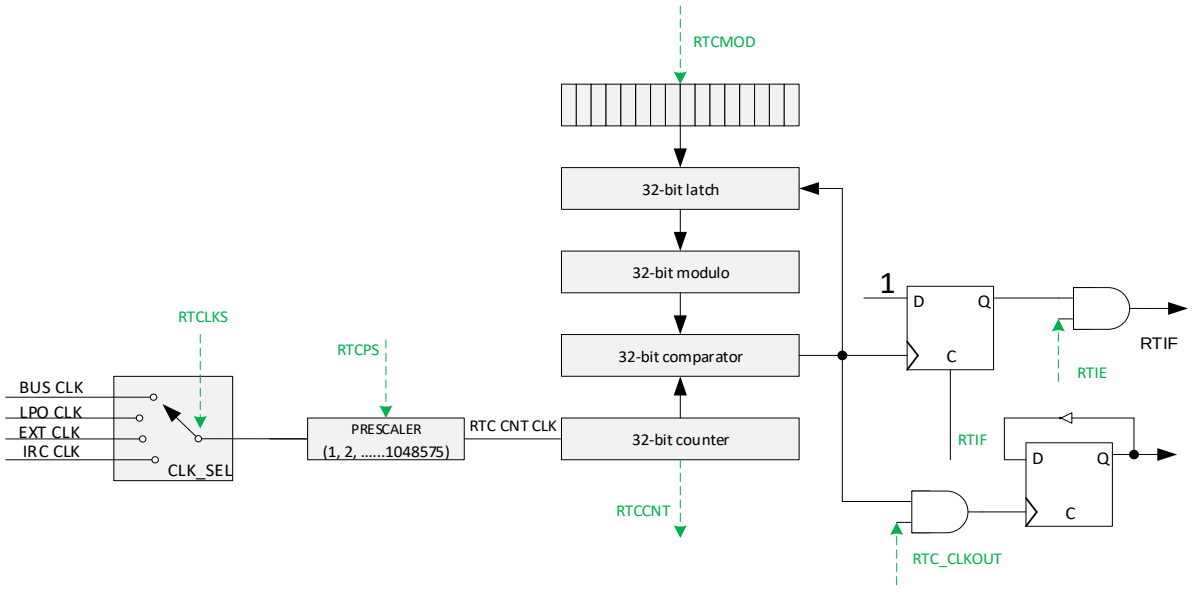

Figure 21-1 RTC block diagram

21.4 Functional description

21.4.1 Clock source selection

The RTC modules has four clock sources: bus clock, internal 32 kHz RC oscillator, external XOSC clock and external RTC_CLKIN pin input clock source.

21.4.2 Counting

RTC is an upwardly increasing counter. After the count reaches the preset modulus value, it will generate the RTC overflow flag. If the RTC overflow interrupt is enabled, RTC interrupt request is generated, and the RTC counter will start a new count from 0 after overflow. The RTC also has a built-in prescaler that upwardly counts. When it reaches the preset prescaler value, the prescaler overflow flag is generated. If the prescaler interrupt is enabled, prescaler interrupt request is generated. After the prescaler count overflows, it will start a new count from 0.

21.4.3 RTC timing signal output

PA13 can be used as RTC_CLKOUT function, which can flip PA13 when RTC overflow.

21.4.4 Low power wake up

In the low-power Stop and Standby mode of the MCU, the RTC can be used as a wake-up source to wake up the MCU. It is necessary to select the internal 32K as the clock source and start the module before entering the low-power mode. In Stop mode, an RTC interrupt is generated after the RTC overflows to wake up the MCU. In Standby mode, the MCU will be waken up directly after the RTC overflows. After waking up from the low power mode, the RTC counter will continue counting.

21.5 Application note

21.5.1 Basic use of RTC

Before using the RTC module, configure the RTC_SC register, initialize the clock, modulus, interrupt, etc. of the RTC module. When the prescaler is configured to a non-zero value finally, the RTC counter and the prescaler counter start counting.

When the RTC prescaler counter overflows, the RTC counter increase and the flag bit RPIF is set, write 1 to clear the flag. When the RTC counter overflows, the flag bit RTIF is set, and write 1 to clear the flag. When the counter is running, modification to RTC_CLK or RTC_PS will clear the counter.

When the RTCO bit is 1, enable the RTC counter overflow to flip the designated GPIO, and the GPIO needs to be multiplexed as the RTC_CLKOUT function.

21.5.2 RTC low power wake up

To use the RTC to wake up the MCU in the low-power Stop or Standby state, it is needed to enable the RTC wake-up in the SPM module, and select the internal 32K clock as the RTC clock source and start timing.

21.6 Register definition

Table 21-1 RTC register mapping

RTC base address: 0x40008400

Address	Name	Width (in bit)	Description
RTC base address + 0x00	RTC_SC	32	RTC Control and Status
RTC base address + 0x04	RTC_MOD	32	RTC Modulo
RTC base address + 0x08	RTC_CNT	32	RTC Count
RTC base address + 0x0C	RTC_PS	32	RTC Clock Prescaler
RTC base address + 0x10	RTC_PSCNT	32	RTC Prescaler Counter

21.6.1 RTC_SC

Table 21-2 RTC_SC register

RTC_SC **Control and Status register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name															RPIF	RPIE
Type															W1C	RW
Reset															0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTCLKS								RTIF	RTIE		RTCO				
Type	RW								W1C	RW		RW				
Reset	0								0	0		0				

Bits	Description
17 RPIF	**Real-Time Prescaler Interrupt Flag** 0: RTC Prescaler counter has not reached the value in the RTC prescaler divider register. 1: RTC Prescaler counter has reached the value in the RTC prescaler divider register. This status bit indicates the RTC Prescaler counter register reached the value in the RTC prescaler divider register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RPIF to 0.
16 RPIE	**Real-Time Prescaler Interrupt Enable** 0: Real-time prescaler interrupt requests are disabled. 1: Real-time prescaler interrupt requests are enabled. This read/write bit enables real-time prescaler interrupts. If RPIE is set, then an interrupt is generated when RPIF is set. Reset clears RPIE to 0.

Bits	Description
15: 14 RTCLKS	**Real-Time Clock Source Select** 00: Bus clock. 01: Internal 32kHz oscillator (LPOCLK). 10: External oscillator (XOSC). 11: External RTC_CLKIN pin input clock. This read/write field selects the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. Reset clears RTCLKS to 00. Freq = RTCLKS / ((MOD +1) * (RTCPS+1))
7 RTIF	**Real-Time Interrupt Flag** 0: RTC counter has not reached the value in the RTC modulo register. 1: RTC counter has reached the value in the RTC modulo register. This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF to 0.
6 RTIE	**Real-Time Interrupt Enable** 0: Real-time interrupt requests are disabled. 1: Real-time interrupt requests are enabled. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE to 0.
4 RTCO	**Real-Time Counter Output** 0: Real-time counter output disabled. 1: Real-time counter output enabled. The read/write bit enables real-time to toggle output on pinout. If this bit is set, the RTC_CLKOUT pinout will be toggled when RTC counter overflows.

21.6.2 RTC_MOD

Table 21-3 RTC_MOD register

RTC_MOD **RTC Modulo** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MOD[31: 16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MOD[15: 0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 MOD	**RTC Modulo** The modulo value is used to compare with the current count value (RTC_CNT). When the count value is equal to the modulo, the count will be reset to 0x0 and set SC[RTIF].

21.6.3 RTC_CNT

Table 21-4 RTC_CNT register

RTC_CNT **RTC count register** **Reset: 0x00000000**

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

Bits	Description
31: 0 CNT	**RTC count** This read-only field contains the current value of the 32-bit counter. Writes have no effect to this register. Reset or writing different values to SC[RTCLKS] and SC[RTCPS] clear the count to 0x0.

21.6.4 RTC_PS

Table 21-5 RTC_PS register

RTC_PS **Prescaler** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													RTCPS[19: 16]			
Type													RW			
Reset													0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTCPS[15: 0]															
Type	RW															
Reset	0															

Bits	Description
19: 0 RTCPS	**RTC Prescaler Select** Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS to 0. When RTCPS equals 0, RTC counter Off. When RTCPS does not equal 0, RTC counter On.

21.6.5 RTC_PSCNT

Table 21-6 RTC_PSCNT register

RTC_PSCNT **Prescaler counter register** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													PSCNT[19: 16]			
Type													RW			
Reset													0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSCNT[15: 0]															
Type	RW															
Reset	0															

Bits	Description
19: 0 PSCNT	**RTC Prescaler Counter** This read-only field contains the current value of the 20-bit counter. Writes have no effect to this register. Reset or writing different values to SC[RTCLKS] and RTC_PS[RTCPS] clear the count to 0x0.

22 Embedded Flash

22.1 Introduction

This chapter introduces the embedded flash controller, which acts as a bridge between the Cortex™-M0+ and the flash memory. In practical application, flash boot mode is the main mode and the user code will be stored in it.

The embedded Flash is hereinafter referred to as eflash.

22.2 Features

- Flash memory
  - Up to 128K bytes.
  - Endurance: ≥ 10000 cycles.
  - Page capacity: 2048 bytes per page.
- Flash controller
  - Operation list
    - Erase: Page Erase, Mass Erase, Option byte page Erase.
    - Program: Page Program, Option byte page Program, the minimum programming bit width is 32bit, and the programming address needs to be aligned with 4 bytes.
    - Read: Read data according to 8bit/16bit/32bit width.
    - Verify: Page Erase Verify, Mass Erase Verify.
  - Contain pre-fetch buffer.

