TX32M2300 Data Manual

泰芯半导体
TaiXin Semiconductor

Revision Record

Date	Version	Description
2023-04-11	V2.2	1. Modify the selection description of the positive and negative input in the comparator section;
2023-02-03	V2.1	1. Solve the garbled code problem when some devices are opened for reading;
2022-08-12	V2.0	1. Reformatted the document;

Notice:

1、The company reserves the right to the final interpretation of the function, performance, scheme, design and improvement of all the following products.

2、The Company reserves the right to copy and change the documents.

Contents

1. Master directory

1.1. Overview

This product uses a high-performance 32-bit microcontroller with a maximum operating frequency of 72MHz, built-in 32KB high-speed Flash memory, 6KB SRAM, rich enhanced I/O ports and peripherals to connect to the external bus. This product contains 1 12-bit ADC, 1 8-bit precision DAC, 1 multi-function comparator, 3 operational amplifiers, 1 16-bit advanced timer, 5 16-bit general purpose timers, 1 32-bit general purpose timer, 1 Watchdog timer, 1 System tick timer. Also includes the standard communication interface: 2 SPI/IIC interface and 2 UART interface, among which UART0 can realize the code upgrade from any pin, built-in a 32 bit division 16 unsigned divider.

The operating voltage of this product series is 2.0V~5.5V, and the operating temperature range is -40℃ ~ 105℃. A variety of power saving operating modes ensure the requirements of low power applications.

The product is available in four packages, including LQFP32, SSOP28, SSOP24 and TSSOP20. The peripherals in the device are configured differently depending on the package form.

A basic introduction to all peripherals in this family is given below.

These rich peripheral configurations make this product microcontroller suitable for a variety of applications:

- Fan, fan, etc
- Consumer Electronics
- Smart home
- Motor drives and application controls
- Medical and handheld devices

- Industrial Controls
- Industrial applications: Programmable controllers (PLCS), frequency converters, printers and scanners

1.2. Product Features

➢ Kernel and System

- 32-Bit RISC architecture CPU
- Maximum working frequency: 72MHz
- Single cycle 32-bit multiplication instruction
- 32 interrupt sources, configurable 4-layer interrupt priority, support interrupt entry address Remap
- Supports bitband operation
- Support double pin debugging interface

➢ Memory

- 32K Byte Flash program memory (NO EEPROM Flash), Sector erase 20,000 times
- Internal 6K Byte SRAM
- Boot Loader supports on-chip Flash, supports single/dual pin UART Online User Programming (IAP)/Online System Programming (ISP)

➢ Clock, reset, and power management

- 2V to 5.5V power supply
- Power on/power off reset (POR/PDR), Programmable Voltage Monitor (PVD)
- External 1-32MHz crystal oscillator
- Factory-tuned 26MHz(+/ -1.5%) high speed oscillator embedded
- Embedded 128KHz low speed oscillator
- PLL outputs 72MHz clock
- Built-in Clock Security System (CSS)

- WDT reset

- **DMA support**
  - Supported peripherals: EFLASH, UART, SPI/I2C, CRC, TIMER, ADC
- **GPIO**
  - Supports up to 30 GPIOs
  - All I/O ports can trigger edge or level response interrupts to wake up low power mode
  - All ports can input and output 5V signals
  - Support for key detection
- **Communication interface peripherals**
  - 2 SPI high-speed serial interface, Master under the maximum support system clock frequency half transmission, support 1/2/4 line master-slave mode, support I2C mode
  - 2 UART interface, support RS232/RS485 protocol, UART0 can support any IO program upgrade
- **Timer**
  - 1 16-bit advanced timer that supports 4 pairs of complementary outputs or 8 independent PWM outputs, supports dead zone insertion and event brake functions, and supports monopulse mode
  - 5 16-bit universal timers, 1 32-bit timer, each with capture function
  - 1 watchdog timer
  - 1 system tick timer
- **Hardware acceleration unit**
  - Hardware signed divider (32bit/16bit)
- **High security**
  - Support 5/7/8/16/32 bit CRC validation to ensure data accuracy
  - Support code scrambling and hardware encryption and decryption

- Low power consumption
  - Supports IDLE, STOP, and SLEEP low-power modes
  - Static power consumption <20uA@25 ° C
  - Low power wake-up time 10us fastest
- 1 12-bit high speed analog-to-digital converter
  - Supports up to 1.2Mhz sampling rate
  - Up to 10 input channels
- 1 comparator
  - Supports 7 positive end inputs and 3 negative end inputs
  - Support hardware channel polling
- 3 operational amPs
  - An op-amp in the same direction
  - Built-in 4/6/8/10/12x gain available
  - Closed-loop gain bandwidth optional
- Built-in temperature sensor
- High reliability
  - ESD HBM 8KV
  - EFT ±4500V
  - Latch-up ±100mA @105℃
- 96-bit chip Unique ID (UID)
- Package
  - Die Form
  - LQFP32/SSOP28/SSOP24/TSSOP20
- Industrial grade temperature range
  - -40℃ ~ 105℃
- Apply
  - Fan, etc.

- Motor drive and application control
- Medical and handheld devices
- PC gaming peripherals and GPS platforms
- Industrial applications: Programmable controllers (PLCS), frequency converters, printers and scanners
- Alarm systems, video intercom, and heating ventilation and air conditioning systems

2. System and memory architecture

TX32M2300 series devices are based on RISC processor 32-bit general purpose microcontroller memory organization using Harvard structure, pre-defined memory mapping and up to 4 GB of storage space, fully ensure the flexibility and scalability of the system.

2.1. 32 bit RISC processor

The 32-bit RISC processor used in the TX32M2300 series is a 32-bit processor with low interrupt latency and low cost debugging features. The high integration and enhanced features make this RISC processor suitable for those market segments that require high performance and low power microcontrollers.

2.2. System Architecture

The TX32M2300 series devices adopt a 32-bit multi-layer bus structure, which makes parallel communication between multiple hosts and slaves in the system possible. The multi-layer bus architecture consists of an AHB interconnection matrix,

two AHB buses and two APB buses. The interconnection of the AHB interconnection matrix is described next. In the interconnection list of the system master/slave interconnection matrix, "1" indicates that the corresponding host can access the corresponding slave through the AHB interconnection matrix. Blank cells indicate that the corresponding host cannot access the corresponding slave through the AHB interconnection matrix.

The TX32M2300 series master system consists of the following parts:

- Two drive units:
  - CPU core system Bus (S-bus)
  - DMA controller
- Three passive units
  - Internal flash memory
  - Internal SRAM
  - Bridge from AHB to APB (AHB2APB0/1), which connects all the APB devices

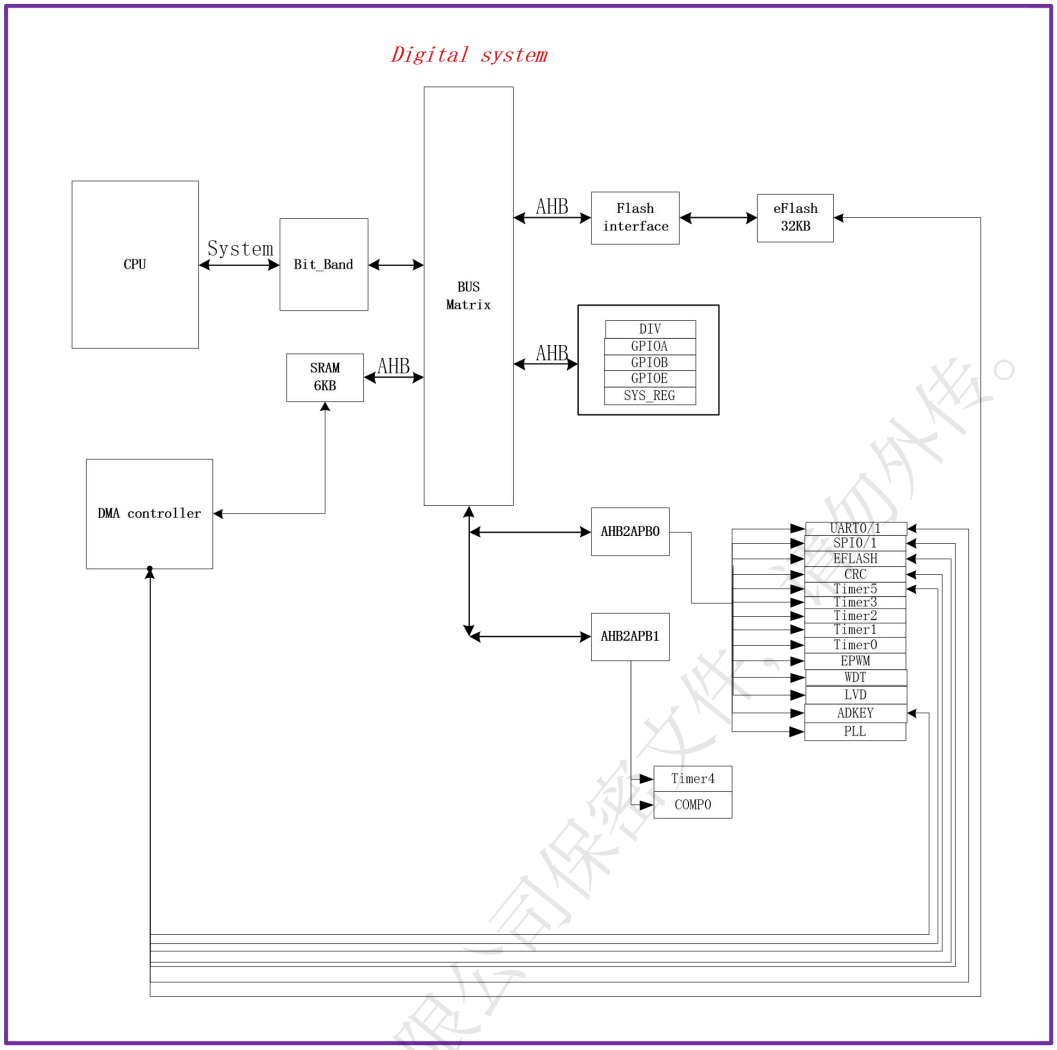

Figure 2-1 Architecture of the TX32M2300 series bus system

System Bus

This bus connects the CPU core's system bus (peripheral bus) to the bus matrix, which coordinates access between the core and various high-speed components.

DMA controller

This bus connects the CPU and peripheral modules to compete for access, coordinates access priorities, arbitrates, etc.

Table 2-1 Priority arrangement used by DMA module

Modules	Priority	Instructions
CPU	0	Highest
SPI0	1	
SPI1	2	
UART0	3	
ADC	4	

UART1	5	
CRC	6	
TIMER5	7	
Eflash	8	Minimum

BusMatrix (BusMatrix)

The bus matrix manages the access arbitration between the kernel system bus and each peripheral module. The bus matrix consists of the master module bus and the slave module bus. AHB peripherals are connected to the system bus through the bus matrix.

AHB to **APB** bridge (**AHB2APB bridge-APB**)

The AHB to APB bridge provides a synchronous connection between the AHB and APB buses.

Note: When an *8-bit* or *16-bit* access is made to an *APB* register, the access is automatically converted to *32-bit* access: the bridge automatically expands the *16* - or *8-bit* data to match the *32-bit* width.

2.3. Memory Mapping

This 32-bit RISC processor uses the same set of buses to read instructions and load/store data. Both instruction code and data are located in the same memory address space, but in different address ranges. Program memory, data memory, registers, and I/O ports all reside in the same linear 4 GB address space. This is the maximum address range for 32-bit RISC, as its address bus width is 32 bits. Also, in order to reduce software complexity for different customers in the same application, the storage map is pre-defined according to the rules provided by the 32-bit RISC processor. In the memory mapping table, a portion of the address space is occupied by 32-bit RISC system peripherals and cannot be changed. In addition, the rest of the address space can be defined by the chip vendor.

The Memory mapping table of the TX32M2300 series device shows the memory mapping of the TX32M2300 series device, including code, SRAM, peripherals, and other pre-defined areas. Simplifies the address decoding of each peripheral.

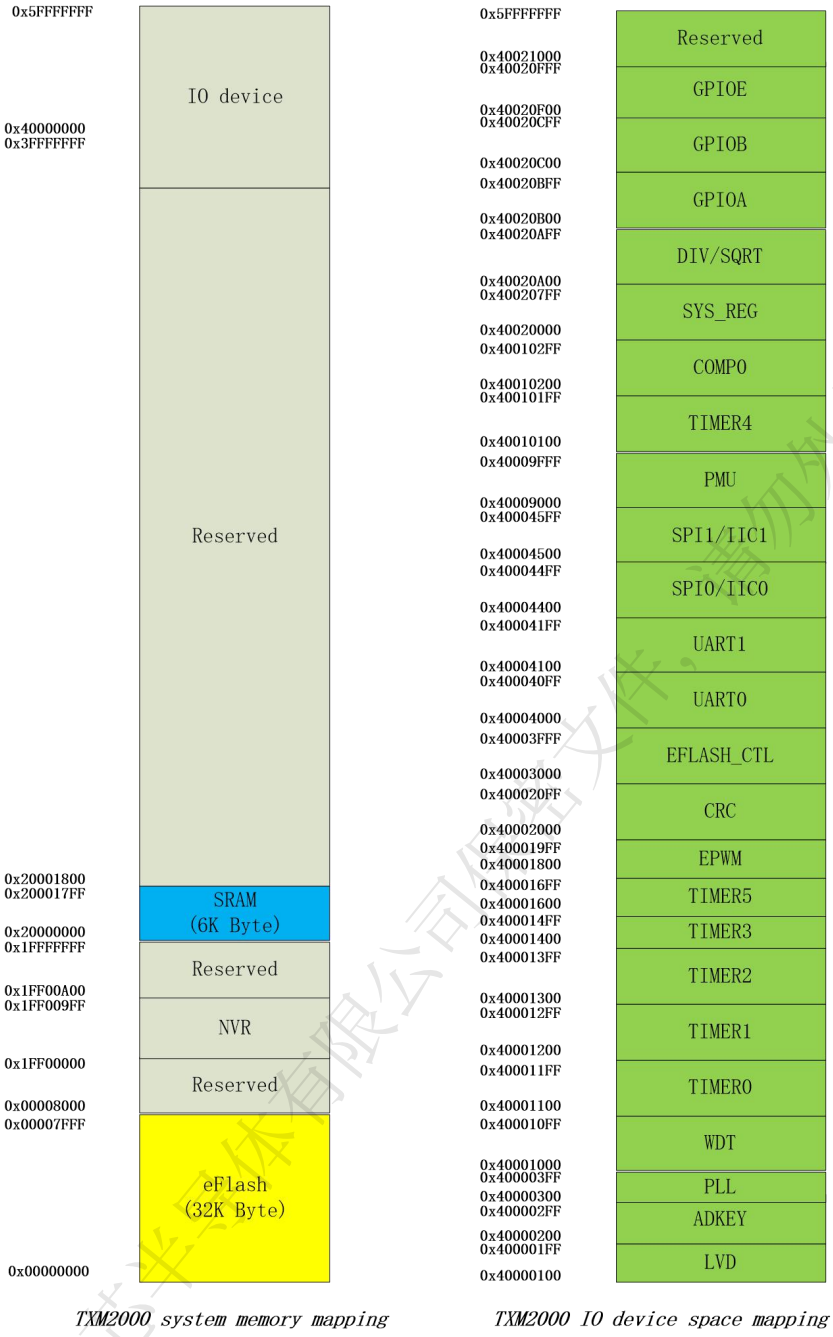

Figure 2-2 Memory mapping table of TX32M2300 series

2.3.1. Bitband operation

To reduce the number of read-change-write operations, 32-bit RISC processors provide a bitband function that can perform single-atom bit operations. The memory map contains two areas that support bitband operations. One is the minimum 1MB range of the SRAM area, and the second is the minimum 1MB range of the on-chip and off-chip

areas. The addresses in these two zones have their own "bitband alias zone" in addition to normal applications. The bitband alias area expands each bit into a 32-bit word. When the user accesses the bitband aliasing area, the purpose of accessing the original bit is achieved.

The following formula shows how each word in the bitband alias area corresponds to the corresponding bit or destination bit in the bitband alias area.

bit_word_addr =bit_band_base +(byte_offset×32)+(bit_number×4) (Equation 31)

Where:

- bit_word_addr refers to the address of the bitband area target bit corresponding to the bitband alias area;
- bit_band_base refers to the start address of the bitband alias area;
- byte_offset refers to the byte address offset of the byte in which the target bit of the bitband is located;
- bit_number refers to the location of the target bit in the corresponding byte (0-7).

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

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

If you write to 0x2200 401C, the 7th bit of 0x2000 0200 will change accordingly; If a read is done to 0x2200 401C, then 0x01 or 0x00 is returned as the 7th bit state of 0x2000 0200.

2.3.2. On-chip SRAM memory

The TX32M2300 series has built-in static SRAM up to 6K bytes. It can be accessed in bytes (8 bits), half words (16 bits), or words (32 bits). The SRAM starts at 0x2000 0000. Up to 6K bytes of SRAM on the data bus. Can be accessed by CPU or DMA using the fastest system clock without inserting any waits. DMA supports access with timer5, ADC, CRC, SPI0/1, UART0/1, EFLASH.

2.3.3. Overview of on-chip FLASH memory

Flash memory has two different storage areas:

- The main flash storage block, which includes application and user data areas (if needed)
- Information block, which consists of two parts:

- Option bytes - Contains hardware and storage protection user configuration options.

- System memory - contains the Boot loader code. See section on Built-in Flash Memory.

The flash memory interface executes instructions and data access based on the AHB protocol. The function of prefetch buffering can accelerate the speed of CPU code execution.

2.3.4. Boot Configuration

After the chip is reset, through the customer's own configuration in the main flash storage block, the boot mode pin can be selected as PC6 or PC0, and the default operation is pull-up or pull-down. After the boot delay, the CPU takes the address at the top of the stack from address 0x0000 0000 and executes the code from the address indicated by 0x0000 0004 of the boot memory.

The embedded bootstrap program is stored in the system memory and written by the manufacturer at production time. The program can be reprogrammed to the flash memory via UART0.

2.4. MCLR features

The TX32M2300 series supports the MCLR reset function by default. MCLR means that the input of a low level on a specific IO pin lasting more than 1.7ms causes the system to reset, as if it were reset after a power-on. When the MCLR function is disabled by the user using the user-defined bit in eflash. PE2 becomes the normal I/O function. Refer to the description of user-defined areas in Flash memory for details.

3. Embedded Flash Memory

3.1. Flash Memory Main Features

- 32K byte flash memory
- Memory structure:
  - Main flash memory space: 32K bytes
  - Secondary flash memory space (system memory) : 2K bytes
- Read interface with prefetch buffer
- Flash programming and erase operations
- Access and write protection
- Low power mode

3.2. Flash Memory Function Description

3.2.1. Flash Memory structure

The flash memory space consists of 32-bit wide storage units that can store both code and data. The main flash block is divided into 32 pages (1K bytes per page) with write protection set on a page basis (see storage protection related content).

Modules	Name	Address	Size (bytes)
Main flash memory space	Page 0	0x0000_0000 – 0x0000_03FF	1K
	Page 1	0x0000_0400 – 0x0000_07FF	1K
	Page 2	0x0000_0800 – 0x0000_0BFF	1K
	Page 3	0x0000_0C00 – 0x0000_0FFF	1K
	...	...	1K
	Page 30	0x0000_7800 – 0x0000_7BFF	1K
	Page 31	0x0000_7C00 – 0x0000_7FFF	1K
Sub-flash space	Sector 0	0x1FF0_0000 – 0x1FF0_01FF	512
	Sector 1	0x1FF0_0200 – 0x1FF0_03FF	512
	Sector 2	0x1FF0_0400 – 0x1FF0_057F	384
	User Parameters area	0x1FF0_0580 – 0x1FF0_06BF	64
	User profile area	0x1FF0_05C0 – 0x1FF0_05FF	64
	Chip information area	0x1FF0_0600 – 0x1FF0_064F	80
	EOTP chip area	0x1FF0_0650 – 0x1FF0_068F	64
	EOTP User area	0x1FF0_0690 – 0x1FF0_06EF	96
Flash register interface	CTRLR0	0x4000_3000 – 0x4000_3003	4
	KST	0x4000_3004 – 0x4000_3007	4
	DONE	0x4000_3008 – 0x4000_300B	4

	PROG_ADDR	0x4000_3010 – 0x4000_3013	4
	PROG_DATA	0x4000_3018 – 0x4000_301B	4
	ERASE_CTRL	0x4000_3020 – 0x4000_3023	4
	TIME_REG0	0x4000_3030 – 0x4000_3033	4
	TIME_REG1	0x4000_3034 – 0x4000_3037	4
	NVR_PASSWORD	0x4000_3050 – 0x4000_3053	4
	MAIN_PASSWORD	0x4000_3054 – 0x4000_3057	4
	CRC_ADDR	0x4000_3058 – 0x4000_305B	4
	CRC_LEN	0x4000_305C – 0x4000_305F	4
	CRC_OUT	0x4000_3060 – 0x4000_3063	4

3.2.2. Flash read protection

The read operation can be completed in the entire product operating voltage range for storing instructions or data. Flash runs at the working frequency of 36MHz. If the working frequency is increased to more than 36MHz, the Flash reading sequence needs to perform frequency division.

The chip is equipped with cache buffer and prefetch buffer to improve the access efficiency of Flash.

If protection is enabled in the user configuration area, the Flash protection mechanism is automatically implemented when the SWD and other debugging interfaces are connected.

The read operation is completed by the following two registers:

① Configuration register (CTRLR0)

② Timing 0 register (TIME_REG0)

3.2.3. Flash burn and erase operations

Write and erase operations can be done over the entire product operating voltage range.

The burn and erase operation is completed by the following 6 registers, first configure the burn time sequence (TIME_REG1) according to the burn clock, then configure the burn password, configure the programming address, and finally configure the programming data, you can start to execute the burn, and then wait for the end of the burn.

As long as the CPU does not access the Flash space, or has a cache hit when accessing Flash, the ongoing Flash write operation will not hinder the operation

of the CPU. In other words, while writing/erasing the Flash, any access to the Flash will cause the bus to stop until the write/erasing operation is complete, which means that the Flash can not be written/erasing at the same time, it can not get fingers and access data

3.3. Registers

3.3.1. Register base address

Name	Base Address	Description
Eflash	0x40003000	Base address of Eflash

3.3.2. Register list

Offset Address	Name	Description
0x0	EFLASH_CTRLR0	Control register
0x04	EFLASH_KST	Trigger register
0x08	EFLASH_DONE	Status register
0x10	EFLASH_PROGADDR	Programming address register
0x18	EFLASH_PROGDATA	Programming data register
0x20	EFLASH_ERASECTRL	Erasing the control register
0x24	EFLASH_INTFERASEENA	Interrupt interrupts EFlash working status
0x30	EFLASH_TIMEREG0	Timing 0 configures register
0x34	EFLASH_TIMEREG1	Timing 1 Configure registers
0x50	EFLASH_NVRPASSWORD	NVR area secret key register
0x54	EFLASH_MAINPASSWORD	Main area key register
0x58	EFLASH_CRCADDR	CRC address register
0x5C	EFLASH_CRCLEN	CRC length register
0x60	EFLASH_CRCOUT	CRC output Register

3.3.3. Register Details

3.3.3.1. EFLASH_CTRLR0

Bit(s)	Name	Description	R/W	Reset
31:20	Reserved		RO	0
19	Error Interrupt IE	**The enable bit for an exception interrupt** After the interrupt is enabled, the following exceptions are triggered: ① Unaligned address programming ② Programming data error ③ No write permission ④ Erasing failed	RW	0
18	Normal Interrupt IE	**Normal interrupt enable bit for normal function** If the interrupt is enabled, the following conditions will trigger: ① Programming ② Erase ③CRC check ④ Automatic programming	RW	0
17	Reserved		RO	0
16	Program Clock select	**Eflash burns clock source selection. RC clock is recommended** 0x0: High speed RC clock 2 frequency division 0x1: Crystal oscillator, only used if the RC is not on time	RW	0
15:12	Reserved		RO	0
11	LVD Program Disable	**LVD power down burn control bit** When LVD power off, whether to allow to interrupt eflash program and erase, generally when power off, immediately save important data in eflash, turn on this function, used to quickly let eflash in the idle state 0x0: Off 0x1: On	RW	0
10:6	Reserved		RO	0
5	Block request Mode	**Request buffering mode for Program/Erase (for test use)** 0x0: Off 0x1: On	RW	0
4	Cache Write Back Mode	**Program/Sector Erase Write back to the cache automatically (Used for test)** 0x0: Off 0x1: On	RW	1

3	Reserved		RO	1
2	Prefetch Enable	**Prefetch enable bit** 0x0: Off 0x1: On	RW	0
1	Reserved		RO	0
0	Cache Enable	**Cache enable bit** 0x0: Off 0x1: On	RW	0

3.3.3.2. EFLASH_KST

Bit(s)	Name	Description	R/W	Reset
31	EFLASH Error Clear Enable	**Eflash Error Clear Enable bit** Note: Needs to be configured at the same time as BIT[15] to be valid 0x0: Off 0x1: Open	WO	0
30	EFLASH Pending Clear Enable	**Complete Status flag Clear enable bit** Note: Needs to be configured at the same time as BIT[14] to be effective 0x0: Off 0x1: On	WO	0
29:27	Reserved		RO	0
26	CRC Kick Enable	**CRC kick enable bit** Note: Needs to be configured at the same time as BIT[10] to be effective 0x0: Off 0x1: On	WO	0
25	Auto Program EFLASH Enable	**Auto program eflash Enable** Note: Needs to be configured at the same time as BIT[9] to be effective 0x0: Off 0x1: On	WO	0
24	Auto Program RAM Enable	**SRAM auto Program Enable bit** Note: Needs to be configured at the same time as BIT[8] to be effective 0x0: Off 0x1: On	WO	0
23:21	Reserved		RO	0
20	Cache Clear Kick Enable	**Cache clear kick enable bit** Note: It needs to be configured with BIT[4] to be valid 0x0: Off 0x1: Open	WO	0
19:11	Reserved		RO	0

15	EFLASH Error Clear	**Exception flag clear bit** Note: Needs to be configured at the same time as BIT[31] to be valid Write 1 Clear	WO	0
14	EFLASH Pending Clear	**Complete status flag clear bit** Note: Needs to be configured at the same time as BIT[14] to be valid Write 1 Clear	WO	0
13:11	Reserved		RO	0
10	CRC Kick Start	**CRC kick start** Note: Needs to be configured at the same time as BIT[26] to be valid 0x0: Off 0x1: On	WO	0
9	Auto Program EFLASH Kick start	**Auto-program trigger bits** Note: Needs to be configured at the same time as BIT[25] to be effective 0x0: Off 0x1: On	WO	0
8	Auto Program RAM Kick start	**SRAM automatically programs trigger bits** Note: Needs to be configured at the same time as BIT[24] to be effective 0x0: Off 0x1: On	WO	0
7:5	Reserved		RO	0
4	Cache Clear Kick Start	**Cache clear bit** Note: needs to be configured at the same time as BIT[20] to be valid 0x0: Off 0x1: On	WO	0
3:0	Reserved		RO	0

3.3.3.3. EFLASH_DONE

Bit(s)	Name	Description	R/W	Reset
31	EFLASH Error	**Error status flag bit** 0x0: Normal 0x1: Abnormal	RO	0
30:20	Reserved		RO	0
19	Chip Erase Error	**Full chip erase abnormal status** 0x0: Works properly 0x1: Erase exception, error is valid	RO	0
18	Write permission	**Programming permission exception status**	RO	0

	Error	0x0: Working properly 0x1: No programming permissions, error is valid		
17	Program Address Error	**Program address abnormal state** 0x0: Working properly 0x1: Unaligned address, error is valid	RO	0
16	Program Data Error	**Abnormal state of programming data** Whether the programmed data is consistent with the data configured by the software 0x0: Works properly 0x1: Data is inconsistent, error is valid	RO	0
15	EFLASH Busy	Eflash working status flag 0x0: Idle status 0x1: In progress	RO	0
14:11	Reserved		RO	0
10	CRC Done	**CRC DONE flag** 0: In progress 1: Idle	RO	1
9	Auto Program EFLASH Done	**Auto program eflash Done flag** 0x0: Idle 0x1: Finished	RO	0
8	Auto Program RAM Done	**Auto program RAM done flag** 0x0: Idle 0x1: Finished	RO	0
7	Reserved		RO	0
6	Program Done	**Program end flag** 0x0: In progress 0x1: Idle	RO	1
5	Reserved		RO	0
4	Cache Clear Done	**Cache initialization flag** 0x0: In progress 0x1: Idle	RO	1
3:2	Reserved		RO	0
1	Chip Erase Done	**The Main field completely erases the flag** 0x0: Running 0x1: Idle	RO	1
0	Sector Erase Done	**Sector (512 byte) Erase flag** 0x1: Running 0x0: Idle	RO	1

3.3.3.4. EFLASH_PROGADDR

Bit(s)	Name	Description	R/W	Reset
31:30	Program Byte	**Bit width Settings for programming** eflash is divided into code area and RAM	RW	-

		area through the DATA RAM setting of the functional area, where the RAM area can perform 1/2/4 byte programming, while the code area can only perform 4 byte programming 0x0:1 byte 0x1:2 byte 0x2:4 byte		
29	Program NVR Select	**The address of the program is NVR zone or not** 0x0: MAIN area 0x1: NVR area	RW	-
28:0	Program Address	**Address for eflash programming** In automatic programming mode, as the destination address: In auto-programmed RAM mode, this address serves as the RAM address; In auto-programmed EFLASH mode, this address acts as the EFLASH address. To program in the code field: Only 4 byte programming is supported Programming in the DATARAM area: The programming address is half word aligned and supports 1/2 byte programming The programming address is aligned in word and supports 1/2/4 byte programming Programming addresses are odd and only 1 byte programming is supported	RW	-

3.3.3.5. EFLASH_PROGDATA

Bit(s)	Name	Description	R/W	Reset
31:0	Program Data	**eflash programming data, need to configure the address can be carried out**	RW	0

3.3.3.6. EFLASH_ERASECTRL

Bit(s)	Name	Description	R/W	Reset
31	Chip Erase Kick Start	**Trigger of Chip erase full** Write "1" trigger, you need to configure the password first	RO	0
30	Sector Erase Kick Start	**Trigger of Sector erase** Write "1" to trigger, you need to configure the password first	RO	0

29	NVR Sector Enable	**The sector enable bit of the NVR** 0: The MAIN area 1: NVR area	RW	0
28:7	Reserved		RO	0
6:0	Erase Sector Address	**Erase sector selection, range 0-127**	RW	0

3.3.3.7. EFLASH_TIMEREG0

Bit(s)	Name	Description	R/W	Reset
31:20	Reserved		RO	0
19:16	PGH	WEb low to PROG2 high hold min time is 15ns	RW	1
15:12	ADS	BYTE/Address/data setup min time is 15ns	RW	1
11:8	ADH	BYTE/Address/data hold min time is 15ns	RW	1
7:4	RW	Latency to next operation after PROG/ERASE low min time is 100ns	RW	8
3:0	RC	Read Cycle Min Time is 25/30ns	RW	0

3.3.3.8. EFLASH_TIME_REG1

Bit(s)	Name	Description	R/W	Reset
31:20	Reserved		RO	0
18:8	1ms unit	1ms time configuration value, in 1us	RW	1000
7:0	1us unit	1us time configuration value, the system default is 26MHz	RW	26

3.3.3.9. EFLASH_NVR_PASSWORD

Bit(s)	Name	Description	R/W	Reset
31:10	NVR password	The password is 0x20150931 and NVR can only be erased and programmed once the password is opened	RW	0

3.3.3.10. EFLASH_MAIN_PASSWORD

Bit(s)	Name	Description	R/W	Reset
31:10	Main password	The password is 0x20170230, and you can only erase and program Main once you open the password	RW	0

3.3.3.11. EFLASH_CRC_ADDR

Bit(s)	Name	Description	R/W	Reset
31:30	Reserved		RO	0
29	NVR Select	**Address is NVR zone** 0x0: The MAIN zone 0x1: NVR area	RW	0
28:2	DMA Address	**The start address of the CRC DMA, as the source address** In auto-programmed RAM mode, this address serves as the EFLASH address; In auto-programmed EFLASH mode, this address serves as the RAM address.	RW	0
1:0	Reserved	Reserved, lower 2-bit address fixed to 0, word aligned	RO	0

3.3.3.12. EFLASH_CRC_LEN

Bit(s)	Name	Description	R/W	Reset
31:2	DMA length	The length of the CRC DMA	RW	0
1:0	Reserved	Reserved, lower 2-bit address fixed to 0, word aligned	RO	0

3.3.3.13. EFLASH_CRC_OUT

Bit(s)	Name	Description	R/W	Reset
31:00	CRC Result	**CRC Result Output** Polynomial CRC-32 is as follows, the result of the inverse is the official result, CRC written to EFLASH needs the official value to take the inverse representation!! x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1	RO	0

4. Interrupts and events

4.1. Nested vector interrupt controller

Basic functions:

- All interrupts can be masked (except NMI)
- 16 programmable priority levels (4-bit interrupt priority is used)
- Low latency exception and interrupt handling
- Power management Controls
- Implementation of system control register

The interface of nested vector interrupt controller (NVIC) and processor core is closely connected, which can realize low latency interrupt processing and efficiently deal with late interrupts.

The nested vector interrupt controller manages interrupts, including kernel exceptions. Refer to the CPU Technical Reference Manual for more instructions on exceptions and NVIC programming.

4.2. System Tick (SysTick) calibration value register

This chip does not support using an external clock to time 1ms.

4.3. Interrupt Function Description

The processor and the nested vector Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler mode. When an exception occurs, the system automatically stacks the current working state of the processor, and automatically unstacks it after executing the Interrupt service subroutine (ISR).

The orientation is carried out in parallel with the current working state, thus improving the interrupt entry efficiency. The processor supports tail-biting interrupts, which can realize back-to-back interrupts, greatly reducing the

overhead caused by repeated switching of working states. The NVIC Exception types list all the exception types.

Table 4-1 NVIC exception types in the CPU

Types of exceptions	Vector number	Priority	Vector address	Description
-	0	-	0x00000000	Retain
Reset	1	- 3	0x00000004	Reset
NMI	2	2 -	0x00000008	Non-maskable interrupts
HardFault	3	- 1	0x0000000C	Various hardware level failures
MemManage	4	Settable	0x00000010	Memory management
BusFault	5	Settable	0x00000014	Prefetch refers to failure, memory access failure
UsageFault	6	Settable	0x00000018	Undefined instruction or illegal status
-	7-10	-	0x0000001C-0x0000002B	reserve
SVcall	11	Settable	0x0000002C	Implementation of system service call via SWI instruction
DebugMonitor	12	Settable	0x00000030	Debug Monitor
-	13	-	0x00000034	Retain
PendSV	14	Settable	0x00000038	Pendable system service request
Systick	15	Settable	0x0000003C	System Beat Timer

The SysTick calibration value is set to 0x1196E and the SysTick clock frequency is set to HCLK. If the HCLK clock is set to 72MHz, the SysTick interrupt will respond once every 1ms.

Table 4-2 Interrupt vector table

Interrupt Sequence number	Vector number	Peripheral interrupt description	Vector address
IRQ0	16	lvd_int	0x0000_0040
IRQ1	17	uart0_int	0x0000_0044
IRQ2	18	uart1_int	0x0000_0048
IRQ3	19	spi0_int	0x0000_004C
IRQ4	20	spi1_int	0x0000_0050
IRQ5	21	gpioa_int	0x0000_0054
IRQ6	22	gpiob_int	0x0000_0058
IRQ7	23	gpioe_int	0x0000_005C
IRQ8	24	wkpnd_int	0x0000_0060
IRQ9	25	timer0_int	0x0000_0064
IRQ10	26	timer1_int	0x0000_0068
IRQ11	27	timer2_int	0x0000_006C
IRQ12	28	timer3_int	0x0000_0070

IRQ13	29	timer4_int	0x0000_0074
IRQ14	30	timer5_int	0x0000_0078
IRQ15	31	epwm_tzint	0x0000_007C
IRQ16	32	epwm_etint	0x0000_0080
IRQ17	33	adkey_interrupt	0x0000_0084
IRQ18	34	div_ovf_int	0x0000_0088
IRQ19	35	crc_dma_int	0x0000_008C
IRQ20	36	comp_int	0x0000_0090
IRQ21	37	wdt_interrupt	0x0000_0094
IRQ22	38	eflash_int	0x0000_0098
IRQ23	39	adkey_int1	0x0000_009C
IRQ24	40	-	0x0000_00A0
IRQ25	41	-	0x0000_00A4
IRQ26	42	-	0x0000_00A8
IRQ27	43	-	0x0000_00AC
IRQ28	44	-	0x0000_00B0
IRQ29	45	-	0x0000_00B4
IRQ30	46	-	0x0000_00B8
IRQ31	47	-	0x0000_00BC

4.4. External Interrupt/Event Controller (EXTI)

The External Interrupt and Time Controller (EXTI) manages external and internal asynchronous events/interrupts and generates corresponding event requests to the CPU/ interrupt controller and wake-up requests to power management. Each input line can be independently configured with the input type (pulse or suspend) and the corresponding trigger event (both rising or falling edges or both sides firing). Each input line can be masked independently. The suspend register holds the interrupt request for the status line.

4.4.1. Key features

The main features of the EXTI controller are as follows:

- Each interrupt/event has an independent trigger and mask
- Each interrupt has a dedicated status bit

- Supports interrupt/event requests for up to 20 pieces of software
- Detect external signals with pulse widths below the APB0 clock width

See the relevant parameters in the Electrical Characteristics section of the data book

4.4.2. Wake up Event Management

The TX32M2300 series can handle external or internal events to wake up the kernel (WFE). Wake events can be generated with the following configuration:

- An interrupt is enabled in the peripheral's control register, but not in the NVIC, while the SEVONPEND bit is enabled in the CPU's system control register. When the CPU recovers from the WFE, it needs to clear the interrupt suspend bit of the corresponding peripheral and the peripheral NVIC interrupt Channel suspend bit (in the NVIC Interrupt Clear Suspend register).
- Configure an external or internal EXTI line to be in event mode. When the CPU recovers from WFE, it does not have to clear the interrupt suspend bit or NVIC interrupt channel suspend bit of the corresponding peripheral because the suspend bit of the corresponding event line is not set.

5. Power Control

5.1. Power supply

The operating voltage (VCC) of the chip is 2.0V to 5.5V. This chip is designed with CAPLESS,

and there is no need to attach external capacitors to the built-in LDO output.

5.1.1. Voltage regulator

Regulator is always enabled after reset. In the case of low power consumption, you can enable the low power mode, PMUCON0.lpen as the power mode switching control bit.

5.2. Power Manager

5.2.1. Power-on reset (POR) and power-off reset (PDR)

The TX32M2300 series has a complete power-on reset (POR) and power-off reset (PDR) circuit inside, when the supply voltage reaches 2.7V

When the system can work normally. When VDD is below the specified limit voltage VPOR/VPDR, the system remains in the reset state without the need for an external reset circuit.

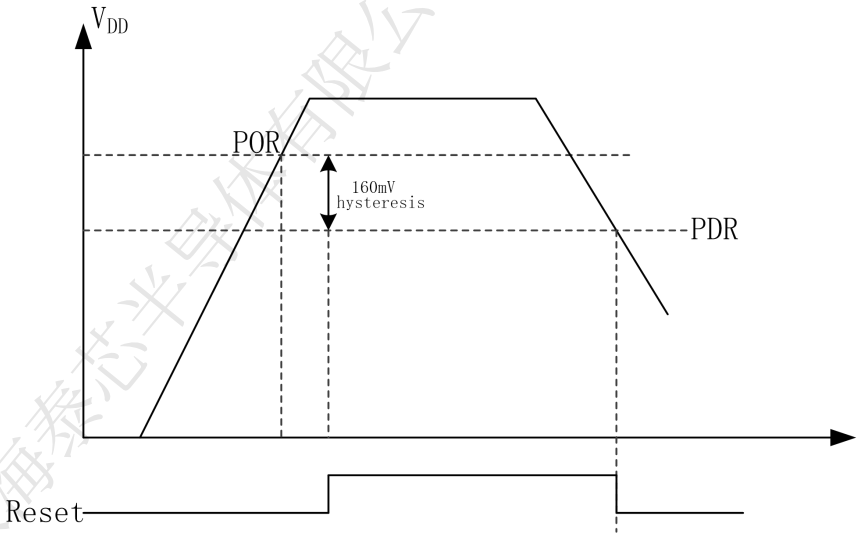

Figure 6-1 power-on reset and power-off reset

5.2.2. Programmable Voltage Monitor (PVD)

Two voltage detectors are integrated inside the TX32M2300 series, one detects the external

powered VCC and one detects the internal LDO output VDD. The LDO samples the capless structure and the VDD is not visible on the package. Both detection voltage thresholds are optional. When the system detects that the VCC or VDD voltage is lower than the configured voltage value, it can choose to trigger the system reset or enter the interrupt subfunction by enabling the PVD interrupt. This feature can be used to perform emergency shutdown tasks. The detection signal can be selected to pass through a burr filter circuit or directly detected, controlled by LVD_CON.lvdvcc_bps_en and LVD_CON.lvdvdd_bps_en.

