User Guide

megawin

MG32F10x

User Guide

Verson 1.0.2

Date 2022/7/28

Table of Contents

List of Figures

List of Tables

1. Documentation conventions

1.1. List of abbreviations for registers

The following abbreviations are used in register descriptions:

Acronym	Description
read/write (RW)	Software can read and write to these bits
read-only (R)	Software can only read these bits.
write-only (W)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (R_W1)	Software can read as well as clear this bit by writing 1. Writing '0' has no effect on the bit value.
read/clear (R_W0)	Software can read as well as clear this bit by writing 0. Writing '1' has no effect on the bit value.
read/clear by read (RC_R)	Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value.
read/set (RS)	Software can read as well as set this bit. Writing '0' has no effect on the bit value.
read-only write trigger (RT_W)	Software can read this bit. Writing '0' or '1' triggers an event but has no effect on the bit value.
toggle (T)	Software can only toggle this bit by writing '1'. Writing '0' has no effect
Reserved (Res)	Reserved bit, must be kept at reset value.

1.2. Peripheral availability

MG32F10xx support below peripherals:

TIM1 TIM2 TIM3 TIM4

UART1 UART2 UART3

QSPI SPIS1 SPIM2 SPIS2

I2C1 I2C2

USB

I2S

ADC

The availability and number should refer to the datasheet of MG32F10xx：

2. System and memory architecture

The devices of MG32F10xx series are 32-bit general-purpose microcontrollers based on the ARM® Cortex™-M3 processor. The ARM® Cortex™-M3 processor includes three AHB buses known as I-Code, D-Code and System buses. All memory accesses of the ARM® Cortex™- M3 processor are executed on the three buses according to the different purposes and the target memory spaces. The memory organization uses a Harvard architecture, predefined memory map and up to 4 GB of memory space, making the system flexible and extendable.

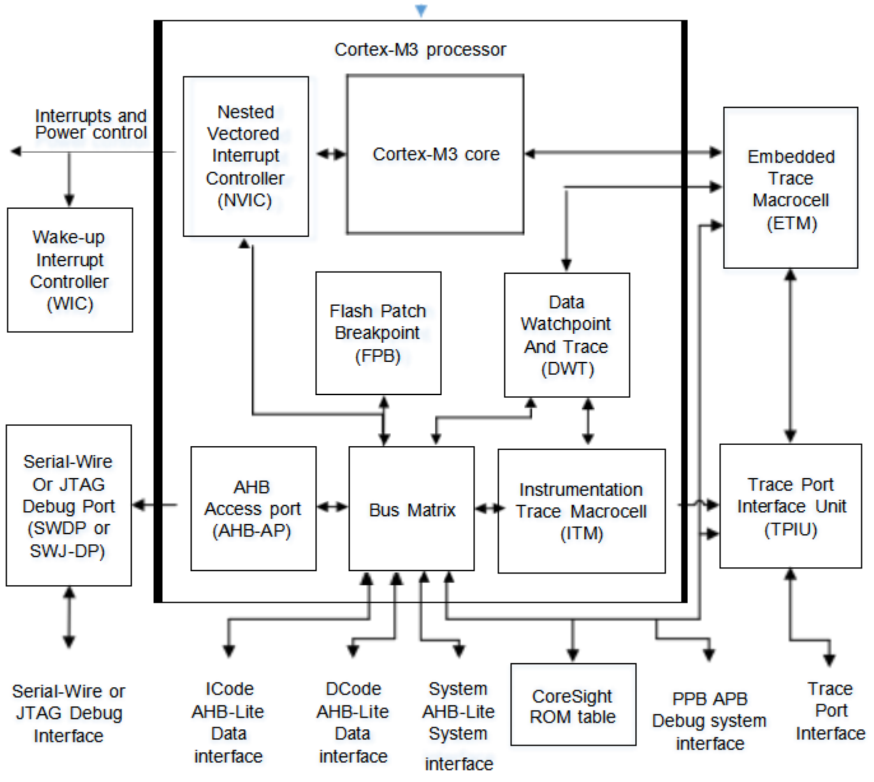
2.1. ARM Cortex-M3 processor

The Cortex™-M3 processor is a 32-bit processor that features low interrupt latency and low-cost debug. Integrated and advanced features make the Cortex™-M3 processor suitable for market products that require micro controllers with high performance and low power consumption. The Cortex™-M3 processor is based on the ARMv7 architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks and advanced data processing bit field manipulations. Some system peripherals listed below are also provided by Cortex™-M3:

- Internal Bus Matrix connected with I-Code bus, D-Code bus, System bus, Private Peripheral Bus (PPB) and debug access.
  - Nested Vectored Interrupt Controller (NVIC)
  - Flash Patch and Breakpoint (FPB)
  - Data Watchpoint and Trace (DWT)
  - Instrumentation Trace Macrocell (ITM)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Trace Port Interface Unit (TPIU)

The figure below shows the Cortex™-M3 processor block diagram. For more information, refer to the ARM® Cortex™-M3 Technical Reference Manual.

Figure 2-1. The structure of the Cortex™ -M3 processor


Cortex-M3 processor
Interrupts and Power control
Nested Vectored Interrupt Controller (NVIC)
Cortex-M3 core
Embedded Trace Macrocell (ETM)
Wake-up Interrupt Controller (WIC)
Flash Patch Breakpoint (FPB)
Data Watchpoint And Trace (DWT)
Serial-Wire Or JTAG Debug Port (SWDP or SWJ-DP)
AHB Access port (AHB-AP)
Bus Matrix
Instrumentation Trace Macrocell (ITM)
Trace Port Interface Unit (TPIU)
Serial-Wire or JTAG Debug Interface
ICode AHB-Lite Data interface
DCode AHB-Lite Data interface
System AHB-Lite System interface
CoreSight ROM table
PPB APB Debug system interface
Trace Port Interface

2.2. System architecture

A 32-bit multi-layer bus is implemented in the MG32F10xx devices, which enables parallel access paths between multiple masters and slaves in the system. The multi-layer bus consists of an AHB interconnect matrix, one AHB bus and two APB buses. The interconnection relationship of the AHB interconnect matrix is shown below. In the following table, “1” indicates the corresponding master is able to access the corresponding slave through the AHB interconnect matrix, while the blank means the corresponding master cannot access the corresponding slave through the AHB interconnect matrix.

Table 2-1. The interconnection relationship of the AHB interconnect matrix

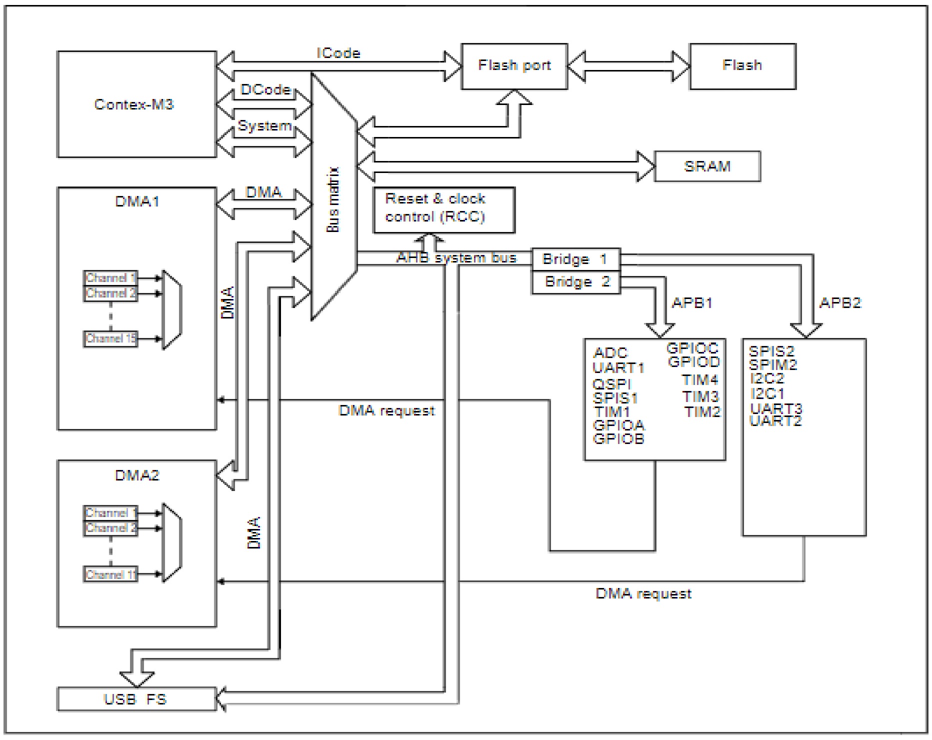
	IBUS	DBUS	SBUS	DMA1	DMA2	USB
FMC-I	1					
FMC-D		1		1	1	
SRAM			1	1	1	1(1)
AHB			1	1	1	
APB1			1	1		
APB2			1		1	