22.3 Block diagram

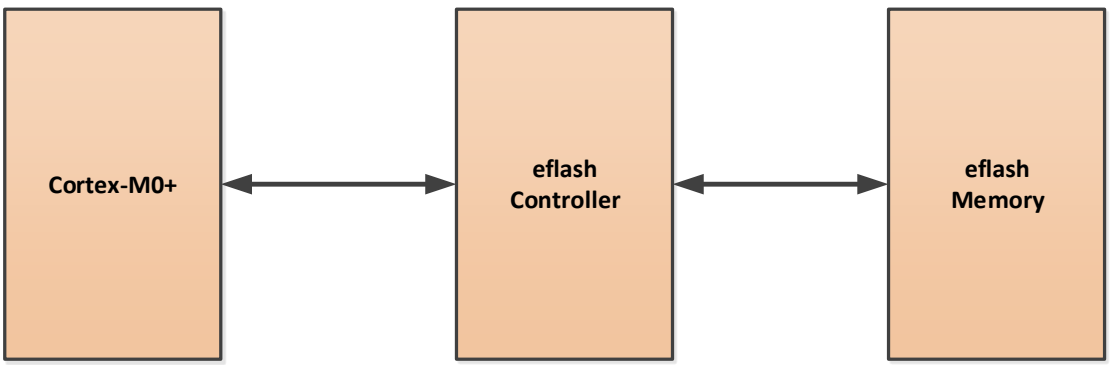

Figure 22-1 Block diagram for eflash and eflash controller

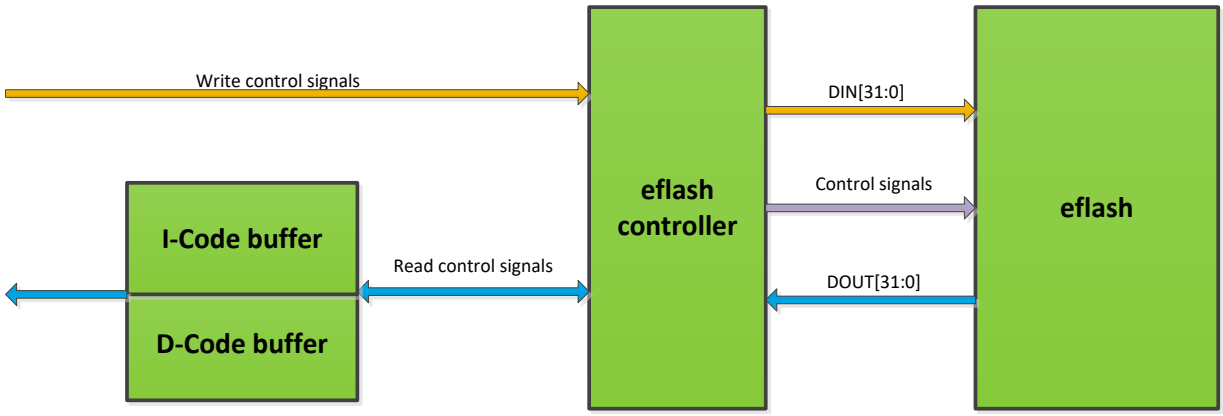

Figure 22-2 Data flow for eflash and eflash controller

22.4 Functional description

22.4.1 Embedded flash memory organization

Before introducing the commands, user should learn about the flash memory organization in Table 22-1. The whole flash memory is composed of two parts: one is main memory and the other is information memory. Every 2Kbytes are called one page in the flash memory in Table 22-1.

- Main memory is used for storing the user code including code and data, and user can put erase, program, verify and read command operations on main memory.
- The information memory is divided into two parts: ISP Firmware and Option byte.
  - The ISP Firmware section is used for solidify the code of semiconductor producer and user can't erase and program.
  - The first 24 bytes in Option byte section are the writing and reading protection information to the main memory. For the 8 bytes (from the 41st byte to the 48th byte), it is used to save user special data. For option byte section, user can erase, program, verify and read.

Table 22-1 Embedded Flash memory organization

FLASH Memory	Name	Address	Size (bytes)	User permission (note)
Main memory (user pages)	Page 0	0x0800 0000 ~ 0x0800 07FF	2K	Erase/program/read/verify
	Page 1	0x0800 0800 ~ 0x0800 0FFF	2K	
	Page 2	0x0800 1000 ~ 0x0800 17FF	2K	
	Page 3	0x0800 1800 ~ 0x0800 1FFF	2K	
	…		2K	
	Page 63	0x0801 F800 ~ 0x0801 FFFF	2K	
Information memory	Option byte	0x0804 0000 ~ 0x0804 002F	48	
		0x0804 0030 ~ 0x0804 07FF (Reserved)	2000	
	ISP Code	0x0804 0800 ~ 0x0804 1FFF	6K	Read

22.4.2 Embedded flash protect

The main contents saved in the options byte page are read protection, write protection, watchdog default working state, etc. In order to avoid illegal access to eflash, the controller has the function of protecting the writing and reading of the main memory. The related information is stored in the following option bytes and writing protection information also be loaded to the register EFLASH_WPRT_EN0 ~ EFLASH_WPRT_EN1**.** After modifying the content in the option byte, it will take effect after reset or power-on again .Specially, the complement codes(such as nRDP / nWDGEN / nWPRT_EN / nDATAx) are implemented by the hardware automatically.

When the write protection is effective, erase (page erase / block erase) and programming operations area of the write protected page are not supported, but the normal reading of eflash data is not affected.

When the disable write protection takes effect, the corresponding eflash area can be erased and programmed correctly.

After the read protection takes effect, the eflash main storage area data cannot be correctly read through JTAG / SWD. When the disabled read protection takes effect, the eflash main storage area data will be erased.

Both enable and disable read-write protection are through the programming option byte address, and both take effect after reset. Refer to the following for detailed settings.

Table 22-2 The content of the key addresses in option byte page

Address	[31: 24]	[23: 16]	[15: 8]	[7: 0]	Default value	Comment
0x0804 0000	0xFF	nRDP	0xFF	RDP	0xFF53 FFAC	
0x0804 0004	0xFF	nWDGEN	0xFF	WDGEN	0xFFFFFFFF	
0x0804 0008	nWPRT_EN[15: 0]		WPRT_EN[15: 0]		0xFFFFFFFF	page 15 ~ 0
0x0804 000C	nWPRT_EN[31: 16]		WPRT_EN[31: 16]		0xFFFFFFFF	page 31 ~ 16

0x0804 0010	nWPRT_EN[47: 32]	WPRT_EN[47: 32]	0xFFFFFFFF	page 47 ~ 32
0x0804 0014	nWPRT_EN[63: 48]	WPRT_EN[63: 48]	0xFFFFFFFF	page 63 ~ 48
0x0804 0028	nDATA0	DATA0	0xFFFFFFFF	user data
0x0804 002C	nDATA1	DATA1	0xFFFFFFFF	user data

22.4.2.1 Read and write protection

As shown in Table 22-2, it can be seen that the address of option byte for read protection is from 0x0804 0000 to 0x0804 0003, and the address of option byte for write protection is located is from 0x0804 0008 to 0x0804 0017. The read-write protection settings of the main memory area are shown in Table 22-3 and Table 22-4.

Table 22-3 Read protection setting

Conditions	RDP	nRDP	Read protection status
Case1	0xFF	0xFF	Protected
Case2	0xAC	0x53	Not protected
Other cases	Except case1 and case2		Protected

Table 22-4 Write protection setting

Conditions	WPRT_EN[x]	nWPRT_EN[x]	Write protection status
Case1	0	1	Protected
Other cases	Except case1		Not protected

Note

In the write protection value, one bit corresponds to one page.

22.4.2.2 Watchdog

Referring to Table 22-2, it can be seen that the address of the option byte where the watchdog default state is located is 0x0804 0004 ~ 0x0804 0007. If WDGEN is programmed to 0xCC and nWDGEN is programmed to 0x33, the watchdog is enabled by default. Otherwise, the watchdog is disabled by default.

Table 22-5 Watchdog default state setting

Conditions	WDGEN	nWDGEN	Status
case1	0xCC	0x33	Enabled watchdog by default
others	Except the above		Disable watchdog by default

22.4.2.3 User data

Referring to Table 22-2, we can see that the option byte address of user data DATAx / nDATAx is 0x0804 0028 ~ 0x0804 002F. Among them, the low 16 bits DATAx is used to store data freely, and the high 16 bits nDATAx is the user data complement, which is automatically calculated by hardware.

22.5 Application note

In this section, the introduction by flow chart will be described about Page erase, Mass erase, Page program, Option byte page erase, Option byte page program, Page erase verify and Mass erase verify. All the command operation mainly refers to the following registers: EFLASH_CTRL0, EFLASH_CTRL1, EFLASH_ADR_CMD, EFlASH_SR0. For read operation, user can directly read the desired address needed for eflash in flash memory according to the 8bit/16bit/32bit access mode, so the related description will be ignored in the document.

In particular, the following flow charts just illustrate the single command operation. You only need to unlock once before multiple command operations. For on chip flash memory, there are 68 pages in total, including 1 option byte page, 3 ISP firmware pages and 64 user pages.