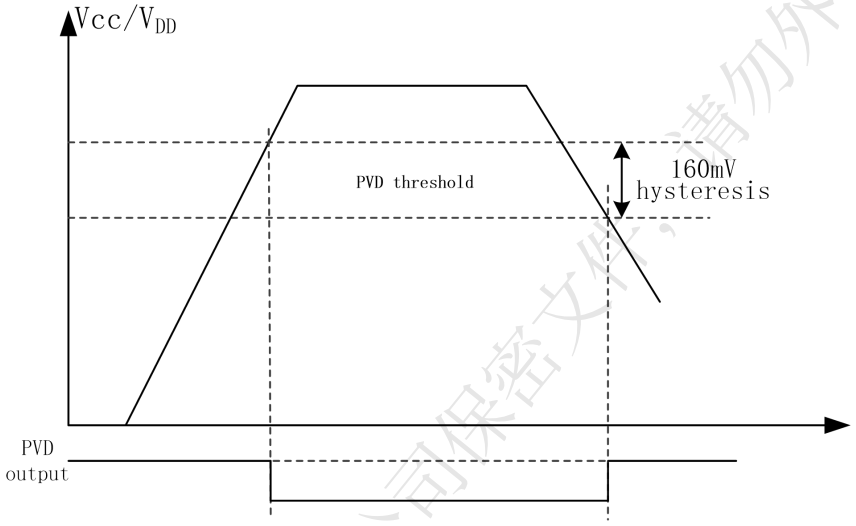

Figure 6-2 Schematic diagram of PVD

5.3. Low Power Mode

After the system or power is reset, the microcontroller is running and the system clock is 256Khz internal RC. A variety of low-power modes can be used to save power when the CPU does not need to continue running, such as while waiting for an external event. The user needs to choose the best low-power mode based on the lowest power consumption, the fastest startup time, and the available wake source.

The TX32M2300 series is available in three low power modes:

- IDLE mode (CPU stops and all peripherals including the peripherals of the CPU, such as NVIC, system clock (SysTick), etc. are still running)
- STOP mode (CPU, most peripherals stop, can rely on other times to wake this mode after the chip continues to run)
- SLEEP mode (all clock sources stop, rely on external IO wake up, can choose

to reset the chip or continue running)

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

- Reduce the system clock
- Turn off unused peripheral clocks on the APB and AHB buses
- Configure the frequency relationship between APB and AHB properly

5.4. Registers

5.4.1. Register base address

Name	Base Address	Description
LVD	0x40000100	LVD's base address

5.4.2. Register list

Offset Address	Name	Description
0x00	LVD_CON	Configuration register

5.4.3. Register Details

5.4.3.1. LVD_CON

Bit(s)	Name	Description	R/W	Reset
31	lvdvdd_pending	**The VDD voltage below the set threshold triggers pending** write 0 clean this bit	R	0
30	lvdvcc_pending	**The VCC voltage below the set threshold triggers pending** write 0 clean this bit	R	0
29	Reserved		-	-
28:22	dbs_lo_limit	**Filter detects voltage signal low burr width** The unit is LVDDBS_CLK	RW	0
21:15	dbs_hi_limit	**Filter detects voltage signal high burr width**	RW	0

		The unit is LVDDBS_CLK		
14	lvdvcc_sync_dis	**Detection of VCC low voltage signal triggers LVDVCC low power wake-up source whether synchronization is required** 0x0: Synchronization is required 0x1: Out of sync	RW	0
13	lvdvdd_bps_en	**Enable the VDD voltage status signal after sampling debounce** 0x0: Closed 0x1: On	RW	0
12	lvdvcc_bps_en	**Enable the VCC voltage status signal after sampling debounce** 0x0: Off 0x1: On	RW	0
11	lvd_oe	**Enable interrupt after threshold judgment triggers condition, and reset function** 0x0: Off 0x1: On	RW	1
10	lvdvdd_rst_en	**The system is enabled after the VDD threshold is triggered** 0x0: Interrupted, no bit 0x1: Reset, not interrupted	RW	1
9	lvdvcc_rst_en	**Reset the system after the VCC threshold is triggered** 0x0: Interrupted, no more bit 0x1: Reset, not interrupted	RW	1
8:4	hlvd_s	**VCC LVD gear setting** 0x00:1.7 0x01:1.8 0x02:1.9 0x03:2 0x04:2.1 0x05:2.2 0x06:2.3 0x07:2.4 0x08:2.55 0x09:2.65 0x0A: 2.8 0x0B: 2.95 0x0C: 3.1 0x0D: 3.25 0x0E: 3.4 0x0F: 3.55 0x10:3.35 0x11:3.55 0x12:3.75	RW	0x9

		0x13:3.95 0x14:4.15 0x15:4.35 0x16:4.55 0x17:4.75		
3:2	llvd_s	**VDD LVD gear setting** 0x0:0.99 0x1:1.09 0x2:1.19 0x3:1.29	RW	0x2
1	llvd_en	**VDD LVD Enable bit** 0x0: Off 0x1: On	RW	1
0	hlvd_en	**VCC LVD Enable bit** 0x0: Closed 0x1: On	RW	1

6. Reset and clock control

6.1. Reset

The TX32M2300 series supports power reset and system reset and master reset.

6.1.1. System reset

A system reset will reset all registers except certain reset status registers and special function registers.

A system reset is produced when one of the following events occurs:

External IO port /LVDVCC low power /COMP0 wake up in SLEEP mode;

WDT count overflow reset

System reset Request reset

UART0 upgrade program reset

System lock reset

6.2. Master Reset

Primary reset can reset some registers that cannot be reset by system reset.

The following events can trigger a master reset:

Software reset

The PVD detects a low voltage event and the controller is in reset mode

When the chip supports MCLR reset, a low level on the MCLR pin lasting more than 1ms triggers the reset

6.2.1. Power reset

A power on/power off reset (POR/PDR reset) is a power reset. A power reset will reset all of the logical and analog modules.

The reset entry vector is fixed at address 0x0000_0004.

6.3. Clock

6.3.1. XOSC clock

High Speed External clock signals (XOSC) are generated by two types of clock sources:

- External crystal/ceramic resonators
- User External clock

6.3.2. HIRC clock

The HIRC clock signal is generated by an internal 26MHz oscillator and can be used directly as a system clock or as a PLL input. HIRC oscillators are capable of providing a system clock without the need for any external devices. It has a shorter startup time than an XOSC crystal oscillator. However, its clock frequency accuracy is poor even after calibration. For factory

calibration, the calibration values are written in the flash system storage area. Before using this clock, the program can read and configure the high precision HIRC clock. After factory validation, the HIRC accuracy is 26MHz(+/ -1.5%) at room temperature. The exact frequency is described in the 0x1FF0_0600-0x1FF0_06FF flash memory area, which can be read by the user to get the exact HIRC frequency.

If the XOSC crystal oscillator fails, the HIRC clock is used as a backup clock source. Refer to the Clock Safety System (CSS).

6.3.3. PLL

The internal PLL can use the internal clock source such as XOSC,HIRC, etc., as a reference, the system can configure the fractional frequency ratio to get the desired arbitrary clock frequency, and can be used as the system clock after the frequency division circuit.

6.3.4. LIRC clock

The LIRC oscillator acts as a low power clock source, serving as a system start clock for the watchdog and other units. The LIRC clock frequency is about 128KHz (between 90KHz and 166KHz). Refer to the section on electrical characteristics in your data manual for further information.

6.3.5. System Clock (SYSCLK) selection

Four different clock sources can be used to drive System Clock (SYSCLK) :

- Internal low speed 128Khz LIRC
- Internal high speed 26Mhz HIRC
- External high speed Crystal oscillator XOSC
- On-chip High speed PLL clock

6.3.6. DBSCLK selection

Four different clock sources can be used to drive GPIO's glitch debounce clock (DBSCLK) :

- External high speed crystal oscillator XOSC
- Internal high speed 26Mhz HIRC frequency division clock
- SYSCLK
- Internal low speed 128Khz LIRC

6.3.7. LVDDBS_CLK clock selection

Four different clock sources can be used to drive the LVD VCC/VDD debounceclock (LVDDBS_CLK), which works independently of the system clock:

- APBOCLK
- Internal high speed 26Mhz HIRC frequency division clock
- Internal low speed 128Khz LIRC
- External high speed crystal oscillator XOSC

6.3.8. CMPCLK Clock Selection

Four different clock sources can be used to drive compartor's clock (CMPCLK) :

- Internal low speed 128Khz LIRC
- Split clock for internal high speed 26Mhz HIRC
- System Clock
- External high speed crystal oscillator XOSC

When not in use, either clock source can be turned on or off independently to optimize system power consumption.

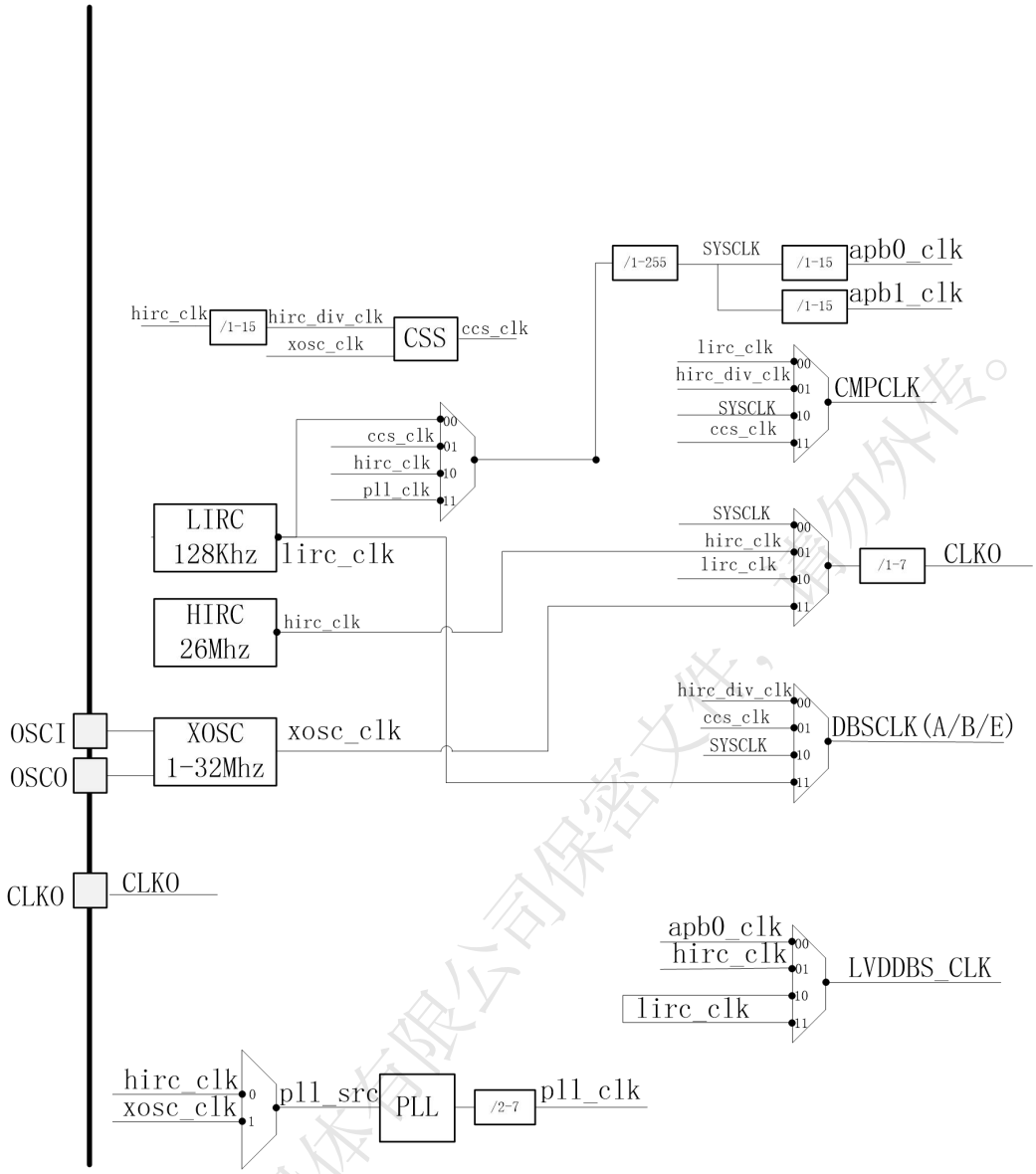

Figure 7-1 Clock structure

6.3.9. Clock Safety System (CSS)

A clock safety system can be activated by software. Make sure that XOSC has been successfully enabled when using this feature. If the XOSC clock fails, the clock safety interrupt CSSI is generated, allowing the software to complete the rescue operation. This CSSI interrupt is connected to the CPU's NMI interrupt. Note: Once the *CSS* is activated and the *XOSC* clock fails, the *CSS* interrupt is generated, and the *NMI* is also generated automatically. The *NMI* will be executed continuously until the *CSS* interrupt suspension bit is cleared. Therefore, the *NMI* handler must write 1 to clear the *CSS* interrupt by setting the hosc_loss_pending bit in the clock interrupt register, (HOSC_MNT). With register enable, clock failure will cause the internal XOSC clock to automatically switch to hirc_div_clk.

6.4. Registers

6.4.1. Register base address

Name	Base Address	Description
System	0x40020000	Base address of the System

6.4.2. Register list

Table 7-1 Register list

Offset Address	Name	Description
0x0000	SYS_KEY	System configuration key register
0x0004	SYS_CON0	System control register 0
0x0008	SYS_CON1	System control register 1
0x000c	SYS_CON2	System control Register 2
0x0010	SYS_CON3	System control Register 3
0x0014	SYS_CON4	System control Register 4
0x0018	SYS_CON5	System control Register 5
0x001c	SYS_CON6	System control register 6
0x0020	SYS_CON7	System control Register 7
0x0024	CLK_CON0	Clock control register 0
0x0028	CLK_CON1	Clock control register 1
0x002c	CLK_CON2	Clock control Register 2
0x0030	CLK_CON3	Clock control register 3
0x0034	CLK_CON4	Clock control Register 4
0x0038	CLK_CON5	Clock control register 5
0x003c	CLK_CON6	Clock control register 6
0x0040	CLK_CON7	Clock control register 7
0x0044	HOSC_MNT	XOSC control register
0x0048	SYS_ERR	System exception error register

0x004c	WKUP_CON	Wakeup register
0x0050	LP_CON	Low power register
0x0054	MBIST_CON	Memory Bist register
0x0058	MBIST_MISR	Memory Bist Rom result register
0x005c	RESERVED	
0x0060	MODE	Mode configuration register
0x0064	PMU_CON	PMU controller
0x0068	RPCON	System pending recording
0x006c	AMP_CON0	AMP control register 0
0x0070	AMP_CON1	AMP control register 1
0x0074	PMUBK	PMU lower power mode backup

6.4.3. Register definition

6.4.3.1. SYS_KEY

Bit(s)	Name	Description	R/W	Reset
31:0	sys_key	**System register configuration key** Write 0x3fac87e4 to make all system registers write enable, Setsys_keyother value will clear this bit other value will clear this bit. Read returns sys_key status 0x0: Locks all system register writes 0x1: Unlocks all system register writes	RW	0x1

6.4.3.2. SYS_CON0

Bit(s)	Name	Description	R/W	Reset
31	fast_rst_en	**Fast reset enabled** The reset time ranges from 1ms to 62us	R/W	0x0
30	sleep_goon_en	**Whether to continue running after wakeup is triggered** 0x0: Reset 0x1: Continue running	R/W	0x0
29:27	sleep_dly_cnt	**Delay goon run-time configuration after Wakeup** 0x0: two 128K RC cycles	R/W	0x0

		0x1: three 128K RC cycles ... 0x7: nine 128K RC cycles		
26	dbs_soft_rst_	**The GPIO debounce software resets** Write the 0 and then the 1 to complete the reset operation	R/W	0x1
25	crc_soft_rst_	**CRC software reset** Write the 0 first and then the 1 to complete the reset operation	R/W	0x1
24:21	Reserved		R/W	0xF
20	gpio_soft_rst_	**The GPIOA/GPIOB/GPIOE software resets** Write the 0 and then the 1 to complete the reset operation	R/W	0x1
19	adkey_soft_rst_	**ADCKey software reset** Write zeros and then ones to complete the reset operation	R/W	0x1
18	uart1_soft_rst_	**UART1 software reset** Write the 0 and then the 1 to complete the reset operation	R/W	0x1
17	uart0_soft_rst_	**UART0 software reset** Write 0 first, then 1 to complete the reset operation	R/W	0x1
16	spi1_soft_rst_	**SPI1 software reset** Write the 0 and then the 1 to complete the reset operation	R/W	0x1
15	spi0_soft_rst_	**SPI0 software resets** Write 0 first, then 1 to complete the reset operation	R/W	0x1
14	epwm_soft_rst_	**The EPWM software resets** Write the 0 and then the 1 to complete the reset operation	R/W	0x1
13:9	Reserved		R/W	0x1F
8	wdt_soft_rst_	**The WDT software resets** Write the 0 first and then the 1 to complete the reset operation	R/W	0x1
7	wdt_sys_soft_rst_	**The WDT SYS software resets** Write the 0 first and then the 1 to complete the reset operation	R/W	0x1
6	timer_soft_rst_	**TIMER software reset** Write the 0 first and then the 1 to complete the reset operation	R/W	0x1
5:0	Reserved	For future usage	R/W	0x3F

6.4.3.3. SYS_CON1

Bit(s)	Name	Description	R/W	Reset
31:14	Reserved		-	-
13	uart_update_dis	**Serial single-pin upgrade** 0x0: Open 0x1: Off	R/W	0x0
12	debug_en	**Sleep mode and debug linkage configuration** The Sleep and stopclk modes will wake up automatically when debugging the connection. 0x0: On 0x1: Off	R/W	0x0
11	int_remap_en	**Interrupts entry address remapping enabled** 0x0: EFLASH 0x1: SRAM	R/W	0x0
10	nmi_inv_sel	**NMI Pin detects signal reversal configuration** 0x0: High level triggers NMI 0x1: Low level triggers NMI	R/W	0x0
9	cp_mode_sysclk_en	**Switch system clock in CP mode** 0x0: Toggles 0x1: Keep ate-clk	R/W	0x1
8	swd_en	**SWD enabled** 0x0: Off 0x1: On	R/W	0x1
7	lvdvcc_wkup_en	**VCC LVD Wake up enabled** 0x0: Off 0x1: On	R/W	0x0
6	clk_test_oe	**Internal clock output to PA0** 0x0: Off 0x1: On	R/W	0x0
5	sys_err_resp_en	**Exception response feedback is enabled when the system bus accesses memory** 0x0: Off 0x1: On	R/W	0x0
4	sys_err_int_en	**The NMI interrupt is triggered when the system bus fails to access the memory** 0x0: Closed 0x1: On	R/W	0x0
3	hosc_loss_nmi_en	**Monitor XOSC loss triggers NMI interrupt** 0x0: Off 0x1: On	R/W	0x0
2	rxev_enable	**Enable PA1 as the CPU's receive event** 0x0: Off 0x1: On	R/W	0x0
1	nmi_int_enable	**Enable PA0 as the external NMI input to the**	R/W	0x0

		CPU 0x0: Off 0x1: On		
0	lockup_enable	Enabling system lock triggers system reset 0x0: Off 0x1: On	R/W	0x0

6.4.3.4. SYS_CON2

Bit(s)	Name	Description	R/W	Reset
31:30	Reserved	-	-	-
29:27	pe_deb_en	GPIOE debounce is enabled It corresponds to control GPIOE0 to GPIOE2	R/W	0x0
26:16	pb_deb_en	GPIOB debounce Is enabled It corresponds to control GPIOB0 to GPIOB10	R/W	0x0
15:0	pa_deb_en	GPIOA debounce is enabled It corresponds to control GPIOA0 to GPIOA15	R/W	0x0

6.4.3.5. SYS_CON3

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.6. SYS_CON4

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.7. SYS_CON5

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.8. SYS_CON6

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.9. SYS_CON7

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.10. CLK_CON0

Bit(s)	Name	Description	R/W	Reset
31:18	Reserved	-	R/W	-
17:16	lvd_debclk_sel	**Lvd debounce clock selection** 0x0: apb0_clk 0x1: HIRC 26M 0x2: LIRC 128K 0x3: LIRC 128K	R/W	0x0
15:14	comp_clk_sel	**Comparator clock selection** 0x0: LIRC 128K 0x1: divider output from HIRC 26M 0x2: sys_clk 0x3: XOSC	R/W	0x0
13:12	Reserved	-	R/W	-
11:10	gpioe_dbs_sel	**GPIOE debounce clock selection** 0x0: divider output from HIRC 26M 0x1: XOSC 0x2: sys_clk 0x3: LIRC 128K	R/W	0x0
9:8	gpiob_dbs_sel	**GPIOB debounce clock selection** 0x0: divider output from HIRC 26M 0x1: XOSC 0x2: sys_clk 0x3: LIRC 128K	R/W	0x0
7:6	gpioa_dbs_sel	**GPIOA debounce clock selection** 0x0: divider output from HIRC 26M 0x1: XOSC 0x2: sys_clk 0x3: LIRC 128K	R/W	0x0
5	Reserved	-	R/W	-
4	pll_refclk_sel	**Pll reference Clock selection** 0x0: HIRC 26M 1:XOSC	R/W	0x0
3:2	clock_to_io_sel	**Output clock Selection** 0x0: sys_clk 0x1: HIRC 26M 0x2: LIRC 32K 0x3: XOSC	R/W	0x0
1:0	sysclk_sel	**System Clock Selection**	R/W	0x0

		0x0: LIRC 128K 0x1: XOSC 0x2: HIRC 26M 0x3: pll_clk		

6.4.3.11. CLK_CON1

Bit(s)	Name	Description	R/W	Reset
31	Reserved	-	-	-
30:28	clk_to_io_div	**Clock to IO allocation ratio** 0x0: divide by 1 0x1: divide by 2 0x2: divide by 3 ... 0x6: divide by 7 0x7: close clock output	R/W	0x0
27:24	hirc_clk_div	**HIRC 26M clock frequency division ratio** 0x0: divide by 1 0x1: divide by 2 0x2: divide by 3 ... 0xE: divide by 15 0xF: close HIRC divide clock	R/W	0x0
23:19	Reserved	-	-	-
18:16	pll_clk_div	**pll clock frequency division ratio** when n=[1,6], pll_clk divide by n+1 when n=7,stoppll_clk n=0 is forbidden	R/W	0x1
15:12	apb1clk_div	**PB1 clock frequency division ratio** 0x0: divide by 1 0x1: divide by 2 0x2: divide by 3 ... 0xE: divide by 15 0xF: close apb1_clk	R/W	0x0
11:8	apb0clk_div	**APB0 clock division ratio** 0x0: divide by 1 0x1: divide by 2 0x2: divide by 3 ... 0xE: divide by 15 0xF: close apb0_clk	R/W	0x0
7:0	sysclk_div	**System clock frequency division ratio** 0x0: divide by 1 0x1: divide by 2	R/W	0x0

		0x2: divide by 3 ... 0xFE: divide by 255 0xFF: close sys_clk		

6.4.3.12. CLK_CON2

Bit(s)	Name	Description	R/W	Reset
31	reserved	-	-	-
30	clk_source_en_bps	**pll_clk and xosc clock glitch free??** **Pll_clk and XOSC Clock glitch free??** 0x0: Open 0x1: Off	R/W	0x1
29	cp_clk_en	**CP mode enables CPU clock** 0x0: Off 0x1: On	R/W	0x0
28	test_clk_en	**ATE Clock output Enable** 0x0: Off 0x1: On	R/W	0x0
27	comp_clk_en	**Comparator clock enabled** 0x0: Off 0x1: On	R/W	0x1
26	Reserved	-	-	-
25	crc_clk_en	**CRC clock is enabled** 0x0: Off 0x1: On	R/W	0x1
24	eflash_mem_clk_en	**eflash erase/program Clock enabled** 0x0: Off 0x1: On	R/W	0x1
23	Reserved	-	-	-
22	hwdiv_clk_en	**Hardware divider clock enabled** 0x0: Off 0x1: On	R/W	0x0
21:19	Reserved	-	-	-
18	uart1_clk_en	**UART1 Clock enabled** 0x0: Off 0x1: On	R/W	0x1
17	uart0_clk_en	**UART0 Clock enabled** 0x0: Off 0x1: On	R/W	0x1
16	spi1_clk_en	**SPI1 Clock enabled** 0x0: Off 0x1: On	R/W	0x1
15	spi0_clk_en	**SPI0 Clock enabled** 0x0: Off	R/W	0x1

		0x1: On		
14	epwm_clk_en	**EPWM clock is enabled** 0x0: Off 0x1: On	R/W	0x1
13	timer5_clk_en	**TIMER5 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
12	timer3_clk_en	**TIMER3 Clock is enabled** 0x0: Closed 0x1: On	R/W	0x1
11	timer2_clk_en	**TIMER2 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
10	timer1_clk_en	**TIMER1 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
9	timer0_clk_en	**TIMER0 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
8	wdt_clk_en	**WDT clock enabled** 0x0: Off 0x1: On	R/W	0x1
7	wdt_sys_clk_en	**WDT sys clock is enabled** 0x0: Off 0x1: On	R/W	0x1
6	timer4_clk_en	**TIMER4 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
5:3	Reserved	-	-	-
2	sram0_clk_en	**SRAM0 Clock enabled** 0x0: Off 0x1: On	R/W	0x1
1	ahb1_clk_en	**AHB1 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1
0	ahb0_clk_en	**AHB0 Clock is enabled** 0x0: Off 0x1: On	R/W	0x1

6.4.3.13. CLK_CON3

Bit(s)	Name	Description	R/W	Reset
31	hrcosc_en_flag	**Working status of HIRC 26M** 0x0: not ready 0x1: ready	RO	0x1

30:29	Reserved	-	-	-
28	hrcosc_vtest2	**The VDD test is enabled** Output VTSOUT 0x0: Off 0x1: On	R/W	0x0
27	hrcosc_vtest1	**The VDDOSC test is enabled** Output VTSOUT 0x0: Off 0x1: On	R/W	0x0
26	hrcosc_vsel	**LDO Reference Selection** 0x0: Voltage bias 0x1: Current	R/W	0x0
25:19	hrcosc_sc	**HRCOSC frequency control** 0x0: lowest ... 0x7F: highest	R/W	0x0
18	hrcosc_ldos	**RCOSC LDO voltage selection** 0x0:1.5v 0x1:1.6v	R/W	0x0
17	hrcosc_en	**RCOSC is enabled** 0x0: disable 0x1: enable	R/W	0x1
16:15	hxosc_fbres		R/W	0x3
14:10	hxosc_dr		R/W	0x10
9:6	hxosc_cto		R/W	0x0
5:2	hxosc_cti		R/W	0x0
1	hxosc_hy		R/W	0x1
0	hxosc_en	**XOSC enabled** 0x0: Off 0x1: On	R/W	0x0

6.4.3.14. CLK_CON4

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.15. CLK_CON5

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.16. CLK_CON6

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.17. CLK_CON7

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.18. HOSC_MNT

Bit(s)	Name	Description	R/W	Reset
31:16	high_limit	**XOSC work limit**	R/W	0x1450
15	hosc_mnt_en	**XOSC monitoring is enabled** 0x0: Disabled 0x1: On	R/W	0x0
14	hosc_loss_pending	**Read register** 0x0: Normal state 0x1: Suspended, indicating that external xosc failed **Write register** 0x0: Clear pending 0x1: Invalid	R/W	0x0
13	hosc_loss_sw_en	**The hardware automatically switches the osc clock from xosc to HIRC when an external osc failure is detected** 0x0: Off 0x1: On	R/W	0x0
12:0	low_limit	**XOSC working lower limit limit**	R/W	0xA7

6.4.3.19. SYS_ERR

Bit(s)	Name	Description	R/W	Reset
31:1	Reserved	-	-	-
1	clk_err	**Clock exception flag** 0x0: Working properly 0x1: When clock is used incorrectly, this bit can be set by hardware and cleared by software	R/W	0
0	sys_err0	**System access exception** This hang causes an NMI interrupt if sys_err_int_en is set	R/W	0

clk_err trigger situation			
	lirc disable	hirc disable	xosc disable
sys_clk_sel_lirc	Square root		
sys_clk_sel_hirc		Square root	
sys_clk_sel_xosc			Square root
sys_clk_sel_pll& pll_ref_sel_lirc			
sys_clk_sel_pll& pll_ref_sel_hirc		Square root	
sys_clk_sel_pll& pll_ref_sel_xosc			Square root

6.4.3.20. WKUP_CON

Bit(s)	Name	Description	R/W	Reset
31:1	Reserved	-	-	-
29:24	wkup_pnd	**Edge detection trigger of Wakeup IO** 0x0: No trigger 0x1: Triggered Corresponding relationship: BIT24->port0, BIT25->port1 and so on	RO	-
23:22	Reserved	-	-	-
21:16	clr_pnd	**Wakeup pending Clear** **Write 1 Clear**	WO	-
15:14	Reserved	-	-	-
13:8	wkup_edge	**Wakeup's edge wake up selection** 0x0: Rising edge trigger 0x1: Falling edge triggered	R/W	0x0
7:6	Reserved	-	-	-
5:0	wkup_en	**Wakeup enabled** 0x0: Turned off 0x1: On BIT0~BIT3: External GPIO wakes up BIT4: Comparator wakes up BIT5: LVD wakes up	R/W	0x0

6.4.3.21. LP_CON

Bit(s)	Name	Description	R/W	Reset
31:9	Reserved	-	-	-
8	lp_mode	**LP Enable** 0x0: Off	0x0	R/W

		0x1: On		
7	lp_mode_auto_en	**LP mode is automatically enabled in Sleep mode** 0x0: Off 0x1: Open	0x0	R/W
6	hirc_auto_enable	**When XOSC detects an exception, it automatically enables and switches HIRC26M** 0x0: Off 0x1: On	0x0	R/W
5	rc32k_soft_en	**LIRC Enabled** 0x0: Disables 0x1: On	0x1	R/W
4	rc32k_auto_dis	**Go to sleep mode to automatically turn off LIRC** 0x0: Off 0x1: On	0x0	R/W
3	hirc_auto_dis	**Enter sleep/stop mode to automatically shut down LIRC** 0x0: Off 0x1: On	0x0	R/W
2	sram0_auto_dis	**Enter sleep/stop mode to automatically shut down SRAM** 0x0: Off 0x1: On	0x0	R/W
1	stopclk	**Go into stop mode** 0x0: Off 0x1: On	0x0	R/W
0	sleep	**Go into sleep mode** 0x0: Closed 0x1: On	0x0	R/W

6.4.3.22. MBIST_CON

Bit(s)	Name	Description	R/W	Reset
31:21	Reserved	-	-	-
20	mbist_fail_h	**MBIST exception flag** Read action: 0x1: MBIST exception Write operation: 0x1: Clear the exception flag	R/W	0x0
19	mbist_tst_done	**MBIST Finish flag** Read action: 0x1: MBIST test complete Write operation: 0x1: Clear complete flag	R/W	0x0

18	mbist_clk_en	**MBIST clock is enabled** 0x0: Off 0x1: On	R/W	0x0
17:12	Reserved	-	-	-
11	mbist_rf1p_debugz	**MBIST rf1p debugging enabled** 0x0: Closed 0x1: On	R/W	0x0
10	mbist_rf1p_hold_l	**MBIST rf1p debugging enabled** 0x0: Off 0x1: On	R/W	0x0
9	mbist_rf1p_test_h	**MBIST rf1p debugging enabled** 0x0: Off 0x1: On	R/W	0x0
8:0	Reserved	-	-	-

6.4.3.23. MBIST_MISR

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.24. MODE

Bit(s)	Name	Description	R/W	Reset
31:0	Reserved	-	-	-

6.4.3.25. PMU_CON

Bit(s)	Name	Description	R/W	Reset
31:23	Reserved		-	-
22:20	dset		R/W	0x4
19:16	pdcore	**VDD drop down Signal selection** 0x0:304pa 0x1:68ua 0x2:263ua 0x3:330ua 0x4:1.31 ma 0x5:1.38ma 0x6:1.57ma 0x7:1.64 ma 0x8:5.24 ma 0x9:5.31ma 0xA: 5.51ma 0xB: 5.57ma	R/W	0x4

		0xC: 6.56ma 0xD: 6.62ma 0xE: 6.81ma 0xF: 6.89ma		
15:12	lpset	**LP mode Close module** 0x0: Off 0x1: On	R/W	0x0
11	lpen	**LP mode is enabled** 0x0: Disabled 0x1: On	R/W	0x0
10	vrefaen	**VREF is enabled** 0x0: Off 0x1: On	R/W	0x1
9	tsensen	**Internal temperature sensor** 0x0: Off 0x1: On	R/W	0x0
8	irefen	**Bias current Enable** It needs to be set to 0 for low power mode 0x0: Off 0x1: On	R/W	0x1
7	bor5en	**VCC POR enabled** 0x0: Disabled 0x1: On	R/W	0x1
6:3	vddset	**VDD voltage gear selection** 0x0:1.2v 0x1:1.25v 0x2:1.3v 0x3:1.35v 0x4:1.4v 0x5:1.45v 0x6:1.5v 0x7:1.55v 0x8:1.6v 0x9:1.65v 0xA: 1.7v 0xB: 1.75v 0xC: 1.8v 0xD-0xF: 1.85v	R/W	0x6
2:0	vbgset	**BGR voltage gear selection** 0x0:1.118v 0x1:1.145v 0x2:1.174v 0x3:1.206v 0x4:1.230v 0x5:1.259v 0x6:1.287v	R/W	0x3

		0x7:1.287v		

Note: vbgset/ vddset will load trim values during power-on, so they may not read back the default values refer to the table above.

6.4.3.26. RPCON

Bit(s)	Name	Description	R/W	Reset
31:20	Reserved		-	-
19	uart0_reset_clr	**Uart0 Reset pending Clear** Write 1 Clear	WO	0
18	lockup_reset_clr	**Lockup resets pending Clear** Write 1 Clear	WO	0
17	soft_reset_clr	**Software reset pending purge** Write 1 Clear	WO	0
16	sleep_sta_clr	**Sleep pending Clear** Write 1 Clear	WO	0
15:4	Reserved		-	-
3	uart0_update_pending	**Uart0 reset pending**	RO	0
2	lock_reset_pending	**Lockup resets pending**	RO	-
1	soft_reset_pending	**Software reset pending** Write 1 Reset	RW	-
0	sleep_pending	**Sleep pending**	RO	-

6.4.3.27. AMP_CON0

Bit(s)	Name	Description	R/W	Reset
31:27	qcsel3	**AMP3 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1
26:24	gainsel3	**AMP3 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
23:19	qcsel2	**AMP2 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1

18:16	gainsel2	**AMP2 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
15:11	qcsel1	**AMP1 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1
10:8	gainsel1	**AMP1 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
7	ampldo_en	**AMP LDO enabled** 0x0: Off 0x1: On	R/W	0x0
6	bias_en	**BIAS Enable** 0x0: Off 0x1: On	R/W	0x0
5	biasadd_en	**BIASADD is enabled** 0x0: Closed 0x1: On	R/W	0x0
4	rcchan_en	**RCCHAN enabled** 0x0: Disabled 0x1: On	R/W	0x0
3	vcmbuff_en	**VCMBUFF is enabled** 0x0: Disabled 0x1: On	R/W	0x0
2	amp3_en	**AMP3 is enabled** 0x0: Off 0x1: On	R/W	0x0
1	amp2_en	**AMP2 is enabled** 0x0: Disabled 0x1: On	R/W	0x0
0	ampl_en	**AMP1 enabled** 0x0: Off 0x1: On	R/W	0x0

6.4.3.28. AMP_CON1

Bit(s)	Name	Description	R/W	Reset
31:13	Reserved	-	-	-
12	vout2pad_en	**AMP2 output to PB2 Enable** 0x0: Off 0x1: On	R/W	0x0
11	vout1pad_en	**AMP1 output to PB1 Enable** 0x0: Off 0x1: On	R/W	0x0
10:9	rcrsel	**AMP3 RC Filter resistor selection** 0x0:1K 0x1:10K 0x2:50K 0x3:100K	R/W	0x0
8:6	restrim3	**AMP3 feedback Configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X 0x5-0x7: disable feedback loop	R/W	0x2
5:3	restrim2	**AMP2 feedback configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X 0x5-0x7: disable feedback loop	R/W	0x2
2:0	restrim1	**AMP1 feedback configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X 0x5-0x7: disable feedback loop	R/W	0x2

6.4.3.29. PMUBK

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	lpen_bk	**backup register of lpen** 0x0: Off 0x1: On	R/W	0x1
14:11	lpset_bk	**backup register of lpset**	R/W	0xF

		0x0: Off 0x1: Open		
10:7	pdcore_bk	**backup register of pdcore** 0x0:304pa 0x1:68ua 0x2:263ua 0x3:330ua 0x4:1.31 ma 0x5:1.38ma 0x6:1.57ma 0x7:1.64 ma 0x8:5.24 ma 0x9:5.31ma 0xA: 5.51ma 0xB: 5.57ma 0xC: 6.56ma 0xD: 6.62ma 0xE: 6.81ma 0xF: 6.89ma	R/W	0x0
6:3	vddset_bk	**backup register of VDD** 0x0:1.2v 0x1:1.25v 0x2:1.3v 0x3:1.35v 0x4:1.4v 0x5:1.45v 0x6:1.5v 0x7:1.55v 0x8:1.6v 0x9:1.65v 0xA: 1.7v 0xB: 1.75v 0xC: 1.8v 0xD-0xF: 1.85v		0x6
2:0	vbgset_bk	**backup register of BGR** 0x0:1.118v 0x1:1.145v 0x2:1.174v 0x3:1.206v 0x4:1.230v 0x5:1.259v 0x6:1.287v 0x7:1.287v	R/W	0x3

6.4.3.30. Chip Info 0 (0x1FF00600)

Bit(s)	Name	Description	R/W	Reset
31:0	UID0	UID information 0	RO	0xFFFF FFFF

6.4.3.31. Chip Info 1 (0x1FF00604)

Bit(s)	Name	Description	R/W	Reset
31:0	UID1	UID Information 1	RO	0xFFFF FFFF

6.4.3.32. Chip Info 2 (0x1FF00608)

Bit(s)	Name	Description	R/W	Reset
31:0	UID2	UID Information 2	RO	0xFFFF FFFF

7. GPIO

Each set of GPIO ports has four 32-bit configuration registers (GPIOx_MODER,GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), One 32-bit bit/reset register (GPIOx_BSRR) and one 32-bit flip register (GPIOx_TGL). In addition, all GPIOs have one 32-bit lock register (GPIOx_LCKR) and two multiplexing function select registers (GPIOx_AFRH and GPIOx_AFRL).

7.1. Gpios main features

- Output status: push-pull or open leak + pull up and down
- Output register status (GPIOx_ODR) or peripheral functionoutput (alternatefunctionoutput)
- Each I/O driver capability is configurable
- Input status: Float, pull up, analog

- Enter data into a data register (GPIOx_IDR) or peripheral (alternate function input)
- Independent set/reset/flip IO state (GPIOx_BSRR, GPIOx_TGL)
- Lock the IO state by locking configuration (GPIOx_LCKR)
- Analog features
- Reuse function

7.2. GPIO function description

Each port of GPIO can be configured to the following states independently by software:

- Input float
- Input pull-up
- Enter drop-down
- Analog function
- Open leakage output (pull-up or pull-down)
- Push-pull output (pull-up or pull-down)
- Reuse function (open drain or push-pull, pull-up or pull-down)

Table 8-1 Mode configuration table of GPIO

MODE	OTYPE(i)	OSPEED(i)	PUPD(i)		IO configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	Reserved	
	0		1	0		
	0		1	1		
	1		0	0	GP output	OD
	1		0	1	GP output	OD+PU
	1		1	0	Reserved	
	1		1	1		
10	0	SPEED[1:0]	0	0	AF	PP
	0		0	1	Reserved	
	0		1	0		
	0		1	1		

	1			0	0	AF	OD
	1			0	1	AF	OD+PU
	1			1	0	Reserved	
	1			1	1		
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved	
11	x	x	x	0	0	Input/Output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

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

7.2.1. General Purpose IO (GPIO)

During and just after the reset, the multiplexing function is not enabled and the I/O port is configured to float input mode. DBG_CLK The IO can be configured to pull up or pull down, depending on the customer's own configuration.

When configured as an output, the value written to the output data register (GPIOx_ODR) is output to the corresponding I/O pin. The output drive can be used in push-pull or open-leak mode.

The input data register (GPIOx_IDR) captures data on the I/O pins on each AHB clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down, which can be activated or disconnected when configured as input.

7.2.2. Separate bit operation

When programming individual bits of GPIOx_ODR, the software need not disable interrupts: only one or more bits can be changed in a single AHB write operation. You only need to write "1" to the bit you want to change in the "set/reset register"

(GPIOx_BSR) or the "take back register" (GPIOx_TGL). Bits that are not selected will not be changed.

7.2.3. Multiplexing function (AF)

The chip IO pin is connected to the on-chip device through a multiplexer, and only one multiplexing function can be gated on each IO at a time. Each IO pin has a 4-input multiplexer connected to the multiplexing function (AF0~AF3) by configuring GPIOx_AFRH/L to select the input function. If the port is configured as the multiplexed output function, the pin is disconnected from the output register and connected to the output signal of the on-chip peripheral. If the software configs a GPIO pin to multiplexed output, but the peripheral is not activated, its output will be uncertain.

7.2.4. GPIO lock mechanism

The lock mechanism allows the GPIO control register GPIOx_LCKR to execute a string of lock procedures and then lock the GPIO state. Once the GPIO state is locked, it will not be changed until the CPU is reset. The locked registers are (GPIOx_MODER, GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH).

Refer to the GPIOx_LCKR register description for the locked sequence.

7.2.5. Input Configuration

When the I/O port is configured as input:

- Output buffers are disabled
- Schmidt trigger input is activated
- Depending on the input configuration (pull-up, pull-down, or float), weak pull-up and pull-down resistors are connected

- The data that appears on the I/O pin is sampled into the input data register at each AHB clock
- Read access to the input data register yields the I/O status

7.2.6. Output Configuration

When the I/O port is configured to output:

- The output buffer is activated
  - ✓ Open leak mode: A "0" on the output register activates the N-MOS, while a "1" on the output register puts the port in a high resistance state (P-MOS is never activated).
  - ✓ Push-pull mode: A "0" on the output register activates the N-MOS, while a "1" on the output register will activate the P-MOS.
- Schmidt triggers the input to be activated
- Weak pull-up and pull-down resistors are activated according to configuration
- The data that appears on the I/O pin at each AHB clock is sampled to the input data register
- In open leak mode, read access to the input data register yields the I/O status
- In push-pull mode, read access to the output data register gets the value of the last write.