As is shown above, there are several masters connected with the AHB interconnect matrix, including IBUS, DBUS, SBUS, DMA1, DMA2 and USB. IBUS is the instruction bus of the Cortex™-M3 core, which is used for instruction/vector fetches from the Code region (0x0000_0000 ~ 0x1FFF_FFFF). DBUS is the data bus of the Cortex™-M3 core, which is used for loading/storing data and also for debugging access of the Code region. Similarly, SBUS is the system bus of the Cortex™-M3 core, which is used for instruction/vector fetches, data loading/storing and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. DMA1 and DMA2 are the buses of DMA1 and DMA2 respectively. USB is the USB device, and it can only access part of SRAM region (1KB).

There are also several slaves connected with the AHB interconnect matrix, including FMC- I, FMC-D, SRAM, AHB, APB1 and APB2. FMC- I is the instruction bus of the flash memory controller, while FMC-D is the data bus of the flash memory controller. SRAM is on- chip static random access memories. AHB is the AHB bus connected with all of the AHB slaves, while APB1 and APB2 are the two APB buses connected with all of the APB slaves. The two APB buses connect with all the APB peripherals. APB1 and APB2 operate at full speed (up to 128MHz depending on the device).

These are interconnected using a multilayer AHB bus architecture as shown in figure below:

Figure 2-2. MG32F10xx Medium-density series system architecture



2.3. Memory map

The ARM® Cortex™-M3 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. The instruction code and data are both located in the same memory address space but in different address ranges. Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space which is the maximum address range of the Cortex™-M3 since the bus address width is 32-bit. Additionally, a pre-defined memory map is provided by the Cortex™-M3 processor to reduce the software complexity of repeated implementation of different device vendors. In the map, some regions are used by the ARM® Cortex™-M3 system peripherals which can not be modified. However, the other regions are available to the vendors. Table 2-2. Memory map of MG32F10xx devices shows the memory map of the MG32F10xx series devices, including Code, SRAM, peripheral, and other pre-defined regions. Almost each peripheral is allocated 1KB of space. This allows simplifying the address decoding for each peripheral.

Table 2-2. Memory map of MG32F10xx devices

Pre-defined Regions	Bus	Boundary Address	Peripheral
Peripheral	AHB	0x4002 0000 - 0x4002 FFFF	Reserved
		0x4001 7C00 - 0x4001 FFFF	Reserved
		0x4001 7800 - 0x4001 7BFF	FMC
		0x4001 6800 - 0x4001 77FF	Reserved
		0x4001 6400 - 0x4001 67FF	SYS
		0x4001 6000 - 0x4001 63FF	Reserved
		0x4001 5C00 - 0x4001 5FFF	BKP
		0x4001 5800 - 0x4001 5BFF	RTC
		0x4001 5400 - 0x4001 57FF	CACHE
		0x4001 5000 - 0x4001 53FF	Reserved
		0x4001 4C00 - 0x4001 4FFF	SFM
		0x4001 4800 - 0x4001 4BFF	CRC
		0x4001 4400 - 0x4001 47FF	Reserved
		0x4001 4000 - 0x4001 43FF	USB
		0x4001 1000 - 0x4001 3FFF	Reserved