The following focuses on the process of page erasure, page erasure verify and page programming. Other command operations can refer to these processes.

22.5.1 Page erase

Page erase operation just acts on main memory in eflash memory. Page erase operation can't reach the information memory. In the following paragraphs, detailed descriptions are introduced for page erase and other command operations can refer to it.

1. Before configuring the EFlASH_CTRL0 and EFlASH_CTRL1, we must check whether they are locked by observing the status of LOCK. If the register is in lock state, we must sequentially write 0xac7811 and 0x01234567 to EFlASH_KEY register to unlock. If they are not in lock state, we can go to the next step directly.
2. After unlocking, we had better check whether there are some command operations in process by reading out the status of CMD_BSY. CMD_BSY equal to 1 represents some command operations is not finished and we must wait until CMD_BSY is equal to 0. In fact, unlocking process can be done before or after checking CMD_BSY and the following chart just illustrate the "before" case.
3. When CMD_BSY turns to 0, we can configure the two registers: EFlASH_CTRL0 and EFlASH_CTRL1. For all the erase and program command operations, user must configure the CKDIV value in EFLASH_CTRL1, the configuration formula is: CKDIV=eflash controller clock frequency/1. For example, if the eflash controller clock frequency is 48MHZ, CKDIV=48/1=48=0x30, but it is recommended that the value is configured larger than 0x30, such as 0x31.

4. Then, user should configure the page erase start address in EFLASH_ADR_CMD, which value is the start address of the desired erase page.
5. After the basic configuration above, user can launch the command and trigger it to start by controlling the EFLASH_CTRL0. User should guarantee that bit CMD_ST must turn from 0 to 1 after other bits have been configured in EFLASH_CTRL0 in order to get a valid trigger. In details, EOPIE is configured to 1 to make related status occurs after the incoming command operation will be completed. CMD_CTL is configured to 0x1 to determine the incoming command operation is page erase. Other bits should be 0 and will be used in other command operations.
6. After a valid trigger, the command operation starts. User should check whether the command operation is finished by reading out the bit CMD_BSY and EOP. When CMD_BSY is equal to 0 and EOP is equal to 1, it represents that the command operation has been finished and user should clear the EOP by writing 0 to this bit. In fact, it is ok for user to just use CMD_BSY to indicate whether the operation is finished for all command operations.
7. Clear EFLASH_CTRL0.
8. Then user can read out other status to check whether there are some errors based on user's requirement, such as OPT_ERR, especially for write protection case, user can not erase the page with write protection character.

The page erase command description is finished completely as the following flow chart. To point out, the process of other command operations is similar to the page erase, just with some difference, which will be described later.

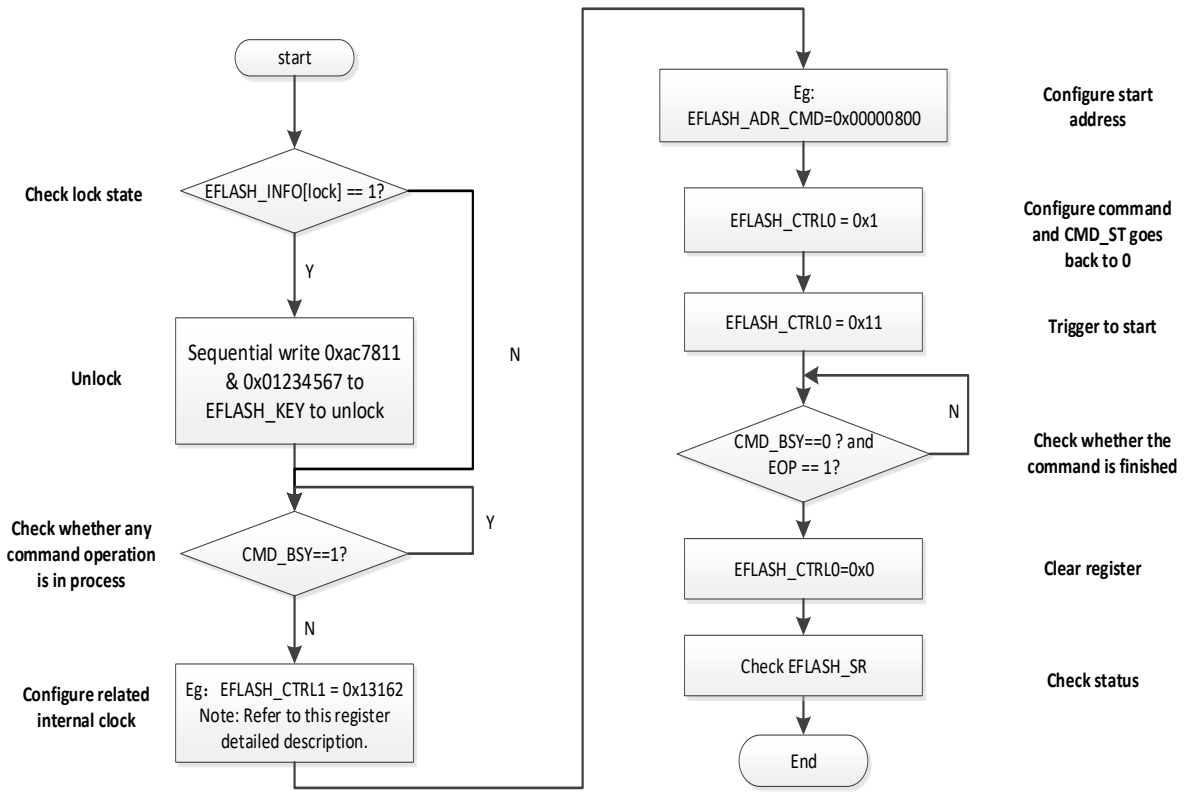

Figure 22-3 Page erase command operation flow

22.5.2 Mass erase

Mass erase can erase the whole user 64 pages eflash. The flow is illustrated in Figure 22-4.

Compared with the page erase command flow, the mass erase process has two differences: one is that EFLASH_CTRL0[CMD_CTL] bit is set to a different value, and the other is that the erase address does not need to be specified in the EFLASH_ADR_CMD register.

Specially, when the main memory attribute is changed from read-protect to read-not-protect, a mass erase will be performed automatically in order to protect the user code from illegal reading. For example, when user program the RDP in option byte page from the default 0XFF to 0xAC, a mass erase performs automatically.

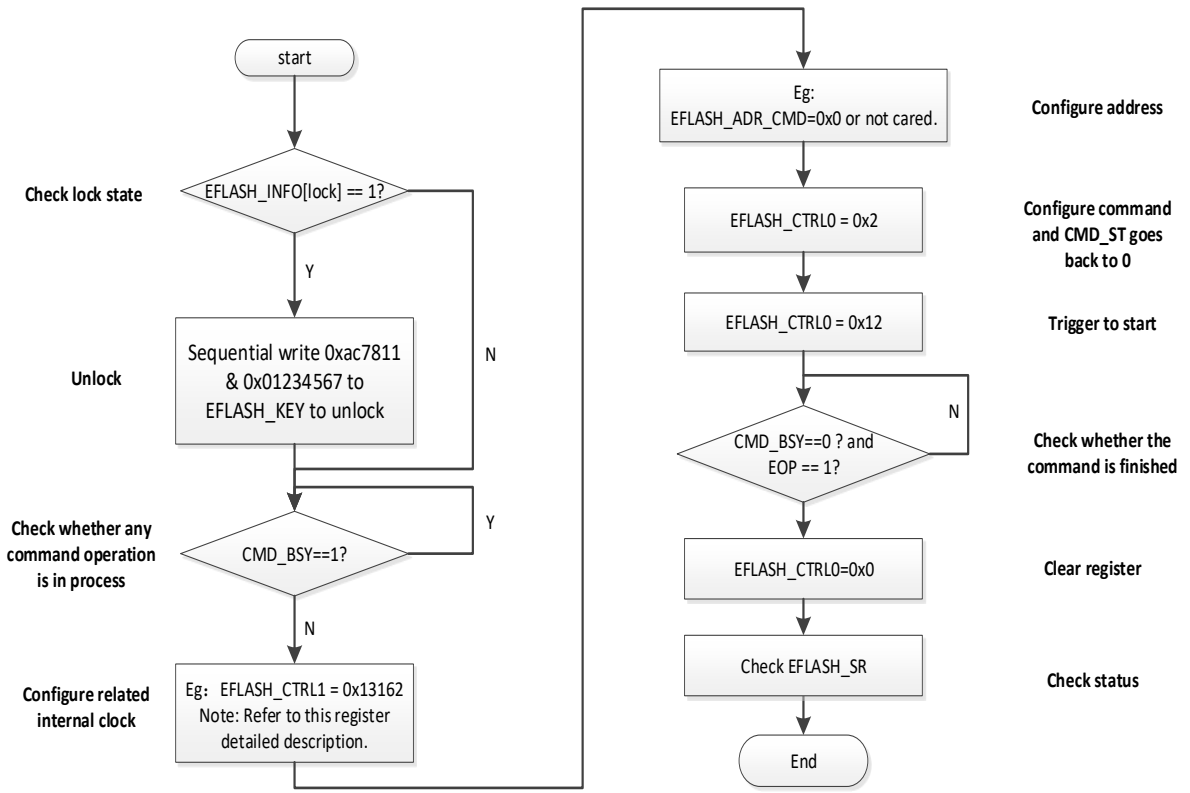

Figure 22-4 Mass erase command operation flow

22.5.3 Page program

Page program command can reach the whole user 64 pages eflash. The flow is illustrated in Figure 22-5. The page program command flow is also similar to the page erase. But there are two differences: one is CMD_CTL bits settings in the EFLASH_CTRL0 register are different. and the other is PROG_LENGTH[9:0] in the EFLAH_CTRL0 register. The bits PROG_LENGTH[9:0] are configured to a specified value to determine the program length by word. For example, if PROG_LENGTH[9:0] are configured to 150, user should write not more than 150 words. If the user write more than 150 words, such as 180 words, the last 30 words will not be written into flash in fact. Meanwhile, if the user writes less than 150 words, such as 110 words, the write operation will be validated normally, but user should keep in mind that program process will be not finished until the data number equal to length or you should use flush command to force program be terminated.