7.2.7. Analog input configuration

When the I/O port is configured to simulate input configuration:

- Output buffers are disabled;
- Disables Schmidt trigger input, achieving zero consumption on each analog I/O pin. The Schmidt trigger output value is forced to be "0";
- Weak pull-up and pull-down resistors are prohibited;

- The value is "0" when reading the input data register.

7.3. Registers

7.3.1. Register base address

Name	Base Address	Description
GPIOA	0x40020B00	GPIOA base address
GPIOB	0x40020C00	GPIOB base address
GPIOC	0x40020F00	GPIOE base address

7.3.2. Register list

Offset Address	Name	Description
0x0000	GPIOA_MODE	GPIOA mode Select Register
0x0004	GPIOA_OTYPE	GPIOA Output mode register
0x0008	GPIOA_OSPEEDL	GPIOA Driver Capability Configuration Register
0x000c	GPIOA_OSPEEDH	GPIOA Drive capability configuration register
0x0010	GPIOA_PUPD	GPIOA pull down the configuration register
0x0014	GPIOA_IDR	GPIOA input register
0x0018	GPIOA_ODR	GPIOA output register
0x001c	GPIOA_BSR	GPIOA BIT Set and BIT Reset register
0x0020	GPIOA_LCK	GPIOA lock register
0x0024	GPIOA_AFRL	GPIOA Multi-function Configuration Register L
0x0028	GPIOA_AFRH	GPIOA Multi-function Configuration Register H
0x002c	GPIOA_TGL	GPIOA flip register
0x0030	GPIOA_IMK	GPIOA interrupt register

Offset Address	Name	Description
0x0000	GPIOB_MODE	GPIOB mode selection register

0x0004	GPIOB_OTYPE	GPIOB Output mode register
0x0008	GPIOB_OSPEEDL	GPIOB Drive capability Configuration register
0x000c	GPIOB_OSPEEDH	GPIOB Drive capability configuration register
0x0010	GPIOB_PUPD	GPIOB Pull-down configuration register
0x0014	GPIOB_IDR	GPIOB input register
0x0018	GPIOB_ODR	GPIOB Output register
0x001c	GPIOB_BSR	GPIOB BIT Set and BIT Reset registers
0x0020	GPIOB_LCK	GPIOB lock register
0x0024	GPIOB_AFRL	GPIOB Multi-function configuration register L
0x0028	GPIOB_AFRH	GPIOB Multi-function Configuration register H
0x002c	GPIOB_TGL	GPIOB flip register
0x0030	GPIOB_IMK	GPIOB interrupt register

Offset Address	Name	Description
0x0000	GPIOE_MODE	GPIOE mode Select register
0x0004	GPIOE_OTYPE	GPIOE output mode register
0x0008	GPIOE_OSPEEDL	GPIOE driver capability configuration register
0x000c	GPIOE_OSPEEDH	GPIOE Driver capability configuration register
0x0010	GPIOE_PUPD	GPIOE pull-down configuration register
0x0014	GPIOE_IDR	GPIOE input register
0x0018	GPIOE_ODR	GPIOE output register
0x001c	GPIOE_BSR	GPIOE BIT Set and BIT Reset register
0x0020	GPIOE_LCK	GPIOE lock register
0x0024	GPIOE_AFRL	GPIOE multi-function configuration register L
0x0028	GPIOE_AFRH	GPIOE multi-function configuration register H
0x002c	GPIOE_TGL	GPIOE flip register
0x0030	GPIOE_IMK	GPIOE interrupt register

7.3.3. Register Detail Definition

7.3.3.1. GPIOx_MODE

Bit(s)	Name	Description	R/W	Reset
31:30	MODER15	**GPIOx15 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
29:28	MODER14	**GPIOx14 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
27:26	MODER13	**GPIOx13 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
25:24	MODER12	**GPIOx12 mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
23:22	MODER11	**GPIOx11 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog input	RW	0x0
21:20	MODER10	**GPIOx10 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
19:18	MODER9	**GPIOx9 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
17:16	MODER8	**GPIOx8 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0

15:14	MODER7	**GPIOx7 mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
13:12	MODER6	**GPIOx6 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
11:10	MODER5	**GPIOx5 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
9:8	MODER4	**GPIOx4 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
7:6	MODER3	**GPIOx3 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
5:4	MODER2	**GPIOx2 mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF Multi-function mode 0x3: Analog Input	RW	0x0
3:2	MODER1	**GPIOx1 Mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0
1:0	MODER0	**GPIOx0 mode selection** 0x0: Input mode 0x1: Output mode 0x2: AF multifunction mode 0x3: Analog Input	RW	0x0

7.3.3.2. GPIOx_OTYPE

Bit(s)	Name	Description	R/W	Reset

31:16	Reserved	-	-	-
15	OT15	**GPIOx15 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
14	OT14	**GPIOx14 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
13	OT13	**GPIOx13 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
12	OT12	**GPIOx12 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
11	OT11	**GPIOx11 Type of output** 0x0: push-pull output 0x1: Open leakage output	RW	0x0
10	OT10	**GPIOx10 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
9	OT9	**GPIOx9 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
8	OT8	**GPIOx8 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
7	OT7	**GPIOx7 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
6	OT6	**GPIOx6 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
5	OT5	**GPIOx5 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
4	OT4	**GPIOx4 Output type** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
3	OT3	**GPIOx3 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
2	OT2	**GPIOx2 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0
1	OT1	**GPIOx1 Type of output** 0x0: Push-pull output	RW	0x0

		0x1: Open leakage output		
0	OT0	**GPIOx0 Type of output** 0x0: Push-pull output 0x1: Open leakage output	RW	0x0

7.3.3.3. GPIOx_OSPEEDL

Bit(s)	Name	Description	R/W	Reset
31:28	OSPEED7	**GPIOx7 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
27:24	OSPEED6	**GPIOx6 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
23:20	OSPEED5	**GPIOx5 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
19:16	OSPEED4	**GPIOx4 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
15:12	OSPEED3	**GPIOx3 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
11:8	OSPEED2	**GPIOx2 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
7:4	OSPEED1	**GPIOx1 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
3:0	OSPEED0	**GPIOx0 Drive capability configuration**	RW	0x0

		0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3		

7.3.3.4. GPIOx_OSPEEDH

Bit(s)	Name	Description	R/W	Reset
31:28	OSPEED15	**GPIOx15 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
27:24	OSPEED14	**GPIOx14 Driver capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
23:20	OSPEED13	**GPIOx13 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
19:16	OSPEED12	**GPIOx12 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
15:12	OSPEED11	**GPIOx11 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
11:8	OSPEED10	**GPIOx10 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
7:4	OSPEED9	**GPIOx9 Drive capability configuration** 0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3	RW	0x0
3:0	OSPEED8	**GPIOx8 Drive capability configuration**	RW	0x0

		0x0: low speed 0x1: speed1 0x2: speed2 0x3: speed3		

7.3.3.5. GOIOx_PUPD

Bit(s)	Name	Description	R/W	Reset
31	PD15	**GPIOx15 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
30	PD14	**GPIOx14 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
29	PD13	**GPIOx13 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
28	PD12	**GPIOx12 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
27	PD11	**GPIOx11 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
26	PD10	**GPIOx10 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
25	PD9	**GPIOx9 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
24	PD8	**GPIOx8 Pull-down Resistor (10K) selection** 0x0: Closed 0x1: On	RW	0x0
23	PD7	**GPIOx7 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
22	PD6	**GPIOx6 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
21	PD5	**GPIOx5 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
20	PD4	**GPIOx4 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
19	PD3	**GPIOx3 Pull-down Resistor (10K) selection**	RW	0x0

		0x0: Off 0x1: On		
18	PD2	**GPIOx2 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
17	PD1	**GPIOx1 pull-down resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
16	PD0	**GPIOx0 Pull-down Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
15	PU15	**GPIOx15 Pull-up Resistor (10K) selection** 0x0: Off 0x1: Open	RW	0x0
14	PU14	**GPIOx14 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
13	PU13	**GPIOx13 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
12	PU12	**GPIOx12 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
11	PU11	**GPIOx11 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
10	PU10	**GPIOx10 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
9	PU9	**GPIOx9 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
8	PU8	**GPIOx8 Pull-up Resistor (10K) selection** 0x0: Closed 0x1: On	RW	0x0
7	PU7	**GPIOx7 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
6	PU6	**GPIOx6 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
5	PU5	**GPIOx5 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
4	PU4	**GPIOx4 Pull-up Resistor (10K) selection**	RW	0x0

		0x0: Off 0x1: On		
3	PU3	**GPIOx3 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
2	PU2	**GPIOx2 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
1	PU1	**GPIOx1 pull-up resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0
0	PU0	**GPIOx0 Pull-up Resistor (10K) selection** 0x0: Off 0x1: On	RW	0x0

7.3.3.6. GPIOx_IDR

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	IDR15	**GPIOx15 Input data read**	RO	0x0
14	IDR14	**GPIOx14 Input data read**	RO	0x0
13	IDR13	**GPIOx13 Input data read**	RO	0x0
12	IDR12	**GPIOx12 Input data read**	RO	0x0
11	IDR11	**GPIOx11 Input data read**	RO	0x0
10	IDR10	**GPIOx10 Input data read**	RO	0x0
9	IDR9	**GPIOx9 Input data read**	RO	0x0
8	IDR8	**GPIOx8 Input data read**	RO	0x0
7	IDR7	**GPIOx7 Input data read**	RO	0x0
6	IDR6	**GPIOx6 Input data read**	RO	0x0
5	IDR5	**GPIOx5 Input data read**	RO	0x0
4	IDR4	**GPIOx4 Input data read**	RO	0x0
3	IDR3	**GPIOx3 Input data read**	RO	0x0
2	IDR2	**GPIOx2 Input data read**	RO	0x0
1	IDR1	**GPIOx1 Input data read**	RO	0x0
0	IDR0	**GPIOx0 Input data read**	RO	0x0

7.3.3.7. GPIOx_ODR

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	ODR15	**GPIOx15 output register**	RW	0x0
14	ODR14	**GPIOx14 output register**	RW	0x0
13	ODR13	**GPIOx13 Output register**	RW	0x0

12	ODR12	**GPIOx12 output register**	RW	0x0
11	ODR11	**GPIOx11 output register**	RW	0x0
10	ODR10	**GPIOx10 output register**	RW	0x0
9	ODR9	**GPIOx9 output register**	RW	0x0
8	ODR8	**GPIOx8 output register**	RW	0x0
7	ODR7	**GPIOx7 output register**	RW	0x0
6	ODR6	**GPIOx6 Output register**	RW	0x0
5	ODR5	**GPIOx5 output register**	RW	0x0
4	ODR4	**GPIOx4 output register**	RW	0x0
3	ODR3	**GPIOx3 output register**	RW	0x0
2	ODR2	**GPIOx2 output register**	RW	0x0
1	ODR1	**GPIOx1 Output register**	RW	0x0
0	ODR0	**GPIOx0 output register**	RW	0x0

7.3.3.8. GPIOx_BSR

Bit(s)	Name	Description	R/W	Reset
31	BR15	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
30	BR14	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
29	BR13	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
28	BR12	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
27	BR11	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
26	BR10	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
25	BR9	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
24	BR8	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
23	BR7	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
22	BR6	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
21	BR5	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
20	BR4	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
19	BR3	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
18	BR2	**Bit Reset configuration**	WO	0x0

		Write 1 valid, when valid, GPIO outputs low		
17	BR1	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
16	BR0	**Bit Reset configuration** Write 1 valid, when valid, GPIO outputs low	WO	0x0
15	BS15	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
14	BS14	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
13	BS13	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
12	BS12	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
11	BS11	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
10	BS10	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
9	BS9	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
8	BS8	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
7	BS7	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
6	BS6	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
5	BS5	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
4	BS4	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
3	BS3	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
2	BS2	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
1	BS1	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0
0	BS0	**Bit Set configuration** Write 1 valid, when valid, GPIO outputs high	WO	0x0

7.3.3.9. GPIOx_LCK

Bit(s)	Name	Description	R/W	Reset
31:17	Reserved	-	-	-
16	LCKEN	**GPIO Lock is effectively enabled** 0x0: Off	RW	0x0

		0x1: On		
15	LCK15	**GPIOx15 Lock enabled** 0x0: Off 0x1: On	RW	0x0
14	LCK14	**GPIOx14 Lock enabled** 0x0: Off 0x1: On	RW	0x0
13	LCK13	**GPIOx13 Lock enabled** 0x0: Off 0x1: On	RW	0x0
12	LCK12	**GPIOx12 Lock enabled** 0x0: Off 0x1: On	RW	0x0
11	LCK11	**GPIOx11 Lock enabled** 0x0: Off 0x1: On	RW	0x0
10	LCK10	**GPIOx10 Lock enabled** 0x0: Closed 0x1: On	RW	0x0
9	LCK9	**GPIOx9 Lock enabled** 0x0: Off 0x1: On	RW	0x0
8	LCK8	**GPIOx8 Lock enabled** 0x0: Off 0x1: On	RW	0x0
7	LCK7	**GPIOx7 Lock enabled** 0x0: Off 0x1: On	RW	0x0
6	LCK6	**GPIOx6 Lock Enable** 0x0: Off 0x1: On	RW	0x0
5	LCK5	**GPIOx5 Lock enabled** 0x0: Off 0x1: On	RW	0x0
4	LCK4	**GPIOx4 Lock enabled** 0x0: Off 0x1: On	RW	0x0
3	LCK3	**GPIOx3 Lock is enabled** 0x0: Off 0x1: On	RW	0x0
2	LCK2	**GPIOx2 Lock enabled** 0x0: Off 0x1: On	RW	0x0
1	LCK1	**GPIOx1 Lock enabled** 0x0: Off	RW	0x0

		0x1: Open		
0	LCK0	**GPIOx0 Lock enabled** 0x0: Off 0x1: On	RW	0x0

To lock GPIO, you need to do the following (using lock GPIOA15 as an example) :

①GPIOA_LCK = (1<<16) | (1<<15);

②GPIOA_LCK = (0<<16) | (1<<15);

③GPIOA_LCK = (1<<16) | (1<<15);

④ read register GPIOA_LCK;

⑤ read register GPIOA_LCK to check whether BIT16 is 1.

7.3.3.10. GPIOx_AFRL

Bit(s)	Name	Description	R/W	Reset
31:28	AFR7	**The GPIOx7's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
27:24	AFR6	**The GPIOx6's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
23:20	AFR5	**The GPIOx5's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
19:16	AFR4	**The GPIOx4's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
15:12	AFR3	**GPIOx3's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2	RW	0x0

		0x3: AF3 Other: reserved		
11:8	AFR2	**The GPIOx2's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
7:4	AFR1	**The GPIOx1's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
3:0	AFR0	**AF multifunction configuration for GPIOx0** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0

7.3.3.11. GPIOx_AFRH

Bit(s)	Name	Description	R/W	Reset
31:28	AFR15	**The GPIOx15's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
27:24	AFR14	**The GPIOx14's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
23:20	AFR13	**The GPIOx13's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3	RW	0x0

		Other: reserved		
19:16	AFR12	**The GPIOx12's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
15:12	AFR11	**The GPIOx11's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
11:8	AFR10	**The GPIOx10's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
7:4	AFR9	**The GPIOx9's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0
3:0	AFR8	**The GPIOx8's AF multifunction configuration** 0x0: AF0 0x1: AF1 0x2: AF2 0x3: AF3 Other: reserved	RW	0x0

7.3.3.12. GPIOx_TGL

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	TG15	**GPIOx15 Rollover configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
14	TG14	**GPIOx14 Flip configuration** 0x0: Invalid	WO	0x0

		0x1: Output flipped		
13	TG13	**GPIOx13 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
12	TG12	**GPIOx12 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
11	TG11	**GPIOx11 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
10	TG10	**GPIOx10 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
9	TG9	**GPIOx9 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
8	TG8	**GPIOx8 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
7	TG7	**GPIOx7 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
6	TG6	**GPIOx6 Flip configuration** 0x0: invalid 0x1: Output flipped	WO	0x0
5	TG5	**GPIOx5 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
4	TG4	**GPIOx4 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
3	TG3	**GPIOx3 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
2	TG2	**GPIOx2 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
1	TG1	**GPIOx1 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0
0	TG0	**GPIOx0 Flip configuration** 0x0: Invalid 0x1: Output flipped	WO	0x0

7.3.3.13. GPIOx_IMK

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	IMK15	**GPIOx15 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
14	IMK14	**GPIOx14 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
13	IMK13	**GPIOx13 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
12	IMK12	**GPIOx12 Interrupt is enabled** 0x0: Off 0x1: On	RW	0x0
11	IMK11	**GPIOx11 Flip configuration** 0x0: Invalid 0x1: Output flipped	RW	0x0
10	IMK10	**GPIOx10 Interrupt Enable** 0x0: Off 0x1: Open	RW	0x0
9	IMK9	**GPIOx9 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
8	IMK8	**GPIOx8 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
7	IMK7	**GPIOx7 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
6	IMK6	**GPIOx6 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
5	IMK5	**GPIOx5 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
4	IMK4	**GPIOx4 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
3	IMK3	**GPIOx3 Interrupt Enable** 0x0: Closed 0x1: On	RW	0x0
2	IMK2	**GPIOx2 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0

1	IMK1	**GPIOx1 Interrupt Enable** 0x0: Off 0x1: On	RW	0x0
0	IMK0	**GPIOx0 Interrupt enabled** 0x0: Off 0x1: On	RW	0x0

8. Communication interface peripheral CSI

8.1. SPI_I2C

This module realizes SPI/IIC share fill a controller function, when using SPI, IIC function can not be used, at the same time, only one protocol can be used.

8.1.1. SPI function description

1, support host mode and slave mode

2, support the following four sampling methods:

mode 0: Clock idel is 0, the rising edge is sampled, and the falling edge is removed

mode 1: Clock idel is 0. Data is collected along the falling edge and output along the rising edge

mode 2: Clock idle is set to 1, sampling along the falling edge and exiting data along the rising edge

mode 3: Clock idel is 1, rising edge sampling, falling edge data out

3. Support normal mode, 3-wire mode, 2-wire mode and 4-wire mode.

Normal mode: CLK, CS, IO0(MOSI), IO1(MISO)

3-wire mode: CLK, CS, IO0(both receive and send via IO0)

Two-wire mode: CLK, CS, O0, IO1(both receive and send via IO0, IO1)

Four-wire mode: CLK, CS, IO0, IO1, IO2, IO3(both receive and send via IO0, IO1, IO2, IO3)

4, data size support 1bit to 32bit, configurable, but different modes have limitations. In normal mode and 3-line mode, the data size can be 1 to 32 bits configurable; In two-wire mode, the data size must be divisible by 2; In four-wire mode, the data size must be divisible by 4.

1、You can choose to send low or high data first in a data frame. When the high is first, IO3 transmits the highest bit, IO2 transmits the second highest, IO1 transmits the third highest, and IO0 transmits the fourth highest. When low is first, IO3 transmits the lowest bit, IO2 transmits the second low, IO1 transmits the third low, and IO0 transmits the fourth low.

6, support DMA function.

8.1.2. I2C function description

1, support host mode and slave mode

2, master support clock synchronization and arbitration

slave supports pulling down SCL when sending data is not ready or receiving buffer is full

4. slave supports 7bit address or 10bit address

5. DMA support

8.1.3. SPI timing chart

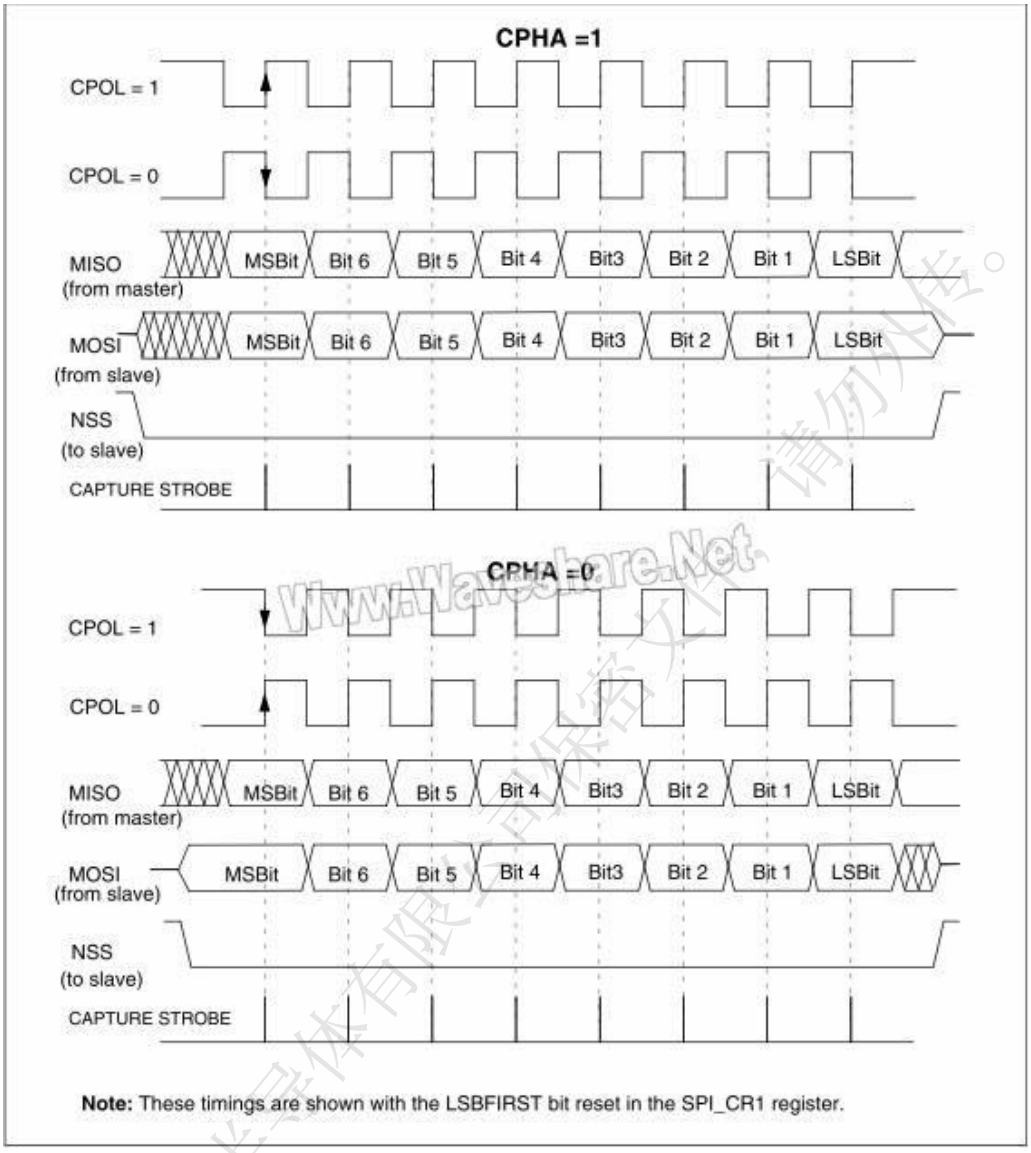

Figure 9-1 Timing diagram in normal mode

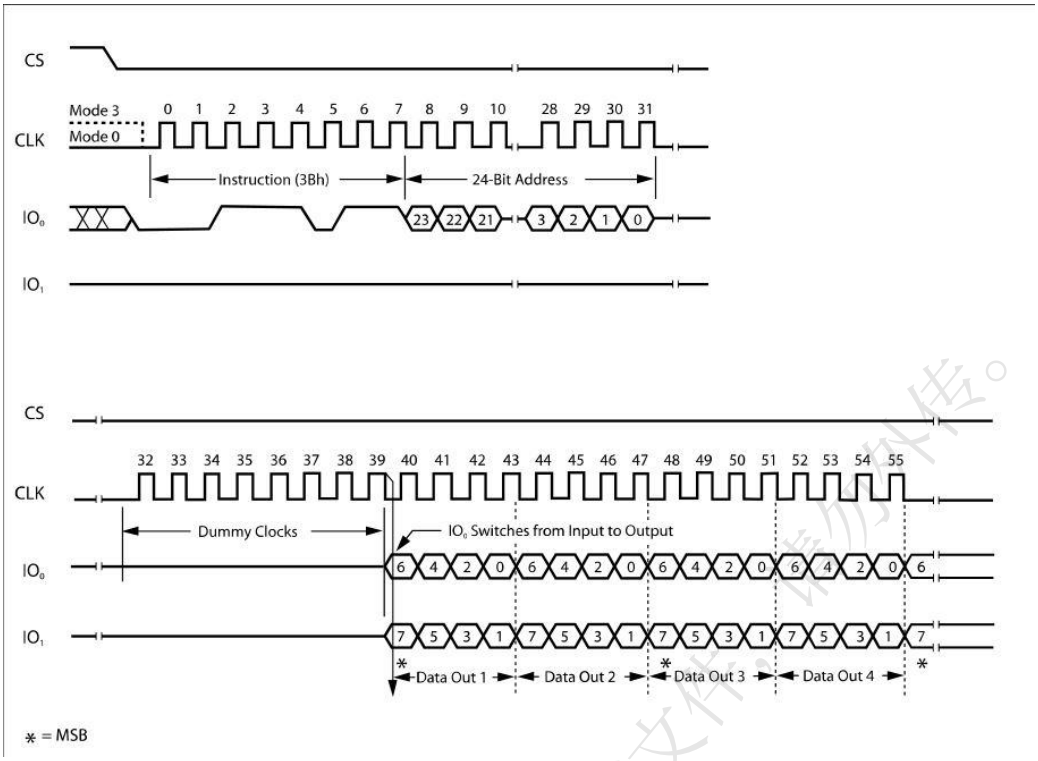

Figure 9-2 Timing diagram in dual-wire mode

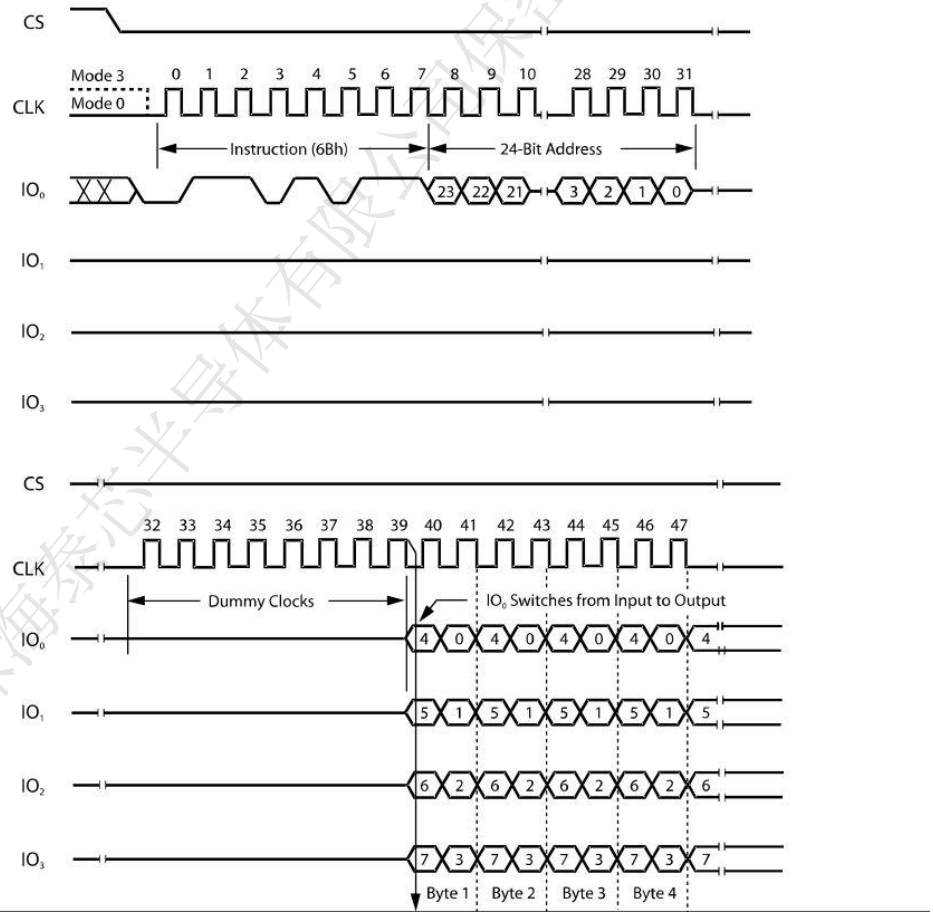

Figure 9-3 Timing diagram in four-wire mode

8.1.4. IO MAPPING

SPI_I2C0:

SPI0_NSS: PA0(AF2)

SPI0_SCK/I2C0_SCL/MOSI: PA1(AF2), PA10(AF1), PB3(AF3)

SPI0_IO0/I2C0_SDA/MISO: PA2(AF2), PA11(AF1), PE2(AF0)

SPI0_IO1: PA3(AF2), PA12(AF1), PB4(AF3)

SPI0_IO2: PA4(AF2)

SPI0_IO3: PA5(AF2)

SPI_I2C1:

SPI1_NSS: PB5(AF1), PA6(AF2)

SPI1_SCK/I2C1_SCL/MOSI: PB6(AF1), PA7(AF2)

SPI1_IO0/I2C1_SDA/MISO: PB7(AF1), PA8(AF2)

SPI1_IO1: PB8(AF1), PA9(AF2)

SPI1_IO2: PB9(AF1), PA10(AF2)

SPI1_IO3: PB10(AF1), PA11(AF2)

8.2. Register

8.2.1. Register base address

Name	Base Address	Description
SPI_I2C0	0x40004400	SPI0/I2C0 base address
SPI_I2C1	0x40004500	SPI1/I2C1 Base address

8.2.2. Register list

Offset Address	Name	Description
0x0000	SPIx_CON0/I2Cx_CON0	SPI/I2C control register 0
0x0004	SPIx_CON1/I2Cx_CON1	SPI/I2C control register 1
0x0008	SPIx_DATA/I2Cx_DATA	SPI/I2C data register

0x000c	SPIx_BAUD/I2Cx_BAUD	SPI/I2C baud rate register
0x0010	SPIx_DAMLEN/I2Cx_DAMLEN	SPI/I2C DMA length register
0x0014	SPIx_DMACNT/I2Cx_DMACNT	SPI/I2C DMA count register
0x0018	SPIx_DMASTART/I2Cx_DMASTART	SPI/I2C DMA trigger register
0x001c	SPIx_STA/I2Cx_STA	SPI/I2C status register

8.2.3. Register Details

8.2.3.1. SPIx_CON0

Bit(s)	Name	Description	R/W	Reset
31:20	Reserved	-	-	-
19:14	SPI_FRAME_SIZE	**SPI frame size configuration** 0x00: Invalid 0x01: Send 1bit per frame 0x02:2 bits are sent per frame ... 0x20: Send 32 bits per frame Other: Invalid	RW	0x0
13	NSS_POS_IE	**SPI slave mode, SPI_NSS pin interrupt enabled** 0x0: Off 0x1: On	RW	0x0
12	SPI_NSS_EN	**SPI slave mode, SPI_NSS pin enabled** 0x0: Off 0x1: On	RW	0x0
11	SPI_NSS	**SPI NSS pin controls the output** This bit is only valid in SPI master mode 0x0: Output low 0x1: Output high	RW	0x0
10:8	RXSEL	**Configuration of SPI host sampling delay** 0x0: no delay 0x1: delay 1 cycle 0x2: delay 2 cycle ... 0x7: delay 7 cycle	RW	0x0
7	SLAVE_SYNC_EN	**In SPI SLAVE mode, the selection bit of whether the input data needs to be synchronized** 0x0: Off 0x1: On	RW	0x0
6	MASTER_SYNC_EN	**In SPI MASTER mode, the selection bit of**	RW	0x0

		whether the input data needs to be synchronized 0x0: Off 0x1: On		
5	Reserved	-	-	-
4	LSBFE	**SPI transfer is enabled from low** 0x0: High priority 0x1: Low priority	RW	0x0
3:2	WIREMODE	**SPI communication line mode selection** 0x0: Normal mode 0x1:3 line mode 0x2: Two-wire mode 0x3: Four-wire mode	RW	0x0
1:0	SPIMODE	**SPI Mode Selection** 0x0: Clock idel is 0, rising edge sampling, falling edge out data 0x1: Clock idel is 0, falling edge sampling, rising edge out data 0x2: Clock idle is 1, falling edge sampling, rising edge out data 0x3: Clock idel is 1, rising edge sampling, falling edge out data	RW	0x0

8.2.3.2. IICx_CON0

Bit(s)	Name	Description	R/W	Reset
31:23	Reserved	-	-	-
22	RX_NACK_IE	**Receive NACK interrupt enabled** 0x0: Off 0x1: On	RW	0x0
21	I2C_AL_IE	**Host quorum loss interruption enabled** 0x0: Disabled 0x1: On	RW	0x0
20	STOP_IE	**If the STOP signal is detected online, the stop signal is enabled. The stop signal will get up regardless of the host** 0x0: Off 0x1: On	RW	0x0
19	ADR_MATCH_IE	**Description Interrupted SLAVE address matching was enabled** 0x0: Disabled 0x1: On	RW	0x0
18:14	I2C_FILTER_CNT	I2C every (I2C_FILTER_CNT +1) I2C module clock, sampled once SCL and SDA, used to filter out a certain width of line burrs. The	RW	0x0

		larger the I2C_FILTER_CNT configuration, the larger the width of the burrs that are filtered out.		
13	BROADCAST_IE	**Broadcast interrupt enabled** 0x0: Off 0x1: On	RW	0x0
12	BROADCAST_EN	**Whether to enable I2C to receive broadcast addresses when i2C functions as the slave** 0x0: Ignore the broadcast address 0x1: When a broadcast address is received, respond with ACK and set BROADCAST_PEND to an interrupt	RW	0x0
11:2	SLAVE_ADR	**I2C SLAVE address when serving as SLAVE**	RW	0x0
1	TX_NACK	**I2C Indicates whether to respond to NACK or ACK when receiving data** 0x0: ACK 0x1: NACK	RW	0x0
0	SLAVE_ADR_WIDTH	**SLAVE address width when I2C is used as SLAVE** 0x0:7bit 0x1:10bit	RW	0x0

8.2.3.3. SPIx_CON1 / IICx_CON1

Bit(s)	Name	Description	R/W	Reset
31:10	Reserved		-	-
9	DMA_IE	**DMA completes interrupt enablement** 0x0: Closed 0x1: On	RW	0x0
8	BUF_OV_IE	**buffer overflow. Data is lost. Interrupt Enable** 0x0: Disabled 0x1: Open	RW	0x0
7	RBUF_NEMPTY_IE	**The receiving buffer is enabled without air disconnection** 0x0: Off 0x1: On	RW	0x0
6	TBUF_NFULL_IE	**The send buffer is not satisfied. Interrupt is enabled** 0x0: Off 0x1: On	RW	0x0
5	SSP_IE	**I2C interface:** I2Cmaster Enables the STOP function after receiving or sending a frame of data (START + WRITE/READ(8bit+ack) + STOP) I2C slave Enable to stop receiving or sending	RW	0x0

		a frame of data (8bit+ack). 0x0: Off 0x1: On		
4	DMA_EN	**Send DMA Enable** 0x0: Off 0x1: On Note: If SSP_TX_EX_EN==0, DMA send; If SSP_TX_EX_EN==1, DMA receive;	RW	0x0
3	SSP_TX_RX_EN	**The interface sending function was enabled. Procedure** 0x0: RX_EN 0x1: TX_EN	RW	0x0
2	SLAVE	**SLAVE mode enabled** 0x0: master mode 0x1: slave mode	RW	0x0
1	SPI_I2C_SEL	**SPI and I2C select bits** 0x0: SPI interface 0x1: indicates the I2C interface	RW	0x0
0	SSP_EN	**Module Enable** 0x0: Off 0x1: On	RW	0x0

8.2.3.4. SPIx_DATA / IICx_DATA

Bit(s)	Name	Description	R/W	Reset
31:0	SSP_CMD_DATA	**SPI interface:** Write: Write the data you want to send to this register, triggering SPI sending; Read: reads this register and gets the received data. **I2C interface:** Write: [7:0] Write the data you want to send to these 8 bits; [8] START enable bit, insert a start bit (only valid in I2C_MASTER mode) before sending the byte; [9] STOP enable bit, followed by a stop bit after sending the byte (valid only in I2C_MASTER mode). Read: Read this register to get the received data (bit7~bit0)	RW	0x0

		[31:10] Not used		

8.2.3.5. SPIx_BAUD / IICx_BAUD

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	BAUD	**MASTER mode** ①SPI BAUD rate =apb0_clock/(2*(BAUD+1)) ②I2C BAUD rate = apb0_clock/(4*(BAUD+1) **SLAVE mode** ①SPI is useless ②I2C is used to define the slave is ready to send data, and then delay(BAUD+1) apb0_clock period before releasing SCL.	RW	0x0

8.2.3.6. SPIx_DMALEN / IICx_DMALEN

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	SPI_DMA_LEN	**Configure the length to receive and send DMA data** Receive unit byte	RW	0x0

8.2.3.7. SPIx_DMACNT / IICx_DMACNT

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	SPI_DMA_CNT	Receive and Send DMA mode: byte length of data actually received and saved to SRAM In master mode, when RX_DMA_EN is 0, receive data: indicates the byte length of the data actually received and has been read by the CPU; Note: when RX_DMA_EN rising edge, TX_DMA_EN rising edge, and MSTER read rising edge, it is cleared to zero.	RO	0x0

8.2.3.8. SPIx_DMASTADR / IICx_DMASTADR

Bit(s)	Name	Description	R/W	Reset

31:13	Reserved	-	-	-
12:0	SPI_DMA_STADR	DMA address, when a frame data is 32bit, need 32bit alignment, other times can be arbitrarily configured	RW	0x0

8.2.3.9. SPIx_STA / IICx_STA

Bit(s)	Name	Description	R/W	Reset
31:28	Reserved	-	-	-
27	SLV_ADDRED	**Slave mode, slave is addressed flag bit** 0x0: The slave is not addressed 0x1: The slave has been addressed	RO	0x0
26:24	STATE	**I2C status** 0x0: IDLE Idle 0x1: START Send start start Has received start, waiting for SCL to change to 0 0x2: TX Send 1byte of data Send 1byte of data 0x3: RX receives 1byte data Receives 1byte data 0x4: STOP Send stop no use 0x5: ADR0 no use Wait to receive 1byte address 0x6: ADR1 no use waiting to receive a 2byte address 0x7: no use	RO	0x0
23:20	Reserved	-	-	-
19	CLR_BUF_CNT	Write 1 Clear BUF_CNT Write 0 does nothing Read: Always returns 0	WO	0x0
18:16	BUF_CNT	How many bytes of valid data are in the buffer, the CPU reads (RX_EN)/ writes (TX_EN) a CMD_DATA register, subtract/add a frame data width, 8bit data subtract 1,16 bit data subtract 2,24 bit and 32bit data subtract 4	RO	0x0
15	MASTER_RX_BUSY	**In MASTER Read slave mode, indicates whether continuous reading of DMA_LEN byte data configured by the software is complete** 0x0: Finished reading 0x1: Not finished reading	RO	0x0
14	I2C_RX_NACK	**The I2C host or slave detects an ACK or NACK during the 9bit handshake phase when sending data** 0x0: ACK 0x1: NACK	RO	0x0

13	I2C_SLAVE_RW	**IIC Slave, in the address phase, receives the read/write flag from the host** 0x1: The host will read the slave 0x0: The host will write to the slave	RO	0x0
12	I2C_BUS_BUSY	**Whether the IIC line is busy indicates a bit** 0x0: No START appears on the line, or STOP appears after START, and the line is idle 0x1: START is detected on the line, and STOP is not detected. The line is busy Note: Hardware clear only	RO	0x0
11	SPI_SLAVE_CS	**SPI SLAVE, the status of CS**	RO	0x01
10	SSP_BUSY	**The SPI or IIC is busy** 0x0: The MASTER is idle SLAVE is not sending data, not in use when receiving 0x1: The MASTER is receiving or sending a frame of data The SLAVE is sending a frame of data, which is not used when receiving Note: Hardware clearing, either turn off SPI and IIC (CON1[0]==0), or CLR_BUF_CNT will also clear SSP_BUSY when slave	RO	0x0
9	AL_PEND	**IIC host, quorum loss detected** 0x0: No quorum loss is detected 0x1: Arbitration is lost Note: Only software clear, must clear the IIC to work properly	RC	0x0
8	STOP_PEND	**IIC interface that detects that a STOP bit is generated on the line** 0x0: The STOP bit is not detected. 0x1: The STOP bit is detected. Note: Only software clear	RC	0x0
7	ADR_MTCH_PEND	**IIC slave, receives the correct slave address sent from the host** 0x1: The slave address matches 0x0: The slave address does not match Note: Only software clear	RC	0x0
6	I2C_BROADCAST_PEND	**When the IIC functions as the SLAVE, the broadcast address flag bit is detected. Procedure** 0x0: No broadcast address is detected 0x1: The broadcast address is detected Note: Only software clear	RC	0x0
5	SPI_NSS_POS	**SPI NSS pin rising edge detected** 0x0: No rising edge detected 0x1: Rising edge detected	RC	0x0

		Note: Software clear only		
4	DMA_PEND	**Send DMA or receive DMA completion flag** 0x0: DMA is not complete 0x1: DMA completed Note: Only software clear	RC	0x0
3	BUF_OV	**buffer overflow, some data is lost** 0x0: The buffer did not overflow 0x1: Another data is received when the buffer is full (the buffer capacity is 5 bytes, but full does not necessarily mean 5 bytes, but the space cannot accommodate a frame of data). Discard the subsequent data. Note: Only software clear	RC	0x0
2	BUF_EMPTY	**buffer empty flag** 0x0: Not empty 0x1: Empty	RO	0x01
1	BUF_FULL	**buffer full flag bit** 0x0: Not satisfied 0x1: Full	RO	0x0
0x0	SSP_DONE	**Done flag** 0x0: Not finished 0x1: SPI interface The SPI has received or sent a frame of data (8bit/16bit/24bit/32bit) I2C interface The I2C MASTER has received or sent a frame of data (START (optional) + WRITE/READ (8bit+ack) + STOP (optional)). The I2C SLAVE has received or sent a frame of data (8bit+ack). Note: Only software clear	RC	0x0

8.2.4. Instructions

8.2.4.1. SPI module use

SPI MASTER initialization:

1. Configure IO mapping to open the data path
2. Configure BAUD
3. Configure CON0
4. Configure configure CON1

Enable SPI MASTER: SPIx->CON1 | = 0x01;

SPI MASTER non-DMA send:

1.SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); //tx enable

2. If SPI_NSS_EN is initialized, you need to lower the NSS: SPIx-> CON0&= ~(1<<11); //nssnegedge

2. Check whether the send buffer is full. If the buffer is not full, enter data into CMD_DATA

3. Fill all the data to be sent into CMD_DATA, and wait until the send buffer is empty and ssp_busy is set to 0.

4. If SPI_NSS_EN is initialized, raise the NSS: SPIx->CON0 |= (1<<11); //nssposedge

5 Wrap up.

SPI MASTER DMA send:

1.SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); //tx enable

2. If SPI_NSS_EN is initialized, you need to lower the NSS: SPIx-> CON0&= ~(1<<11); //nssnegedge

2. Configure DMA_STADR,DMA_LEN

Enable DMA:SPIx->CON1 |= (1<<4); 3. // DMA EN\

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5. If initializing SPI_NSS_EN, you need to stretch NSS higher: SPIx->CON0 |= (1<<11); //nssposedge

6 Wrap up.

SPI MASTER non-DMA reception:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. If SPI_NSS_EN is initialized, NSS needs to be lowered: SPIx-> CON0&= ~(1<<11); //nssnegedge

2. Set DMA_LEN to the number of bytes of data you want to receive

3. Write CMD_DATA to trigger the start of receiving.

4. Check that the received buffer is not empty. Fetch the data by reading DMA_DATA.

5. Repeat 4 until DMA_LEN data is read away

6. If initializing SPI_NSS_EN, pull NSS higher: SPIx->CON0 |= (1<<11);

//nssposedge

7 Wrap up.

SPI MASTER DMA Reception:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. If SPI_NSS_EN is initialized, NSS needs to be lowered: SPIx-> CON0&= ~(1<<11); //nssnegedge

2. Set DMA_LEN to the number of bytes of data you want to receive

Enable DMA:SPIx->CON1 |= (1<<4); 3. //

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5. If initializing SPI_NSS_EN, you need to stretch NSS higher: SPIx->CON0 |= (1<<11); //nssposedge

6 Wrap up.

SPI SLAVE initialization:

1. Configure IO mapping to open the data path

2. Configure CON0

3. Configure CON1

Enable the SPI SLAVE: SPIx->CON1 | = 0x05; 4.