Pre-defined Regions	Bus	Boundary Address	Peripheral
		0x4001 0C00 - 0x4001 0FFF	RCC
		0x4001 0800 - 0x4001 0BFF	IWDG
		0x4001 0400 - 0x4001 07FF	ANCTL
		0x4001 0000 - 0x4001 03FF	PWR
	APB2	0x4000 FC00 - 0x4000 FFFF	DMAC2
		0x4000 BC00 - 0x4000 FBFF	Reserved
		0x4000 B800 - 0x4000 BBFF	RNG
		0x4000 B400 - 0x4000 B7FF	I2S
		0x4000 9C00 - 0x4000 B3FF	Reserved
		0x4000 9800 - 0x4000 9BFF	WWDG
		0x4000 9400 - 0x4000 97FF	SPIS2
		0x4000 9000 - 0x4000 93FF	SPIM2
		0x4000 8C00 - 0x4000 8FFF	I2C2
		0x4000 8800 - 0x4000 8BFF	I2C1
		0x4000 8400 - 0x4000 87FF	UART3
		0x4000 8000 - 0x4000 83FF	UART2
	APB1	0x4000 7C00 - 0x4000 7FFF	DMAC1
		0x4000 4000 - 0x4000 7BFF	Reserved
		0x4000 3C00 - 0x4000 3FFF	ADC
		0x4000 3800 - 0x4000 3BFF	UART1
		0x4000 3400 - 0x4000 37FF	SPIS1
		0x4000 3000 - 0x4000 33FF	QSPI
		0x4000 2C00 - 0x4000 2FFF	Reserved
		0x4000 2800 - 0x4000 2BFF	TIM4
		0x4000 2400 - 0x4000 27FF	TIM3
		0x4000 2000 - 0x4000 23FF	TIM2
		0x4000 1C00 - 0x4000 1FFF	TIM1
		0x4000 1800 - 0x4000 1BFF	EXTI
		0x4000 1400 - 0x4000 17FF	AFIO
		0x4000 1000 - 0x4000 13FF	Reserved
		0x4000 0C00 - 0x4000 0FFF	GPIOD
		0x4000 0800 - 0x4000 0BFF	GPIOC
		0x4000 0400 - 0x4000 07FF	GPIOB

Pre-defined Regions	Bus	Boundary Address	Peripheral
		0x4000 0000 - 0x4000 03FF	GPIOA
SRAM	AHB	0x2007 0000 - 0x3FFF FFFF	Reserved
		0x2006 0000 - 0x2006 FFFF	Reserved
		0x2003 0000 - 0x2005 FFFF	Reserved
		0x2002 0000 - 0x2002 FFFF	Reserved
		0x2001 C000 - 0x2001 FFFF	Reserved
		0x2001 9000 - 0x2001 BFFF	Reserved
		0x2000 1000 - 0x2001 8FFF	SRAM
		0x2000 0000 - 0x2000 0FFF	
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F000 - 0x1FFF FFFF	Option Bytes
		0x1FFF E000 - 0x1FFF EFFF	Boot loader
		0x1FFF 7A10 - 0x1FFF DFFF	Reserved
		0x1FFF 7800 - 0x1FFF 7A0F	Reserved
		0x1FFF 0000 - 0x1FFF 77FF	Reserved
		0x1FFE C010 - 0x1FFE FFFF	Reserved
		0x1FFE C000 - 0x1FFE C00F	Reserved
		0x1001 0000 - 0x1FFE BFFF	Reserved
		0x1000 0000 - 0x1000 FFFF	Reserved
		0x083C 0000 - 0x0FFF FFFF	Reserved
		0x0830 0000 - 0x083B FFFF	Reserved
		0x0800 0000 - 0x0804 FFFF	Main Flash
		0x0004 0000 - 0x07FF FFFF	Reserved
		0x0002 0000 - 0x0003 FFFF	Aliased to Main Flash or Boot loader
		0x0000 0000 - 0x0001 FFFF	

2.3.1. Bit-banding

In order to reduce the time of read-modify-write operations, the Cortex™-M3 processor provides a bit-banding function to perform a single atomic bit operation. The memory map includes two bit-band regions. These occupy the SRAM and Peripherals respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

A mapping formula shows how to reference each word in the alias region to a corresponding bit, or target bit, in the bit-band region. The mapping formula is:

bit_word_addr = bit_band_base + (byte_offset×32) + (bit_number×4) (1-1)

where:

- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region.
- byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- bit_number is the bit position (0-7) of the targeted bit.