About page program command, there are some auto pre-check mechanisms to do content protection, such as write protection check, blank content check and address boundary check. So user should check EFLASH_SR0 register carefully after each program command to confirm whether the write operation is successful.

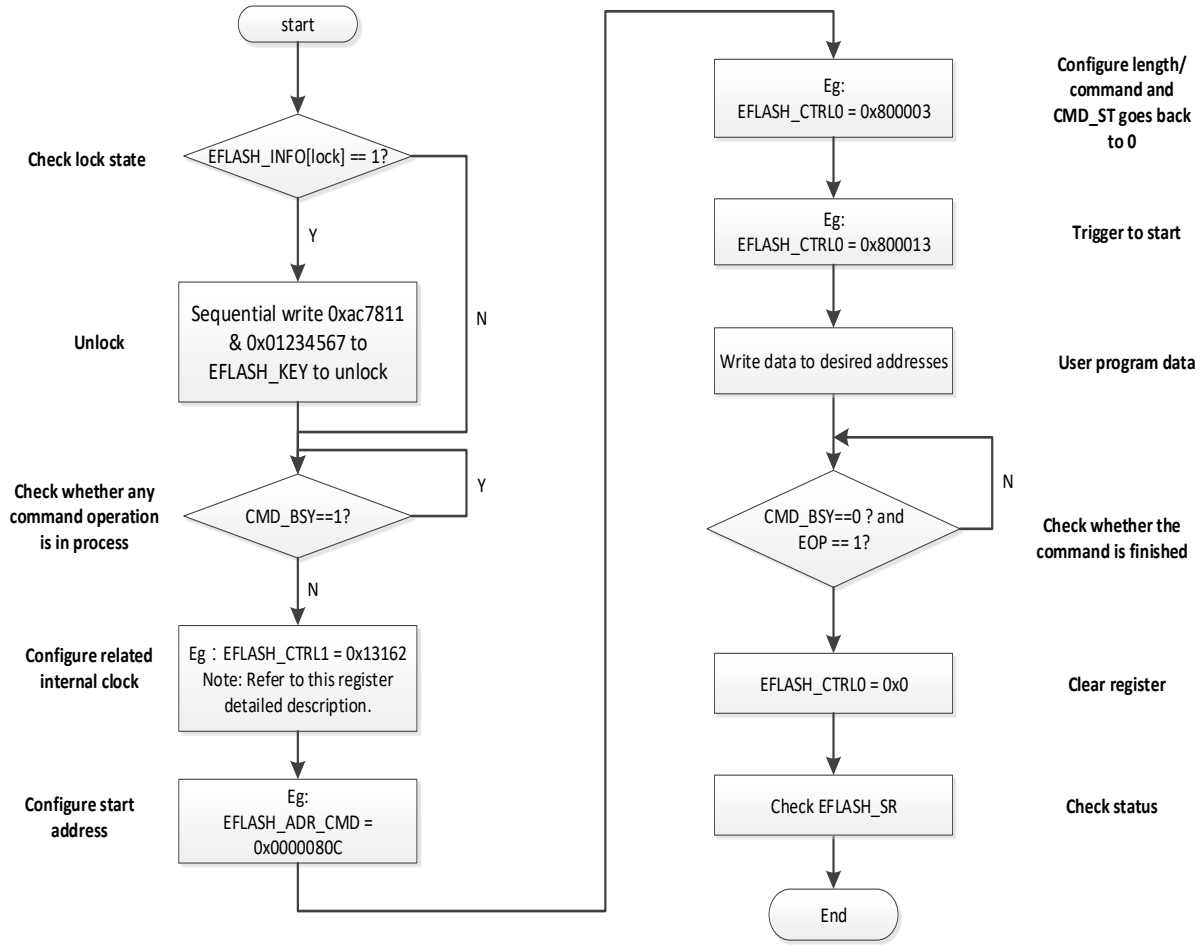

Figure 22-5 Page program command operation flow

22.5.4 Page erase verify

Page verify command can reach the whole user 64 pages eflash. This operation is usually executed after erase operation in order to verify whether the erase operation is performed successfully. The flow is illustrated in Figure 22-6. Compared with the page erase command flow, the page erase verify flow has only one difference: CMD_CTL bits settings in the EFLASH_CTRL0 register are different.

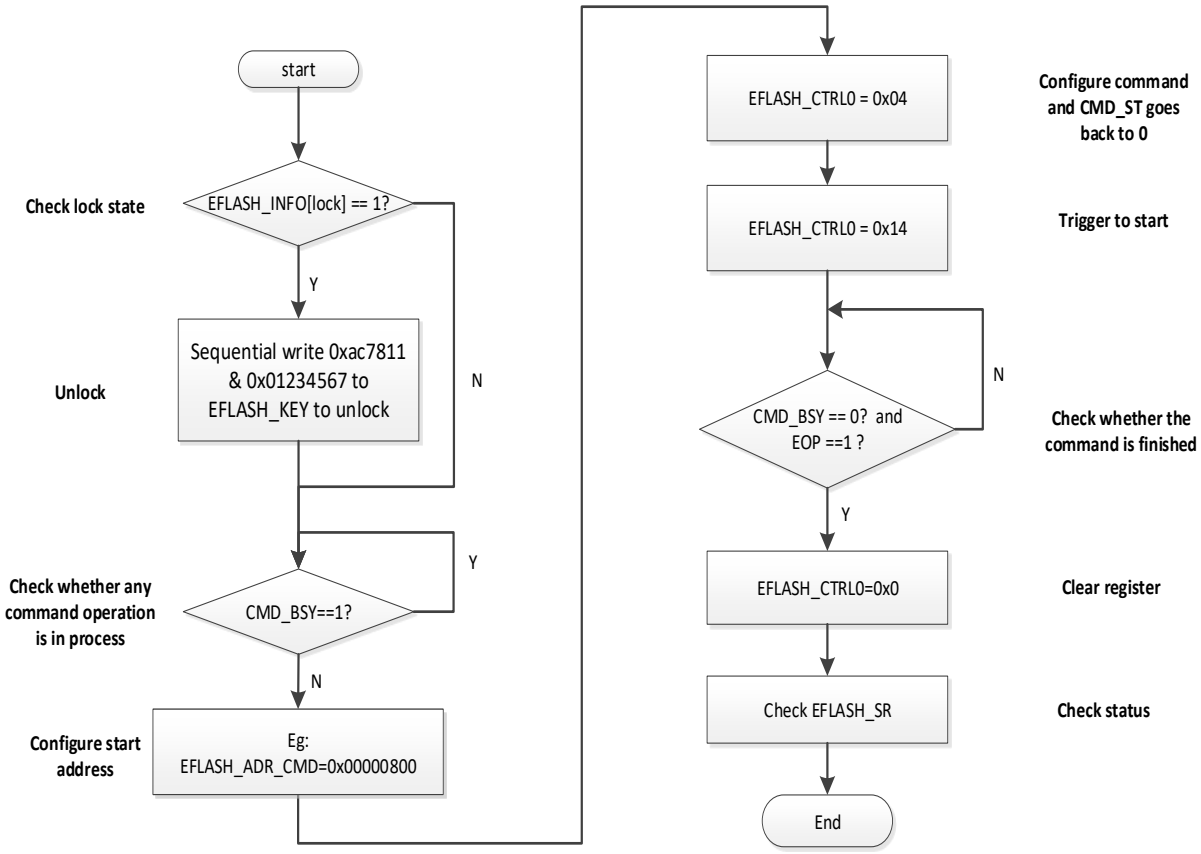

Figure 22-6 Page erase verify command operation flow

22.5.5 Mass erase verify

Mass erase verify command can reach the whole user 64 pages memory. This operation is usually executed after mass erase operation in order to verify whether the mass erase operation is finished successfully. The flow is illustrated in Figure 22-7. Compared with the page erase command flow, the whole erase verify flow has two differences: one is CMD_CTL bits settings in the EFLASH_CTRL0 register are different, and the second is that it does not need to specifies the erase address in the EFLASH_ADR_CMD register. Since the entire 64 pages of storage have been reached, there is no need to specify erase address in the EFLASH_ ADR_CMD register.