SPI SLAVE Non-DMA send:

SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); 2. //tx enable

2. Check whether the send buffer is full. If the buffer is not full, enter data in CMD_DATA

3. Fill all the data to be sent into CMD_DATA, and wait until the send buffer is empty and ssp_busy is set to 0.

4. End.

SPI SLAVE DMA Send:

1.SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); 2. //tx enable

2. Configure DMA_STADR,DMA_LEN

Enable DMA:SPIx->CON1 |= (1<<4); 3. // DMA EN\

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

SPI SLAVE Non-DMA receive:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. Check that the received buffer is not empty. Fetch the data by reading DMA_DATA.

3. Repeat 2 until you have read all the required data

4 Finish.

SPI SLAVE DMA receives:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. Set DMA_LEN to the number of bytes of data to receive

Enable DMA:SPIx->CON1 |= (1<<4); 3.

Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

8.2.4.2. Description of the I2C module

I2C MASTER initialization:

1.I/O mapping configuration to open the data path

2. Configure BAUD

3. Configure CON0

4. Configure configure CON1

Enable SPI MASTER: SPIx->CON1 | = 0x03;

I2C MASTER non-DMA send:

1.SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); //tx enable

2. Check whether the send buffer is full. If the buffer is not full, enter data in CMD_DATA

3. Fill all the data to be sent into CMD_DATA, and wait until the send buffer is empty and ssp_busy is set to 0.

4. End.

I2C MASTER DMA send:

1.SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); //tx enable

2. Configure DMA_STADR,DMA_LEN

Enable DMA:SPIx->CON1 |= (1<<4); 3.

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

I2C MASTER non-DMA reception:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. Set DMA_LEN to the number of bytes of data to receive

3. Write CMD_DATA to trigger the start of receiving.

4. Check that the received buffer is not empty. Fetch the data by reading DMA_DATA.

5. Repeat 4 until DMA_LEN's data is read away

6. End.

I2C MASTER DMA reception:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. Set DMA_LEN to the number of bytes of data to receive

Enable DMA:SPIx->CON1 |= (1<<4); 3.

Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

I2C SLAVE initialization:

1. Configure I/O mapping to open the data path

2. Configure CON0

3. Configure CON1

Enable the SPI SLAVE: SPIx->CON1 | = 0x07; 4.

I2C SLAVE non-DMA send:

SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); 2. //tx enable

2. Check whether the send buffer is full. If the buffer is not full, enter data in CMD_DATA

3. Fill all the data to be sent into CMD_DATA, and wait until the send buffer is empty and ssp_busy is set to 0.

4. End.

I2C SLAVE DMA send:

SSP_TX_EN Enable: SPIx->CON1 |= (1<<3); 2. //tx enable

2. Configure DMA_STADR,DMA_LEN

Enable DMA:SPIx->CON1 |= (1<<4); 3. // DMA EN\

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

I2C SLAVE Non-DMA receive:

Enable SSP_RX_EN: SPIx-> CON1&= ~(1<<3); //rx enable

2. Check that the received buffer is not empty. Fetch the data by reading DMA_DATA.

3. Repeat 2 until you have read all the required data

4 Finish.

I2C SLAVE DMA receives:

1.SSP_RX_EN Enable: SPIx-> CON1&= ~(1<<3); //rx enable

2. Set DMA_LEN to the number of bytes of data to receive

Enable DMA:SPIx->CON1 |= (1<<4); 3. // DMA EN\

4. Wait for DMA to end: while((SPIx->STA & (1<<4))==0); //WAIT DMA PEND

5 End.

9. UART

9.1. Overview

UART0:

- Supports both 8bit data and 9bit data modes
- Parity is supported, and parity/parity is optional
- Support for detecting system update data string (0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4->0xec)
- It has a receiving cache of 4 frames and a sending cache of 1 frame
- Hardware detection receiving time out, time out length can be configured,

configuration range: 1~65536 bit rate time.

- GPIO mapping:

  UART0 RX: PB8(fun2), PE1(fun3), PA7(fun3)

  UART0 TX: PB7(fun2), PE0(fun3), PA6(fun3)

UART1:

- Supports both 8bit data and 9bit data modes
- Parity is supported, and parity/parity is optional
- It has a receiving cache of 4 frames of data and a sending cache of 1 frame of data
- DMA support for receiving and sending
- Support RS485 mode
- Hardware detection receives time out. The time out length can be configured. The configuration range is 1 to 65536 bit rate time.
- GPIO mapping:

  UART1 RX: PB3(fun2), PE1(fun2), PA12(fun3)

  UART1 TX: PB4(fun2), PE0(fun2), PA11(fun3), PE2(fun1)

  UART1 DE: PB5(fun2), PB10(fun2)

  UART1 RE: PB6(fun2), PE2(fun2)

9.2. Register

9.2.1. Register base address

Name	Base Address	Description
UART0	0x40004000	UART0 Base address
UART1	0x40004100	UART1 Base Address

9.2.2. Register list

Offset Address	Name	Description

0x0000	UART0_CON	UART0 control register
0x0004	UART0_BUAD	UART0 Baud rate register
0x0008	UART0_DATA	UART0 Data register
0x000c	UART0_STA	UART0 Status register

Offset Address	Name	Description
0x0000	UART1_CON	UART1 Control register
0x0004	UART1_BUAD	UART1 Baud rate register
0x0008	UART1_DATA	UART1 Data register
0x000c	UART1_STA	UART1 Status register
0x0010	UART1_TSTADR	UART1 Send the DMA starting address
0x0014	UART1_RSTADR	UART1 receives the DMA start address
0x0018	UART1_TDMALEN	UART1 Send the length of the DMA
0x001c	UART1_RDMALEN	UART1 receives the length of the DMA
0x0020	UART1_TDMACNT	UART1 has sent DMA length
0x0024	UART1_RDMACNT	UART1 has received the DMA length
0x0028	UART1_DMACON	UART1 DMA control register
0x002c	UART1_DMASTA	UART1 DMA status register
0x0030	UART1_RS485CON	UART1 RS485 control register
0x0034	UART1_RS485DET	UART1 RS485 DET register
0x0038	UART1_RS485TAT	UART1 RS485 TAT register

9.2.3. Register Details

9.2.3.1. UARTx_CON

Bit(s)	Name	Description	R/W	Reset
31:16	TO_BIT_LEN	**Configure timeouts** The unit is bit rate time Time_out_time= (TO_BIT_LEN+1) * bit_rate_time	RW	54
15	TCIE	**Transmission complete interrupt enabled** 0x0: Off	RW	0x0

		0x1: On, produces a UART interrupt when TC=1 in the UART_STA register		
14	TMR_PWM_EN	The output of UART is calculated with the PWM of TIMER before output the enable bit (UART0 outputs the PWM of timer0, UART1 outputs the PWM of timer1). 0x0: Off 0x1: On	RW	0x0
13	TO_IE	**Time out Interrupt enabled** 0x0: Off 0x1: On	RW	0x0
12	TO_EN	**Time out Detects the enable bit** Check whether no data is received after 5 * (1 start_bit + 8 data bit + 2 stop bit) time. After each TO_EN, you need to wait until one byte of data is received before the detection begins. The 1byte flag will be cleared each time TO_PEND is cleared, or if TO_EN is equal to 0. 0x0: Off 0x1: On	RW	0x0
11	FERR_IE	**Frame error interrupt enable bit** 0x0: Off 0x1: On	RW	0x0
10	TXBUF_EMPTY_IE	**buffer air break enabled bit was sent** 0x0: Off 0x1: On	RW	0x0
9	RXBUF_NEMPTY_IE	**Receive buffer Air break Enable bit** 0x0: Off 0x1: On	RW	0x0
8	TX_INV	**Send output signal take inverse selection bit** 0x0: Off 0x1: Open	RW	0x0
7	RX_INV	**Receive the input signal to take the inverse selection bit** 0x0: Does not take invert 0x1: Take the inverse	RW	0x0
6	ODD_EN	**Parity selection** 0x0: Parity check 0x1: Parity check	RW	0x0
5	PARITY_EN	**Parity enabled** 0x0: Off 0x1: On Note: Parity check and BIT9_EN can only be one of two	RW	0x0

4	BIT9_EN	**UART transfers 9bit data select bits** 0x0: UART transfers 8bit data 0x1: UART transfers 9bit data Note: Parity check and BIT9_EN can only be one of two	RW	0x0
3	STOP_BIT	**End bit Enable bit** 0x0: An end bit 0x1: Two end bits	RW	0x0
2:1	WORK_MODE	**work_mode** 0x0: Full duplex, can send and receive at the same time 0x1: Simplex send 0x2: Simplex receive 0x3: A data cable, the hardware automatically switches the IO direction; In non-RS485M mode, the software triggers the send IO direction is the output, no data send IO direction is the input. In RS485 mode, IO is output when DE is valid, and IO is input when DE is invalid	RW	0x0
0	UART_EN	**UART module enable bit** 0x0: Off 0x1: On	RW	0x0

9.2.3.2. UARTx_BAUD

Bit(s)	Name	Description	R/W	Reset
31:18	Reserved	-	-	-
17:0	UARTx_BAUD	**UART baud rate Settings** Baud rate =SYS_CLK /(UARTx_BAUD+1) Note: UARTx_BAUD must be configured to be greater than or equal to 6, otherwise the input signal will be filtered out by the internal filter.	RW	0xa93

9.2.3.3. UARTx_DATA

Bit(s)	Name	Description	R/W	Reset
31:9	Reserved	-	-	-
8:0	UARTx_DATA	**UART transfers data** Write 8/9 bits of data to UARTx_DATA: triggers the UART module to start sending data.	RW	0x0

		Read UARTx_DATA: Get the received low 8/9bit data, before reading, read PERR[0] to get parity information		

9.2.3.4. UARTx_STA

Bit(s)	Name	Description	R/W	Reset
31:16	-	-	-	-
15	TC	**Transfer complete** If the transfer of a frame is complete and TX_BUF_EMPTY is set, then this bit is set by the hardware. If TCIE=1 in the UARTx_CON register, an interrupt is generated. It is cleared by a software sequence (written to the usart_dr register). The TC bit can also be cleared by writing "1". 0x0: indicates that the transfer is not complete 0x1: Transfer completed	RO	0x0
14	UPDATE_DETECT_EN	**System upgrade detection enable bit** Detects whether the following string of continuous data is received 0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4->0xec 0x0: Off 0x1: On Note: when UPDATE_DETECT_EN is 1, and UPDATE_DETECT_PEND is 0 UARTx_CON: will be fixed to 0x1005 (simplex receive, 1stop, 8bit mode, no parity check, no inverting, all interrupts not enabled); UARTx_BAUD: will be configured by the EFLASH module, the default value will be 0xa93; UART_DMACON: will be fixed to 0x00 (DMA disabled); UART_RS485_CON: will be fixed to 0x00 (disable rs485); Software cannot be changed.	RO	0x0
13	UPDATE_DETECT_PEND_G	This bit is the same as UPDATE_DETECT_PEND, except that once it is set to 1, it will never be set to 0 unless reset UART0.	RO	0x0
12	UPDATE_DETECT_PEND	**System upgrade flag bit was detected** Indicates that the following string of data 0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4-	WC/R	0x0

		>0xec was received. Write 1 to clear zero. 0x0: No system upgrade detected 0x1: System upgrade detected Note: Only UART0 has this bit. Software clear only.		
11	TO_PEND	**Time out flag bit** Indicates that after 5 * (1 start_bit + 8 data bit + 2 stop bit) time, no data is received. Write 1 to clear zero. 0x0: time out does not appear 0x1: time out appears	WC/R	0x0
10:7	PERR	**Error in receiving data parity** 4 bits, corresponding to the 4 data in BUF. Before reading UARTx_DATA, you need to read this register first, the hardware will automatically correspond to PERR[0] you read UARTx_DATA to obtain data.	RO	0x0
6:4	RX_CNT	**RX How many data does BUF have** 0x0:0 data 0x1:1 data 0x2:2 pieces of data 0x3:3 data 0x4:4 data Other: invalid	RO	0x0
3	FERR	**Frame Error** Indicates that a low level of RX_IN was detected during STOPbit. Write 1 to clear zero 0x0: There are no frame errors 0x1: A frame error has occurred	WC/R	0x0
2	RX_BUF_OV	Receiving BUF can hold up to 4 8/9bit data, after receiving 4 data, if the software has not had time to read away, and receive another data, this sign will be up. Write 1 to clear zeros. 0x0: (0 to 4 bytes) data was received 0x1: After receiving (>4 bytes) of data, BUF saves only the first 4 bytes and discards the rest	WC/R	0x0
1	RX_BUF_NOT EMPTY	**Receives the BUF not empty flag bit** Write 1 to clear zero 0x0: Receive BUF without data 0x1: Received BUF has data	RO	0x0
0	TX_BUF_EMPTY	**Send finish flag bit** Write 1 to clear zero, or write UARTx_DATA to clear zero	RO	0x0

		0x0: Send did not complete 0x1: Sent completed		

9.2.3.5. UART1_TSTADR

Bit(s)	Name	Description	R/W	Reset
31:13	Reserved	-	-	-
12:0	TSTADRT	**Send the DMA starting address** If it's 9bit data, you need 16bit alignment; If it is 8bit data, you can configure it any way you want.	RW	0x0

9.2.3.6. UART1_RSTADR

Bit(s)	Name	Description	R/W	Reset
31:13	Reserved	-	-	-
12:0	RSTADRT	**Receive the DMA starting address** If it is 9bit data, 16bit alignment is required; If it is 8bit data, you can configure it any way you want.	RW	0x0

9.2.3.7. UART1_TDMALEN

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	TDMALEN	**Plan to send DMA byte length** 8bit data mode, uart one frame of data equals 1byte; 9bit data, uart one frame data is equal to 2byte, TDMALEN needs to be configured as a multiple of 2.	RW	0x0

9.2.3.8. UART1_RDMALEN

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	RDMALEN	**Plan to receive the DMA byte length** 8bit data mode, uart one frame of data is equal to 1byte;	RW	0x0

		9bit data, uart a frame data is equal to 2byte, RDMALEN needs to be configured as a multiple of 2.		

9.2.3.9. UART1_TDMACNT

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	TDMACNT	**DMA byte length has been sent** 8bit data mode, uart a frame of data is equal to 1byte; 9bit data, uart a frame of data is equal to 2byte.	RO	0x0

9.2.3.10. UART1_RDMACNT

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	RDMACNT	**The DMA byte length has been received** 8bit data mode, uart one frame data is equal to 1byte; 9bit data, uart a frame of data is equal to 2byte.	RO	0x0

9.2.3.11. UART1_DMACON

Bit(s)	Name	Description	R/W	Reset
31:10	Reserved	-	-	-
9	RX_DMA_EN_CFG_EN	**The RX_DMA_EN configuration is enabled** This parameter can be configured only when RX_DMA_EN is configured. 0 will be cleared automatically. 0x0: Off 0x1: On	WO	0x0
8	TX_DMA_EN_CFG_EN	**TX_DMA_EN configuration Enabled** When configuring TX_DMA_EN, this parameter can be set to 1 at the same time. 0 is automatically cleared. 0x0: Off 0x1: On	WO	0x0
7:5	Reserved	-	-	- *
4	RX_DMA_PERR_IE	If at least one parity error occurs in a	RW	0x0

		packet of received DMA data, interruption is enabled 0x0: Disabled 0x1: On		
3	RX_DMA_IE	**Receive DMA interrupt enabled** 0x0: Off 0x1: On	RW	0x0
2	TX_DMA_IE	**Send DMA interrupt enabled** 0x0: Off 0x1: On	RW	0x0
1	RX_DMA_EN	**Enable DMA for receiving** Configure this bit to also write 1 to RX_DMA_EN_CFG_EN 0x0: Off 0x1: On	RW	0x0
0	TX_DMA_EN	**Send DMA Enable** Configure this bit to write 1 to TX_DMA_EN_CFG_EN at the same time 0x0: Off 0x1: On	RW	0x0

9.2.3.12. UART1_DMASTA

Bit(s)	Name	Description	R/W	Reset
31:3	Reserved	-	-	-
2	RX_DMA_PERR	**At least one parity error flag bit occurs in the received DMA packet** 0x0: No data has a parity error 0x1: At least one data parity error has occurred	RC	0x0
1	RX_DMA_PEND	**Receive the DMA completion flag bit** Data is received complete and saved to SRAM 0x0: DMA is not complete 0x1: DMA is complete	RC	0x0
0	TX_DMA_PEND	**Send the DMA complete flag bit** The data is taken out of SRAM and sent out in its entirety via uart. 0x0: DMA is not complete 0x1: DMA completed	RC	0x0

9.2.3.13. UART1_RS485CON

Bit(s)	Name	Description	R/W	Reset

31:10	Reserved	-		-
9	RE_EN	RE Enable 0x0: Off 0x1: On	RW	0x0
8	DE_EN	**DE Enable** 0x0: Off 0x1: Open	RW	0x0
7:4	Reserved	-	-	-
3	RS485_MODE	**RS485 working mode** 0x0: Software configuration DE_EN, DE will be valid until the configuration DE_EN is 0; If software is configured with RE_EN, RE will remain valid until RE_EN is configured with 0; Hardware will automatically insert DE_DAT, DE_AT, RE2DE_T, DE2RE_T times. When this bit is equal to 0, DE_EN and RE_EN cannot be 1 at the same time 0x1: The hardware automatically switches between DE and RE modes, there is data to be sent to switch to DE, there is no data to be sent to keep RE	RC	0x0
2	RE_POL	**RE Polarity** 0x0: Active high 0x1: Low level is valid	RC	0x0
1	DE_POL	**DE Polarity** 0x0: High is valid 0x1: Low level is valid	RW	0x0
0	RS485_EN	**RS485 Enable** 0x0: Closed 0x1: On	RW	0x0

9.2.3.14. UART1_RS485DET

Bit(s)	Name	Description	R/W	Reset
31:25	Reserved	-	-	-
24:16	DE_DAT	STOP BIT The time interval between the end of the DE and the invalidation, in uart module clock	RW	0x0
15:9	-	-	-	-
8:0	DE_AT	DE The time interval between valid and sending START BIT, in uart module clock	RW	0x0

9.2.3.15. UART1_RS485TAT

Bit(s)	Name	Description	R/W	Reset
31:16	RE2DE_T	The time interval between when RE is valid and when DE is valid, in uart module clock	RW	0x0
15:0	DE2RE_T	The time interval between when DE is valid and when RE is valid, in uart module clock	RW	0x0

9.2.4. Instructions

1) Sending UART with DMA mode not enabled:

1. Configure UARTx_BAUD.

2. Configure UARTx_CON.

TX_IE: If you need to interrupt, this one matches 1

TX_INV: If you need to invert the output, this is 1

UART_EN: matches 1

BIT9_EN: If you need to transfer 9bit data, this one matches 1

BIT9: If BIT9_EN, this writes the 9th-bit data that needs to be transferred

3. Configure UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;

4. Wait for UARTx_STA[0] (TX_BUF_EMPTY) bit to be 1, and then write the 8bit/9bit data to be sent to UARTx_DATA

5. Repeat 4 if there is more data to be sent.

6. If there is no data to be sent, wait for all data to be sent: Wait for UARTx_STA[15] : TC set to 1;

2) UART sending enabled by DMA mode:

1. Configure UARTx_BAUD.

2. Configure UARTx_CON

TX_IE: If you need to interrupt, this one matches 1

TX_INV: If you need to invert the output, this is 1

UART_EN: match 1

BIT9_EN: If you need to transfer 9bit data, this one matches 1

BIT9: If BIT9_EN, this writes the 9th-bit data that needs to be transferred

3. Configure UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;

4. Configure DMACON,DMA_TSTADR,DMA_TDMALEN;

5. Wait for DMA completion :UART_DMA_CON[0] to clear zero, or UART_DMASTA[0] to 1;

3) UART reception not enabled by DMA mode:

1. Configure UARTx_BAUD.

2. Configure UARTx_CON

RX_IE: If you need to interrupt, this one matches 1

RX_INV: If you need to invert the input, this gets 1

UART_EN: matches 1

BIT9_EN: If you need to transfer 9bit data, this one matches 1

3. Configure UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;

4. Wait to receive the data: After receiving the data, UARTx_STA[1] : RX_BUF_NOTEMPTY will set 1, and you can understand how many byte data received by viewing UARTx_STA[6:4] : RX_CNT.

5. Read the received data, and obtain the received data by reading UARTx_DATA.

6. If there is still data to receive, repeat 4 and 5.

4) UART receiving enabled by DMA mode:

1. Configure UARTx_BAUD.

2. Configure UARTx_CON

RX_IE: If you need to interrupt, this one matches 1

RX_INV: If you need to invert the input, this gets 1

UART_EN: matches 1

BIT9_EN: If you need to transfer 9bit data, this one matches 1

3. Configure UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;

4. Configure DMACON,DMA_RSTADR,DMA_RDMALEN;

5. Wait for the DMA to complete :UART_DMA_CON[1] is cleared, or UART_DMASTA[1] is 1;

10. CRC count cell

10.1. Key features

- Supports polynomials of different lengths such as 5/7/8/16/32
- Custom polynomials are supported

10.2. Registers

10.2.1. Register base address

Name	Base Address	Description
CRC	0x40002000	The base address of CRC

10.2.2. Register list

Offset Address	Name	Description
0x00	CRC_CFG	Config register
0x04	CRC_INIT	Initialize the configuration register
0x08	CRC_INV	Invert register
0x0C	CRC_POLY	Polynomial configuration register
0x10	CRC_KST	Trigger register
0x14	CRC_STA	Status register
0x18	CRC_ADDR	Address register
0x1C	CRC_LEN	Length register
0x20	CRC_OUT	Result output register

10.2.3. Register Details

10.2.3.1. CRC_CFG

Bit(s)	Name	Description	R/W	Reset
31:14	Reserved		RO	0
13:8	POLY_WIDTH	**Poly's bit width configuration item** The 5/7/8/16/32 configuration is supported, and the default is 32 Bit	RW	32
7:2	Reserved		RO	0
1	BIT_ORDER_EN	**Whether the output data is inverted** 0x0: Off 0x1: On, DATA[7:0] toDATA[0:7]	RW	0
0	INT_EN	**CRC interrupt Enable** 0x0: Off 0x1: On	RW	0

10.2.3.2. CRC_INIT

Bit(s)	Name	Description	R/W	Reset
31:0	INIT_VALUE	**CRC initial value**	RW	0xFFFFFFFF

10.2.3.3. CRC_INV

Bit(s)	Name	Description	R/W	Reset
31:0	INV_VALUE	**CRC output result inversion**	RW	0xFFFFFFFF

10.2.3.4. CRC_POLY

Bit(s)	Name	Description	R/W	Reset
31:0	POLY_VALUE	**Configure the CRC ploy value** The default value is 0xhedb88320	RW	0xhedb883320

10.2.3.5. CRC_KST

Bit(s)	Name	Description	R/W	Reset

31:1	Reserved		RW	0
1	Pending Clear	**Clear the flag bits of CRC** Write 1 Clear	WO	0

10.2.3.6. CRC_STA

Bit(s)	Name	Description	R/W	Reset
31:1	Reserved		RO	0
1	Pending	**CRC completion flag bit**	RO	0

10.2.3.7. CRC_POLY

Bit(s)	Name	Description	R/W	Reset
31:0	POLY_VALUE	**Configure the CRC ploy value** The default value is 0xhedb88320	RW	0xhedb88320

10.2.3.8. CRC_ADDR

Bit(s)	Name	Description	R/W	Reset
31:2	DMA Start Address	**CRC DMA start address** Note: Physical address, such as address 0x0 redirected 0x2000_0000	RO	0
1:0	Reserved	Reserved, for 4 byte alignment	RO	0

10.2.3.9. CRC_LEN

Bit(s)	Name	Description	R/W	Reset
31:0	DMA Length	**CRC DMA length configuration** 4 byte Alignment	RW	0x0

10.2.3.10. CRC_OUT

Bit(s)	Name	Description	R/W	Reset
31:0	CRC Result	**CRC result Output**	RO	0x0

10.3. Operation flow

2、Configure CRC initialization (CRC_INIT) and inverse (CRC_INV);

3、Configure the CRC polynomial (CRC_POLY), the bit width of the CRC polynomial (CRC_CFG);

4、Set the start address (physical address) of the DMA and the length of the DMA. If the length is not 0, the CRC check starts.

5、Wait for CRC pending(CRC_STA), get CRC result (CRC_OUT), and finally clear pending(CRC_KST).

11. Hardware acceleration unit

11.1. Acceleration Unit Introduction

This acceleration unit contains a hardware divider. Hardware division is useful in high performance applications for automatically performing signed or unsigned 32-bit / 16-bit integer division operations.

11.2. Main features of hardware division

- Signed or unsigned integer division operations
- 32-bit divisor and 16-bit dividend, output 32-bit quotient and 16-bit remainder
- 8 HCLK cycles complete
- If the divisor is zero, the overflow interrupt flag bit is generated
- Write the divisor to automatically perform division operations
- Automatically wait for the end of the operation when reading quotient and remainder registers, without checking the status bit

11.3. Hardware division function introduction

The hardware division unit includes 4 32-bit data registers, which are dividend, divisor, quotient and remainder, and can do signed or unsigned 16-bit division operations. By hardware division control register SIGN can choose whether to do signed or unsigned division. Each time you write to the divisor register, the division operation is automatically triggered. At the end of the operation, the result is written to the quotient and remainder register. If you read the quotient, remainder, or status register before the end of the operation, the operation is paused and the result is not returned until the end. If the divisor is zero, the overflow interrupt flag bit is generated

11.4. Registers

11.4.1. Register base address

Name	Base Address	Description
DIV	0x40020A00	Hardware divider base address

11.4.2. Register list

Offset Address	Name	Description
0x0000	DIV_DVD	Dividend register
0x0004	DIV_DVS	Divisor register
0x0008	DIV_QUO	Quotient register
0x000c	DIV_REM	Remainder register
0x0010	DIV_SR	Status register
0x0014	DIV_CON	Control register

11.4.3. Register details

11.4.3.1. DIV_DVD

Bit(s)	Name	Description	R/W	Reset
31:0	Dividend	Dividend register bit (Dividend data)	RW	0x0

11.4.3.2. DIV_DVS

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	DIVISOR	Divisor register bit (Divisor data) After writing this register, the divisor operation is triggered automatically.	RW	0x0

11.4.3.3. DIV_QUO

Bit(s)	Name	Description	R/W	Reset
31:0	QUOTIENT	Quotient register bit (Quotient data)	RW	0x0

11.4.3.4. DIV_REM

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	REMAINDER	Remainder data (Remainder data)	RW	0x0

11.4.3.5. DIV_SR

Bit(s)	Name	Description	R/W	Reset
31:1	Reserved	-	-	-
0	DIV0_FLAG	read 1: The current division operation has a divisor of 0 read 0: The current divisor is not 0 Writing 1 clears this bit as well as writing the DVSR Write 0: No operation	RW	0x0

11.4.3.6. DIV_CON

Bit(s)	Name	Description	R/W	Reset

31:2	Reserved	-	-	-
1	INT_EN	0x0: A divisor of 0 does not produce an interrupt signal 0x1: A divisor of 0 produces an interrupt signal	RW	0x1
0	SIGN	0x0: Unsigned division 0x1: Signed division	RW	0x0

12. Comparator (COMP)

12.1. Introduction

A universal comparator COMP is embedded in the chip, which can be used independently or in combination with timers and EPWM.

12.2. Key features

- Rail-to-rail comparator
- Reusable I/O with the internal end attached to the DAC
- Programmable hysteresis voltage
- Filter function to support comparison results
- The output can be redirected to an I/O port
- The output is connected to multiple timer inputs and can trigger the capture event of the timer
- The output is connected to multiple EPWM inputs and can trigger the EPWM brake
- The COMP has 7 positive phase inputs and 4 reverse inputs
- The comparator generates interrupts and supports waking the CPU from sleep

mode and stop mode

12.3. Function Description

12.3.1. Comparator function block diagram

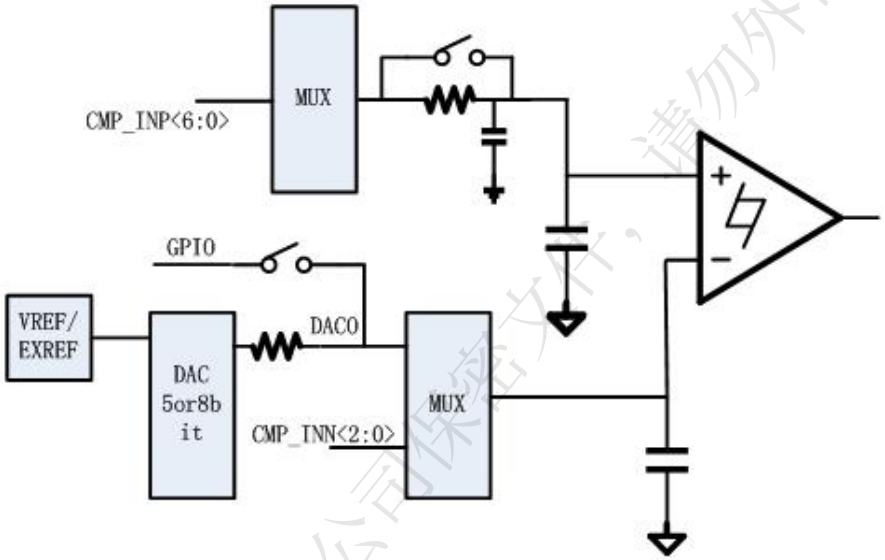

Figure 12-1 Channel structure diagram

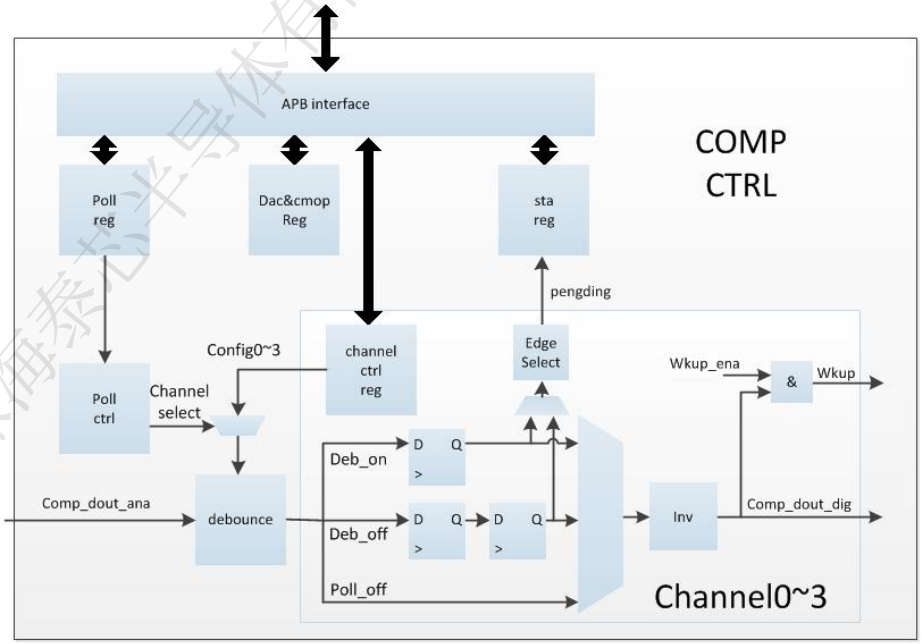

Figure 12-2 Control structure

12.3.2. Comparator input and output

The I/O pin that serves as the comparator input must be set to analog mode in the GPIO register.

The output of the comparator can be redirected to multiple I/O ports.

The output of the comparator is internally connected to the input of multiple timers: captured as input, measuring the timing.

The output of the comparator is internally connected to the input of multiple EPWM: as the brake.

12.3.3. Comparator operating mode

Comparators are divided into normal working mode and polling working mode.

In normal mode, the comparator can only compare the channels set by the software, and the operation is done by configuring CMP_CHAN0. The software operation flow is as follows:

1. Configure CMP_CON register, set the working conditions of DAC and CMP;
2. Configure the POLL_INPSEL and POLL_INNSEL bits of the CMP_CHAN0 register to select the signal for comparison;
3. Configure CMP_CHAN0 register interrupt enable, reverse enable, filter period, wake edge and wake up enable;
4. Configure the CMPEN bit of the CMP_CON register to enable the comparator.

In polling operation mode, the comparator compares different channels at intervals, polling up to four channels with selectable inputs for each channel. You can configure the positive and inverse end to poll at the same time, or you can select one end to poll, the other end of the fixed channel, by configuring CMP_POLL and CMP_CHAN0~3 to complete the operation. The software operation flow is as follows:

1. Configure CMP_CON register, set the working conditions of DAC and CMP;

2. Configure the POLL_INPSEL and POLL_INNSEL bits of CMP_CHAN0~3 register to select and compare the signal;

3. Configure CMP_CHAN0~3 register interrupt enable, reverse enable, filter period, wake edge and wake up enable;

4. Configure CMP_POLL register polling channel number, polling cycle, positive and reverse polling enable and polling MASK;

5. Configure the CMPEN bit of the CMP_CON register to enable the comparator.

Note: When poll_num is 0, it defaults to normal operating mode.

12.3.4. Comparator filter control

Due to the unstable state of the output caused by the change of the input end of the comparator, the comparator provides two methods to filter the output of the comparator, digital filtering and hardware MARK.

1. Digital filtering is to filter the output result of the comparator directly filt_num (COMP_CHANx register) clock cycle.

2. Hardware MARK means that the output of the comparator is masked by the poll_mark (COMP_POLL register) clock cycle before the polling cycle of the CHANNEL. If the output stability of the comparator requires 3 clock cycles, poll_mark can be matched to 4, and the hardware will take the results of the comparator after 4 clock cycles for storage, and directly shield the unstable results.

In the ordinary working mode, only the digital filtering method can be used. In polling work mode, both methods can be used.

12.3.5. Comparator polling cycle

In polling working mode, the polling period needs to satisfy the following formula:

1. Use a digital filter: Poll_period>= 2 * filt_num + 6

2. Use hardware MARK: Poll_period>=poll_mark + 2

3 Check whether ①Poll_period>=poll_mark + 2 *filt_num+ 6②Filt_num + 5 <= poll_mark

In polling mode, the signal holding time at the input end of the comparator should meet the following formula:

Hold_time>poll_num * poll_period

Note: When polling and channel inverting output, hardware MARK mode must be used.

12.3.6. Comparator interrupt and wake up

The output of the comparator can be internally connected to external interrupt and time controllers. The comparator outputs a wake signal that can generate an interrupt, event, or be used to exit low power mode.

12.3.7. Comparator lock mechanism

Comparators can be used for safety purposes, such as overcurrent protection or overheat protection. In some applications with specific security requirements, it is necessary to ensure that the comparator Settings cannot be changed by invalid register access or program counter corruption.

For this purpose, the comparator control and status registers can be set to write protection.

Once set, the LOCK bit must be set to 1, which causes the entire register to become read-only, including the LOCK bit.

Write protection can only be cleared by MCU reset.

12.3.8. Comparator hysteresis

The configurable hysteresis voltage of the comparator prevents unwanted output

changes from generating noisy signals. Hysteresis can be suppressed without the need to force hysteresis voltage.

12.4. Registers

12.4.1. Register base address

Name	Base Address	Description
COMP	0x40010200	Comparator base address

12.4.2. Register list

Offset Address	Name	Description
0x0000	COMP_CON	Control register
0x0004	COMP_POLL	Polling register
0x0008	COMP_CHAN0	Channel 0 register
0x000c	COMP_CHAN1	Channel 1 register
0x0010	COMP_CHAN2	Channel 2 register
0x0014	COMP_CHAN3	Channel 3 register
0x0018	COMP_CLR	Clear register
0x001C	COMP_STA	Status register

12.4.3. Register Details

12.4.3.1. COMP_CON

Bit(s)	Name	Description	R/W	Reset
31	Lock	**Comparator lock** This bit can only be written once, set to "1" by the software and cleared by the system reset. It makes all control bits of the comparator	RW	0x0

		read-only. 0x0: COMP_CON can be read and written 0x1: COMP_CON is read-only		
30:29	reserve	-	-	-
28:21	Dak	DAC data entry	RW	0x0
20:19	Cmphyst	**CMP hysteresis function selection** 0x0: There is no hysteresis 0x1:20mV 0x2:40mV 0x3:70mV	RW	0x0
18:17	Cmpimir	**CMP bias current ratio** 0x0:2:1 0x1:3:1 0x2:4:1 0x3:5:1	RW	0x2
16:15	Cmpmode	**Comparator working mode selection** 0x0: Lowest power mode 0x1: This low power mode 0x2: Regular mode 0x3: Maximum speed mode	RW	0x2
14:13	Daflres	**DAC Filter res select** 0x0:0 0x1:10K 0x2:20K 0x3:30K	RW	0x0
12:11	Dafssel	**8 bit DAC range selection** 0x0:1.6V 0x1:2.4V 0x2:3.2V 0x3:4.8V	RW	0x0
10:8	Isumres	**Fitter RES select on the positive input side of the comparator** 0x0:50K 0x1:100K 0x2:150K 0x3:200K Other: bypass RES Filter capacitance is about 3.5pf	RW	0x6
7	Cmpen	**Comparator Enable** 0x0: Off 0x1: Open	RW	0x0
6	Cmpnen	**CMP inverting input** 0x0: Off 0x1: On	RW	0x1
5	Cmppen	**CMP positive phase input** 0x0: Off	RW	0x1

		0x1: On		
4	Dabitsel	**DAC Mode Selection** 0x0:8 bit 0x1:5 bit	RW	0x1
3	Dacoutsw	**DAC output to GPIO** 0x0: Off 0x1: On	RW	0x0
2	Daen	**DAC Enable** 0x0: Off 0x1: On	RW	0x0
1	Dafpen	**8 bit DAC fast path enable** 0x0: Off 0x1: On	RW	0x0
0	Darefsel	**5 bit DAC reference source (range) selection** 0x0: inside ref,1.2V 0x1: outside ref	RW	0x1

12.4.3.2. COMP_POLL

Bit(s)	Name	Description	R/W	Reset
31	lock	**Comparator lock** This bit can only be written once, set to "1" by the software and cleared by the system reset. It makes all control bits of the comparator read-only. 0x0: COMP_POLL can be read and written 0x1: COMP_POLL Read-only	RW	0x0
30:12	reserved	-	-	-
11:7	Poll_mark	**Channel mark Cycle** 0x00: Disable mark 0x01.1 CMP CLK 0x02:2 CMP CLK ... 0x1F: 31 CMP CLK	RW	0x0
6:5	Poll_num	**Number of channels per polling pass** 0x0:1 channel 0x1:2 channels 0x2:3 channels 0x3:4 channels	RW	0x0
4:2	Poll_period	**Polling period** 0x0:1 CMP CLK 0x1:2 CMP CLK 0x2:4 CMP CLK	RW	0x0

		.. 0x7:128 CMP CLK		
1	npoll_en	**Inverted end polling enabled** 0x0: Off 0x1: On	RW	0x0
0	ppoll_en	**Phase end polling enabled** 0x0: Off 0x1: Open	RW	0x0

12.4.3.3. COMP_CHANx

Bit(s)	Name	Description	R/W	Reset
31	Lock	**Comparator lock** This bit can only be written once, set to "1" by the software and cleared by the system reset. It makes all control bits of the comparator read-only. 0x0: COMP_CHAN0 can be read and written 0x1: COMP_CHAN0 is read-only	RW	0x0
30:16	Reserve	-	-	-
15:13	poll_innsel	**< 2 > Channel x enable bit** 0x0: Off 0x1: On **<1:0> Inverting end channel x input selection** 0x0: PA9 0x1: PA10 0x2: PA11 0x3: DAC	RW	0x0
12:10	poll_inpsel	**Input selection for positive phase end channel x** 0x0: PA0 0x1: PA3 0x2: PA5 0x3: PA6 0x4: AMP_VOUT1(op-amp 1 output) 0x5: AMP_VOUT2(Opamp 2 Output) 0x6: AMP_VOUT3(OPAMP 3 Output) 0x7: Disconnect from GPIO	RW	0x0
9:8	Cpu_wkup_sel	**Wake up edge selection** 0x0: Comparator 0 wakes up with rising edge 0x1: Comparator 0 wakes up along a falling edge 0x2: The comparator wakes up with both	RW	0x0

		rising and falling edges others: Retain		
7:3	filt_num	**Filter period** 0x00:0CMP CLK 0x01:1 CMP CLK 0x02:2 CMP CLK .. 0x1F: 31 CMP CLK	RW	0x0
2	Inv_en	**Inverting Enable** 0x0: Off 0x1: On	RW	0x0
1	wkup_en	**Wake up Enable** 0x0: Off 0x1: On	RW	0x0
0	Int_en	**Channel x interrupt enabled** 0x0: Off 0x1: On	RW	0x0

12.4.3.4. COMP_CLR

Bit(s)	Name	Description	R/W	Reset
31:4	reserved	-	-	-
3	Ch3_pend_clr	Write 1 Clear the comparator channel 3 output flag signal	WO	0x0
2	Ch2_pend_clr	Write 1 Clear the comparator channel 2 output flag signal	WO	0x0
1	Ch1_pend_clr	Write 1 Clear comparator channel 1 Output flag signal	WO	0x0
0	Ch0_pend_clr	Write 1 Clear the comparator channel 0 output flag signal	WO	0x0

12.4.3.5. COMP_STA

Bit(s)	Name	Description	R/W	Reset
31:4	reserved	-	-	-
3	Ch3_pend	Comparator channel 3 outputs the flag signal	RO	0x0
2	Ch2_pend	Comparator channel 2 outputs the flag signal	RO	0x0
1	Ch1_pend	Comparator channel 1 outputs the flag signal	RO	0x0
0	Ch0_pend	Comparator channel 0 outputs flag signal	RO	0x0

13. Operational Amplifier (OPA)

13.1. Introduction

The chip is embedded with three operational amplifiers, the input and output of the operational amplifier are connected to I/O, and the ADC and comparator can be connected through shared I/O.