For example, to access bit 7 of address 0x2000_0200, the bit-band alias is:

Writing to address 0x2200_401C will cause bit 7 of address 0x2000_0200 change while a read to address 0x2200_401C will return 0x01 or 0x00 according to the value of bit 7 at the SRAM address 0x2000_0200.

2.3.2. On-chip SRAM memory

The MG32F10xx series of devices contain up to 36 KB of on-chip SRAM which starts at the address 0x2000_0000. It supports byte, half-word (16 bits), and word (32 bits) access.

2.3.3. On-chip flash memory overview

The devices provide high density on-chip flash memory, which is organized as follows:

- Up to 256KB of main flash memory.
- Up to 4KB of information blocks for the boot loader.
- Option bytes to configure the device.

Flash Access

The instruction and data in flash is accessed through IBUS and DBUS on AHB bus. There are two 128bit slots cache line for both IBUS and DBUS to speed up the flash access, When IBUS and DBUS request to access flash on the same cycle, the DBUS always have the higher priority to access the bus. The read access time should be configured according to the system clock frequency:

1. Wait cycle: this parameter control the wait cycle number for each read operation.

2. Pre-fetch: the pre-fetch function is turned on by default, the pre-fetch buffer has two 128bit slots which is the same width as the flash bus. Since CPU fetches 32-bit instruction each time so there are four instructions in the buffer for a pre-fetch operation.

Note: These options should be configured according to the system clock frequency:

A. 0 wait cycle, when			system frequency ≤ 32Mhz
B. 1 wait cycle, when	32Mhz	<	system frequency ≤ 48Mhz
C. 2 wait cycle, when	48Mhz	<	system frequency ≤ 72Mhz
D. 3 wait cycle, when	72Mhz	<	system frequency ≤ 96Mhz
E. 4 wait cycle, when	96Mhz	<	system frequency ≤ 128Mhz

When system frequency is higher than 96MHz, the HIFREQ bit in CACHE_CR SFR should be set.

2.4. Program and Erase

The flash can be erased by page or the whole chip, the option byte is not erased during whole chip erase operation.

The flash should be program by page (256 byte), byte program operation is not supported.

The flash page should be erased before program, it is not recommend to program the same page multiple times without erase.

2.5. Boot configuration

The MG32F10xx devices provide three kinds of boot sources which can be selected by the BOOT0 and BOOT1 pins. The details are shown in the following table. The value on the two pins is latched on the 4th rising edge of system clock after a reset. It is up to the user to set the BOOT0 and BOOT1 pins after a power-on reset or a system reset to select the required boot source. Once the two pins have been sampled, they are free and can be used for other purposes.

Table 2-3. Boot modes

Selected boot source	Boot mode selection pins	
	Boot1	Boot0
Main Flash Memory	x	0
Boot loader	0	1
On-chip SRAM	1	1

Note: *When the boot source is hoped to be set as "Main Flash Memory", the Boot0 pin has to be connected with GND definitely and can not be floating.*

After power-on sequence or a system reset, the ARM® Cortex™-M3 processor fetches the top-of-stack value from address 0x0000_0000 and the base address of boot code from 0x0000_0004 in sequence. Then, it starts executing code from the base address of boot code.

Due to the selected boot source, either the main flash memory (original memory space beginning at 0x0800_0000) or the system memory (original memory space beginning at 0x1FFF_E000) is aliased in the boot memory space which begins at the address 0x0000_0000. When the on-chip SRAM whose memory space is beginning at 0x2000_0000 is selected as the boot source, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

The embedded boot loader is located in the System memory, which is used to reprogram the Flash memory. In MG32F10xx devices, the boot loader can be activated through the UART interface.