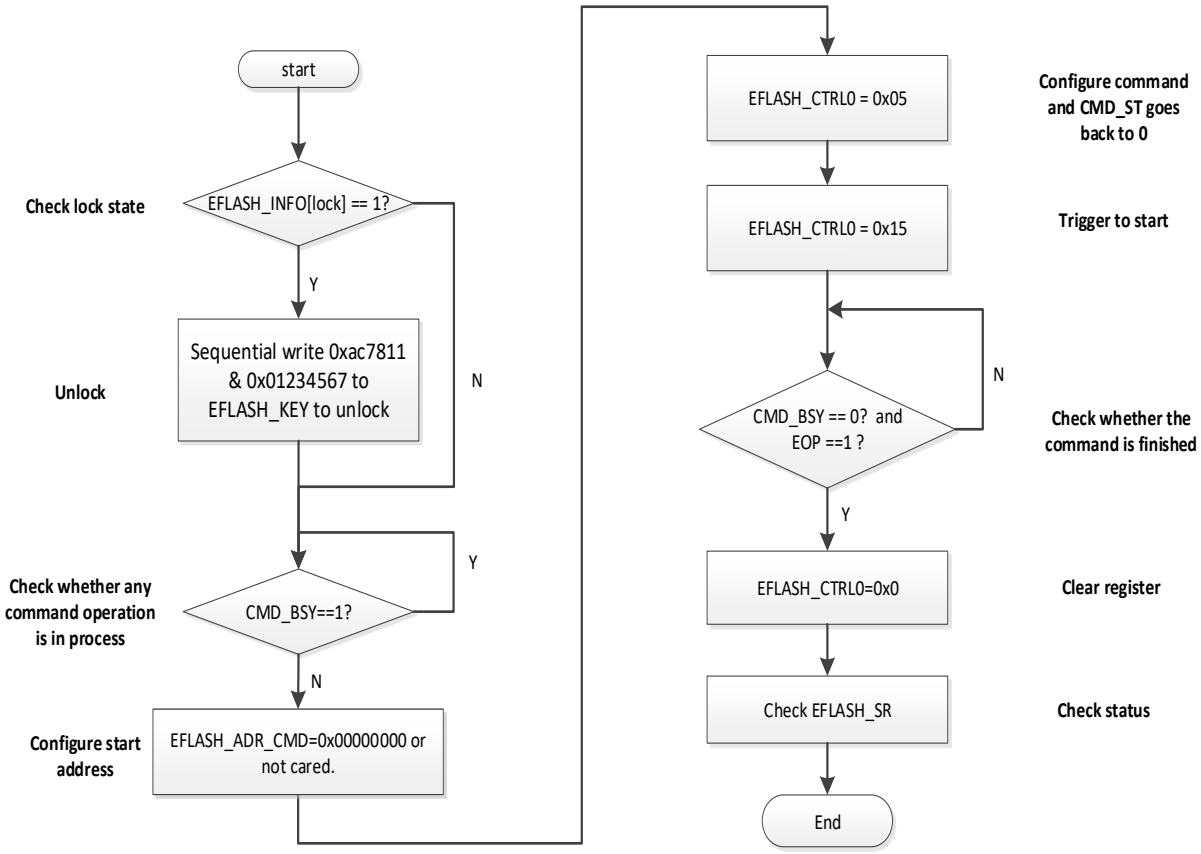

Figure 22-7 Mass erase verify command operation flow

22.5.6 Option byte page erase

Option byte page erase command aims at the one Option byte page. The flow is illustrated in Figure 22-8. Compared with the page erase command flow, there are two differences in the option byte erase flow: one is CMD_CTL bits settings in the EFLASH_CTRL0 register are different, the second is the address which specified in the EFLASH_ ADR_CMD register is the option byte address.

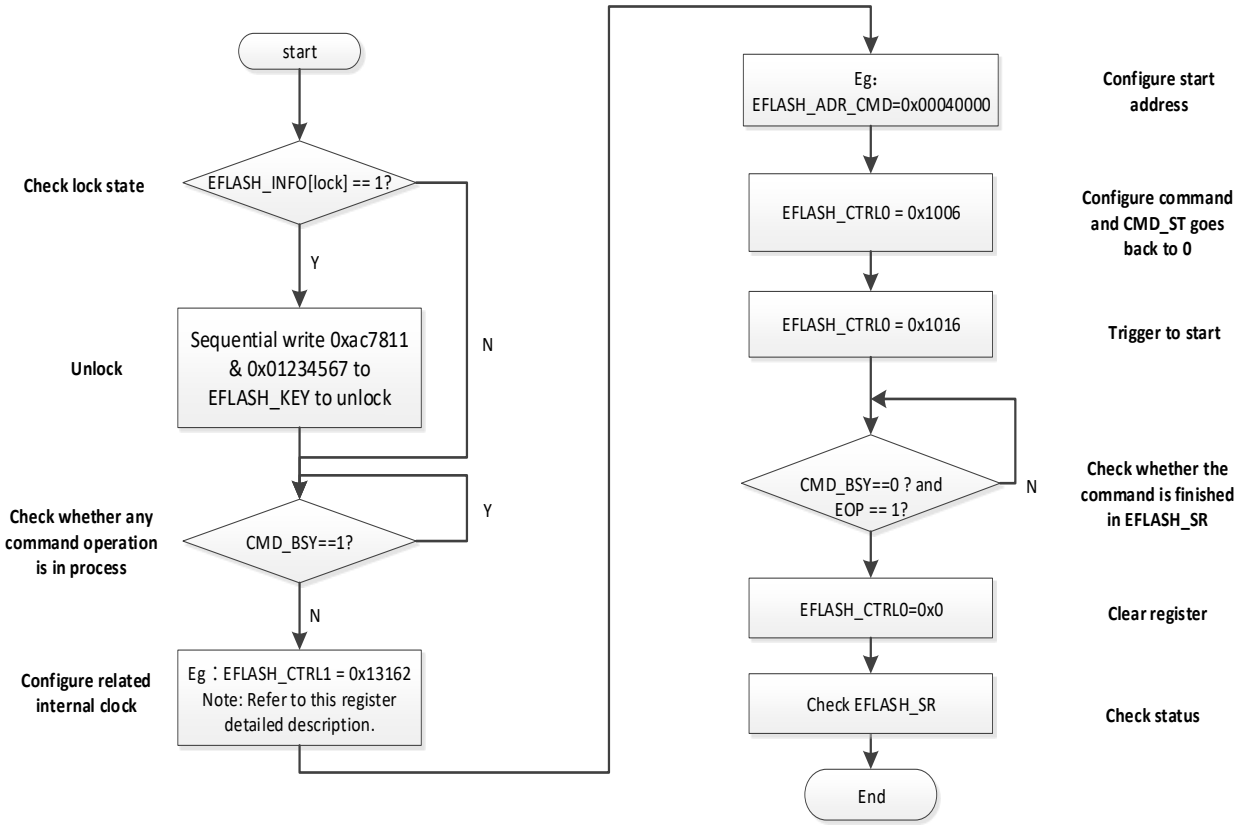

Figure 22-8 Option byte page erase command operation flow

22.5.7 Option byte page program

Option byte page erase command aims at the Option byte page. The flow is illustrated in Figure 22-9. Compared with page programming command flow, there are two differences in option byte programming flow: one is CMD_CTL bits settings in the EFLASH_CTRL0 register are different, the second is the address which specified in the EFLASH_ ADR_CMD register is the option byte address.

User should pay high attention to read protection character change when do option byte page program, if you change read protect character from protect to un-protection, eflash controller will do mass erase command automatically to erase the content of whole user pages.