13.2. Key features

- Track-to-track input/output
- The output is connected to the I/O

13.3. Register Trace

13.3.1. Register base address

Name	Base Address	Description
AMP	0x4002006C	Op amp base address

13.3.2. Register list

Offset Address	Name	Description
0x0000	AMP_CON0	Control register 0
0x0004	AMP_CON1	Control register 1

13.3.3. Register Details

13.3.3.1. AMP_CON0

Bit(s)	Name	Description	R/W	Reset
31:27	qcsel3	**AMP3 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1
26:24	gainsel3	**AMP3 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
23:19	qcsel2	**AMP2 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1
18:16	gainsel2	**AMP2 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
15:11	qcsel1	**AMP1 filter cap selection** 0x01:300fp 0x02:300fp 0x04:400fp 0x08:400fp 0x10:600fp	R/W	0x1
10:8	gainsel1	**AMP1 gain selection** 0x0: x4 0x1: x6 0x2: x8 0x3: x10 0x4: x12 0x5-0x7: disable feedback loop	R/W	0x2
7	ampldo_en	**AMP LDO enabled**	R/W	0x0

		0x0: Off 0x1: On		
6	bias_en	**BIAS Enable** 0x0: Off 0x1: On	R/W	0x0
5	biasadd_en	**BIASADD is enabled** 0x0: Disabled 0x1: On	R/W	0x0
4	rcchan_en	**RCCHAN enabled** 0x0: Disabled 0x1: On	R/W	0x0
3	vcmbuff_en	**VCMBUFF is enabled** 0x0: Disabled 0x1: On	R/W	0x0
2	amp3_en	**AMP3 is enabled** 0x0: Off 0x1: On	R/W	0x0
1	amp2_en	**AMP2 is enabled** 0x0: Disabled 0x1: On	R/W	0x0
0	amp1_en	**AMP1 enabled** 0x0: Off 0x1: On	R/W	0x0

13.3.3.2. AMP_CON1

Bit(s)	Name	Description	R/W	Reset
31:13	Reserved	-	-	-
12	vout2pad_en	**AMP2 output to PB2 Enable** 0x0: Off 0x1: On	R/W	0x0
11	vout1pad_en	**AMP1 output to PB1 is enabled** 0x0: Off 0x1: On	R/W	0x0
10:9	rcrsel	**AMP3 RC Filter resistor selection** 0x0:1K 0x1:10K 0x2:50K 0x3:100K	R/W	0x0
8:6	restrim3	**AMP3 feedback Configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X	R/W	0x2

		0x5-0x7: disable feedback loop		
5:3	restrim2	**AMP2 feedback configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X 0x5-0x7: disable feedback loop	R/W	0x2
2:0	restrim1	**AMP1 feedback configuration** 0x0:1.17X 0x1:1.05X 0x2:1X 0x3:0.95X 0x4:0.87X 0x5-0x7: disable feedback loop	R/W	0x2

14. ADC

14.1. Function introduction

This module is a 12bit successive approximation ADC controller. The ADC supports multiple modes of operation: single conversion and continuous conversion, selectable channel automatic scanning, and multi-channel trigger scanning. ADC startup includes software startup, external pin trigger, and other on-chip startup (timer trigger startup, epwm).

14.2. Key features

SAR ADC with up to 12-bit programmable resolution, up to 13 external input channels and 5 internal channels

up to 1.2Msps conversion rate

supports multiple modes of operation

- ✓ Single conversion mode: A/D conversion Completes a conversion in the specified channel
- ✓ Single cycle scan mode: A/D conversion completes one cycle (from a low serial number channel to a high serial number channel or from a high serial number channel to a low serial number channel) in all the specified channels
- ✓ Continuous scan mode: A/D conversion Performs the single-cycle scan mode continuously until the software stops the A/D conversion

- ✓ Multi-channel independent trigger mode: can be configured with 6 channels, each channel independent trigger source, trigger to start A/D sampling

channel sampling time, resolution can be configured by software

supports DMA transmission.

A/D conversion start condition

- ✓ Software startup
- ✓ External IO trigger boot
- ✓ Internal timer triggers start
- ✓ Internal epwm triggers startup

14.3. Structural block diagram

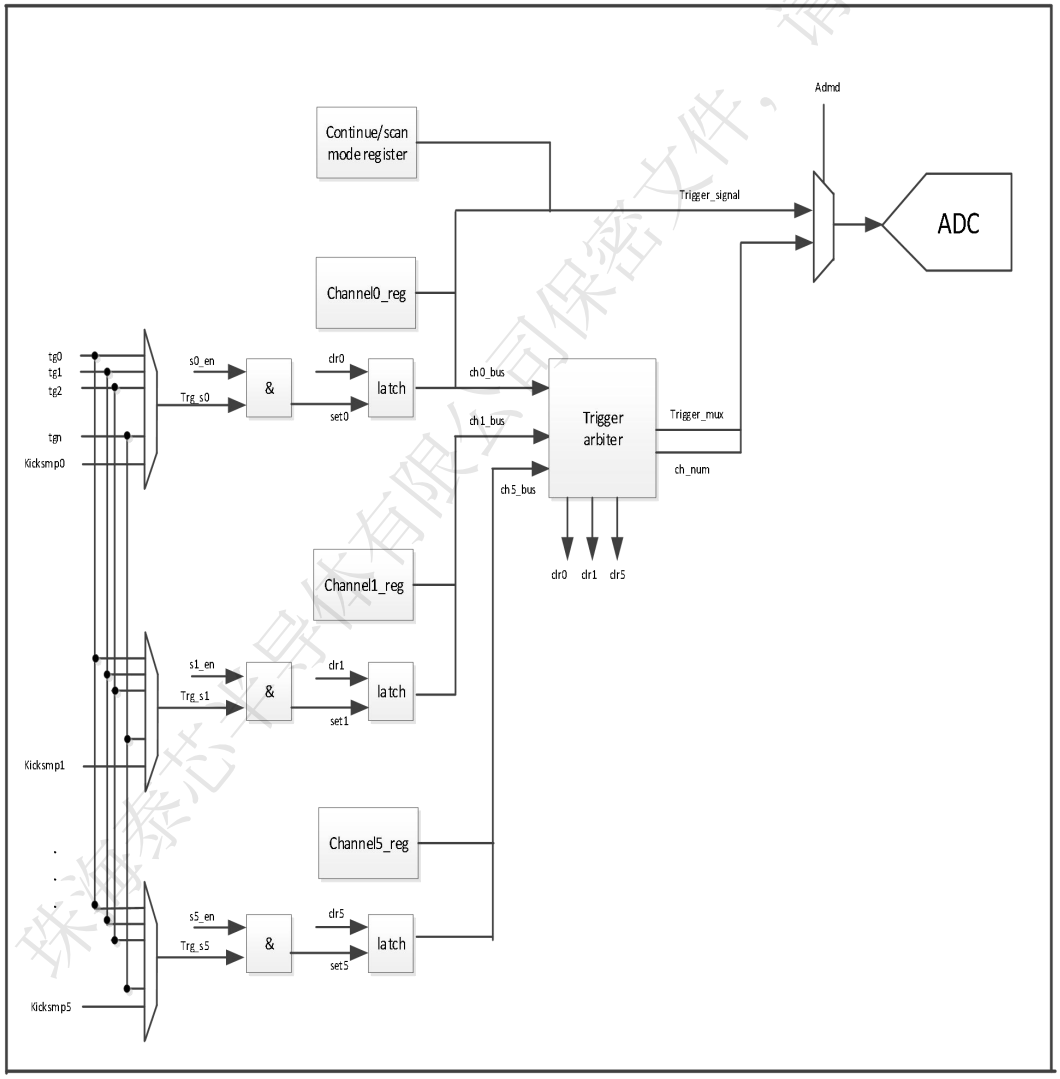

Figure 14-1 ADC functional structure diagram

14.4. Function Description

14.4.1. ADC Switch Control

Power on the ADC by setting the ADEN bit of the ADCFG register. When the ADEN bit is set for the first time, it wakes up the ADC from the power off state. After the ADC is powered on for a delay (tSTAB), after the sampling mode and channel are set, the ADST bit is set to begin the conversion. The ADST bit can be cleared to stop the conversion and the ADEN bit can be set to power off mode.

14.4.2. Channel selection

Contains 15 external input channels, internal temperature sensor channels, and internal 1.2V reference voltage channels. Each external input channel has its own independent enable bit, which can be set by setting the corresponding bit of the ADCHS register.

14.4.3. ADC operating mode

14.4.3.1. Single transition mode

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

- In software configuration, the conversion mode ADMD is set to 0 and ADEN is enabled.
- Configure the TRGCON0 registers PADSEL0 and INSSEL0 to select the sampling channel.
- Start ADC by setting ADST to 1 via software.
- When A/D conversion is complete, the data value of A/D conversion will be stored in the data register ADC_DATA0 of A/D.
- When A/D conversion is complete, the DONE0 position of the status register ADC_STA is' 1 '. If the TRGIE0 position of the control register ADC_IE is' 1 'at this time, the AD conversion end interrupt request will be generated.
- During A/D conversion, the ADST bit remains 1. When the A/D conversion ends, the ADST bit automatically clears to 0, and the A/D converter enters idle mode.

14.4.3.2. Single cycle scan mode

In the single cycle scan mode, A/D conversion will be performed from the enabled minimum serial number channel to the maximum serial number channel. The scanning channel direction can be selected by configuring register bit SCAN_DIR. The operation steps are as follows:

- The software configs the conversion mode ADMD to 1 and enables ADEN.
- Configure the DMA base address and length, and enable DMAEN.

- Configure ADC_CHS to select sampling channels.
- Software sets ADST to 1, or external trigger sets ADST to start ADC. External trigger can only be configured with TRGCON0. The startup delay can be configured by software.
- After each channel A/D conversion is completed, the A/D conversion value will be loaded into the corresponding SRAM in an orderly manner, and the DONE0 conversion end flag of ADC_STA is set. If the conversion end interrupt is set, an interrupt request will be generated after all channels have completed the conversion.
- After the conversion, the ADST bit automatically clears to 0, and the A/D converter enters the idle state.

14.4.3.3. Continuous Scan mode

In continuous scan mode, the A/D conversion is performed sequentially on the channel where CHENn bit is enabled in the ADCHS register (the scanning channel direction can be selected by configuring register bit SCAN_DIR), as follows:

- The software configures the conversion mode ADMD to 2 and enables ADEN.
- Configure the DMA base address and length, and enable DMAEN.
- Configure ADC_CHS to select sampling channels.
- The software sets ADST to 1 to start ADC.
- After each A/D conversion is completed, the A/D conversion values are loaded into the corresponding SRAM in an orderly manner.
- When the A/D conversion of all channels completes one round, the DONE0 conversion end flag of ADC_STA is set to 1; When the ADST bit is cleared to 0, the DONE0 flag is also set to 1.
- As long as the ADST bit remains 1, the ADC scans the sampling channel repeatedly according to SCANDIR. When the ADST bit is cleared to 0, the hardware waits for the current channel A/D conversion to complete and stops.

14.4.3.4. Multi-channel independent trigger mode

In this mode, 1~3 independent channels can be selected at the same time, and the corresponding ADC sampling channel can be selected by PADSELx and INSSELx of TRGCONx, and the sampling trigger source of each channel can be selected by SRCSELx register of TRGCONx. The operation steps are as follows:

- Convert mode ADMD to 3 and enable ADEN.
- Configure the PADSELx and INSSELx of TRGCONx to select corresponding sampling channels.
- Configure SRCSELx for TRGCONx to select the sampling trigger source.
- Configure KICKSMPx for TRGCONx to start sampling, or start sampling with a trigger source.
- Configure TRGENx select trigger Enable for TRGCONx.
- Read the ADC_STA corresponding DONEx bit and ADC_DATAx to get the corresponding ADC sampling data.

14.4.4. DMA request

The results of the most recent conversion are saved in the ADDATA register for single-cycle and continuous scans. During DMA transmission, the results of all scanning channels are sequentially stored in the corresponding SRAM. The dma start address and length are configured through ADC_DMAADR and ADC_DMALEN, and the number of samples stored is indicated by the DATCNT register.

14.4.5. Sampling frequency setting

The ADC clock ADCLK is obtained by PCLK frequency division, and the frequency division coefficient can be determined by setting the ADCPRE bit of the ADCFG register, that is, PCLK/(N+1) is used as the ADC clock after frequency division.

ADC fastest sampling rate =/(ADCPRE+1)/20 F_{pclk}

14.4.6. The continuous sampling interval time is configurable

For continuous scanning mode, a sampling delay can be configured between two consecutive channel samples (that is, after the completion of one channel scan, wait for a period of time before the next channel scan).

14.4.7. External Trigger Transition

ADC conversions can be triggered by external events (e.g. Timer, IO). If the TRGENx bit of the ADC_TRGCONx register is set, the conversion can be triggered using external events. An external trigger source can be selected by setting the SRCSELx bit.

For specific external trigger source selection, refer to the description of the ADC_TRGCONx control register.

The external trigger can set the delay control, the specific reference ADC_CR TRGSHIFT description.

After the trigger signal is generated, the clock cycle of N PCLK is delayed before sampling is started. If it is triggered scan mode, only the first channel sampling is delayed, the other channels are started immediately after the end of the previous sampling.

14.5. ADC Interface Timing

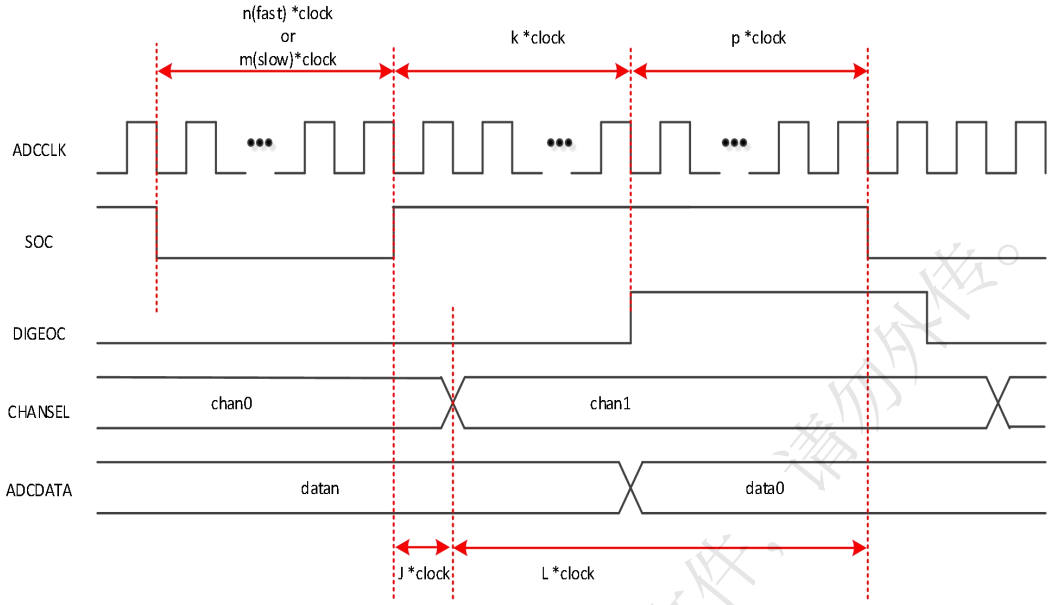

Figure 14-2 ADC timing diagram

N: Sampling period (high speed), FSMPCYC "4 to 31", at least five ADCCLKs

M: Sampling period (low speed), SSMPCYC "4 to 31", at least 5 ADCCLKs can be configured

k: Conversion time, configurable RSLTCYC "9~13"

J: Channel hold time, configurable CHHOLDCYC "3~RSLTCYC", at least 4 ADCCLK

L: channel establishment time, configurable CHSETUPCYC "3~(RSLTCYC-CHHOLDCYC)"

P: Continuous sampling interval time, can be configured D2DCYC "0~15" at least 1 ADCCLK

14.5.1. ADC power-on time sequence

Delay time	Enable signal
100us	BIASEN_VDD
300us	VCMEN_VDD
100us	CMPEN_VDD
100us	ADCEN_VDD
500us	SOC_VDD/CK_VDD

14.6. Register

14.6.1. Register base address

Name	Base Address	Description
ADC	0x40000200	ADC Base Address

14.6.2. Register list

Offset Address	Name	Description
0x0000	ADC_CFG0	Configure register 0
0x0004	ADC_CFG1	Config Register 1
0x0008	ADC_TRGCON0	Trigger register 0
0x000c	ADC_TRGCON1	Trigger register 1
0x0010	ADC_TRGCON2	Trigger register 2
0x0014	ADC_TRGCON3	Trigger register 3
0x0018	ADC_TRGCON4	Trigger register 4
0x001c	ADC_TRGCON5	Trigger register 5
0x0020	ADC_CR	Control register
0x0024	ADC_CHS	Channel register
0x0028	ADC_IE	Interrupt register
0x002c	ADC_STA	Status register
0x0030	ADC_DATA0	Data register 0
0x0034	ADC_DATA1	Data register 1
0x0038	ADC_DATA2	Data Register 2
0x003c	ADC_DATA3	Data Register 3
0x0040	ADC_DATA4	Data Register 4
0x0044	ADC_DATA5	Data Register 5
0x0048	ADC_DMAADR	DMA address register
0x004c	ADC_DMACNT	DMA count register
0x0050	ADC_DMALEN	DMA length register

0x0054	ADC_ANACON	Analog control register
0x0058	ADC_ANAPAR0	Analog configuration register 0
0x005c	ADC_ANAPAR1	Analog configuration Register 1
0x0060	ADC_ANAPAR2	Analog configuration Register 2

14.6.3. Register Details

14.6.3.1. ADC_CFG0

Bit(s)	Name	Description	R/W	Reset
31:28	CHHOLDCYC	**Channel hold time** N=n+1 ADC clock cycle width	RW	0x3
27:23	CHSETCYC	**Channel establishment time** N=n+1 ADC clock cycle width	RW	0xA
22:18	Reserve	-	-	-
17:13	FSMPCYC	**High speed sampling time configuration** N=n+1 ADC clock cycle width	RW	0x4
12:8	RSLTCYC	**Conversion cycle configuration (affects data effective accuracy)** N=n+1 clock cycle width, n is the ADC conversion data resolution 5'h9: 8-bit valid 5'ha: 9-bit valid 5'hb: 10 bits effective 5'hc: 11 bits effective 5'hd: Effective for 12 bits	RW	0xD
7:0	ADCPRE	**ADC Clock Pre-Division (ADC prescaler)** n = (n+1) frequency division	RW	0x3

14.6.3.2. ADC_CFG1

Bit(s)	Name	Description	R/W	Reset
31:27	Reserved	-	-	-
26:18	SSMPCYC	**Low speed sampling time configuration** N=n+1 ADC clock cycle width	RW	0x4
17	SMPCSEL17		RW	0x0
16	SMPCSEL16		RW	0x0
15	SMPCSEL15		RW	0x0
14	SMPCSEL14		RW	0x0
13	SMPCSEL13		RW	0x0

12	SMPCSEL12		RW	0x1
11	SMPCSEL11		RW	0x1
10	SMPCSEL10		RW	0x1
9	SMPCSEL9		RW	0x1
8	SMPCSEL8		RW	0x1
7	SMPCSEL7		RW	0x1
6	SMPCSEL6		RW	0x1
5	SMPCSEL5		RW	0x1
4	SMPCSEL4		RW	0x1
3	SMPCSEL3		RW	0x1
2	SMPCSEL2		RW	0x1
1	SMPCSEL1		RW	0x1
0	SMPCSEL0	**Channel sampling speed selection** 0x0: Low speed 0x1: High speed [12:0] Corresponds to external channel AINPAD[12:0] [17:13] corresponds to the inner channel AIN[4:0] 0= low speed, corresponding channel sampling period is SSMPCYC 1= High speed, corresponding channel sampling period is FSMPCYC	RW	0x1

14.6.3.3. ADC_TRGCON0/1/2/3/4/5

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15	OUTSEL	ADC result output range selection 0x0: [0,2^12-1] 0x1: [-2^11,2^11-1]	RW	0x0
14:13	Reserved	-	-	-
12:8	CHANSEL	Sampling channel selection: 0x0: AINPAD[0] = AMP_VOUT1 0x1: AINPAD[1] = AMP_VOUT2 0x2: AINPAD[2] = AMP_VOUT3 0x3: AINPAD[3] = PA2 0x4: AINPAD[4] = PA5 0x5: AINPAD[5] = PA8 0x6: AINPAD[6] = PA9 0x7: AINPAD[7] = PA10 0x8: AINPAD[8] = PA11 0x9: AINPAD[9] = PA12	RW	0x0

		0xA: AINPAD[10] = PA13 0xB: AINPAD[11] = PA14 0xC: AINPAD[12] = PB3 0xD: AIN[0] = Select internal PLL_CPOUT_ADC 0xE: AIN[1] = Select internal PMU_VPTAT 0xF: AIN[2] = Select internal PMU_VREF_BGBUF 0x10: AIN[3] = Select internal vdd voltage 0x11: AIN[4] = Select the internal 0.5*VCCA voltage 0x12~0x1F: Do not select any channel		
7:4	SRCSELx	Channel triggers source selection 0x0: epwn0_soca 0x1: epwn0_socb 0x2: epwn1_soca 0x3: epwn1_socb 0x4: epwn2_soca 0x5: epwn2_socb 0x6: epwn3_soca 0x7: epwn3_socb 0x8: timer1 0x9: timer2 0xA: timer3 0xB: timer4 0xC: timer5 0xD: adkey_trgio0 = TRG13_SEL? timer0 :PA11; 0xE: adkey_trgio1 = PA14 0xF: software KICKSMPx Trigger condition is rising edge	RW	0xC
3	TRGENx	Trigger channel Enable High active 0x0: Closed 0x1: On	RW	0x0
2:1	Reserved	-	-	-
0	KICKSMPx	Software enables sampling High active 0x0: Off 0x1: On	WO	0x0

Note: Currently there are 3 trigger sampling channels with each register configured for the corresponding channel. Priority: TRG0 > TRG1 > TRG2

14.6.3.4. ADC_CR

Bit(s)	Name	Description	R/W	Reset
31:22	Reserved	-	-	-

21:20	CAL_CTRL1	ADC channels PAD[3]~PAD[12],AIN[0]~AIN[4] Calibration control bit 0x0: ADC result Bypass 0x1: ADC result performs ADC calibration, and the calibration parameter value selects the initialization data in eflash 0x3: ADC result is ADC calibration, and calibration parameter value is selected as software configuration value ADC_ANAPAR0	RW	0x1
19:17	CAL_CTRL0	ADC channels PAD[0],PAD[1],PAD[2] Calibrate control bit 0x0: ADC result Bypass 0x1: ADC result only performs ADC calibration, and the calibration parameter value selects the initialization data in eflash 0x3: ADC result is ADC+AMP calibrated, and AMP Gain value selects initialization data in eflash 0x7: ADC result is ADC+AMP calibrated, and AMP Gain value is ADC_ANAPAR1/2	RW	0x3
16	SW_RST	The software resets the internal state machine of the module. The high level is valid	WO	0x0
15:12	D2DCYC	The interval time period between two consecutive samples, N=n+1 ADC clock period width	RW	0x0
11	Reserved		-	-
10:8	TRGSHIFT	External trigger shift sample After the trigger signal is generated, delay N PCLK clock cycles before sampling begins. 0x0: No delay 1:4 cycles 0x2:8 cycles 3:16 cycles 0x4:32 cycles 5:64 cycles 0x6:128 cycles 7:25 6 cycles	RW	0x0
7	SCANDIR	ADC scan Channel Sequence (ADC scan direction) Set the sequence of scan channels when scanning in periodic or continuous mode 0x0: ADC channel selector register is scanned in order from low to high I.e. : PAD[0],PAD[1]... PAD[12],IN[0],IN[2]... IN[4] 0x1:ADC channel selector registers are scanned in order from high to low Namely: IN[4],IN[3]...	RW	0x0

		IN[0],PAD[12],PAD[11]... PAD[0]		
6	TRG13_SEL	Trigger source 13 select bits: 0x0: Select IOPA11 trigger 0x1: Select time1 trigger	RW	0x0
5	TESTMD	Conduct the voltage of internal AIN[0]~AIN[4] to PA2 for test use only, AIN[0]~AIN[4] selected by ADC_TRGCON0[12:8] configuration 0x0: test is invalid, ADC is working properly 0x1: test is valid, ADC is not working	RW	0x0
4:3	ADMD	A/D Conversion mode (ADC mode) 0x0: Single conversion 0x1: Periodic scan 0x2: Continuous scan 0x3: Trigger mode When changing the conversion mode, the software first disables the ADST bit.	RW	0x0
2	ADST	ADST: A/D Conversion start (ADC start) 0x0: Conversion ends or enters idle state 0x1: The transition begins ADST can be set in the following two ways: In single mode or single cycle mode, the ADST will be automatically cleared by the hardware after the conversion is completed; In continuous scan mode, the A/D conversion will continue until the software writes 0 to the bit or the system resets. Single conversion, periodic scan, Continuous scan uses this bit to start ADC sampling	RW	0x0
1	DMAEN	DMA enable (Direct memory access enable) 0x0: DMA disabled 0x1: DMA request enabled	RW	0x0
0	ADEN	A/D Conversion enable (ADC enable) 0x0: disabled 0x1: Enabled	RW	0x0

14.6.3.5. ADC_CHS

Bit(s)	Name	Description	R/W	Reset
31:18	Reserved	-	-	-
17:0	CHxEN	**ADC input channel enabled in single cycle/continuous scan mode** 0x0: Disabled	RW	0x0

		0x1: Enabled Corresponding channel mapping: CHxEN[0] : AINPAD[0] = AMP_VOUT1 CHxEN[1] : AINPAD[1] = AMP_VOUT2 CHxEN[2] : AINPAD[2] = AMP_VOUT3 CHxEN[3] : AINPAD[3] = PA2 CHxEN[4] : AINPAD[4] = PA5 CHxEN[5] : AINPAD[5] = PA8 CHxEN[6] : AINPAD[6] = PA9 CHxEN[7] : AINPAD[7] = PA10 CHxEN[8] : AINPAD[8] = PA11 CHxEN[9] : AINPAD[9] = PA12 CHxEN[10] : AINPAD[10] = PA13 CHxEN[11] : AINPAD[11] = PA14 CHxEN[12] : AINPAD[12] = PB3 CHxEN[13] : AIN[0] = Select internal PLL_CPOUT_ADC CHxEN[14] : AIN[1] = Select internal PMU_VPTAT CHxEN[15] : AIN[2] = Select internal PMU_VREF_BGBUF CHxEN[16] : AIN[3] = Select the internal vdd voltage CHxEN[17] : AIN[4] = Select the internal 0.5*VCCA voltage;		

14.6.3.6. ADC_IE

Bit(s)	Name	Description	R/W	Reset
31:22	Reserved	-	-	-
21:16	INTADR_SEL	**A/D conversion end DONE Interrupt entry address selection bit** 0x0: The interrupt entry address is number 17 0x1: Interrupt entry address is number 23 Bit16: Channel 0 Bit17: Channel 1 Bit18: Channel 2 Bit19: Channel 3 Bit20: Channel 4 Bit21: Channel 5	RW	0x0
15:14	Reserved	-	-	-
13	DMAFIE	**DMA Full interrupt enabled**	RW	0x0
12	DMAHIE	**DMA half-full interrupt enabled**	RW	0x0
11:7	Reserved	-	-	-
6	OVRIE	overrun Interrupt Enable	RW	0x0
5:0	TRGIE0/1/2/3/4/5	**A/D interrupt enable bit** 0x0: Disables A/D interrupt 0x1: Enables A/D interrupts	RW	0x0

		Bit0: Channel 0 Bit1: Channel 1 Bit2: Channel 2 Bit3: Channel 3 Bit4: Channel 4 Bit5: Channel 5		

14.6.3.7. ADC_STA

Bit(s)	Name	Description	R/W	Reset
31:22	Reserved	-	-	-
21:15	CHANNELSEL	**This 7 bit shows the channel to which the current data corresponds** n = Conversion data for channel n	0x0	RO
14	BUSY	**Busy/free (Busy)** 0x0: The A/D converter is idle 0x1: The A/D converter is busy	0x0	RW
13	DMAFF	**DMA Full Full flag** This flag bit writes' 1 'to clear zero	0x0	RW
12	DMAHF	**DMA Half full flag** This flag bit writes' 1 'to clear zero	0x0	RW
11:6	OVRUNO/1/2/3/4/5	**Channel data overwrites flag bits** 0x0: ADC_DATA result of the latest data conversion 0x1: ADC_DATA is overwritten If DATA[15:0] is not read before a new conversion is loaded into the register, OVERRUN will set '1' Write '1' to clear the flag Bit6: Channel 0 Bit7: Channel 1 Bit8: Channel 2 Bit9: Channel 3 Bit10: Channel 4 Bit11: Channel 5	0x0	RW
5:0	DONEO/1/2/3/4/5	**A/D converts the end flag bit** This bit is set by the hardware at the end of the channel group conversion and cleared by the software. 0x0: The A/D conversion is not complete 0x1: The A/D conversion is complete The flag bit is written '1' to clear Bit0: Channel 0 Bit1: Channel 1	0x0	RW

		Bit2: Channel 2 Bit3: Channel 3 Bit4: Channel 4 Bit5: Channel 5		

Note: When the DONE signal clears, the hardware thinks that the software has read ADC_DATA, OVRUN will not set 1.

14.6.3.8. ADC_DATA0

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	DATA0	A/D conversion result (Transfer data) Single sampling, periodic scanning, continuous scanning, and The data that triggers channel 0 is stored in this register	RW	0x0

14.6.3.9. ADC_DATA1/2/3/4/5

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	DATA	A/D conversion results (Transfer data) The data of the trigger channel 1/2/4/5 is stored in this register	RW	0x0

14.6.3.10. ADC_DMAADR

Bit(s)	Name	Description	R/W	Reset
31:0	DMAADR	DMA start address, byte address	RW	0x0

14.6.3.11. ADC_DMACNT

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-
11:0	dmadatcnt	Dmadatcnt Number of ADC data that has been DMA completed	RO	0x0

14.6.3.12. ADC_DMALEN

Bit(s)	Name	Description	R/W	Reset
31:12	Reserved	-	-	-

11:0	dmalen	Configure the dma buffer length: N=n+1 ADC data	RW	0x0

14.6.3.13. ADC_ANACON

Bit(s)	Name	Description	R/W	Reset
31:13	Reserved	-	-	-
12	VOLTAGE_SEL	**ADC calibrates the supply voltage selection** 0x0:3.3V 0x1:5V	RW	0x1
11:10	TENSEL_VDD<1:0>	**Test the path selection signal** TENSEL=2'bxx 0x0: Select VCM to TOT 0x1: Select VREFP to TOT 0x2: Select the input signal path to TOT 0x3: Neither selected	RW	0x3
9	CMPBSSEL_VDD	**Comparator bias current select signal** 0x0:1X 0x1:1.5X	RW	0x0
8:7	BIASSEL_VDD<1:0>	**Internal bias current select signal** BIASSEL=2'bxx 0x0:1.33X 0x1:1X 0x2:1X 0x3:0.8X	RW	0x2
6:4	VREFSEL_VDD<2:0>	**Internal LDO gear selection signal** VREFSEL_VDD<2:0>=3'bxxx 3 'b000~3' b111 corresponds to 1.5~4.3V respectively, with a step size of 0.4V	RW	0x0
3	SELINREF_VDD	**Select the internal VREFP enable signal** 0x0: Select Internal REF 0x1: Select external REF	RW	0x0
2	VCMEN_VDD	**Internal VCMBUFFER module enables signal** 0x0: Off 0x1: On	RW	0x0
1	CMPEN_VDD	**Comparator module enables signal** 0x0: Off 0x1: On	RW	0x0
0	BIASEN_VDD	**Biasen_vdd Bias current module enables signal** 0x0: Off 0x1: On	RW	0x0

14.6.3.14. ADC_ANAPAR0

Bit(s)	Name	Description	R/W	Reset
31:28	Reserved	-	-	-
27:16	OFFSET	Parameter Offset (signed) in ADC calibration -(c*b)/a	RW	0x0
15:0	GAIN	Parameter Gain (unsigned) in ADC calibration (a/c)*2^15	RW	0x0

Note: Calibration formula ADCDATA/GAIN + OFFSET, GAIN range [2^14 2^16-1]

14.6.3.15. ADC_ANAPAR1

Bit(s)	Name	Description	R/W	Reset
31:29	Reserved	-	-	-
28:16	OFFSET0	AINPAD[0] (AMP+ADC) Parameter Offset in calibration (signed) -(D(Vcmm)*(Gaini/Gainm - D(Vcmi) + D(Vos)*(Gaini/Gainm))- D(Vos)*Gaini	RW	0x0
15:0	GAIN	Parameter Gain (unsigned) in AMP+ADC calibration (Gainm/Gaini)*2^15	RW	0x0

Note: Calibration formula ADCDATA/GAIN + OFFSET, GAIN range [2^14 2^16-1]

14.6.3.16. ADC_ANAPAR2

Bit(s)	Name	Description	R/W	Reset
31:26	Reserved			
25:13	OFFSET2	AINPAD[2] (AMP+ADC) Parameter Offset in calibration (signed) -(D(Vcmm)*(Gaini/Gainm - D(Vcmi) + D(Vos)*(Gaini/Gainm))- D(Vos)*Gaini	RW	0x0
12:0	OFFSET1	AINPAD[1] (AMP+ADC) Parameter Offset in calibration (signed) -(D(Vcmm)*(Gaini/Gainm - D(Vcmi) + D(Vos)*(Gaini/Gainm))- D(Vos)*Gaini	RW	0x0

15. TIMER

15.1. Intro

The TX32M2300 series contains a total of 6 normal timers and a watchdog. Among them, timer0/1/2/3/5 is a 16-bit timer, and timer4 is a 32-bit timer.

The timer timer0/1/2/3/5 consists of a 16-bit auto-loading counter, which is driven by a programmable pre-divider.

It is used for many purposes, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output comparison, PWM, etc.).

15.2. Key features

- 16-bit incrementing counter
- Programmable (can be modified in real time) pre-divider
- Support for selecting GPIO as a count clock source
- Different counting clock sources are supported
- Allows updating timing cycle registers after each counter cycle
- Support input capture function,
  - Supports saving up to 4 capture event Pointers at the same time
  - Each capture event polarity is configurable independently
  - You can configure whether to reset the meter value each time a capture event occurs
- PWM Output
- Support period and duty cycle auto-reload

- Use external signals to control timer and timer interconnect synchronization circuits

15.3. Function Description

15.3.1. Time base unit

The main part of the programmable advanced control timer is a 16-bit counter and its associated auto-load register. This counter can be counted upward. This counter clock is obtained by the pre-divider.

The counter, auto-load register, and pre-divider register can be read and written by the software, even if the counter is still running.

Timebase units include:

- Counter register (TIMERx_CNT)
- Period register (TIMERx_RD)
- Frequency division register (TIMERx_CON[10:8])

The autoload register is pre-loaded, and writing or reading the autoreload register will access the pre-loaded register. The contents of the preloaded register are transferred to the shadow register at each update event (the value equals the period value), and an overflow event is generated.

In timer mode, the counter counts from 0 to the period value (TIMERx_PRD), and then starts counting again from 0, and a counter overflow interrupt is generated.

The frequency division register can increase the maximum period of the count by dividing the clock frequency of the counter by 2 to the NTH power.

15.3.2. Count source Selection

Counter clocks can be supplied by the following clock sources:

- System clock
- Internal High speed RC
- External Low Speed RC
- External crystal oscillator
- External GPIO

Configure inc_src_sel of the TIMERx_CTL register to select different count sources, and configure psc of TIMERx_CTL to configure different frequency division coefficients.

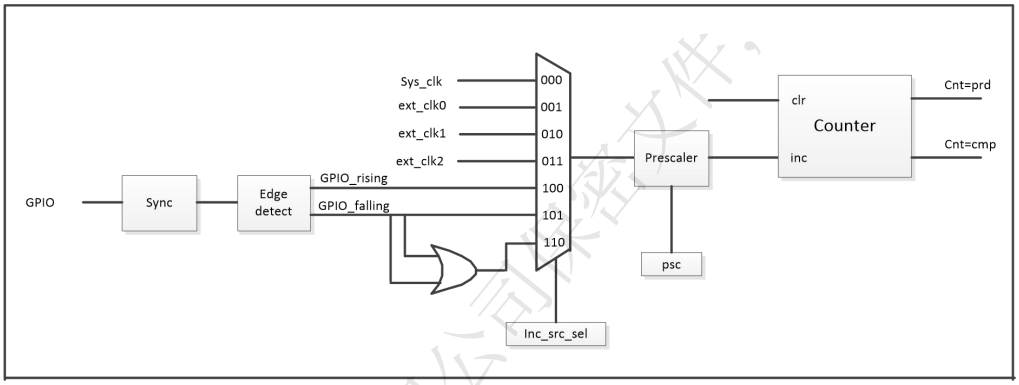

Figure 15-1 Structure of the TImer

15.3.3. Enter the capture source

Input capture supports external GPIO trigger, internal comparator trigger, and external 3 GPIO XOR together as a capture source.

Input capture channel:

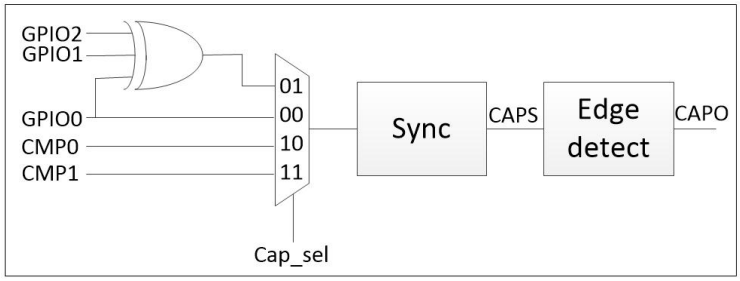

Figure 15-2 Capture structure diagram

The pin in the figure above corresponds to the GPIO input:

	CAP (Cap_sel==0x0)	XOR CAP PIN(Cap_sel==0x1)
TIMER0	PA6/A10	T0 ^ T1 ^ T2
TIMER1	PA3	T0 ^ T1 ^ T2
TIMER2	PE0/PA9	T0 ^ T1 ^ T2
TIMER3	PA13	T0 ^ T1 ^ T3
TIMER4	PA0	PA0 ^ PA1 ^ PA2
TIMER5	PA15/PA10	T0 ^ T1 ^ T5
Note: T0/T1/T2/T3/T5 refers to a CAP PIN of TIMER0/1/2/3/5, respectively		

cmp0 and cmp1 are the comparison output signals of comparator 0/1, respectively. After the input trigger path is selected by a selector, the CAPS are synchronized by a synchronizer, and then the edge detector generates the edge trigger signal, and finally the final captured signal CAP0 is selected by a selector. When the capture signal is effective, the value of the current counter is automatically stored in the TIMx_CAPx register, and the capture interrupt signal is generated.