2.6. CACHE register

2.6.1. Cache Control register (CACHE_CR)

Address offset: 0x00

Reset value: 0x0300 0000

Bits	Fields	R/W	Descriptions
31:26	Reserved	-	Must be kept at reset value
25:24	CHEEN	RW	CACHE Enable control 0x0：the cache is disabled 0x3：the cache is enabled

Bits	Fields	R/W	Descriptions
23:9	Reserved	-	Must be kept at reset value
8	HIFREQ	RW	High speed access auxiliary function, must be enabled when system frequency is higher than 96Mhz. 0: high speed access auxiliary function is turned off 1: high speed access auxiliary function is turned on
7:6	Reserved	-	Must be kept at the reset value
5:4	PREFEN	RW	Pre-fetch function control. 0x0: the instruction prefetch function is turned off 0x3: the instruction prefetch function is turned on
3:0	LATENCY	RW	Flash access wait cycle 0x0: 0 wait cycle, when system frequency ⩽ 32Mhz 0x1: 1 wait cycle, when 32Mhz < system frequency ⩽ 48Mhz 0x2: 2 wait cycle, when 48Mhz < system frequency ⩽ 72Mhz 0x3: 3 wait cycle, when 72Mhz < system frequency ⩽ 96Mhz 0x4: 4 wait cycle, when 96Mhz < system frequency ⩽ 128Mhz Others: Reserved

3. System configuration (SYS)

3.1. Overview

System configuration block provide system information for software. Software will do proper configuration base on the status and hardware configuration based on such information.

Following information can be read from SYS block:

- Chip ID
- SRAM and FLASH memory size
- System boot status and configuration
- Security status and configuration
- The protection status of User Code space, System memory space and Information block

Note: When the system configuration in information block be updated, the system configuration will be only active after system reset.

3.2. Main features of SYS

3.2.1. Security level configuration

MG32F10xx offers different security level options for end user. User code space protection is defined by security configuration. Customer can select security level according to system requirement in different project development stage.

Table 3-1. Security level configuration

Security level	RD_PROT	SRAM_DIS	SWD_DIS	SMEM_DIS	Description
Level 0	0	0	0	0	No protection
Level 1	1	1	0	1	ISP not support, and user program code can not be read out through SWD.
Level 2	1	1	1	0	ISP support
Level 3	1	1	1	1	Highest level and upgrade program defined by customer directly.

Beside these security levels, MG32F10xx offers secondary development space protection while needed. Please refer to section 3.2.5.

Note: While RD_PROT bit be set, DMA read from code space is prohibited.

3.2.2. Information block write protection

MG32F10xx has 8 information blocks. There are 256 bytes for each information block with dedicate protection control and status. Information block and protection setting listed in below:

Table 3-2. Information write protection

Block no	Address space	Protection bit	Description
Block 0	0x1FFF_F000 - 0x1FFF_F0FF	-	Read Only
Block 1	0x1FFF_F100 - 0x1FFF_F1FF	-	Read Only
Block 2	0x1FFF_F200 - 0x1FFF_F2FF	-	Read Only
Block 3	0x1FFF_F300 - 0x1FFF_F3FF	-	Read Only
Block 4	0x1FFF_F400 - 0x1FFF_F4FF	SYS_IF4LCK	Boot control
Block 5	0x1FFF_F500 - 0x1FFF_F5FF	SYS_IF5LCK	User code write protection
Block 6	0x1FFF_F600 - 0x1FFF_F6FF	SYS_IF6LCK	Secondary development protection
Block 7	0x1FFF_F700 - 0x1FFF_F7FF	SYS_IF7LCK	Reboot function configuration and OTP space

Note:Information block 0 -3 is permanently protected, Any modification of block 0-3 is prohibited by hardware. Information block 4 -7 protection is controlled by user. After blocks 4-7 be set as write protected, any erase operation in block 4 7 will cause the data in main array damaged

Address space 0x1FFF_F720 - 0x1FFF_F7FF of Information block 7 can be used as OTP area for user if write protection not be set

3.2.3. The write protection of system memory space

MG32F10xx has 4K bytes system memory.