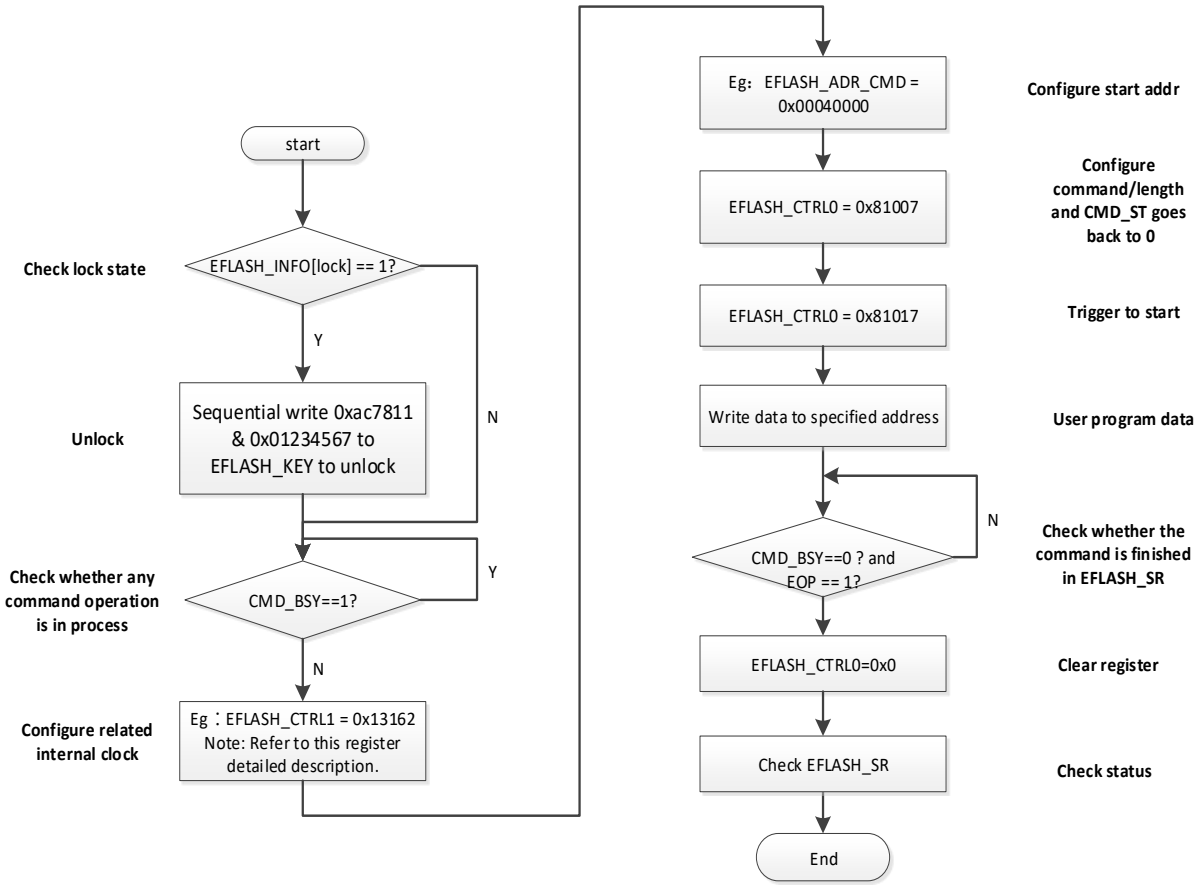

Figure 22-9 Option byte page program command operation flow

22.6 Register definition

Table 22-6 Embedded flash register map

EFLASH base address: 0x40002000

Address	Name	Width (in bit)	Description
EFLASH base address +0x0	EFLASH_KEY	32	Unlock sequence register
EFLASH base address +0x4	EFLASH_INFO	32	Protect info register
EFLASH base address +0x8	EFLASH_ADR_CMD	32	Erase start address
EFLASH base address +0xC	EFLASH_CTRL0	32	Control register 0
EFLASH base address +0x10	EFLASH_SR0	32	Status register
EFLASH base address +0x14	EFLASH_CTRL1	32	Control register 1: clock setting
EFLASH base address +0x18 ~ EFLASH base address +0x1C	EFLASH_WPRT_ENx	32	Write Protect enable bit[63:0]
EFLASH base address +0x40	EFLASH_CTRL2	32	Control register 2

22.6.1 EFLASH_KEY

Table 22-7 EFLASH_KEY register

EFLASH_KEY **Key sequence register** **RESET: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY															
Type	RW															
Reset	0															

Bits	Description
31: 0 KEY	**Unlock the writing protection of the eflash control registers** The eflash control registers are indicated to be not locked when EFLASH _INFO[LOCK] bit is 0 and is locked when EFLASH_INFO[LOCK] bit is 1. Sequential write 0xac7811 and 0x01234567 to unlock the protection, then the control registers can be written again.

22.6.2 EFLASH_INFO

Table 22-8 EFLASH_INFO register

EFLASH_INFO **Protect info register** **RESET: current protect status**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LOCK															
Type	RW															
Reset	1															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															WDG_EN	RD_PRT
Type															R	R
Reset																

Bits	Description
31 LOCK	**Embedded Flash controller lock register** 0: Embedded Flash controller register not lock, then write 1 to lock Embedded Flash controller directly. 1: Embedded Flash controller register lock, it cannot be unlocked by writing 0, only unlock by operating the EFLASH_KEY registers.

Bits	Description
1 WDG_EN	**WDG enable status in option byte.** 0: WDG disabled. 1: WDG enable. **Note: the reset value of this bit depends on the watchdog enable status in the option byte.**
0 RD_PRT	**Embedded Flash read protection status.** 0: read protection disabled. 1: read protection enable. **Note: the reset value of this bit depends on the current read protection enable state.**

22.6.3 EFLASH_ADR_CMD

Table 22-9 EFLASH_ADR_CMD register

EFLASH_ADR_CMD **Erase/program start address** **RESET: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ADR_CMD															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ADR_CMD															
Type	RW															
Reset	0															

Bits	Description
31: 0 ADR_CMD	**Start address for commands** (1) For page erase ADR_CMD[19:0] is the low 20 bits of the eflash address, which is just among this page. For example, if user want to erase page 63 which range from 0x0801 F000 to 0x0801 F7FF, so, user can configure ADR_CMD[31:0] as any value from 0x0001 F000 to 0x0001 F7FF. (2) For program ADR_CMD[19:0] is the program start address, which is the low 20 bits of eflash address. ADR_CMD[19:0] and PROG_LENGTH[9:0] are together to determine the programmable address range. (3) For page verify ADR_CMD[19:0] is the low 20 bits of the first eflash address of this page. (4) For mass erase/verify ADR_CMD[31:0] is not cared.

Bits	Description
	Note: Before erase/program/verify, the start address must be set. ADR_CMD[31:20] must be 12'b0.

22.6.4 EFLASH_CTRL0

Table 22-10 EFLASH_CTRL0 register

EFLASH_CTRL0 **Control register 0** **RESET: 0x0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	HDFEN						PROG_LENGTH									
Type	RW						RW									
Reset	0						0									
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				OPT_CMD_EN								CMD_ST	CMD_CTL			
Type				RW								RW	RW			
Reset				0								0	0			

Bits	Description
31 HDFEN	**Whether a hard fault interrupt generated when a read-write protection rule is violated** 0: disable, hard fault interrupt will not generate when read-write protection rules are violated. 1: enable, hard fault interrupt will generate when read-write protection rules are violated.
25: 16 PROG_LENGTH	**Program length for Program operation** Program length. **Note: the unit is word.**
12 OPT_CMD_EN	**Enable the command operation related with the option bytes' zone** 0: Register CMD_ The CTL values of 0110 or 0111 are invalid, and the values of 0001, 0010 or 0011 are valid 1: Register CMD_ CTL values of 0110 or 0111 are valid, while values of 0001, 0010 or 0011 are invalid **Note: refer to CMD_CTL.**
4 CMD_ST	**Control the command operation to start** Write 1 to trigger the start command and clear the error status bit [11:2] of EFLASH_SR0 register
3: 0 CMD_CTL	**Command**

Bits	Description
	0000: Idle 0001: Page Erase 0010: Mass Erase 0011: Page Program 0100: Page Erase Verify 0101: Mass Erase Verify 0110: Option Byte Page Erase 0111: Option Byte Page Program **Note: Please refer to the Configuration of OPT_CMD_EN for the validity of CMD_CTL value. When CMD_ When the value of CMD_CTL is 0100 or 0101, it has nothing to do with the configuration of OPT_CMD_EN.**

22.6.5 EFLASH_SR0

Table 22-11 EFLASH_SR0 register

EFLASH_SR0 **Status register** **RESET: 0x00800002**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				FLUSH	OPTER				VRER	ERAER	PPADRER	PPERER	RDER	WRER	EOP	CMD_BSY
Type				RW	R				R	R	R	R	W1C	W1C	W1C	R
Reset				0	0				0	0	0	0	0	0	1	0

Note

Writing 1 to the EFLASH_ CTRL0[CMD_ST] bit can clear the error status bit [11:2] of the EFLASH_SR0 register.

Bits	Description
12 FLUSH	**Write 1 to fore the program command operation to be finished** **Note: This bit will not be used when user perform program command operation obeying this application note. But in some cases, the remaining memory doesn't match the value of PROG_LENGTH[9:0] or the actual program number of memory is less than PROG_LENGTH[9:0], So, the program can't be finished. For this condition, write 1 to FLUSH will force the program command operation to be finished.**
11: 8 OPTER	**Indicate the error status for option byte** OPTER [3]: 0: no error, namely, DATAx=~nDATAx 1: exist error, namely, DATAx != ~nDATAx, except DATAx = nDATAx=0xff

Bits	Description
	OPTER [2]: 0: no error, namely, WPRT_EN[x] = ~nWPRT_EN[x] 1: exist error, namely, WPRT_EN[x] != ~nWPRT_EN[x], except WPRT_EN[x] = ~nWPRT_EN[x] = 0x1 OPTER [1]: 0: no error, namely, WDGEN = ~nWDGEN 1: exist error, namely, WDGEN != ~nWDGEN, except WDGEN= nWDGEN=0xff OPTER [0]: 0: no error, namely, RDP = ~nRDP 1: exist error, namely, RDP != ~nRDP, except RDP = nRDP=0xff
7 VRER	**Indicate whether there is error in the verify command operation** 0: data verify is ok 1: data verify is not ok
6 ERAER	**Indicate whether there is error in the erase command operation** 0: no error 1: error occurs, because address in EFLASH_ADR_CMD is illegal
5 PPADRER	**Indicate whether there is error in the page program command operation.** 0: no error 1: error occurs, because program address is illegal or verify operation before program command operation is not OK.
4 PPERER	**Indicate permission error of commands operation** 0: no error 1: error occurs when page/mass erase or program acts on the protected main memory
3 RDER	**Indicate the operation violates the read protection rules** 0: no violation for the read protection rules 1: violations for the read protection rules Write 1 to clear RDER to 0.
2 WRER	**Indicate the operation violates the write protection rules** 0: no violation for the write protection rules 1: violations for the write protection rules Write 1 to clear WRER to 0.
1 EOP	**Indicate whether command operation is finished** 0: not finished

Bits	Description
	1: finished **Note: it is not significant for user.** Write 1 to clear EOP to 0.
0 CMD_BSY	**Indicate whether any of the command operations is in process** 0: all operations are not in process 1: at least one operation in process

22.6.6 EFLASH_CTRL1

Table 22-12 EFLASH_CTRL1 register

EFLASH_CTRL1 **Control register 1** **RESET:0x00002009**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		SPEED_LATENCY						CKDIV_LOCK		CKDIV						
Type		RW						RW		RW						
Reset		2						0		9						

Bits	Description
13:12 SPEED_LATENCY	**FLASH initialization reference frequency value** **Note: After the power-on system clock is initialized, it must be written as 0x2.**
8 CKDIV_LOCK	**Lock the configuration for EFLASH_CTRL1** 0: can configure the register 1: can't configure the register **Note:** if this bit is written to 1 after power on, this register EFlASH_CTRL1 can't be configured again until the chip is powered down.
6: 0 CKDIV	**Clock divisor for generating 1us pulse** Must configure to the reasonable value based on the speed of eflash controller to get 1us period before the following operation: Page program, Page erase, Mass erase based on the command operation flow. For example, if eflash controller speed is 48MHz, CKDIV= 48/1 = 48= 0x30, but it is recommended that the value is configured larger than 0x30, such as 0x31.