15.3.4. Input capture mode

When a capture event occurs, save the current count to the corresponding capture register (TNRx_CAP1/2/3/4).

By configuring the cap_num of register TIMERx_CTL, 1~4 capture registers can be configured. For example, if 3 registers are used, the first capture event is generated, the capture value is saved in TIMERx_CAP1, the second capture event is generated, the capture value is saved in TIMERx_CAP2, and the third capture event is generated. The capture value is saved in TIMERx_CAP3, the fourth capture event is generated, and the capture value is saved in TIMERx_CAP1, and the sequence is repeated.

By configuring the capxpol register TIMERx_CTL, you can independently configure the polarity of each capture event, and choose rising edge or falling edge capture.

By configuring the ctrrst1/2/3/4 register TIMERx_CTL, you can independently configure whether to reset the counter value when corresponding capture events occur.

Each time a capture event occurs, the corresponding capture flag (TIMERx_FLAG)

is generated. The configuration register TIMERx_IE can be used to independently configure whether an interrupt is generated for each capture event.

In capture mode, count overflow interrupt is supported. In this mode, the count period is fixed at 16'hffff. When the count reaches 16'hffff, the overflow flag ovf_flag is generated. The overflow interrupt can be generated by configuring the TIMERx_IE register.

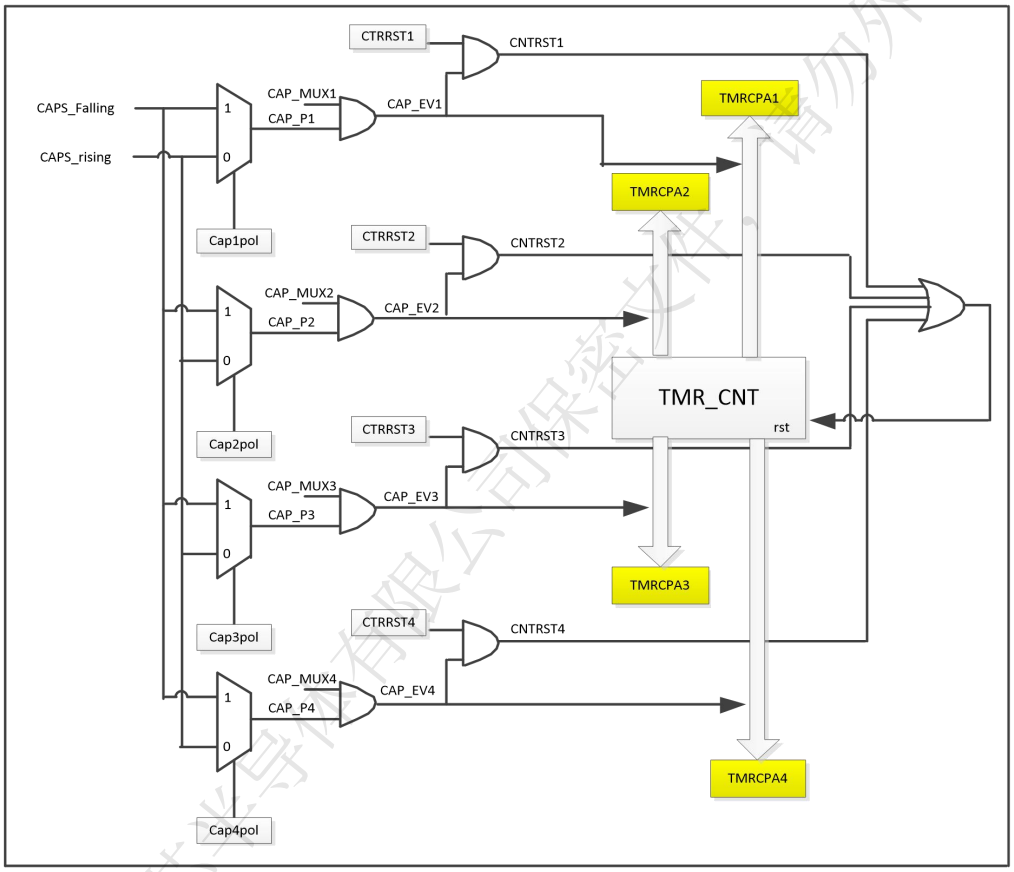

Figure 15-3 Structure block Diagram 1

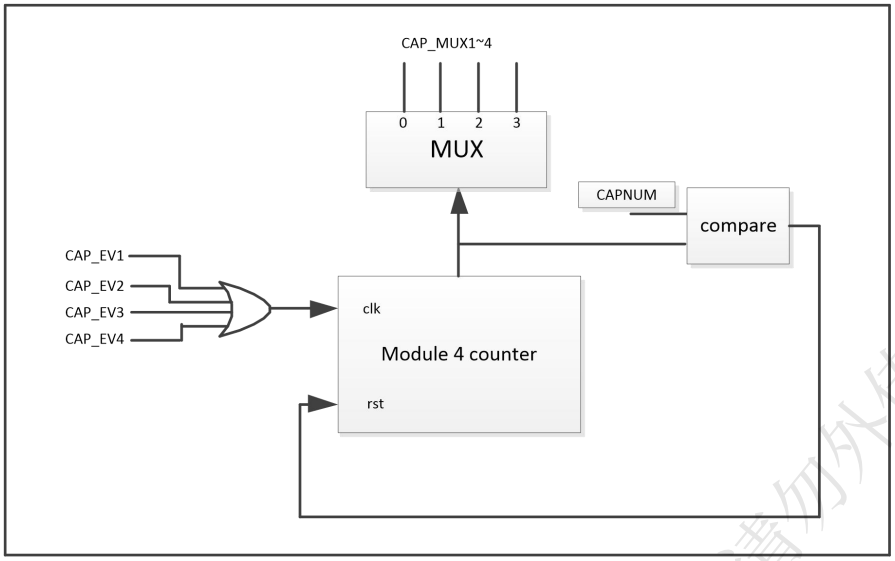

Figure 15-4 Structure block Diagram 2

15.3.5. PWM Mode

PWM mode can produce a TIMERx_PRD (TIMERx_CAP1) register to determine the period, TIMERx_CMP (TIMERx_CAP2) register to determine the duty cycle signal.

When PWM mode is used, TIMERx_CAP3 is the shadow register of TIMERx_CAP1, and TIMERx_CAP4 is the shadow register of TIMERx_CAP2. Each time the value of TIMER_CNT reaches TIMERx_RPD, the value of TIMERx_CAP3 is automatically assigned to TIMERx_CAP1, and the value of TIMERx_CAP4 is assigned to TIMERx_CAP2.

Write TIMERx_CAP1 register, will automatically write the same value to TIMERx_CAP3 register. Writing to the TIMERx_CAP2 register will automatically write the same value to the TIMERx_CAP4 register.

When the count value reaches the TIMERx_CMP value or the count value reaches the TIMERx_PRD value, the corresponding flag bit will be generated. If the corresponding interrupt enable bit is set, the interrupt will be generated. The value of the preload register (TIMERx_CAP3/4) can be modified in the interrupt, so that the next time the count reaches the period value, the pre-configured value will be automatically loaded into the period register and comparator register.

Configure the multiplexing function of the corresponding PWM output GPIO in advance, configure the required PWM period and duty cycle in the TIMERx_PRD and

TIMERx_CMP registers, configure the tmr_mode=0x2 on the TIMx_CTL register, and then start the PWM output. In this way, the desired PWM waveform signal will be generated on the corresponding IO, until the software configuration turns off the PWM output.

15.3.6. Trigger ADC sampling

The Timer also has an enhanced feature that internally connects the synco to the ADC trigger source to trigger the ADC sampling. Refer to the ADC section for details.

15.3.7. timer synchronization output

Output synchronization signal can be selected:

- Sync input SYNCI
- The value is equal to TIMERx_PRD
- The value is equal to TIMERx_CMP

15.3.8. Slave mode

The TIMER can be synchronized with an external trigger SYNCI in external mode: reset mode, trigger mode, and gate mode.

15.3.9. Reset Mode

The counter and its predivider can be reinitialized when a trigger input event occurs; For example, if the counter is operating normally, SYNCI appears an ascending edge, at which point the counter is cleared and the count is restarted from 0. At the same time, the trigger flag (in TIMERx_FLAG) is set, according to the slave_ie

setting in the TIMERx_IE register, resulting in an interrupt.

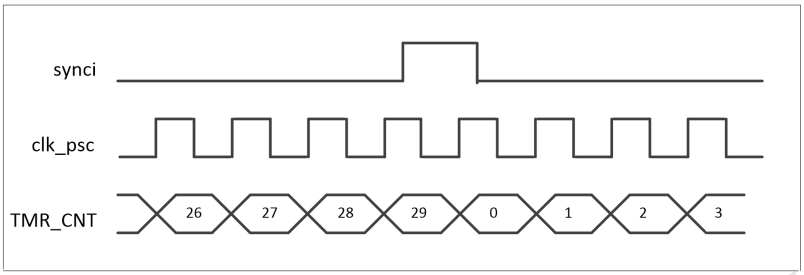

Figure 15-5 Reset sequence

15.3.10. Trigger mode

The enable of the counter depends on the event of SYNCI at the synchronous input.

In the following example, when a rising edge appears in SYNCI, the counter starts counting under the internal clock driver, setting the slave_flag bit and generating an interrupt according to the slave_ie setting in the TIMERx_IE register.

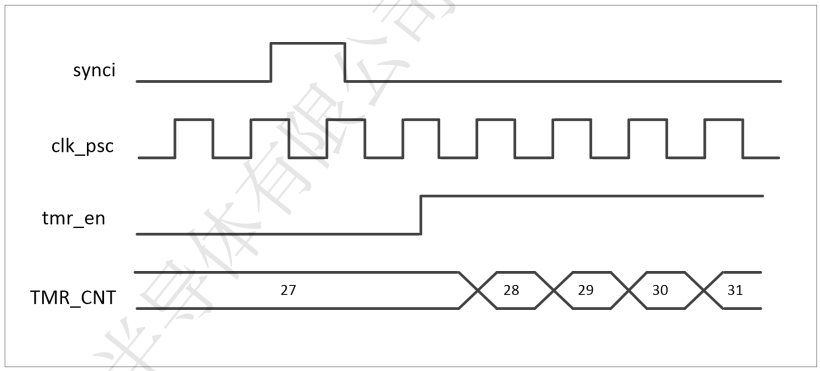

Figure 15-6 Trigger timing diagram

15.3.11. Gating mode

The enabling of the counter depends on the level of the input synchronization signal SYNCI. TIMERx_CNT counts on active SYNCI level, invalid level, stop counting.

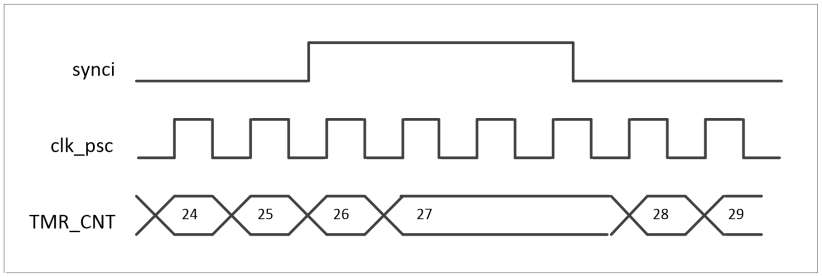

Figure 15-7 Gating timing diagram

15.3.12. DMA Transfer Mode

15.3.12.1. DMA generates PWM

The TIMER supports DMA transmission and can be used to store the period and duty cycle of PWM in advance to the specified position of SRAM. Then, the hardware automatically loads data from the SRAM each period, modifies the period and duty cycle of PWM, and generates the desired PWM signal.

The specific process is as follows:

1) Write the expected PWM period and duty cycle to the SRAM in advance.
2) Configure the TIMERx_DADR and TIMERx_DLEN registers to configure the start address and length of the DMA.
3) Configure the dma_lpbk of the TIMERx_DCTL register in cyclic or single-pass mode. If dma_lpbk=0, PWM automatically stops output after the specified dma length is executed. If dma_lpbk=1, after the execution of the specified dma length, the data is automatically fetched from the start address again until the software shuts down dma_en.
4) The dma transfer mode is enabled by configuring the dma_en of the TIMERx_DCTL register.

15.3.12.2. Captured data is written to SRAM via DMA

Support for automatically saving captured data to SRAM.

1) Clear TIMERX_DCTL to zero.

2) Configure the TIMERx_DADR and TIMERx_DLEN registers to configure the start address and length of DMA.

3) Configure the dma_fl_ie of the TIMERx_IE register to enable DMA completion interrupt.

4) Configure and enable capture.

5) After the interrupt, get the previous capture data in SRAM.

15.3.12.3. DMA interrupt flag

DMA buffer half-full interrupts and fully full interrupts are supported, and interrupts are enabled by dma_hf_ie and dma_fl_ie of the TIMERx_IE register.

Due to the area limit, the dma function is not added for the time being, so the dma function mentioned above is not available, and may be considered to add to a timer later.

15.3.13. Multiple timers are cascaded

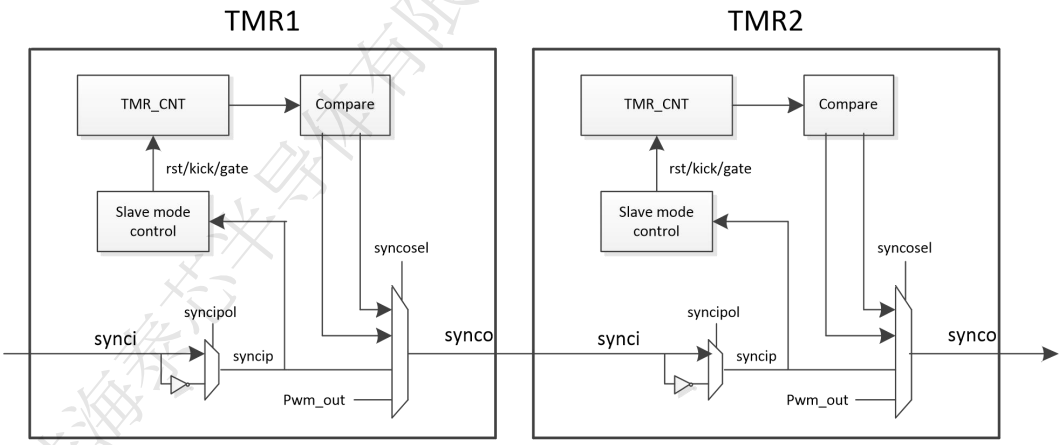

Figure 15-8 timer Cascading multiple timers

15.3.14. Multiple timer counts are cleared to zero at the same time

Supports the simultaneous clearing of all timer counts. The following example, the two TIMER value at the same time clear zero, synchronization configuration is as follows:

1) Configure syncipol = 0 of the TIMER0_CTL register,
2) Configure the syncosel = 2 of the TIMER0_CTL register to connect the synci of TIMER0 directly to the synco of TIMER0.
3) Set slave_mode = 2 to the TIMER0_CTL register for reset mode.
4) Configure syncipol = 0 for the TIMER1_CTL register.
5) Configure slave_mode = 2 for TIMER1_CTL register, reset mode.
6) When a valid edge jump is made on synci, the TIMER0 and TIMER1 counts are reset simultaneously.

15.3.15. A PWM signal with a carrier is generated

A timer acts as the carrier for the later timer; Examples are as follows:

1) Configure TIMER1 to PWM output mode (low frequency);
2) Configure syncosel of TIMER1_CTL register and select PWM_OUT to output to synco.
3) Configure slave_mode = 3 for TIMER2_CTL register, gated mode.
4) Configure TIMER2 bit PWM output mode (high frequency).
5) Enable TIMER1 and TIMER2.
6) This way, a PWM signal with a carrier is generated.

15.4. Registers

15.4.1. Register base address

Name	Base Address	Description
TIMER0	0x40001100	TIMER0 Base address
TIMER1	0x40001200	TIMER1 Base address
TIMER2	0x40001300	TIMER2 Base Address
TIMER3	0x40001400	TIMER3 Base Address
TIMER5	0x40001600	TIMER5 Base address

15.4.2. Register list

Offset Address	Name	Description
0x0000	TIMERx_CON	Control register
0x0004	TIMERx_EN	Enable register
0x0008	TIMERx_IE	Interrupt register
0x000c	TIMERx_FLG	Status register
0x0010	TIMERx_CLR	Clear register
0x0014	TIMERx_CNT	Count register
0x0018	TIMERx_CAP1	Data register 1
0x001c	TIMERx_CAP2	Data register 2
0x0020	TIMERx_CAP3	Data register 3
0x0024	TIMERx_CAP4	Data register 4
0x0028	TIMER5_DCTL	TIMER5 DMA control register
0x002c	TIMER5_DADR	TIMER5 DMA address register
0x0030	TIMER5_DLEN	TIMER5 DMA length register
0x0034	TIMER5_DCNT	TIMER5 DMA count register
0x0038	TIMER_ALLCON	TIMER total register

15.4.3. Register details

15.4.3.1. TIMERx_CON

Bit(s)	Name	Description	R/W	Reset
31:26	reserved	-	-	-
25	pwmpol	**PWM Polarity Selection**	RW	0
24	cap4pol	**Polarity selection for capture Event 4** 0x0: Rising edge 0x1: Falling edge GPIO XOR when this bit is not valid, capture Event 4 default edge detection	RW	0
23	cap3pol	**capture Event 3 Polarity selection** 0x0: Rising edge 0x1: Falling edge GPIO XOR when this bit is not valid, capture Event 3 default edge detection	RW	0
22	cap2pol	**Polarity selection for capture Event 2** 0x0: Rising edge 0x1: Falling edge GPIO XOR when this bit is invalid, capture Event 2 default edge detection	RW	0
21	cap1pol	**Polarity selection for capture Event 1** 0x0: Rising edge 0x1: Falling edge GPIO XOR when this bit is not valid, capture Event 1 default edge detection	RW	0
20	ctrrst4	**capture Event 4 automatically clears the value of CNT to zero**	RW	0
19	ctrrst3	**capture Event 3 automatically clears the value of CNT to zero**	RW	0
18	ctrrst2	**capture Event 2 automatically clears the value of CNT to zero**	RW	0
17	ctrrst1	**capture Event 1 automatically clears the value of CNT to zero**	RW	0
16:15	cap_cnt	**Capture function mode selection** 0x0: data store in CAP1 0x1: data store in CAP1 CAP2 0x2: data store in CAP1 CAP2 CAP3 0x3: data store in CAP1 CAP2 CAP3 CAP4	RW	0
14:13	cap_sel	**Capture mode** 0x0: GPIO 0x1: GPIO XOR 0x2: compare0 output 0x3: compare1 output	RW	0

12:11	syncosel	**Output synchronization signal selection** 0x0: CNT value =PRD value 0x1: CNT value =CMP value 0x2: Output the SYNCI value to SYNCO 0x3: PWM output is assigned to SYNCO	RW	
10	syncipol	**Syncipol 2 Take the polarity backwards** 0x0: Do not invert 0x1: Take the inverse	RW	0
9:8	slave_mode	**synci feature selection** 0x0: disable 0x1: kick start 0x2: reset 0x3: gating	RW	0
7:5	psc	**Timer Pre-split setting** 0x0:0 frequency division 0x1:2 Frequency division 0x2:4 frequency division 0x3:8 frequency division 0x4:16 Frequency division 0x5:32 Frequency division 0x6:64 Frequency division 0x7:128 Frequency division	RW	0
4:2	Inc_src_sel	**Timer Count source selection** 0x1: Internal Low speed RC 2 split (32K/2) 0x2: Internal High speed RC 2 crossover (26M/2) 0x3: External Crystal 2 crossover clock 0x4: timer inc pin rising 0x5: timer inc pin falling 0x6: timer inc pin rising and falling Others: System clock	RW	0
1:0	Mode_sel	**Timer Mode Selection** 0x0: timer counter mode 0x1: timer pwm mode 0x2: timer capture mode Others: reserved	RW	0

15.4.3.2. TIMERx_EN

Bit(s)	Name	Description	R/W	Reset
31:1	reserved	-	-	-
0	tmren	TIMER enables signal, active at high level	RW	0x0

15.4.3.3. TIMERx_IE

Bit(s)	Name	Description	R/W	Reset
31:10	reserved	-	-	-
9	dma_fl_ie	dma buffer full interrupt enabled	RW	0x0
8	dma_hf_ie	dma buffer half-full interrupt enabled	RW	0x0
7	slave_ie	The trigger mode or reset mode of the slave mode is disabled	RW	0x0
6	cmp_ie	When the CNT value is equal to the CMP value, interrupt is enabled, Is valid only in pwm mode	RW	0x0
5	prd_ie	Interrupt enabled when CNT value is equal to PRD value, Is only valid in counter mode/PWM mode	RW	0x0
4	ovf_ie	Interrupt enabled when CNT value overflows (16 'hffff)	RW	0x0
3	cap4_ie	capture Event 4 occurs when interrupt is enabled	RW	0x0
2	cap3_ie	capture Event 3 occurs when interrupt is enabled	RW	0x0
1	cap2_ie	capture When Capture event 2 occurs, interrupt is enabled	RW	0x0
0	cap1_ie	capture Event 1 occurs when interrupt is enabled	RW	0x0

15.4.3.4. TIMERx_FLG

Bit(s)	Name	Description	R/W	Reset
31:10	reserved		-	-
9	dma_fl_flg	dma buffer Full flag	RO	0x0
8	dma_hf_flg	dma buffer half-full flag	RO	0x0
7	slave_flg	slave mode occurrence flag (reset or trigger only)	RO	0x0
6	cmp_flg	CNT value equals CMP value flag, Valid only in pwm mode	RO	0x0
5	prd_flg	CNT value equals PRD value flag, Valid only in counter mode/PWM mode	RO	0x0
4	ovf_flg	CNT Value Overflow (16 'hffff) flag	RO	0x0
3	cap4_flg	capture Event 4 occurs flag	RO	0x0
2	cap3_flg	capture Event 3 occurs flag	RO	0x0
1	cap2_flg	capture Event 2 occurs flag	RO	0x0
0	cap1_flg	capture Event 1 occurs flag	RO	0x0

15.4.3.5. TIMERx_CLR

Bit(s)	Name	Description	R/W	Reset
31:10	reserved	-	-	-
9	dma_fl_clr	dma buffer Full flag cleared	WO	0x0
8	dma_hf_clr	dma buffer Half Full flag cleared	WO	0x0
7	slave_clr	slave mode occurs flag clear	WO	0x0
6	cmp_clr	CNT value equal to CMP value flag cleared	WO	0x0
5	prd_clr	CNT value equal to PRD value flag cleared	WO	0x0
4	ovf_clr	CNT Value Overflow (16'hffff) flag cleared	WO	0x0
3	cap4_clr	capture Event 4 Occurred flag cleared	WO	0x0
2	cap3_clr	capture Event 3 Occurred flag cleared	WO	0x0
1	cap2_clr	capture Event 2 occurs flag cleared	WO	0x0
0	cap1_clr	capture The event 1 occurrence flag is cleared	WO	0x0

15.4.3.6. TIMERx_CNT

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15:0	CNT	Count register	RW	0x0

15.4.3.7. TIMERx_CAP1

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	CAP1/PR	Capture mode: Capture register 1 Timing mode, PWM mode: Count cycle register	RW	0xFFFF

15.4.3.8. TIMERx_CAP2

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved		-	-
15: 0	CAP2/CMP	Capture mode: Capture register 2 Timing mode, PWM mode: Compare register	RW	0x0

15.4.3.9. TIMERx_CAP3

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15: 0	CAP3/PR_SD	Capture mode: Capture register 3 Timing mode, PWM mode: Count cycle shadow	RW	0xFFFF

		register		

15.4.3.10. TIMERx_CAP4

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15:0	CAP4/CMP_SD	Capture mode: Capture register 4 Timing mode, PWM mode: Compare shadow registers	RW	0x0

15.4.3.11. TIMER5_DCTL

Bit(s)	Name	Description	R/W	Reset
31:2	Reserveds			
1	dma_lpbk	dma mode selection 0: indicates the single-timer mode. disable TIMER after the specified dma length is complete 1: In cyclic mode, after the specified dma length is completed, the timer starts from the start address again	RW	0x0
0	dma_en	dma Enable	RW	0x0

15.4.3.12. TIMER5_DADR

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15: 0	dma_stadr	dma starting address (word aligned)	RW	0x0

15.4.3.13. TIMER5_DLEN

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved			
15:0	dma_len	Dma buffer length (32bit), configured as n-1 if buffer is n;	RW	0x0

15.4.3.14. TIMER5_DCNT

Bit(s)	Name	Description	R/W	Reset
31:16	Reserved	-	-	-
15: 0	dma_cnt	dma data valid number	RW	0x0

15.4.3.15. TIMER_ALLCON

Bit(s)	Name	Description	R/W	Reset
31:14	Reserved	-	-	-
13	tmr5_sync	timer5 count value clear to zero	R0	0x0
12	Reserved	-	-	-
11	tmr3_sync	timer3 count value clear to zero	R0	0x0
10	tmr2_sync	timer2 count value clear to zero	R0	0x0
9	tmr1_sync	timer1 count value clear to zero	R0	0x0
8	tmr0_sync	timer0 Count value clears to zero	R0	0x0
7:6	reserved	-	-	-
5	tmr5_kick	timer5 Start counting	R0	0x0
4	reserved	-	-	-
3	tmr3_kick	timer3 Start counting	R0	0x0
2	tmr2_kick	timer2 Start counting	R0	0x0
1	tmr1_kick	timer1 Start counting	R0	0x0
0	tmr0_kick	timer0 Starts counting	R0	0x0

16. EPWM

16.1. Intro

EPWM peripherals are a key module in many commercial and industrial-grade power electronic system controls. These systems include digital motor control, power switching mode control, uninterruptible power UPS, and other forms of power conversion modules. EPWM can sometimes be used as a DAC, and the duty cycle is the analog output of the DAC.

There are four EPWM modules in the TX32M2300, and each EPWM module supports the following functions:

1. A dedicated 16-bit time controller for cycle and frequency control.
2. Two PWM outputs (EPWMxA and EPWMxB) can be configured for the following:
   - Two independent PWM outputs for unilateral control;
   - Two independent PWM outputs for bilateral symmetric control;

- One independent PWM output for bilateral asymmetrical control;

3. And asynchronous coverage control of PWM signal by software.
4. Each EPWM comes with a total of 4 event trigger registers, CMPA,CMPB,CMPC,CMPD, which can be used to trigger PWM flip and ADC sampling.
5. Each EPWM can be programmed with other EPWM modules for lead or lag phase control.
6. Hardware-locked (synchronized) phase relationships can be performed for individual EPWM modules at each cycle.
7. Has independent rising edge and falling edge dead zone delay control.
8. The programmable fault zone (trip zone) is used for cycle-by-cycle trip control and one-shot trip control in case of failure.
9. The PWM output can be forced to a high, low, or high impedance logic level via CPU, IO, or comparator.
10. All events in the EPWM module can trigger a CPU interrupt and start ADC Start Conversion (ADC SOC).
11. Programmable events effectively reduce the burden on the CPU when interrupts occur.

.

16.2. Submodule introduction

16.2.1. Time base Counter submodule (Time-BASE)

The Timebase counter submodule has the following configuration options:

1. Calibrate the time base counter clock related to the system clock;
2. Configure the frequency and period of PWM timebase counter;
3. Set the mode of the timebase counter:
   - Incremental count
   - Decreasing count

- Increment and decrease count

4. Configure the time base phase relationship with another EPWM module;
5. Synchronize timebase counters between different modules through software or hardware;
6. Configure the direction of the timebase counter after the synchronization event has occurred (incrementing, decrement, or incrementing or subtracting counts);
7. Specify the synchronization output signal source for the EPWM module;
   - Synchronous input signal
   - Time base counter equals 0
   - The time base counter is equal to CMPA
   - A time base counter is equal to CMPB
   - No output synchronization signal is generated

16.2.2. Time base Counting comparator submodule (Counter-comparator)

1. Specifies the duty cycle of PWM waves output on EPWMxA and EPWMxB;
2. Specify the switching time on EPWMxA and EPWMxB;
3. To generate event trigger signals, including:
   - The timebase counter equals 0
   - The time base counter is equal to the period value
   - The timebase counter is equal to CMPA
   - A time base counter is equal to CMPB
   - The time base counter equals CMPC

16.2.3. Action generation submodule (action-qualifier)

The action generation submodule can convert the event Settings into various action types, thus output the required waveforms in EPWMxA and EPWMxB.

1. The action generation submodule has the following functions: generate action (set 1, clear 0, reverse);
2. Force control PWM output state through software;

16.2.4. Dead-band submodule

1. Traditional complementary dead zone relationships that control switches;
2. Specify the output rising edge delay value;
3. Specify the output falling edge delay value;
4. Ignore the dead zone module, in which case the PWM waveform passes without any change;

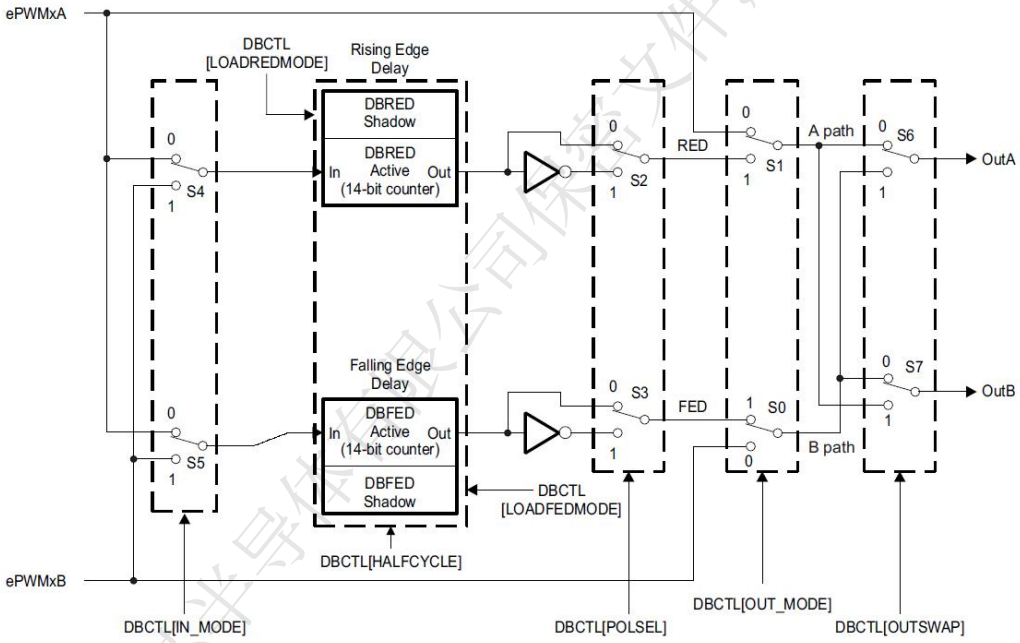

Figure 16-1 deadzone structure block diagram

16.2.5. The Event trigger submodule (Event-trigger)

The main functions of the event trigger sub-module are as follows:

1. Receive the event input generated by the timebase counter or timebase count comparator submodule;
2. Delimit the event up/down by the time base counter direction;
3. Use count logic to generate interrupt requests and initiate adc conversions

in the following cases:

- Per event
- Every two events
- Every third event

4. Provide visualization of events produced through event counters and flags;
5. Allowing software to force interrupts and initiate ADC conversions;
6. When the event is generated, the event trigger submodule manages the generation and counting of the events of the timebase submodule. The comparison submodule generates CPU interrupts and generates the pulse of the ADC to start the conversion;

16.2.6. Fault zone submodule (Trip-zone)

The main functions of the Trip-Zone submodule are as follows:

1. Fault zone input signals TZ1 to TZ6 can be flexibly mapped to any EPWM module;
2. Depending on the fault, the EPWMxA and EPWMxB outputs can be forced out as the following signals:
   - Force a high output
   - Force output low
   - Force a high impedance output
   - No action occurring
3. Supports single trip in case of short circuit or overcurrent;
4. Support cycle cycle (CBC) current limiting operation;
5. Each Trip-zone input pin can be assigned to a single or cyclic operation;
6. An interrupt can occur on any Trip-zone pin;
7. Support software to force trip;
8. If not needed, you can ignore the Trip-zone submodule;

16.3. Function description and common topologies

The synchronization function and main configuration options between EPWM are

as follows:

1. Select Syncin signal source;
2. Enable synchronous gate pulse switch, when the synchronization pulse appears, the value in the phase register is loaded into the counter;
3. Also do not do any operation, turn off the synchronous gate pulse switch;
4. SyncOut connection options:
   - Sync flow-through - connects SyncOut directly to SyncIn;
   - In Master mode, a sync signal is provided at the PWM boundary -- SyncOut is connected to CTR == 0;
   - In Master mode, a sync signal is provided at any programmable time point -- SyncOut is connected to CTR == CMPA or CTR == CMPB;
   - When the module is in a separate mode and out of sync with other modules -- SyncOut is connected to X (disabled);

The two most common types -- Master mode and Slave mode -- are shown below:

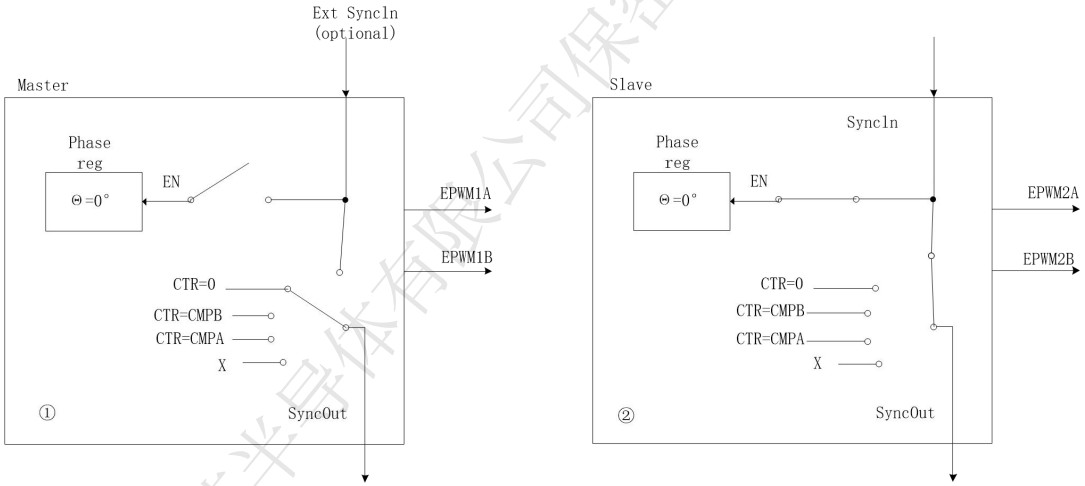

Figure 16-2 Synchronization diagram

16.3.1. Independent frequency control for multiple Buck conversion circuits

One of the simplest power conversion circuit topologies is the buck. When configuring an EPWM module as a master, two buck stages can be controlled with one frequency. If each buck circuit is required to be controlled with an independent frequency, then each converter level will need to be assigned an EPWM module. The

figure below shows the four buck stages and waveforms, each operating on an independent frequency. In this case, all four EPWM modules are configured as Master and are not synchronized. Note that even though there are four buck levels, there are only three waveforms.

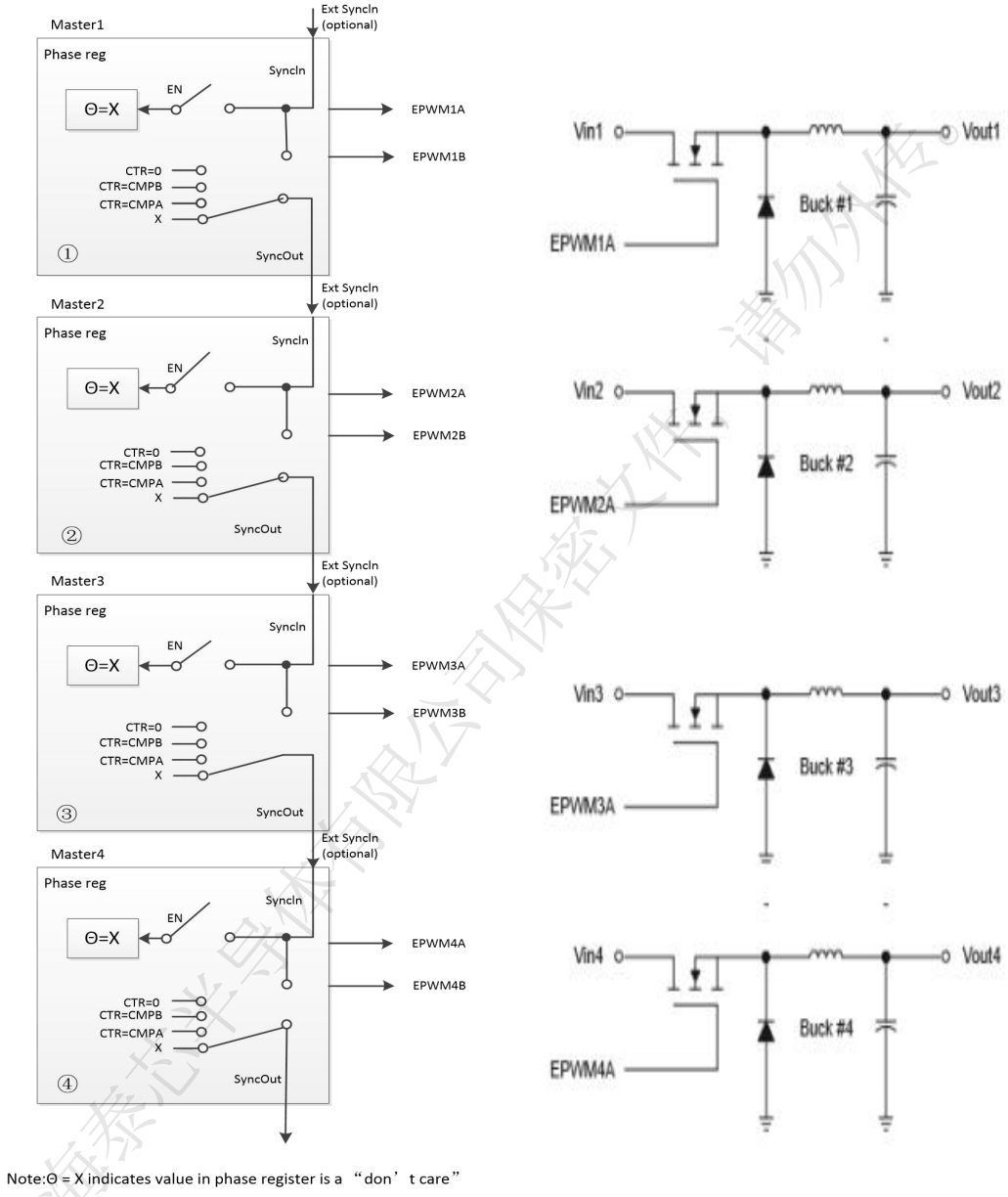

Figure 16-3 Control diagram

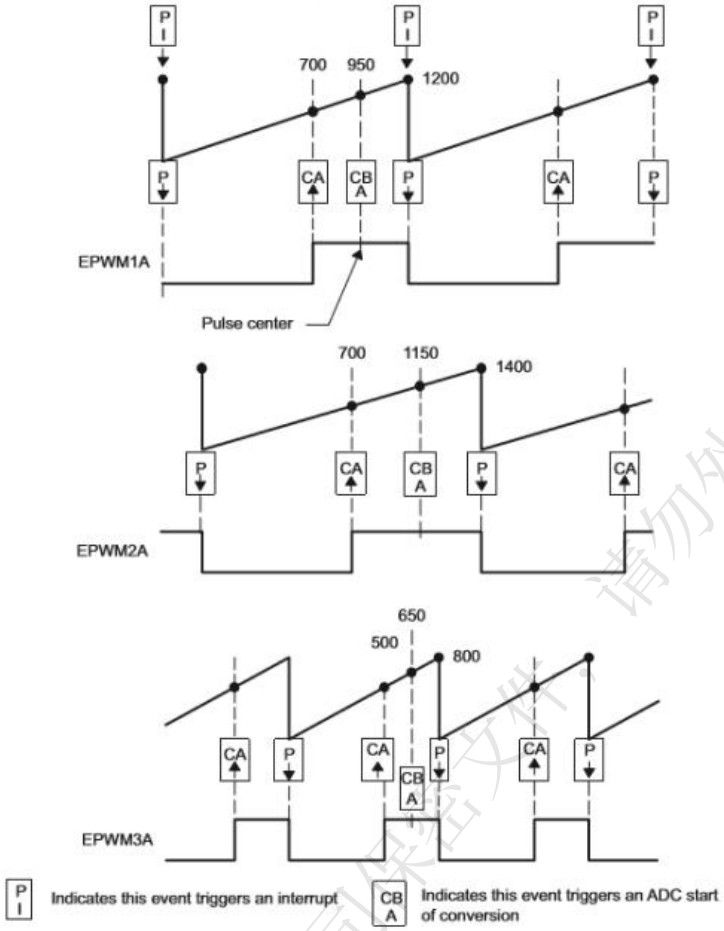

Figure 16-4 Timing diagram

16.3.2. Multiple Buck conversion circuits are controlled at the same frequency

If two Buck circuits are required to be synchronized, then EPWM2 can be configured as a slave and operate at integer multiples of EPWM1 frequencies. A sync signal from master to slave ensures that these modules remain locked. The following figure shows the model and waveform in this setting.