System memory space: 0x1FFF_E000 - 0x1FFF_EFFF

3.2.4. User program space write protection control

MG32F10xx write protection is based on 4K bytes. There are 32 write protection bits for the whole array (bit0-bit31). The bit 31 also controls the user program space beyond 128K bytes.

Protection bit of user program memory space list in below table:

Table 3-3. User program space protection

Block No	Address space	Write protection	Description
Block 0	0x0800_0000 - 0x0800_0FFF	MAIN_WEN[0]	User configure
Block 1	0x0800_1000 - 0x0800_1FFF	MAIN_WEN[1]	User configure
Block 2	0x0800_2000 - 0x0800_2FFF	MAIN_WEN[2]	User configure
Block 3	0x0800_3000 - 0x0800_3FFF	MAIN_WEN[3]	User configure
Block 4	0x0800_4000 - 0x0800_4FFF	MAIN_WEN[4]	User configure
Block 5	0x0800_5000 - 0x0800_5FFF	MAIN_WEN[5]	User configure
Block 6	0x0800_6000 - 0x0800_6FFF	MAIN_WEN[6]	User configure
Block 7	0x0800_7000 - 0x0800_7FFF	MAIN_WEN[7]	User configure
Block 8	0x0800_8000 - 0x0800_8FFF	MAIN_WEN[8]	User configure
Block 9	0x0800_9000 - 0x0800_9FFF	MAIN_WEN[9]	User configure
Block 10	0x0800_A000 - 0x0800_AFFF	MAIN_WEN[10]	User configure
Block 11	0x0800_B000 - 0x0800_BFFF	MAIN_WEN[11]	User configure
Block 12	0x0800_C000 - 0x0800_CFFF	MAIN_WEN[12]	User configure
Block 13	0x0800_D000 - 0x0800_DFFF	MAIN_WEN[13]	User configure
Block 14	0x0800_E000 - 0x0800_EFFF	MAIN_WEN[14]	User configure
Block 15	0x0800_F000 - 0x0800_FFFF	MAIN_WEN[15]	User configure
Block 16	0x0801_0000 - 0x0801_0FFF	MAIN_WEN[16]	User configure
Block 17	0x0801_1000 - 0x0801_1FFF	MAIN_WEN[17]	User configure
Block 18	0x0801_2000 - 0x0801_2FFF	MAIN_WEN[18]	User configure
Block 19	0x0801_3000 - 0x0801_3FFF	MAIN_WEN[19]	User configure
Block 20	0x0801_4000 - 0x0801_4FFF	MAIN_WEN[20]	User configure
Block 21	0x0801_5000 - 0x0801_5FFF	MAIN_WEN[21]	User configure
Block 22	0x0801_6000 - 0x0801_6FFF	MAIN_WEN[22]	User configure
Block 23	0x0801_7000 - 0x0801_7FFF	MAIN_WEN[23]	User configure
Block 24	0x0801_8000 - 0x0801_8FFF	MAIN_WEN[24]	User configure
Block 25	0x0801_9000 - 0x0801_9FFF	MAIN_WEN[25]	User configure
Block 26	0x0801_A000 - 0x0801_AFFF	MAIN_WEN[26]	User configure
Block 27	0x0801_B000 - 0x0801_BFFF	MAIN_WEN[27]	User configure

Block No	Address space	Write protection	Description
Block 28	0x0801_C000 - 0x0801_CFFF	MAIN_WEN[28]	User configure
Block 29	0x0801_D000 - 0x0801_DFFF	MAIN_WEN[29]	User configure
Block 30	0x0801_E000 - 0x0801_EFFF	MAIN_WEN[30]	User configure
Block 31	0x0801_F000 - 0x0801_FFFF	MAIN_WEN[31]	User configure
Block 32	0x0802_0000 - 0x0803_FFFF	MAIN_WEN[31]	Beyond 128K bytes space

When read protection is activated, the write protection function in different boot mode is listed below:

Table 3-4. User space write protection (read protect enabled)(low density product)

Boot mode	0x0800_0000 - 0x0800_0FFF	Other space
Boot from SRAM	Write protection	Write protection
Boot from system program	Write protection	MAIN_WEN
Boot from user program	Write protection	MAIN_WEN

Table 3-5. User space write protection (read protect enabled)(high density product)

Boot mode	0x0800_0000 - 0x0800_1FFF	Other space
Boot from SRAM	Write protection	Write protection
Boot from system program	Write protection	MAIN_WEN
Boot from user program	Write protection	MAIN_WEN

Note: 192K bytes flash product only support lowest 4K bytes write protection when read protection is active.