22.6.7 EFLASH_WPRT_ENx

Table 22-13 EFLASH_WPRT_ENx register

EFLASH_WPRT_ENx **Write Protection enable register x** **RESET:**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	WPRT_ENx															
Type	R															
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WPRT_ENx															
Type	R															
Reset																

Note

The Reset value in the above table will be changed due to the option byte register configuration.

Note

x=0~1, four EFLASH_WPRT_ENx registers compose 64 enable bits, which determine the write protection attribution of each main memory page respectively.

Bits	**Description**
31: 0 WPRT_ENx	**Indicate whether there exists the write protection for the corresponding main memory page.** 0: not in protection 1: in write protection

22.6.8 EFLASH_CTRL2

Table 22-14 EFLASH_CTRL2 register

EFLASH_CTRL2 **Control register 2** **RESET: 0x0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PVD_WRN			PVDMON_EN												
Type	RW			RW												
Reset	0			0												
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																
Type																
Reset																

Bits	**Description**
31 PVD_WRN	**Program voltage detector status** Default value:1'b0 **Note: This bit is always used as a status mark. If write, it can be only written to 0. If the PVD module is enabled and the current IC voltage is lower than the programming voltage in the PVD module, the status bit will be set to 1 before the power supply voltage exceeds the programming voltage.**
28 PVDMON_EN	**Program voltage detector, it can avoid the destruction of EFLASH due to sudden power failure during erasing or programming.** 1'b1: enable monitoring function 1'b0: disable monitoring function. Default value:1'b0.

23 SRAM ECC

23.1 Introduction

The full name of ECC_SRAM is SRAM Error Correcting Code, which is used for SRAM error detection and correction. When an SRAM error occurs, it will generally not cause the entire SRAM to be unreadable or all errors, but only one or a few bits' error in the entire SRAM. ECC_SRAM uses Hamming code ECC single-bit error correction and two-bit detection algorithms. The calculation speed is very fast. Errors above 1 bit cannot be corrected, and errors above 2 bits are not guaranteed to be detected.

23.2 Features

- Support error detection of 1bit and 2 bits.
  - 1 bit and 2 bits' error status can be detected.
  - Software can read the error address and the error status through the register.
- Support error status correction of 1bit.
  - When a 1-bit' error state is detected, the hardware will automatically correct the error
  - For 2 bits' error status, the hardware cannot correct errors.
- Support 2 bits' error status interrupt.
  - The software can be configured to generate an interrupt when ECC_SRAM detects a 2 bits' error.
  - Software can configure whether to enable system reset when ECC_SRAM detects a 2 bits' error.
- Error status of more than 2 bits cannot be guaranteed to be detected.

23.3 Functional description

1. ECC_SRAM is enabled by default and cannot be disabled.
2. ECC_SRAM will generate an interrupt if enabled ERR2_IRQEN =1 and a 2 bits' error is detected. When ECC_SRAM interrupt occurs,
   - The interrupt flag register ERR2_STATUS is set to 1.
   - The ECC_SRAM status register ERR_STATUS is set to 1.
   - The ECC_SRAM error address register ERR2_ADDR will fill with the error address automatically.

- It is needed for the software to write 1 to interrupt flag register ERR2_STATUS to clear the flag.

3. The software can query the value of the register ERR_STATUS to check the current ECC_SRAM error status.
   - If ERR_STATUS=2'b00, it indicates that ECC_SRAM has not detected an error.
   - If ERR_STATUS=2'b01, it indicates that ECC_SRAM has detected a 2 bits' error. At this time, the software obtains the address of the current 2 bits' error by reading the register ERR2_ADDR.
   - If ERR_STATUS=2'b10 or 2b'11, it indicates that ECC_SRAM has detected a 1-bit error. At this time, the software obtains the address of the current 1-bit error by reading the register ERR1_ADDR.
   - The ECC_SRAM error status register ERR_STATUS and the error address register ERR_ADDRx (x=1,2) will retain the error status and error address of the last ECC_SRAM check when the system is not reset. It will clear this error status register ERR_STATUS and the error address register ERR_ADDRx (x=1,2) by writing 2'b11 to ERR_STATUS.
   - The system will generate a reset if programed the 23th bit (ecc2_rst_en) of the register RESET_CRTL (address is 0x4000000C) to be 1 and the ECC_SRAM detects a 2 bits' error.

23.4 Register definition

Table 23-1 SRAM ECC register map

ECC_SRAM base address: 0x40000020

Address	Name	Width (in bit)	Description
ECC_SRAM base address +0x0	ECC_SRAM_CTRL	32	ECC control and status register
ECC_SRAM base address +0x4	ECC_SRAM_ERR1_ADDR	32	ECC 1 bit error address register
ECC_SRAM base address+0x8	ECC_SRAM_ERR2_ADDR	32	ECC 2 bits' error address register

23.4.1 ECC_SRAM_CTRL

Table 23-2 ECC_SRAM_CTRL register

ECC_SRAM_CTRL **ECC_SRAM ctrl and status register** **RESET: 0x00000001**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											ERR_STATUS			ERR2_STATUS	ERR2_IRQEN	
Type											W1C			W1C	RW	
Reset											0			0	0	

Bits	Description
5: 4 ERR_STATUS	**ECC_SRAM Error Status** **Note: write 2'b11 to clear these two status bits and ERRx_ADDR(x=1,2)** 2'b00: no error; 2'b01: 2 bits' error and cannot be corrected; 2'b10/2'b11: 1 bit error and can be corrected;
2 ERR2_STATUS	**ECC_SRAM 2 bits' error status** **Note: write 1 to clear this bit.** **If ERR2_IRQEN is enabled and 2 bits' error is generated, it must write 1 to clear this bit, otherwise this module will always generate 2 bits' error interrupt.**
1 ERR2_IRQEN	**ECC 2 bits' error interrupt enable** 1: enable 0: disable

23.4.2 ECC_SRAM_ERR1_ADDR

Table 23-3 ECC_SRAM_ERR1_ADDR register

ECC_SRAM_ ERR1_ADDR ECC_SRAM 1bit error address register RESET: 0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				ERR1_ADDR [12: 0]												
Type				R												
Reset				0												

Bits	Description
12: 0 ERR1_ADDR	**1-bit error address** **Note: multiply ERR1_ADDR by 4 and then plus 0x20000000 to get the SRAM address of occurring 1 bit error.**

23.4.3 ECC_SRAM_ERR2_ADDR

Table 23-4 ECC_SRAM_ERR2_ADDR register

ECC_SRAM_ERR2_ADDR ECC_SRAM 2 bits' error address register RESET: 0x00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				ERR2_ADDR [12: 0]												
Type				R												
Reset				0												

Bits	Description
12: 0 ERR2_ADDR	**2 bits' error address** **Note: multiply ERR2_ADDR by 4 and then plus 0x20000000 to get the SRAM address of occurring 2 bits' error.**

24 Coprocessor Unit

24.1 Introduction

Coprocessor unit (MMDIVSQRT module) is used to calculate division and square root. The calculation format is as follows:

Division: $(x \ll z)/y$

Square Root: $\sqrt[2]{x^2 + y^2}$

The dividend x and the divisor y are 32bit data, the shift bit z is 5 bits, the range of which is between 0 and 31. The MMDIVSQRT module supports execution of the integer divide operations including signed and unsigned calculations.

24.2 Features

- Supports 32-bit signed and unsigned divide (or remainder) calculations.
- Supports 32-bit signed square root calculations.
- Simple programming model includes an input data register, a result register and a control/status register.
- Supports quotient and remainder can be selected for division.
- The division uses the SRT algorithm and the squared uses the CORDIC algorithm.