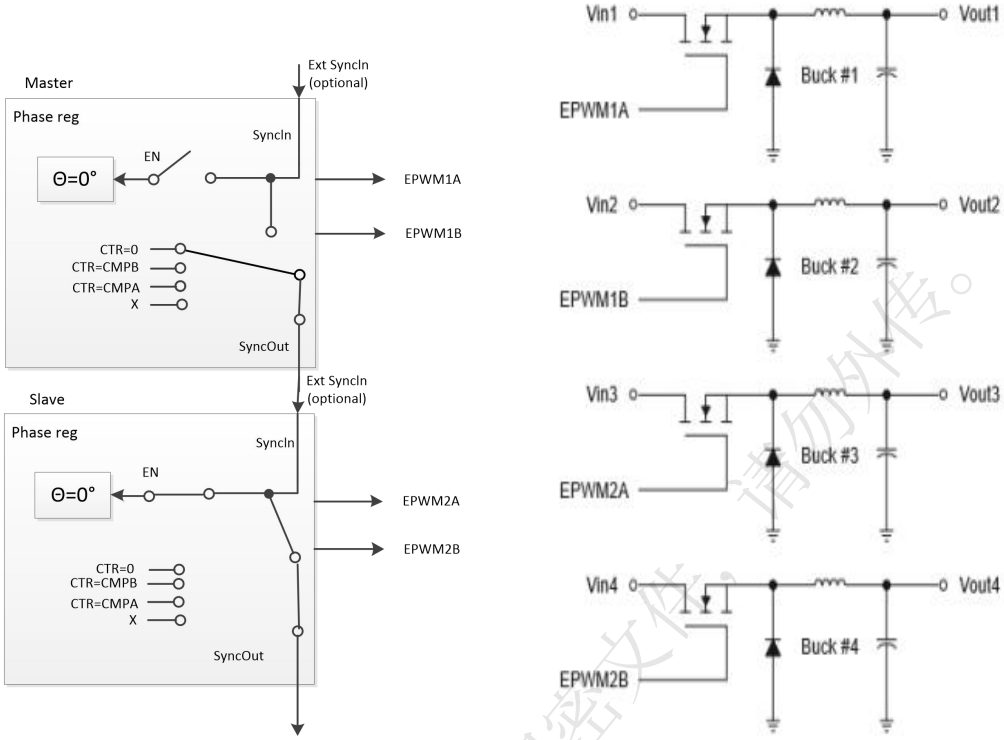

Figure 16-5 Control Diagram 2

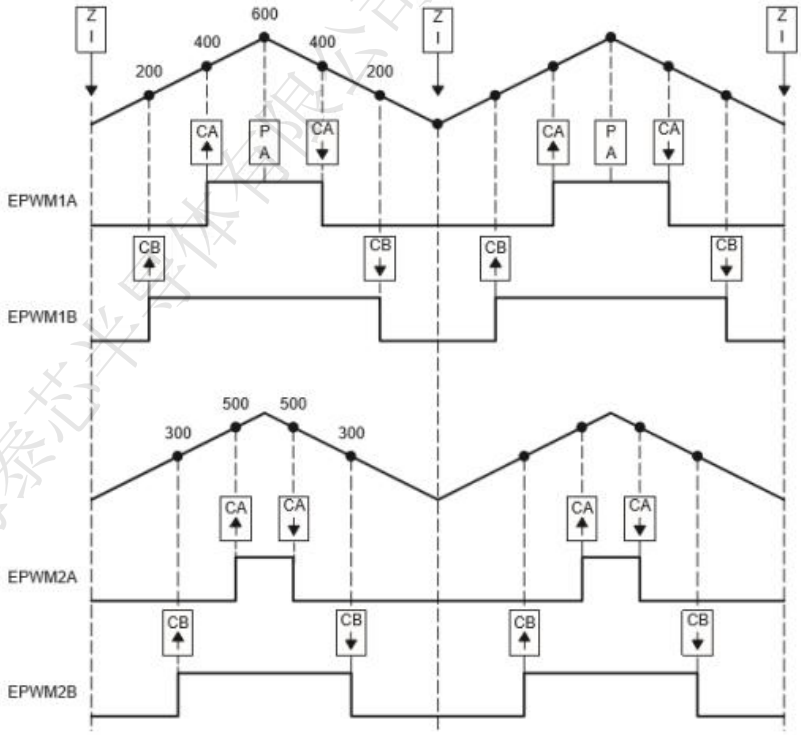

Figure 16-6 Timing Diagram 2

16.3.3. Control multiple H Half Bridge circuit (HHB) conversion circuits

Topologies that control multiple switching elements can also be handled with the same EPWM module. It is possible to control an H half bridge circuit with one EPWM module. This control method can also be extended to several H-half bridge circuits. The following figure shows that when controlling two synchronized H-half bridge circuits, the second stage can be made to operate at integer multiples of the operating frequency of the first stage. And the waveform generated under this setting.

Slave is configured to Sync flow-through. If desired, a third H half bridge can be added to be controlled by a third EPWM module and must be synchronized with the master.

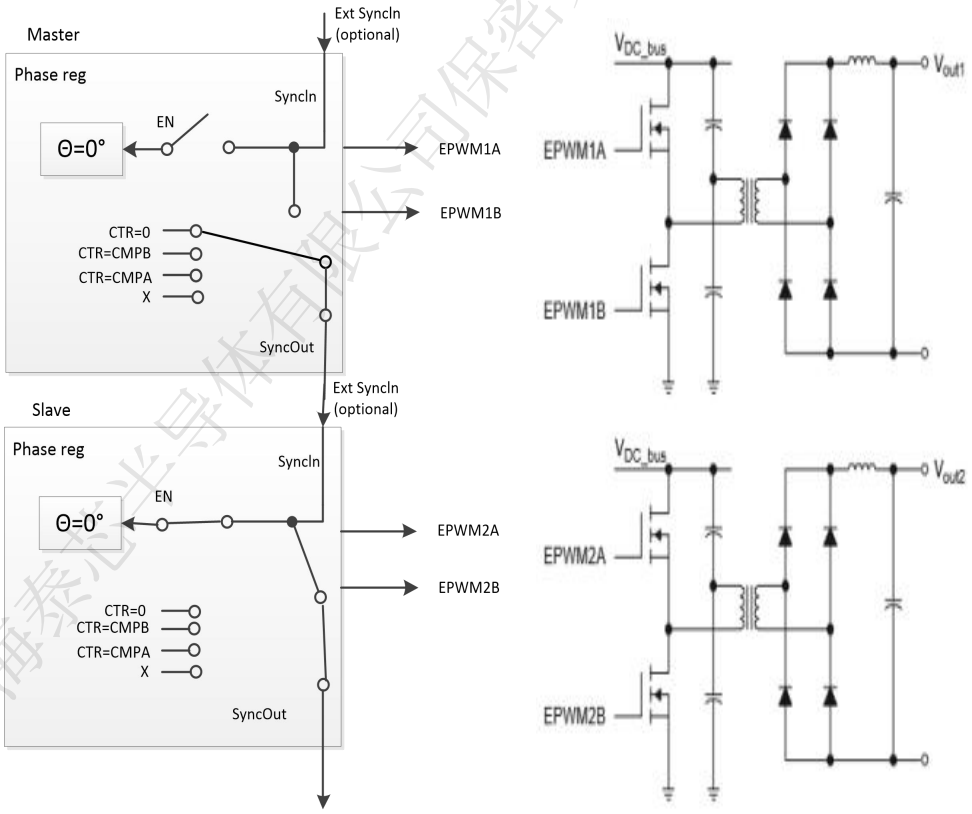

Figure 16-7 Control Diagram 3

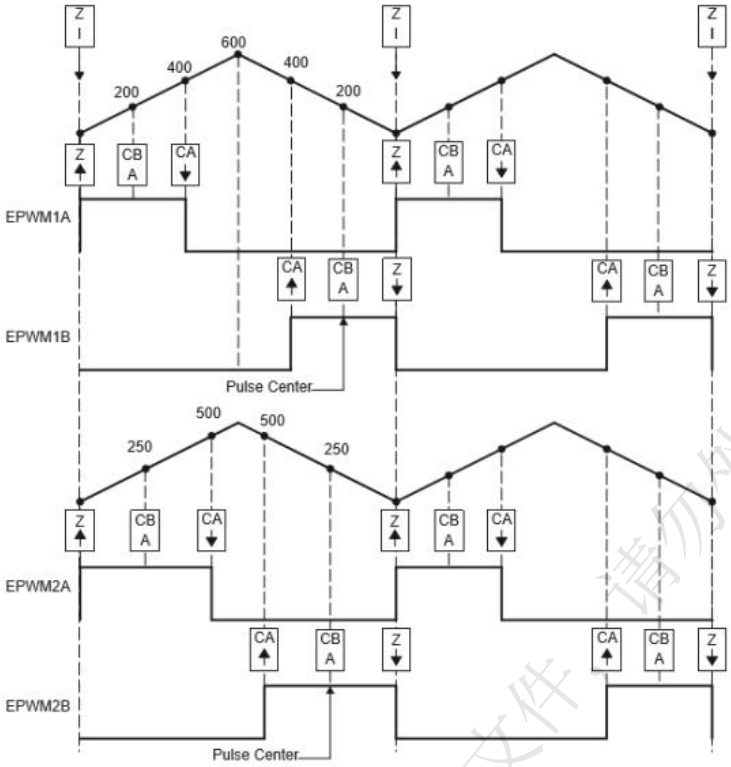

Figure 16-8 Timing Diagram 3

16.3.4. Double three-phase inverters (ACI and PMSM) that control the motor

The idea of multiple modules controlling a single power stage can be extended to three-phase inverters. In this case, the 6 switching elements can be controlled by three PWM modules, one for each branch. Each branch must be switched on and off at the same frequency and all branches synchronized. So, a combination of 1 master+2 slaves can easily solve such a need. The figure below shows how 6 PWM modules control 2 separate three-phase inverters, each of which runs a motor. The following diagram shows the operating waveform.

As shown in the previous sections, we can make each inverter operate at a different frequency (for example, in the example of Figure 3-9, modules 1 and 4 both act as masters); Or set 1 as master and 5 as slave to synchronize the two inverters. In this case, the frequencies of modules 4, 5, 6 (the three modules have equal frequencies) are integer multiples of the frequencies of modules 1, 2, 3 (the three

modules have equal frequencies).

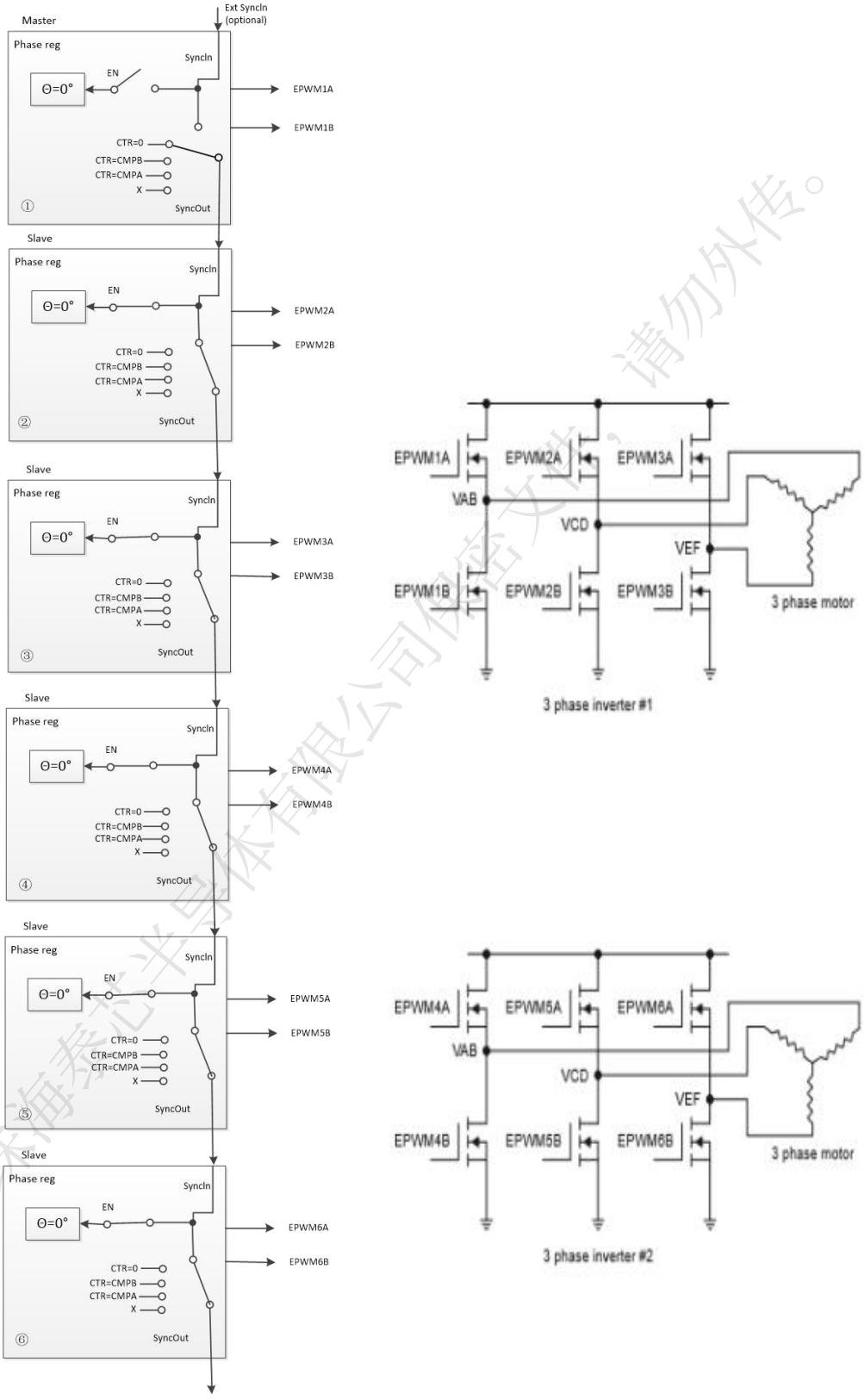

Figure 16-9 Control Diagram 4

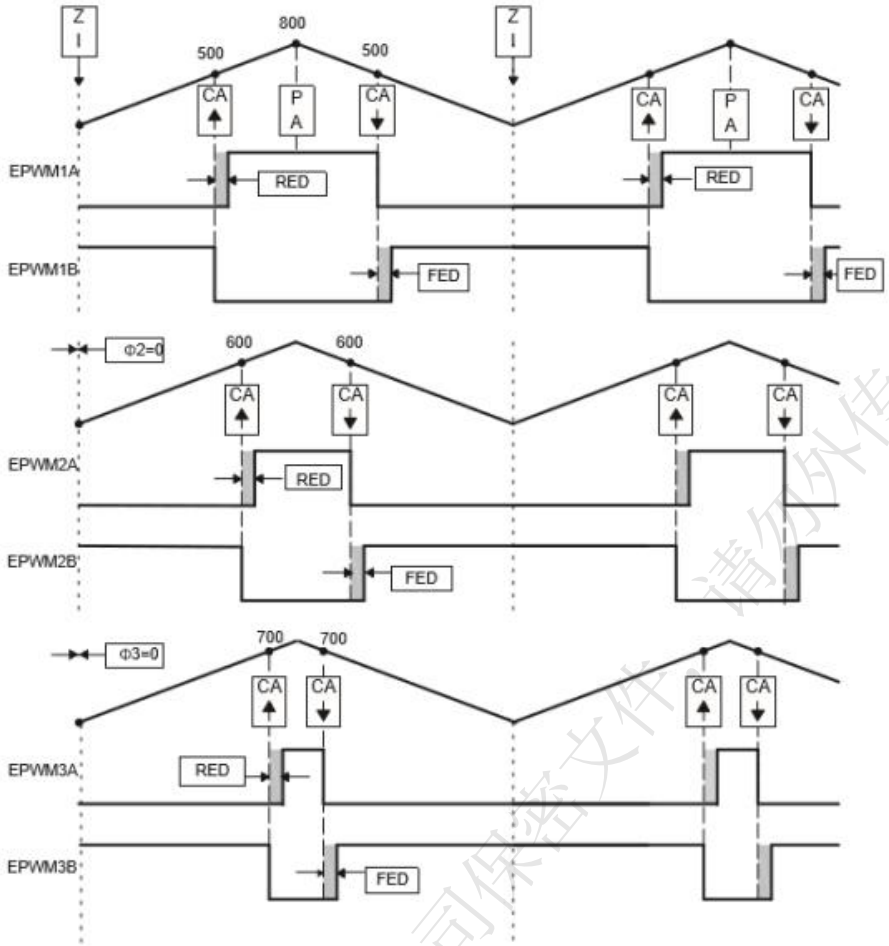

Figure 16-10 Timing Diagram 4

16.3.5. The practical application of phase control between PWM modules

Until now, none of the examples used phase registers (TBPHS). The phase register is either set to 0 or ignored completely. However, by configuring the TBPHS register with appropriate values, multiple PWM modules can handle other power topology levels that operate correctly depending on the phase relationship between the branches. A PWM module can be configured with SyncIn pulses so that the TBPHS register can be loaded into the TBCTR register. As shown in the above point of view, the following figure shows a master and slave with a phase relationship of 120°.

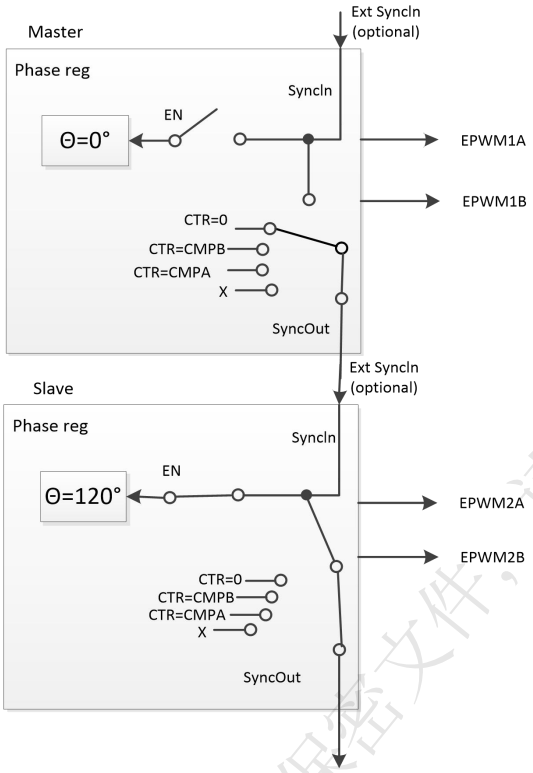

Figure 16-11 Control Diagram 5

The diagram below shows the timing waveform diagram produced under this configuration. Both Master and slave have TBPRD equal to 600. The TBPHS of a Slave = 200 (200/600*360° = 120°). When the master generates a SyncIn pulse, the value of TBPHS = 200 will be loaded into the slave's TBCTR register. So the slave's time base is always 120° ahead of the master's.

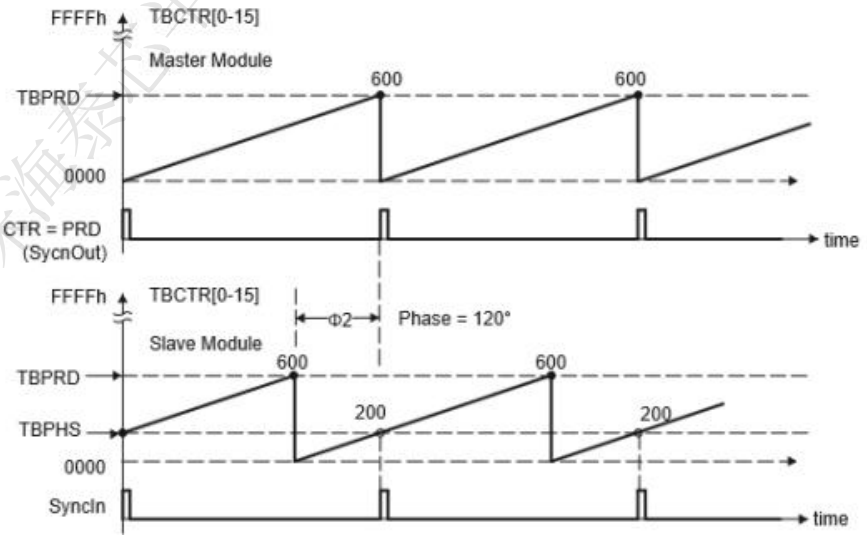

Figure 16-12 Timing Diagram 5

16.3.6. Control the three-phase staggered DC/DC switching circuit

In the figure below, the universal power topology is made use of phase offsets between modules. The system uses 3 PWM modules, of which module 1 is configured as the master. When working, the phase relationship between the adjacent modules is F = 120°. This is done by setting the values of the slave's TBPHS register 2 and TBPHS register 3 to 1/3 and 2/3 of the period, respectively. For example, if the period register is loaded with a value of 900 counts, then TBPHS (slave2) = 300 and TBPHS (slave3) = 600. All slave modules should be synchronized with master1 module.

This idea can be extended by 4 or more phases by properly setting the value of TBPHS register. The following formula provides N phase values for the TBPHS register:

TBPHS (N,M) = (TBPRD/N) * (-1)

Where N is the number of phase values

M is the number of PWM modules

For example, in the case of 3 phase values, TBPRD = 900,

TBPHS (3,2) = (900/3) * (2-1) = 300 (this is the phase value of slave module 2);

TBPHS (3,3) = (600) (this is the phase value of slave module 3).

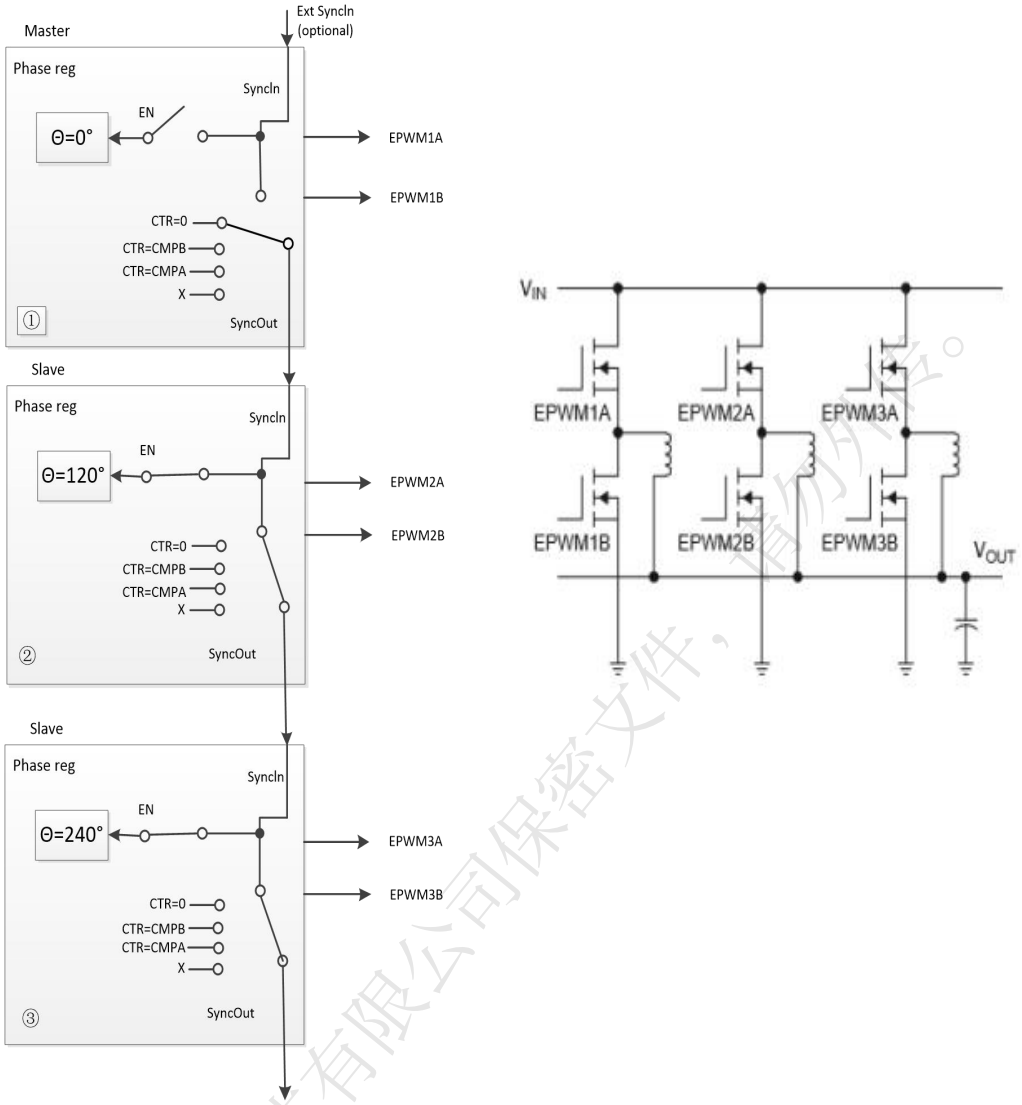

Figure 16-13 Control Diagram 6

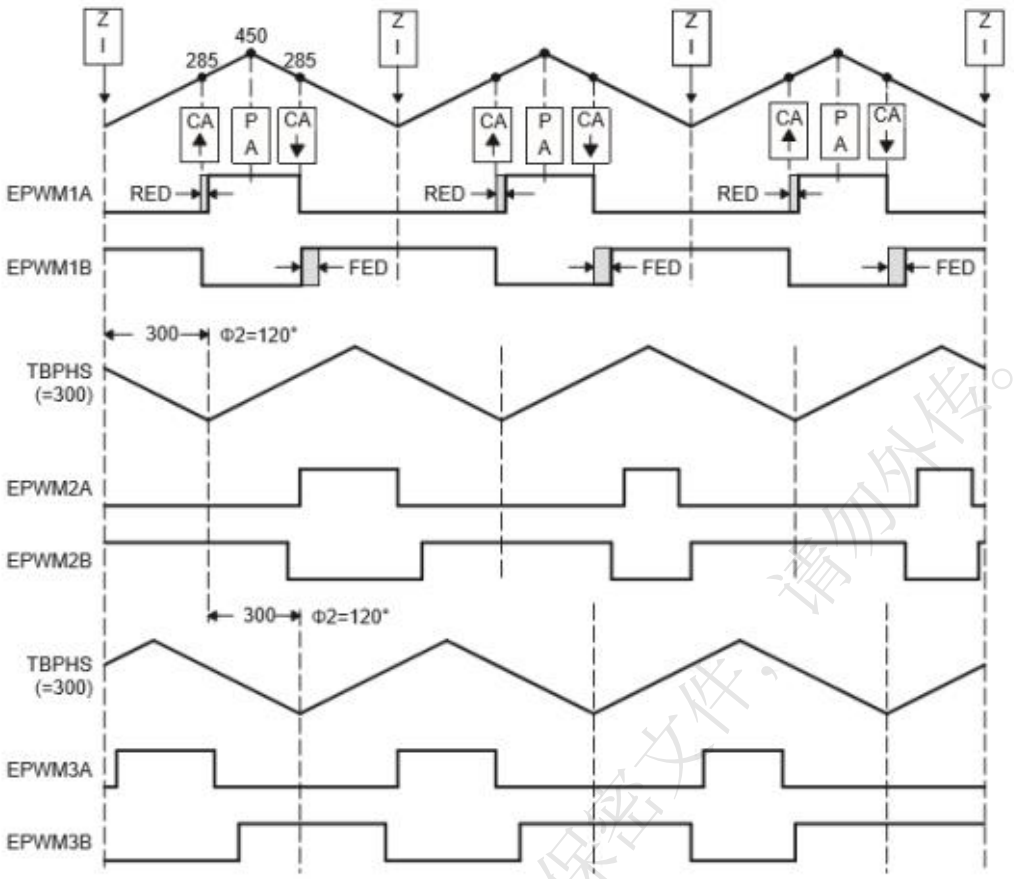

Figure 16-14 Timing Diagram 6

16.3.7. Full bridge conversion circuit (ZVSFB) that controls voltage 0 conversion

The figure below assumes a static or constant phase relationship between the modules. In this case, the control can be achieved by modulating the duty cycle. It is also possible to change the phase values dynamically in the basis of the cycle. This feature allows the module itself to control the power-level topology like a phase-shifted full bridge or a zero-voltage conversion full bridge. The parameter controlled here is not the duty cycle (which remains constant or approximately 50%), but the phase relationship between the branches. Such a system can be achieved by controlling a single power stage through the resource configuration of two PWM modules, which requires the control of four switching elements. The figure below

shows a synchronized master/slave combination working together to control an H-bridge. In this case, both the master and slave are required to convert at the same PWM frequency. The phase can be controlled via the slave's phase register (TBPHS). The phase register of the master is not used, so the phase register of the master should be initialized to 0.

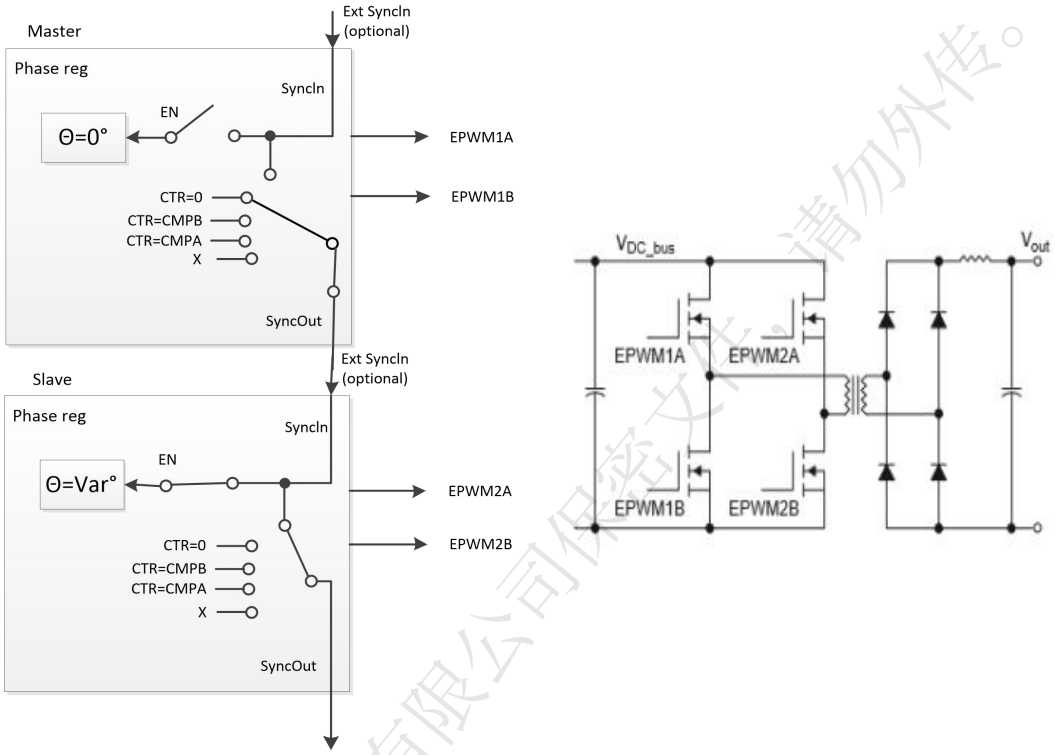

Figure 16-15 Control Schematic 7

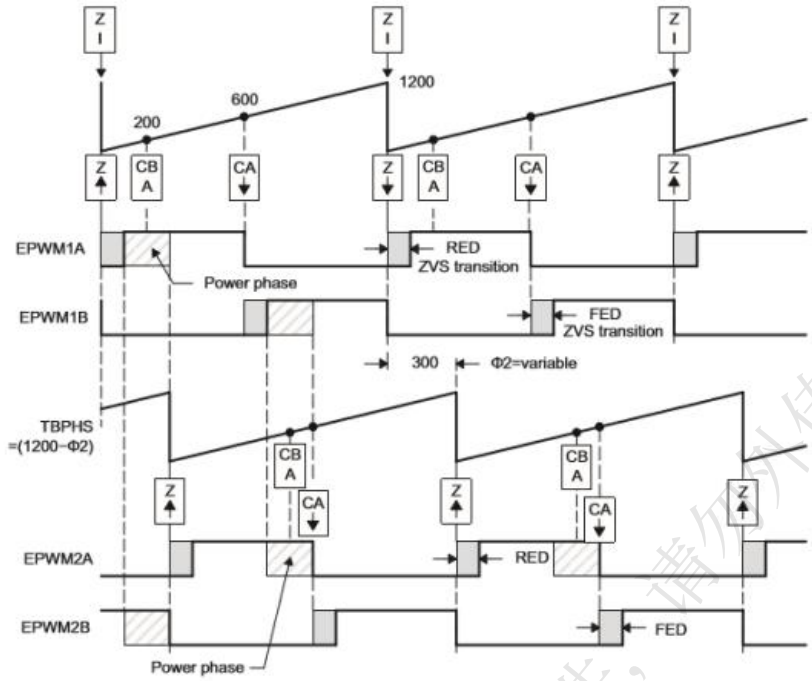

Figure 16-16 Timing Diagram 7

16.3.8. Control The Peak Current Mode (Peak Current Mode) Controls the Buck module

The peak current mode control technology provides the advantages of automatic overcurrent limiting, rapid correction of input voltage changes, and reduction of magnetic saturation. The figure below shows the application of EPWM1A and the on-chip comparator to the Buck converter topology. The output current can be sensed and diverted to the positive electrode of the on-chip comparator through a current sensing resistor. The internally programmable 8-bit DAC provides a reference peak current to the positive electrode of the comparator. Alternatively, an external reference current is also connected to this input. The output of the comparator is the input to the submodule of the digital comparator. The EPWM module is configured so that the output of the EPWM is triggered immediately when the induced current reaches the peak reference current. A cycle cycle mechanism is adopted. The waveform generated under this configuration is also shown.

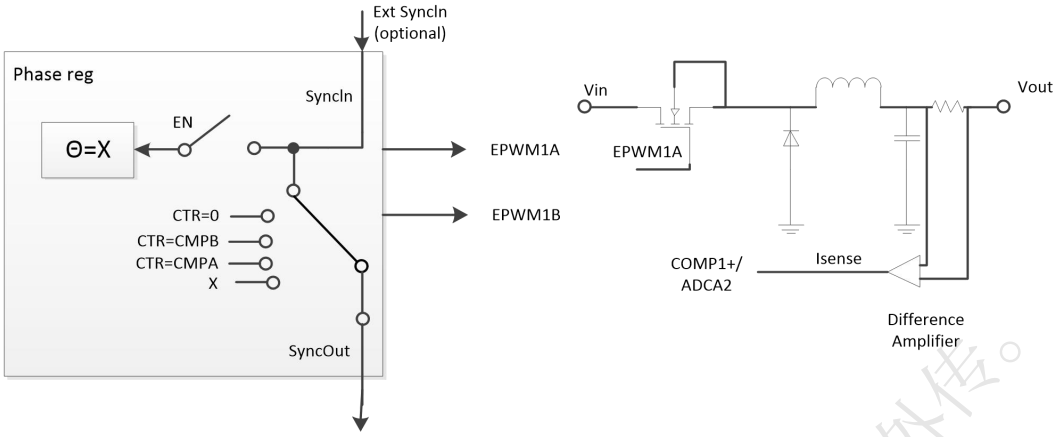

Figure 16-17 Control Diagram 8

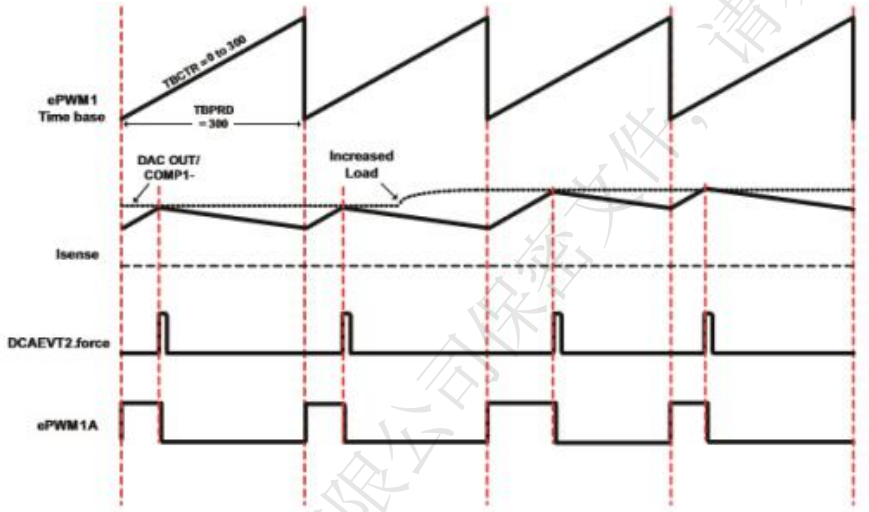

Figure 16-18 Timing Diagram 8

16.3.9. Control the H-bridge LLC resonant converter

Topologies of various resonant converters have been well known in the field of power electronics over the years. In addition, the H-Bridge LLC resonant converter topology has recently gained popularity in consumer electronics applications requiring high efficiency and high power density. For example, the EPWM1 has a very detailed single-channel configuration, but its configuration can be simply expanded to multi-channel. The control parameter is the frequency and not the duty cycle (the duty cycle should be kept constant or at about 50%). Users need to update the dead zone time in real time to improve efficiency by adjusting enough softswitch delay.

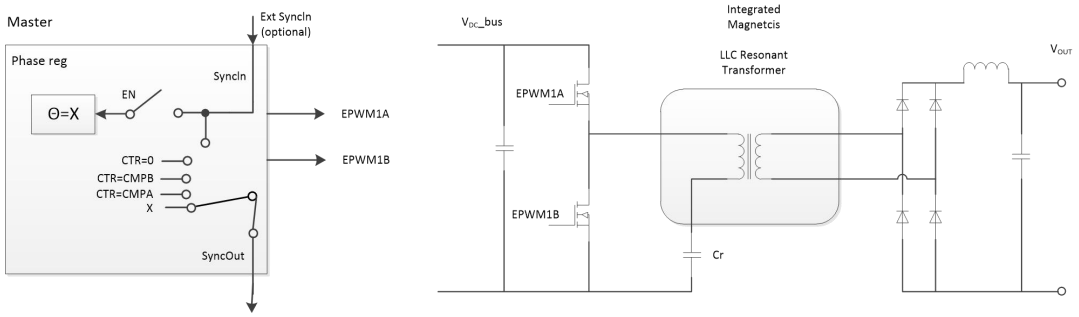

Figure 16-19 Control Diagram 9

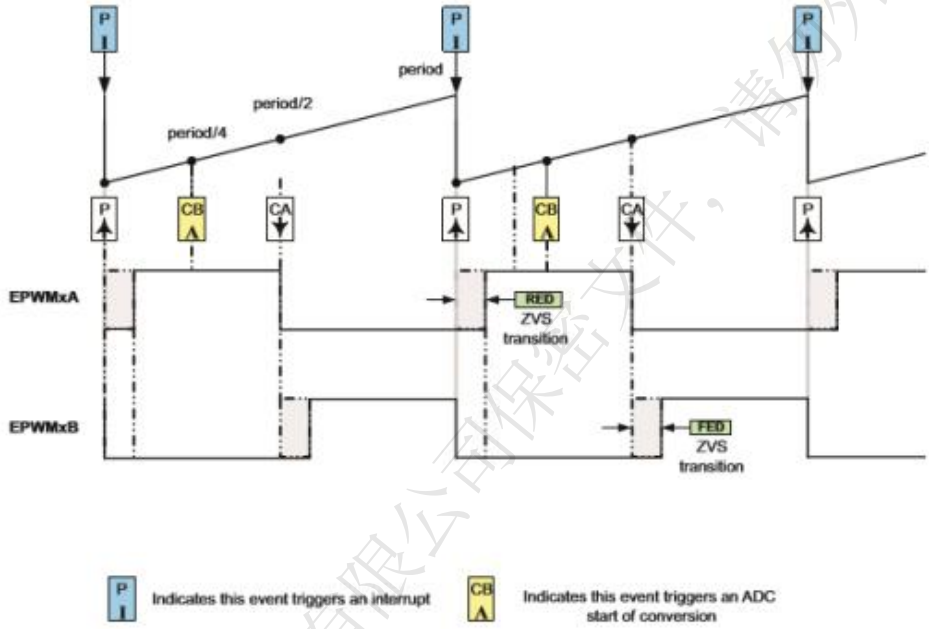

Figure 16-20 Control Diagram 9

16.4. Registers

16.4.1. Register base address

Name	Base Address	Description
EPWM	0x40001800	EPWM base address

16.4.2. Register list

Offset Address	Name	Description

0x0000	EPWMx_TBCTL	TB control register
0x0004	EPWMx_TBPRD	TB cycle register
0x0008	EPWMx_TBPHASE	TB Phase register
0x000c	EPWMx_CMPCTL	CMP control register
0x0010	EPWMx_CMPA	CMPA register
0x0014	EPWMx_CMPB	CMPB register
0x0018	EPWMx_CMPC	CMPC register
0x001c	EPWMx_CMPD	CMPD register
0x0020	EPWMx_AQCTLAB	AQ control register
0x0024	EPWMx_AQSFRC	AQ behavior register 0
0x0028	EPWMx_AQCSFRC	AQ behavior register 1
0x002c	EPWMx_DBCTL	DB control register
0x0030	EPWMx_DEDELAY	DB delay register
0x0034	EPWMx_ETCTL	ET control register
0x0038	EPWMx_ETCTL2	ET control register 2
0x003C	EPWMx_ETFLAG	ET status register
0x0040	EPWMx_DCCTL	DC control register
0x0044	EPWMx_DCTRIPSEL	DC trigger register
0x0048	EPWMx_BLANKOFFSET	DC window register
0x004C	EPWMx_WINDIDTH	DC window length register
0x0050	EPWMx_TZCTL	TZ control register
0x0054	EPWMx_TZFLAG	TZ status register
0x0058	EPWMx_DCCAP	DC capture register
0x005C	EPWM_TTCTL	Total control register

16.4.3. Register details

16.4.3.1. EPWMx_TBCTL

Bit(s)	Name	Description	R/W	Reset
31:16	TBCTR	**Read this register to know what the counter**	R	0x0

		is counting at this time		
15:13	CLK_DIV	**Module clock division** 0x0: epwm clk 0x1: epwm clk div 2 0x2: epwm clk div 4 0x3: epwm clk div 8 0x4: epwm clk div 16	RW	0x0
12:10	Reserved	-	-	-
9	TBCTR_DIR	**TB Indicates the direction status bit** 0x0: TB is in decline 0x1: TB is increasing	R	0x0
8	SYNCI	**Enter the sync latch status bit** **Read:** 0x0: No external synchronization event occurred. 0x1: An external synchronization event occurred (EPWMxSYNCI) **Write:** 0x0: Invalid: 0x1: Writing a 1 to this bit will clear the latch event.	RW	0x0
7:5	SYNCO_SEL	**Sync signal output selection** 0x0: sync in 0x1: tbcnt equal zero 0x2: tbcnt equal cmpa 0x3: tbcnt equal cmpb 0x4: tbcnt equal cmpc 0x5: tbcnt equal cmpd 0x6 to 0x7: invalid	RW	0x0
4	SWF_SYNC	**Software sync enabled** 0x0: Off 0x1: Open	W	0x0
3	PHSEN	**Phase synchronization enabled** 0x0: Off 0x1: On	RW	0x0
2:1	CTRMODE	**Count Mode selection** 0x0: Up mode 0x1: Down mode 0x2: Up Dowm mode 0x3: constant	RW	0x0
0	PRDLD	**Shadow register Enable for cycle register** 0x0: The period register (TBPRD) is loaded from the shadow register when the counter TBCTR equals zero 0x1: Load the TBPRD register immediately without using the shadow register	RW	0x0

16.4.3.2. EPWMx_TBPRD

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15:0	TBPRD	**Cycle configuration of the TB counter** Shading of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default, this register is hidden. ● If TBCTL[PRDLD] = 0, the shadow register is enabled, and any writes or reads will automatically go to the shadow register. In this case, when the TB counter equals zero, the register will be loaded from the shadow register; ● If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the count register, the register that actively controls the hardware;	RW	0x0

16.4.3.3. EPWMx_TBPHASE

Bit(s)	Name	Description	R/W	Reset
31:17	reserved	-	-	-
16	TBDIR	**Set the counter to count in the direction** Set the time base counter orientation of the selected ePWM relative to the time base that provides the synchronized input signal ● If TBCTL[PHSEN] = 0, the synchronization event is ignored and no time base direction for that direction is loaded; ● If TBCTL[PHSEN] = 1, then the time base direction will be loaded direction (TBDIR) when the synchronization event occurs; Synchronization events can be initiated by an input synchronization signal (EPWMxSYNCI) or forced to synchronize by	RW	0x0

		software. Note: This bit is only used for up and down mode		
15:0	TBPHS	**The phase value for synchronizing when the sync signal is triggered** These bits set the time base counter phase of the selected ePWM with respect to the time base that provides the sync input signal. ● If TBCTL[PHSEN] = 0, the synchronization event is ignored and the timing counter is not loaded; ● If TBCTL[PHSEN] = 1, then the timebase counter (TBCTR) will load the phase (TBPHS) when the synchronization event occurs; A synchronization event can be initiated by an input synchronization signal (EPWMxSYNCI) or it can be forcibly synchronized by software.	RW	0x0

16.4.3.4. EPWMx_CMPCTL

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15	SHDWDFULL	**CMPD Shadow register status** 0x0: The shadow register is not full 0x1: The shadow register is full and will be overwritten if the CPU writes again	RO	0x0
14	SHDWDMODE	**CMPD Load mode selection** 0x0: Shadow mode, as double buffering. All writes through the CPU access the shadow register 0x1: In immediate mode, only active compare registers are used. All write and read operations access the active register directly for immediate comparison operations	RW	0x0
13:12	LOADDMODE	**CMPD Load data moment point selection** This feature only works in shadow mode 0x0: Load comparison value data when CTR=0 0x1: Load comparison data when CTR=PRD 0x2: Load comparison value data for both CTR=0 and CTR=PRD 0x3: Invalid	RW	0x0

11	SHDWCFULL	**CMPC Shadow register status** 0x0: The shadow register is not full 0x1: The shadow register is full and will be overwritten if the CPU writes again	RO	0x0
10	SHDWCMODE	**CMPC loading mode selection** 0x0: Shadow mode, as double buffering. All writes through the CPU access the shadow register 0x1: In immediate mode, only active compare registers are used. All write and read operations access the active register directly for immediate comparison operations	RW	0x0
9:8	LOADCMODE	**CMPC Load data moment point selection** This feature only works in shadow mode 0x0: Load comparison value data when CTR=0 0x1: Load comparison data when CTR=PRD 0x2: Load comparison value data for both CTR=0 and CTR=PRD 0x3: Invalid	RW	0x0
7	SHDWBFULL	**CMPB Shadow register status** 0x0: The shadow register is not full 0x1: The shadow register is full and will be overwritten if the CPU writes again	RO	0x0
6	SHDWAFULL	**CMPA Shadow register status** 0x0: The shadow register is not full 0x1: The shadow register is full and will be overwritten if the CPU writes again	RO	0x0
5	SHDWBMODE	**CMPB Load mode selection** 0x0: Shadow mode, as double buffer. All writes through the CPU access the shadow register 0x1: In immediate mode, only active compare registers are used. All write and read operations access the active register directly for immediate comparison operations	RW	0x0
4	SHDWAMODE	**CMPA load mode selection** 0x0: Shadow mode, as double buffer. All writes through the CPU access the shadow register 0x1: In immediate mode, only active compare registers are used. All write and read operations access the active register directly for immediate comparison operations	RW	0x0

3:2	LOADBMODE	**CMPB Load data moment point selection** This feature only works in shadow mode 0x0: Load comparison value data when CTR=0 0x1: Load comparison data when CTR=PRD 0x2: Load comparison value data for both CTR=0 and CTR=PRD 0x3: Invalid	RW	0x0
1:0	LOADAMODE	**CMPA load data moment point selection** This feature only works in shadow mode 0x0: Load comparison value data when CTR=0 0x1: Load comparison data when CTR=PRD 0x2: Load comparison value data for both CTR=0 and CTR=PRD 0x3: Invalid	RW	0x0

16.4.3.5. EPWMx_CMPA

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	–	–	–
15:0	CMPA	**CMPA register value** By default, writes to this register are hidden. Enable and disable shading via the CMPCTL[SHDWAMODE] bit. ● If CMPCTL[SHDWAMODE]=0, shadow is enabled and any writes will automatically go to the shadow register. The CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. ● Before writing, the CMPCTL[shdwfulll] bit can be read to determine if the shadow register is currently full. ● If CMPCTL[SHDWAMODE]=1, then the shadow register is disabled and any writes will go directly to the active register, that is, the register that actively controls the hardware.	RW	0x0

16.4.3.6. EPWMx_CMPB

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	–	–	–
15:0	CMPB	**CMPB register value**	RW	0x0

		By default, writes to this register are hidden. Enable and disable shading via the CMPCTL[SHDWAMODE] bit. ● If CMPCTL[SHDWAMODE]=0, shadow is enabled and any writes will automatically go to the shadow register. The CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. ● Before writing, the CMPCTL[shdwfulll] bit can be read to determine if the shadow register is currently full. ● If CMPCTL[SHDWAMODE]=1, then the shadow register is disabled and any writes will go directly to the active register, the register that actively controls the hardware.		

16.4.3.7. EPWMx_CMPC

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	–	–	–
15:0	CMPC	**CMPC register value** By default, writes to this register are hidden. Enable and disable shading via the CMPCTL[SHDWAMODE] bit. ● If CMPCTL[SHDWAMODE]=0, shadow is enabled and any writes will automatically go to the shadow register. The CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. ● Before writing, the CMPCTL[shdwfulll] bit can be read to determine if the shadow register is currently full. ● If CMPCTL[SHDWAMODE]=1, then the shadow register is disabled and any writes will go directly to the active register, the register that actively controls the hardware.	RW	0x0

16.4.3.8. EPWMx_CMPD

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15:0	CMPD	**CMPD register value** By default, writes to this register are hidden. Enable and disable shading via the CMPCTL[SHDWAMODE] bit. ● If CMPCTL[SHDWAMODE]=0, shadow is enabled and any writes will automatically go to the shadow register. The CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. ● Before writing, the CMPCTL[shdwfulll] bit can be read to determine if the shadow register is currently full. ● If CMPCTL[SHDWAMODE]=1, then the shadow register is disabled and any writes will go directly to the active register, the register that actively controls the hardware.	RW	0x0

16.4.3.9. EPWMx_AQCTLAB

Bit(s)	Name	Description	R/W	Reset
31:28	reserved	-	-	-
27:26	CBDB	**Counter CNT is under decreasing trend, behavior control of EPWMxB when CNT=CMPB** 0x0: remains original 0x1: EPWMxB output low 0x2: EPWMxB outputs high 0x3: Flip	RW	0x0
25:24	CBUB	**Counter CNT is under increasing trend, EPWMxB behavior control when CNT=CMPB** 0x0: remains original 0x1: EPWMxB output low 0x2: EPWMxB outputs high 0x3: Flip	RW	0x0
23:22	CADB	**Counter CNT is under decreasing trend, EPWMxB behavior control when CNT=CMPA** 0x0: remains original 0x1: EPWMxB output low level 0x2: EPWMxB outputs high	RW	0x0

		0x3: Flip		
21:20	CAUB	**Counter CNT is under increasing trend, EPWMxB behavior control when CNT=CMPA** 0x0: remains original 0x1: EPWMxB output low 0x2: EPWMxB outputs high 0x3: Flip	RW	0x0
19:18	PRDB	**Counter CNT in up and down mode, EPWMxB behavior control when CNT=PRD** 0x0: Stays as it was 0x1: EPWMxB output low 0x2: EPWMxB outputs high 0x3: Flip	RW	0x0
17:16	ZROB	**Counter CNT is in up and down mode, behavior control of EPWMxB when CNT=0** 0x0: Stays as it was 0x1: EPWMxB output low 0x2: EPWMxB outputs high 0x3: Flip	RW	0x0
15:12	reserved	-	-	-
11:10	CBDA	**Counter CNT under decreasing trend, behavior control of EPWMxA when CNT=CMPB** 0x0: Remains original 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0
9:8	CBUA	**The behavior control of EPWMxA when CNT=CMPB is in an increasing trend** 0x0: remains original 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0
7:6	CADA	**Counter CNT under decreasing trend, behavior control of EPWMxA when CNT=CMPA** 0x0: Remains original 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0
5:4	CAUA	**Counter CNT is under increasing trend, behavior control of EPWMxA when CNT=CMPA** 0x0: remains original 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0
3:2	PRDA	**Counter CNT is in up and down mode, behavior control of EPWMxA when CNT=PRD**	RW	0x0

		0x0: Remains as it was 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip		
1:0	ZROA	**Counter CNT is in up and down mode, behavior control of EPWMxA when CNT=0** 0x0: Stays as it was 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0

16.4.3.10. EPWMX_AQSFRC

Bit(s)	Name	Description	R/W	Reset
31:8	reserved	-	-	-
7:6	RLDCSF	**AQCSFRC Shadow register load status** 0x0: CNT=0 0x1: CNT=PRD 0x2: CNT=0 or CNT=PRD 0x3: Load immediately	R	0x0
5	OTSFB	**Software triggers One Time to make EPWMxB respond to behavior** 0x0: Invalid 0x1: Immediately triggered a one-off, EPWMxB to make the corresponding behavior	RW	0x0
4:3	ACTSFB	**EWPMxB behavior selection after triggering a one-off signal** 0x0: Stay as is 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0
2	OTSFA	**Software triggers One Time to make EPWMxA respond to behavior** 0x0: Invalid 0x1: One Time is triggered immediately, and EPWMxA makes corresponding behavior	RW	0x0
1:0	ACTSFA	**EWPMxA behavior selection after triggering the One-Time signal** 0x0: Stay as is 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Flip	RW	0x0

16.4.3.11. EPWMX_AQCSFRC

Bit(s)	Name	Description	R/W	Reset
31:4	reserved	-	-	-
3:2	CSFB	**Software Loop Trigger Configuration (EPWMxB)** In immediate mode, continuous triggering takes effect on the next TBCLK edge; In shadow mode, the next TBCLK edge does not take effect until after the shadow register is loaded. 0x0: Remains the original 0x1: EPWMxA output low level 0x2: EPWMxA outputs high 0x3: Stay on	RW	0x0
1:0	CSFA	**Software Loop Trigger Configuration (EPWMxA)** In immediate mode, continuous firing takes effect on the next TBCLK edge; In shadow mode, the next TBCLK edge does not take effect until after the shadow register is loaded. 0x0: Remains the original 0x1: EPWMxA output low 0x2: EPWMxA outputs high 0x3: Stay on	RW	0x0

16.4.3.12. EPWMX_DBCTL

Bit(s)	Name	Description	R/W	Reset
31:19	reserved	-	-	-
18	SHDWDBCTLMODE	**DBCTL load mode** 0x0: Immediate mode 0x1: Shadow mode	RW	0x0
17:16	LOADDBCTLMODE	**DBCTL load point selection** 0x0: CNT = 0 0x1: CNT = PRD 0x2: CNT= 0, or CNT= PRD 0x3: Invalid Note: Only valid in Shadow mode	RW	0x0
15	HALFCYCLE	**Dead zone clock split selection** 0x0: Regardless of frequency 0x1:1/2 EPWM clock	RW	0x0
14	reserved	-	-	-
13:12	OUTSWAP	**Dead zone output swap Settings**	R/W	0x0

		In Figure 16-1, BIT13 corresponds to S7 and BIT12 corresponds to this S6 0x0: OutA = A-path, OutB = B-path 0x1: OutA = A-path, OutB = A-path 0x2: OutA = B-path, OutB = B-path 0x3: OutA = B-path, OutB = A-path		
11	SHDWDBFEDMODE	**FED dead zone loading mode** 0x0: Immediate mode 0x1: Shadow mode	R/W	0x0
10	SHDWDBREDMODE	**RED dead zone loading mode** 0x0: Immediate mode 0x1: Shadow mode	R/W	0x0
9-8 -	LOADFEDMODE	**DBFED Load moment selection** 0x0: CNT = 0 0x1: CNT = PRD 0x2: CNT= 0, or CNT= PRD 0x3: Invalid Note: Only valid in Shadow mode	R/W	0x0
7-6	LOADREDMODE	**DBRED Load time selection** 0x0: CNT = 0 0x1: CNT = PRD 0x2: CNT= 0, or CNT= PRD 0x3: Invalid Note: Only valid in Shadow mode	R/W	0x0
5:4	IN_MODE	**Dead zone input control** The behavior of the dead zone input comes to the AQ module 0x0: The input source for both the rising and falling edge of the dead zone selects EPWMxA 0x1: EPWMxB is the input source for the rising edge of the dead zone, and EPWMxA is the input source for the falling edge of the dead zone 0x2: Select EPWMxA for the rising edge of dead zone and EPWMxB for the falling edge of dead zone 0x3: Select EPWMxA for both rising and falling dead edge input sources	RW	0x0
3:2	POLSEL	**Dead zone output polarity selection** 0x0: EPWMxA and EPWMxB maintain their original polarity outputs 0x1: EPWMxA output polarity reverses, EPWMxB maintains the original polarity output 0x2: EPWMxA maintains the original polarity output, and EPWMxB maintains the polarity	RW	0x0

		output 0x3: Both EPWMxA and EPWMxB output polarity reverses		
1:0	OUT_MODE	**Dead zone output mode selection** 0x0: Deadzone function is turned off 0x1: Turns off rising edge dead zone and turns on falling edge dead zone 0x2: Open rising edge dead zone, close falling edge dead zone 0x3: Open dead zones for both rising and falling edges	RW	0x0

16.4.3.13. EPWMX_DBDELAY

Bit(s)	Name	Description	R/W	Reset
31:30	reserved	--	-	-
29:16	FED	**Dead zone time configuration for the falling edge** 14bit wide	RW	0x0
15:14	reserved	-	-	-
13:0	RED	**Dead zone time configuration for the top falling edge** 14bit wide	RW	0x0

16.4.3.14. EPWMx_ETCTL

Bit(s)	Name	Description	R/W	Reset
31:24	reserved	-	-	-
23:20	SOCBCNT	**Number of SOCB triggers** 0x0: No event was triggered 0x1: Has been triggered once 0x2: Has been triggered twice ... 0xF: Has been triggered 15 times	RO	0x0
19:16	SOCBPRD	**Frequency selection of SOCB triggers** 0x0: Off 0x1: ETCTL[SOCBCNT] = 0x1 triggers an interrupt signal 0x2: ETCTL[SOCBCNT] = 0x2 triggers an interrupt signal ... 0xF: ETCTL[SOCBCNT] = 0xF triggers an interrupt signal	RW	0x0
15:12	SOCACNT	**The number of times the SOCA is triggered**	RO	0x0

		0x0: No event was triggered 0x1: Has been triggered once 0x2: Has been triggered twice ... 0xF: Has been triggered 15 times		
11:8	SOCAPRD	**SOCA trigger frequency selection** 0x0: Off 0x1: ETCTL[SOCACNT] = 0x1 triggers an interrupt signal 0x2: ETCTL[SOCACNT] = 0x2 triggers an interrupt signal ... 0xF: ETCTL[SOCACNT] = 0xF triggers an interrupt signal	RW	0x0
7:4	INTCNT	**The number of times the interrupt was triggered** 0x0: No event was triggered 0x1: Has been triggered once 0x2: Has been triggered twice ... 0xF: Has been triggered 15 times	RO	0x0
3:0	INTPRD	**Frequency selection of interrupt triggers** 0x0: Off 0x1: ETCTL[INTCNT] = 0x1 triggers interrupt signal 0x2: ETCTL[INTCNT] = 0x2 triggers interrupt signal ... 0xF: ETCTL[INTCNT] = 0xF triggers interrupt signal	RW	0x0

16.4.3.15. EPWMx_ETCTL2

Bit(s)	Name	Description	R/W	Reset
31:26	Reserved	-	-	-
25:16	MULTSCSEL	Multipoint trigger signal selection Each 1 bit represents the enable bit of a signal, which can be enabled in multiple channels BIT25: CNT = 0 BIT24: CNT = PRD BIT23: CNT is in increasing mode and CNT = CMPA BIT22: CNT is in decreasing mode and CNT = CMPA	RW	0x0

		BIT21: CNT is in increasing mode and CNT = CMPB BIT20: CNT is in decreasing mode and CNT = CMPB BIT19: CNT is in increasing mode and CNT = CMPC BIT18: CNT is in decreasing mode and CNT = CMPC BIT17: CNT is in increasing mode and CNT = CMPD BIT16: CNT is in decreasing mode and CNT = CMPD		
15	Reserved	-	-	-
14	SOCBEN	**SOCB triggers Enable** 0x0: Off 0x1: On	RW	0x0
13:10	SOCBSEL	**SOCB trigger source selection** 0x0: Invalid 0x1: CNT = 0 0x2: CNT = PRD 0x3: In up-down mode, CNT = 0 or CNT = PRD 0x4: The CNT is increasing and CNT = CMPA 0x5: In a decreasing trend in CNT and CNT = CMPA 0x6: The CNT is in increasing trend and CNT = CMPB 0x7: In decreasing trend at CNT and CNT = CMPB 0x8: The CNT is increasing and CNT = CMPC 0x9: In a decreasing trend at CNT and CNT = CMPC 0xA: The CNT is in an increasing trend and CNT = CMPD 0xB: The CNT is in decreasing trend and CNT = CMPD 0xC: Multi-point trigger signal ETCTL2[MULTSCSEL] Other: invalid	RW	0x0
9	SOCAEN	**SOCA triggers Enable** 0x0: Off 0x1: On	RW	0x0
8:5	SOCASEL	**SOCA trigger source selection** 0x0: Invalid 0x1: CNT = 0 0x2: CNT = PRD 0x3: In up-down mode, CNT = 0 or CNT = PRD	RW	0x0

		0x4: The CNT is increasing and CNT = CMPA 0x5: In a decreasing trend in CNT and CNT = CMPA 0x6: The CNT is in increasing trend and CNT = CMPB 0x7: In decreasing trend at CNT and CNT = CMPB 0x8: The CNT is increasing and CNT = CMPC 0x9: In a decreasing trend at CNT and CNT = CMPC 0xA: The CNT is in an increasing trend and CNT = CMPD 0xB: The CNT is in decreasing trend and CNT = CMPD 0xC: Multi-point trigger signal ETCTL2[MULTSCSEL] Other: invalid		
4	INTEN	**The interrupt trigger function is enabled** 0x0: Off 0x1: On	RW	0x0
3:0	INTSEL	**Interrupt trigger source selection** 0x0: Invalid 0x1: CNT = 0 0x2: CNT = PRD 0x3: In up-down mode, CNT = 0 or CNT = PRD 0x4: The CNT is increasing and CNT = CMPA 0x5: In a decreasing trend in CNT and CNT = CMPA 0x6: The CNT is in increasing trend and CNT = CMPB 0x7: In decreasing trend at CNT and CNT = CMPB 0x8: The CNT is increasing and CNT = CMPC 0x9: In a decreasing trend at CNT and CNT = CMPC 0xA: The CNT is in an increasing trend and CNT = CMPD 0xB: The CNT is in decreasing trend and CNT = CMPD 0xC: Multi-point trigger signal ETCTL2[MULTSCSEL] Other: invalid	RW	0x0

16.4.3.16. EPWMx_ETFLAG

Bit(s)	Name	Description	R/W	Reset
31:12	reserved	-	-	-
11	FRCSOCB	**The software forces SOCB to trigger enable** 0x0: Invalid 0x1: Generates SOCB signal	RW	0x0
10	FRCSOCA	**Software forces SOCA to trigger enable** 0x0: Invalid 0x1: Generates SOCA signal	RW	0x0
9	Reserved	-	-	-
8	FRCINT	**Software force interrupt trigger Enable** 0x0: Invalid 0x1: Generates interrupt	RW	0x0
7	CLRSOCB	**SOCB triggers the clear flag** Write 1 clear	RW	0x0
6	CLRSOCA	**SOCA triggers the clear flag** Write 1 Clear	RW	0x0
5	Reserved	-	-	-
4	CLRINT	**The interrupt triggers the clear flag** Write 1 Clear	RW	0x0
3	SOCB	**SOCB trigger flag bit** 0x0: Not triggered 0x1: Triggered	R	0x0
2	SOCA	**SOCA trigger flag bit** 0x0: Not triggered 0x1: Triggered	RW	0x0
1	reserved		-	-
0	INT	**Interrupt trigger flag bit** 0x0: Not triggered 0x1: Triggered	RW	0x0

16.4.3.17. EPWMx_DCCTL

Bit(s)	Name	Description	R/W	Reset
31:20	reserved	-	-	-
19	AEVT2FRCSYNCSEL	**DCAEVT2 forces sync signal selection** 0x0: Sync signal 0x1: Asynchronous signal	RW	0x0
18	AEVT2SRCSEL	**DCAEVT2 signal source selection** 0x0: DCAEVT2 Signal 0x1: DCEVTFILT Signal	RW	0x0
17	AEVT1SYNCE	**DCAEVT1 synchronization signal is enabled** 0x0: Off 0x1: On	RW	0x0

16	AEVT1SOCE	**DCAEVT1 SOC Enabled** 0x0: Closed 0x1: On	RW	0x0
15	AEVT1FRCSYNCSEL	**DCAEVT1 forces sync signal selection** 0x0: Sync signal 0x1: Asynchronous signal	RW	0x0
14	AEVT1SRCSEL	**DCAEVT1 signal source selection** 0x0: DCAEVT1 Signal 0x1: DCEVTFILT Signal	RW	0x0
13	BEVT2FRCSYNCSEL	**DCBEVT2 Forces sync signal selection** 0x0: Sync signal 0x1: Asynchronous signal	RW	0x0
12	BEVT2SRCSEL	**DCBEVT2 Signal Source Selection** 0x0: DCBEVT2 Signal 0x1: DCEVTFILT Signal	RW	0x0
11	BEVT1SYNCE	**DCBEVT1 Synchronization signal enabled** 0x0: Off 0x1: On	RW	0x0
10	BEVT1SOCE	**DCBEVT1 SOC Enabled** 0x0: Disable 0x1: On	RW	0x0
9	BEVT1FRCSYNCSEL	**DCBEVT1 forces synchronization signal selection** 0x0: Sync signal 0x1: Asynchronous signal	RW	0x0
8	BEVT1SRCSEL	**DCBEVT1 Signal Source Selection** 0x0: DCBEVT1 Signal 0x1: DCEVTFILT Signal	RW	0x0
7:6	PULSESEL	**Blanking pulse select and capture the starting alignment point** 0x0: CNT = PRD 0x1: CNT = 0 Other: invalid	RO	0x0
5	BLANKINV	**The Blanking window is enabled backwards** 0x0: Off 0x1: On	RW	0x0
4	BLANKE	**The Blanking window is enabled** 0x0: Closed 0x1: On	RW	0x0
3:2	FILTSRCSEL	**Filter signal source selection** 0x0: Select DCAEVT1 Signal 0x1: Selects DCAEVT2 Signal 0x2: Select DCBEVT1 Signal 0x3: Select DCBEVT2 Signal	RW	0x0
1	SHDWMODE	**Capture Shadow function** 0x0: Shadow mode	RW	0x0

		0x1: Immediate mode		
0	CAPE	**Capture function enabled** 0x0: Disabled 0x1: On	RW	0x0

16.4.3.18. EPWMx_DCTRIPSEL

Bit(s)	Name	Description	R/W	Reset
31:28	reserved	-	-	-
27:25	DCBEVT2	**DCB EVT2 input trigger condition selection** 0x0: Off 0x1: DCBH = low, DCBL = don't care 0x2: DCBH = high, DCBL = don't care 0x3: DCBL = low, DCBH = don't care 0x4: DCBL = high, DCBH = don't care 0x5: DCBL = high, DCBH = low 0x6: DCBL = high, DCBH = high 0x7: DCBL = low, DCBH = low	RW	0x0
24:22	DCBEVT1	**DCB EVT1 Input trigger condition selection** 0x0: Off 0x1: DCBH = low, DCBL = don't care 0x2: DCBH = high, DCBL = don't care 0x3: DCBL = low, DCBH = don't care 0x4: DCBL = high, DCBH = don't care 0x5: DCBL = high, DCBH = low 0x6: DCBL = high, DCBH = high 0x7: DCBL = low, DCBH = low	RW	0x0
21:19	DCAEVT2	**DCA EVT2 input trigger condition selection** 0x0: Closed 0x1: DCAH = low, DCAL = don't care 0x2: DCAH = high, DCAL = don't care 0x3: DCAL = low, DCAH = don't care 0x4: DCAL = high, DCAH = don't care 0x5: DCAL = high, DCAH = low 0x6: DCAL = high, DCAH = high 0x7: DCAL = low, DCAH = low	RW	0x0
18:16	DCAEVT1	**DCA EVT1 Input trigger condition selection** 0x0: Off 0x1: DCAH = low, DCAL = don't care 0x2: DCAH = high, DCAL = don't care 0x3: DCAL = low, DCAH = don't care 0x4: DCAL = high, DCAH = don't care 0x5: DCAL = high, DCAH = low 0x6: DCAL = high, DCAH = high 0x7: DCAL = low, DCAH = low	RW	0x0

15:12	DCBLCOMPSEL	**DCBL input Source selection** DCBL (Digital Compare B Low Input) is the L end of the DC input source 0x0: TZ1 input 0x1: TZ2 input 0x2: TZ3 input 0x3: TZ4 input 0x8: COMP1OUT input 0x9: COMP2OUT input 0xA: COMP3OUT input 0xB: COMP4OUT input 0xC: COMP5OUT input	RW	0x0
11:8	DCBHCOMPSEL	**DCBH input source selection** DCBH (Digital Compare B High Input) is the H end of the DC input source 0x0: TZ1 input 0x1: TZ2 input 0x2: TZ3 input 0x3: TZ4 input 0x8: COMP1OUT input 0x9: COMP2OUT input 0xA: COMP3OUT input 0xB: COMP4OUT input 0xC: COMP5OUT input	RW	0x0
7:4	DCALCOMPSEL	**DCAL input Source selection** DCAL (Digital Compare A Low Input) is the L end of the DC input source 0x0: TZ1 input 0x1: TZ2 input 0x2: TZ3 input 0x3: TZ4 input 0x8: COMP1OUT input 0x9: COMP2OUT input 0xA: COMP3OUT input 0xB: COMP4OUT input 0xC: COMP5OUT input	RW	0x0
3:0	DCAHCOMPSEL	**DCAH input source selection** DCAH (Digital Compare A High Input) is the H end of the DC input source 0x0: TZ1 input 0x1: TZ2 input 0x2: TZ3 input 0x3: TZ4 input 0x8: COMP1OUT input 0x9: COMP2OUT input 0xA: COMP3OUT input	RW	0x0

		0xB: COMP4OUT input 0xC: COMP5OUT input		

16.4.3.19. EPWMx_BLANKOFFSET

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15:0	OFFSET	**Blanking Window offset** Set the offset time of the Blanking window from start to active. The start point is determined by DCTRL[PULSESEL].	RW	0x0

16.4.3.20. EPWMx_WINWIDTH

Bit(s)	Name	Description	R/W	Reset
31:12	reserved	-	-	-
11:0	WINDOW	**Blanking the length of the window** 0x0: The Blanking window does not work Other: Blanking takes effect in TBCLK time	RW	0x0

16.4.3.21. EPWMx_TZCTL

Bit(s)	Name	Description	R/W	Reset
31	DCBEVT1_OSTEN	**DCB EVT1 OSHT (one-shot-trip) is enabled** 0x0: Disables 0x1: Open	0x0	RW
30	DCAEVT1_OSTEN	**DCA EVT1 OSHT (one-shot-trip) Is enabled** 0x0: Disables 0x1: On	0x0	RW
29	OSHT6	**Trip-zone 6 OSHT (one-shot-trip) is enabled** 0x0: Disables oshT 0x1: On	0x0	RW
28	OSHT5	**Trip-zone 5 OSHT (one-shot-trip) is enabled** 0x0: Disables oshT 0x1: On	0x0	RW
27	OSHT4	**Trip-zone 4 Enable OSHT (one-shot trip)** 0x0: Disables oshT 0x1: On	0x0	RW
26	OSHT3	**Trip-zone 3 Enable OSHT (one-shot trip)** 0x0: Disables oshT 0x1: On	0x0	RW
25	OSHT2	**Trip-zone 2 Enable OSHT (one-shot trip)** 0x0: Disables oshT	0x0	RW

		0x1: On		
24	OSHT1	**Trip-zone 1 Enable OSHT (one-shot-trip)** 0x0: Closed 0x1: On	0x0	RW
23	DCBEVT2_CBCEN	**DCB EVT2 CBC (cycle by cycle) enabled** 0x0: Disabled 0x1: On	0x0	RW
22	DCAEVT2_CBCEN	**DCA EVT2 CBC (cycle by cycle) is enabled** 0x0: Disabled 0x1: On	0x0	RW
21	CBC6	**Trip-zone 6 Enable CBC (cycle by cycle)** 0x0: Off 0x1: On	0x0	RW
20	CBC5	**Trip-zone 5 Enable CBC (cycle by cycle)** 0x0: Off 0x1: On	0x0	RW
19	CBC4	**Trip-zone 4 Enable CBC (cycle by cycle)** 0x0: Off 0x1: On	0x0	RW
18	CBC3	**Trip-zone 3 Enable CBC (cycle by cycle)** 0x0: Off 0x1: On	0x0	RW
17	CBC2	**Trip-zone 2 cycle by cycle (CBC) is enabled** 0x0: Off 0x1: On	0x0	RW
16	CBC1	**Trip-zone 1 Enable CBC (cycle by cycle)** 0x0: Off 0x1: On	0x0	RW
15:12	reserved	-	-	-
11:10	DCBEVT2_FRCEN	**DCB EVT2 software triggers behavior** 0x0: EPWMxB outputs high resistance state 0x1: EPWMxB outputs high 0x2: EPWMxB outputs low 0x3: EPWMxB remains in its original state	0x0	RW
9:8	DCAEVT2_FRCEN	**DCA EVT2 software triggers behavior** 0x0: EPWMxA output high resistance state 0x1: EPWMxA outputs high 0x2: EPWMxA outputs low level 0x3: EPWMxA remains in its original state	0x0	RW
7:6	DCBEVT1_FRCEN	**DCB EVT1 software triggers behavior** 0x0: EPWMxB outputs high resistance 0x1: EPWMxB outputs high 0x2: EPWMxB outputs low 0x3: EPWMxB remains in its original state	0x0	RW
5:4	DCAEVT1_FRCEN	**DCA EVT1 software triggers behavior**	0x0	RW

		0x0: EPWMxA output high resistance state 0x1: EPWMxA outputs high level 0x2: EPWMxA outputs low level 0x3: EPWMxA remains in its original state		
3:2	TZB	**EPWMxB behavior setting when trip-zone is triggered** 0x0: EPWMxB outputs high resistance state 0x1: EPWMxB outputs high 0x2: EPWMxB outputs low 0x3: EPWMxB remains in its original state	0x0	RW
1:0	TZA	**EPWMxA behavior is set when trip-zone is triggered** 0x0: EPWMxA outputs high resistance state 0x1: EPWMxA outputs high 0x2: EPWMxA outputs low level 0x3: EPWMxA remains in its original state	0x0	RW

16.4.3.22. EPWMx_TZFLAG

Bit(s)	Name	Description	R/W	Reset
31	reserved	-	-	-
30	DCBEVT2INTE	**DCB EVT2 interrupt enabled** 0x0: Disables 0x1: On	RW	0x0
29	DCBEVT1INTE	**DCB EVT1 interrupt enabled** 0x0: Disables 0x1: On	RW	0x0
28	DCAEVT2INTE	**DCA EVT2 interrupt enabled** 0x0: Disables 0x1: On	RW	0x0
27	DCAEVT1INTE	**DCA EVT1 interrupt enabled** 0x0: Off 0x1: On	RW	0x0
26	OSTINTE	**One-Shot Trip Event Indicates that the interruption is enabled** 0x0: Off 0x1: On	RW	0x0
25	CBCINTE	**Cycle-by-Cycle Trip Event Interrupt enabled** 0x0: Disabled 0x1: On	RW	0x0
24:23	reserved	-	-	-
22	CLRDCBEVT2	**DCB EVT2 flag bit cleared** Write 1 Clear flag bit	WO	0x0
21	CLRDCBEVT1	**DCB EVT1 flag bit cleared**	WO	0x0

		Write 1 Clear flag bit		
20	CLRDCAEVT2	**DCA EVT2 flag bit cleared** Write 1 to clear flag bits	WO	0x0
19	CLRDCAEVT1	**DCA EVT1 flag bit cleared** Write 1 Clear flag bit	WO	0x0
18	CLROST	**One-Shot Trip Event flag cleared** Write 1 Clear flag bit	WO	0x0
17	CLRCBC	**Cycle-by-Cycle Trip Event flag bit cleared** Write 1 Clear flag bit	WO	0x0
16	CLRINT	**Interrupt flag Clear** Write 1 Clear flag bit	WO	0x0
15	reserved	-	-	-
14	FRCDCBEVT2	**Software forces Digital Compare Output B Event 2** 0x0: Off 0x1: On	WO	0x0
13	FRCDCBEVT1	**Software forces Digital Compare Output B Event 1** 0x0: Off 0x1: On	WO	0x0
12	FRCDCAEVT2	**The software forcibly enables Digital Compare Output an Event 2** 0x0: Off 0x1: On	WO	0x0
11	FRCDCAEVT1	**Software forces Digital Compare Output an Event 1** 0x0: Off 0x1: On	WO	0x0
10	FRCOST	**The software forces the One-Shot Trip Event** 0x0: Off 0x1: Open	WO	0x0
9	FRCCBC	**The Cycle-by-Cycle Trip Event is forcibly enabled by the software** 0x0: Off 0x1: On	WO	0x0
8:7	reserved	-	-	-
6	DCBEVT2_FLAG	**Digital Compare Output B Event 2 Flag bits** 0x0: No trigger 0x1: There is a signal trigger	RW	0x0
5	DCBEVT1_FLAG	**Digital Compare Output B Event 1 Flag bit** 0x0: No trigger 0x1: There is a signal trigger	RW	0x0
4	DCAEVT2_FLAG	**Digital Compare Output an Event 2 Flag bit** 0x0: No trigger 0x1: There is a signal trigger	RW	0x0

3	DCAEVT1_FLAG	**Digital Compare Output an Event 1 Flag bit** 0x0: No trigger 0x1: There is a signal trigger	RW	0x0
2	OST	**One-Shot Trip status flag bit** 0x0: Not triggered 0x1: There is a signal trigger	RW	0x0
1	CBC	**Cycle-By-Cycle Trip Event status flag bit** 0x0: No trigger 0x1: There is a signal trigger	RW	0x0
0	INT	**Flag bit for TZ interrupt** 0x0: No interrupt was triggered 0x1: An interrupt is triggered	RW	0x0

16.4.3.23. EPWMx_DCCAP

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	
15:0	DCCAP	**Numeric values captured by the DC module** After DCCTL[CAPE] is enabled, the captured data value is stored in this register ● If DCCTL[SHDWMODE] = 0, shadow mode is enabled. In this mode, the register is copied to a shadow register on TBCTR = TBPRD or TBCTR = 0 defined by the dcctl [PULSESEL] bit. The CPU reading this register will return the shadow register value. ● If DCCTL[SHDWMODE] = 1, the shadow register is disabled. In this mode, the cpu will return the register value when it reads.	RW	0x0

16.4.3.24. EPWM_TTCTL

Bit(s)	Name	Description	R/W	Reset
31:24	reserved	-	-	-
23	LVDERR	**LVD error flag bit**	RO	0x0
22	WDTERR	**Watchdog error flag bit**	RO	0x0
21	HOCERR	**HighspeedOSC error flag bit**	RO	0x0
20	SYSERR	**System error flag bit**	RO	0x0
19	LVDERR_TZEN	**LVD error to TripZone 5 Enabled** 0x0: Off 0x1: Open	RW	0x0
18	WDTERR_TZEN	**Watchdog error to TripZone 5 Enable**	RW	0x0

		0x0: Off 0x1: On		
17	HOCERR_TZEN	**Highspeed OSC error to TripZone 5 Enable** 0x0: Off 0x1: On	RW	0x0
16	SYSERR_TZEN	**System error to Tripzone 5 Enable** 0x0: Off 0x1: On	RW	0x0
15	EPWM3_TZINT	**EPWM3 Trip zone flag bit**	R	0x0
14	EPWM2_TZINT	**EPWM2 Trip zone flag bit**	R	0x0
13	EPWM1_TZINT	**EPWM1 Trip zone flag bit**	R	0x0
12	EPWM0_TZINT	**EPWM0 Trip zone flag bit**	R	0x0
11	EPWM3_ETINT	**EPWM3 event triger flag bit**	R	0x0
10	EPWM2_ETINT	**EPWM2 event triger flag bit**	R	0x0
9	EPWM1_ETINT	**EPWM1 event triger flag bit**	R	0x0
8	EPWM0_ETINT	**EPWM0 event triger flag bit**	R	0x0
7:5	reserved	-	-	-
4	CPUTZ	**The software controls the TZ value** Write 1 will force tz6 to 0. Write 0 will force tz6 to 1.	RW	0x0
3	EPWM3_EN	**EPWM3 is enabled** 0x0: Disabled 0x1: Open	RW	0x0
2	EPWM2_EN	**EPWM2 is enabled** 0x0: Disabled 0x1: On	RW	0x0
1	EPWM1_EN	**EPWM1 is enabled** 0x0: Turned off 0x1: On	RW	0x0
0	EPWM0_EN	**EPWM0 Is enabled** 0x0: Disabled 0x1: On	RW	0x0

17. WDT

17.1. Intro

Watchdog works on a clock of 32K. The default timing is 2S, and the feeding time can be changed by modifying the frequency division factor. By configuring the register, you can choose to reset the system or generate an interrupt when the timing

overflows.

17.2. Registers

17.2.1. Register base address

Name	Base Address	Description
WDT	0x40001000	WDT base address

17.2.2. Register list

Offset Address	Name	Description
0x0000	WDT_CON	Control register
0x0004	WDT_KEY	Secret key register

17.2.3. Register Details

17.2.3.1. WDT_CON

Bit(s)	Name	Description	R/W	Reset
31:10	reserved		-	-
9	wake_en	**Enable Wakeup** 0x0: Turned off 0x1: On	RO	0x0
8:7	reserved	-	-	-
6	wdt_pend	**WDT counts full flag** WDT_KEY writes 0xaaaa to clear wdt_pending;	RO	0x0
5	int_enable	**Interrupt enable** 0x0: Reset the system when the count is full 0x1: Interrupts when count is full	RO	0x0
4	wdte	Write WDT_KEY=0xCCCC and set Write WDT_KEY=0xDDDD, reset	RW	0x1
3:0	wdt_psr	**Frequency division factor configuration** WDT_KEY=0x5555 must be written before each configuration of this bit field	RW	0x8

		0x0: Regardless of frequency 0x1:2 0x2:4 0x3:8 0x4:16 0x5:32 0x6:64 0x7:128 0x8:256 0x9:512 0xA: 1024 0xB: 2048 0xC: 4096 0xD: 8192 0xE: 16384 0xF: 32768 Watchdog reset time =1/32K*256* frequency division factor		

17.2.3.2. WDT_KEY

Bit(s)	Name	Description	R/W	Reset
31:16	reserved	-	-	-
15:0	KEY	KEY[15:0]：Key value (write register only, read out value is 0x0000) (Key value) The software must write 0xAAAA at regular intervals, otherwise, the watchdog will produce a reset when the counter is 0. Writing 0xAAAA will clear pending when pending is 1. Writing 0x5555 will allow access to WDT_PSR Write 0xCCCC to start watchdog work Write 0xDDDD, turn off watchdog Write 0xaaaa to clear wdt_pending Write 0x55AA to turn on interrupt enable Write to 0xAA55, disable interrupt Write to 0x5A5A, turn on wake up enable Write to 0xA5A5, turn off wake up enable	RO	0x0