24.3 Block diagram

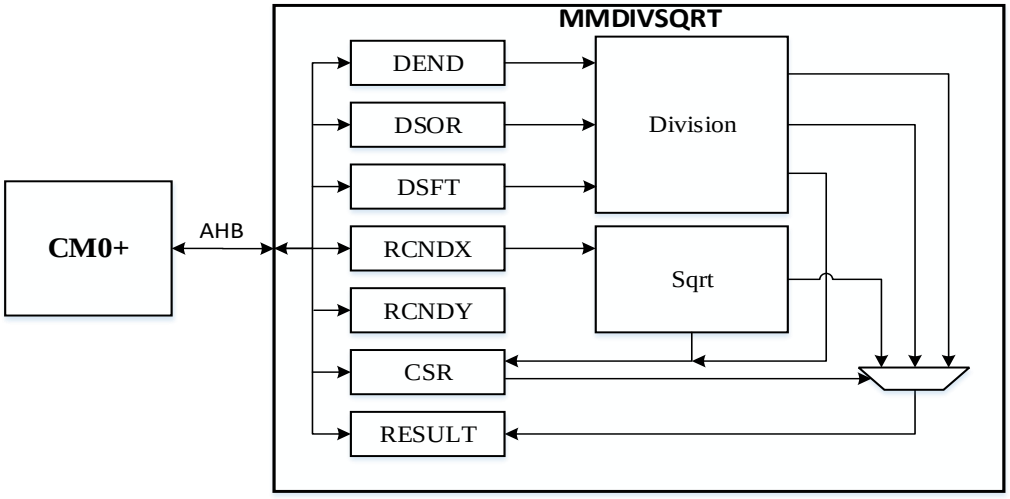

Figure 24-1 MMDIVSQRT block diagram

24.4 Application note

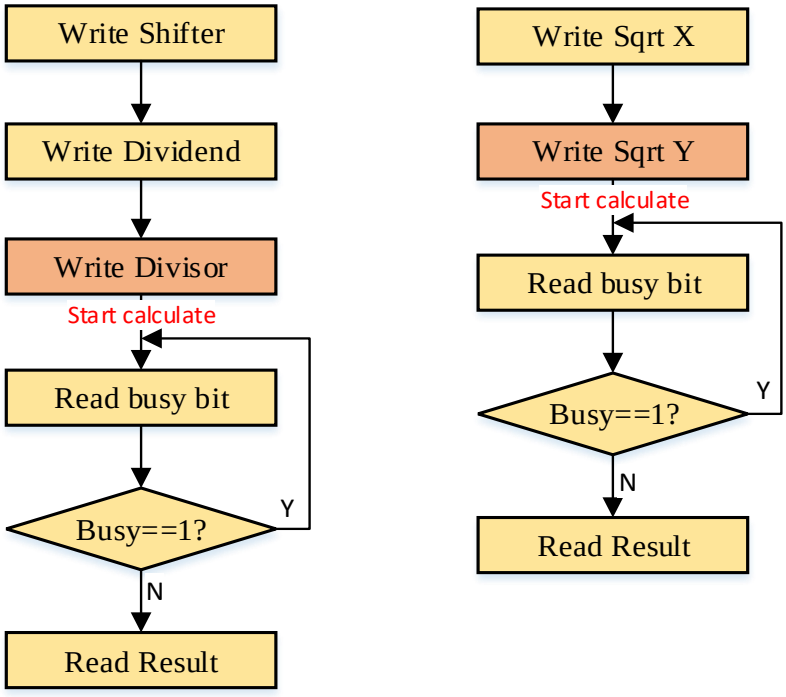

Figure 24-2 MMDIVSQRT programming steps

Division operation configuration steps:

1. The default is a signed division operation. If it is needed to configure an unsigned division operation, please set the **USIGN** bit to 1.
2. By default, the result of division calculation is the quotient. To get the remainder, please set the **MEM** bit to 1.
3. Configure the shift register MMDIVSQRT_DSFT, the range of which is 0~31.
4. Configure the dividend register MMDIVSQRT_DEND.
5. Configure the divisor register MMDIVSQRT_DSOR. After configuration, the calculation is automatically started.
6. Read the busy bit. If it is 1, it means that the division operation is in progress and needs to continue to wait. When the busy bit is 0, it means that the division calculation is completed, and the calculation result can be read from the MMDIVSQRT_RESULT register. The result saved in the MMDIVSQRT_RESULT register is the quotient or remainder, depending on the **MEM** bit.

Root extraction configuration steps:

1. Configure the square root x register MMDIVSQRT_RCNDX.
2. Configure the square root y register MMDIVSQRT_RCNDY. After configuration, the calculation is automatically started.
3. Read the busy bit. If it is 1, it means that the square root operation is in progress and needs to continue to wait. When the busy bit is 0, it means that the square root calculation is completed, and the calculation result can be read from the MMDIVSQRT_RESULT register.

Note

1. **The square root operation is a signed operation, so configuring the USING bit has no effect on the square root operation.**
2. **The precision of the square root is 8. That is, the input range of RCNDX and RCNDY is -8192~8192, the error of the calculation result is guaranteed to be within 8.**

24.5 Register definition

Table 24-1 MMDIVSQRT register map

MMDIVSQRT base address: 0x20081800

Address	Name	Width (in bit)	Description
MMDIVSQRT base address + 0x000	MMDIVSQRT_DEND	32	Input dividend (numerator) for the divide
MMDIVSQRT base address + 0x004	MMDIVSQRT_DSOR	32	Input divisor (denominator) for the divide
MMDIVSQRT base address + 0x008	MMDIVSQRT_DSFT	32	Input Shifter for the divide (0, 31)
MMDIVSQRT base address + 0x00C	MMDIVSQRT_RCNDX	32	Input "square" data x
MMDIVSQRT base address + 0x010	MMDIVSQRT_RCNDY	32	Input "square" data y
MMDIVSQRT base address + 0x014	MMDIVSQRT_CSR	32	Control/Status
MMDIVSQRT base address + 0x018	MMDIVSQRT_RESULT	32	Output result

24.5.1 MMDIVSQRT_DEND

Table 24-2 MMDIVSQRT_DEND register

MMDIVSQRT_DEND **Input dividend (numerator) for the divide** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DEND[31:16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DEND[15:0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 DEND	**DEND** dividend

24.5.2 MMDIVSQRT_DSOR

Table 24-3 MMDIVSQRT_DSOR register

MMDIVSQRT_DSOR **Input divisor (denominator) for the divide** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DSOR[31:16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DSOR[15:0]															
Type	RW															
Reset	0															

Bits	Description
31: 0	**DSOR**
DSOR	divisor

24.5.3 MMDIVSQRT_DSFT

Table 24-4 MMDIVSQRT_DSFT register

MMDIVSQRT_DSFT **Input Shifter for the divide** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												DSFT				
Type												RW				
Reset												0				

Bits	Description
4: 0	**DSFT**
DSFT	Input Shifter for the divide (0, 31)

24.5.4 MMDIVSQRT_RCNDX

Table 24-5 MMDIVSQRT_ RCNDX register

MMDIVSQRT_RCNDX **Input "square" data x** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RCNDX[31:16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RCNDX[15:0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 RCNDX	**RCNDX** Input "square" data x Range: -8192 to +8192

24.5.5 MMDIVSQRT_RCNDY

Table 24-6 MMDIVSQRT_ RCNDY register

MMDIVSQRT_RCNDY **Input "square" data y** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RCNDY[31:16]															
Type	RW															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RCNDY[15:0]															
Type	RW															
Reset	0															

Bits	Description
31: 0 RCNDY	**RCNDY** Input "square" data y Range: -8192 to +8192

24.5.6 MMDIVSQRT_CSR

Table 24-7 MMDIVSQRT_CSR register

MMDIVSQRT_CSR **Control/Status** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BUSY	DIV	SQRT													
Type	R	R	R													
Reset	0	0	0													
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														REM	USIGN	
Type														RW	RW	
Reset														0	0	

Bits	Description
31 BUSY	**Busy** 0: done 1: doing This read-only bit is asserted when the MMDIVSQRT is performing a divide or square root. When an operation is initiated, the hardware sets this flag. It is deasserted when calculation is done.
30 DIV	**Div** 0: divide 1: not divide Current or last operation was a divide. This read-only indicator bit signals if the current or last operation performed by the MMDIVSQRT was a divide. It is automatically set by hardware.
29 SQRT	**SQRT** 0: sqrt 1: not sqrt Current or last operation was a sqrt. This read-only indicator bit signals if the current or last operation performed by the MMDIVSQRT was a sqrt. It is automatically set by hardware.
2 REM	**REM** 0: quotient 1: remainder This indicator selects whether the quotient or the remainder is returned in the REM register.

Bits	Description
1 USIGN	**USIGN** 0: signed 1: unsigned This indicator selects whether unsigned or signed is written in the DIV register (DEND & DSOR).

24.5.7 MMDIVSQRT_RESULT

Table 24-8 MMDIVSQRT_RESULT register

MMDIVSQRT_RESULT **Output result** **Reset: 0x00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RESULT[31:16]															
Type	R															
Reset	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RESULT[15:0]															
Type	R															
Reset	0															

Bits	Description
31: 0 RESULT	**RESULT** Calculation result output, division (quotient/remainder) and square root are shared.

25 Debug

25.1 Introduction

This debug of the device is based on the ARM CoreSight architecture. It provides debug interface, and control program run/stop/reset operations.

This device supports only one debug interface, Serial Wire Debug (SWD).

25.2 Features

- 4 hardware breakpoints.
- 2 data watchpoints.
- SWD interface accessibility.