Note: The flash density of high density product is >= 128K bytes and low density product <=64K bytes.

3.2.5. Secondary Development

MG32F10xx supports secondary development function. While the function is enabled (SENDEV_ EN=1), the main array is separated into user code and special code area.

Only instruction can be stored in special space, and constant data used for special code space has to be stored in user code space. The program in special code space is executive only, any data read operation to special code space will cause hardware exception interrupt.

The user code space is defined by SENDEV_SIZE, and special code space is beyond SENDEV_SIZE. For example: Flash memory size=64KB, SENDEV_SIZE = 48KB, then special code space is from 48KB to 64KB. After user code space/special code space configured, and secondary development function be enabled, user should reset the chip to active it.

3.3. SYS Register Description

3.3.1. ID Register (SYS_ID)

Address bias: 0x00

Reset value: NA

Bits	Fields	R/W	Description
31:28	-	R	It is 0x3.
27:14	-	R	Reserved
13:0	CHIP_ID	R	Chip ID number It is 0x2980.

3.3.2. Memory control Register (SYS_MEMSZ)

Address bias: 0x04

Reset value: NA

Bits	Fields	R/W	Description
31:6	-	-	Reserved
5:4	SRAM_SIZE	R	SRAM density 00: SRAM density is 36K Bytes. 01: SRAM density is 28K Bytes. 10: SRAM density is 20K Bytes. 11: SRAM density is 12K Bytes.
3:0	FLASH_SIZE	R	FLASH density 0000: Reserved configuration. 0010: Reserved configuration. 0011: FLASH density is 256K Bytes. 0100: FLASH density is 128K Bytes. 0110: FLASH density is 96K Bytes. 0111: FLASH density is 64K Bytes. 1111: FLASH density is 192K Bytes. Others: FLASH density is 32K Bytes.

3.3.3. Security control Register (SYS_BTCR)

Address bias: 0x0C

Reset value:NA

Bits	Fields	R/W	Description
31:25	-	-	Reserved
24	SMEM_DIS	R	System memory boot control 0:System memory boot enable 1:System memory boot disable
23:17	-	-	Reserved
16	SRAM_DIS	R	SRAM boot control 0: SRAM boot enable 1: SRAM boot disable
15:9	-	-	Reserved
8	SWD_DIS	R	SWD control 0:SWD enable 1:SWD disable
7:1	-	-	Reserved
0	RD_PROT	R	Read protection control 0:Read protection enable 1:Read protection disable

3.3.4. FLASH write protection register (SYS_MEMWEN)

Address bias: 0x10

Reset value: NA

Bits	Fields	R/W	Description
31:0	MAIN_WEN	R	FLASH Write protection control 0: Main array Erase/Write disable. 1: Main array Erase/Write enable.

Note: Each control bit protect 4K Bytes memory space, for high density product, bit 31 also protect the space beyond 128K Bytes.

3.3.5. Secondary development control Register (SYS_SENDEV)

Address bias: 0x14

Reset value: NA

Bits	Fields	R/W	Description
31:17	-	-	Reserved
16	SENDEV_EN	R	Secondary development control 0:Secondary development disable 1:Secondary development enable
15:12	-	-	Reserved
11:0	SENDEV_SIZE	R	Secondary development code space control, the granuality is 1K Byte. When Secondary development enabled, the memory space beyond SENDEV_SIZE is special code space.

3.3.6. Reboot control Register (SYS_RSTCR)

Address bias: 0x18

Reset value: 0x0000 000x

Bits	Fields	R/W	Description
31:3	-	-	Reserved
2	RST_STBY	R	Standby Mode reset control 0:Reset disabled when enter standby mode 1: Reset enabled when enter standby mode
1	RST_STOP	R	Stop mode reset control 0:Reset disabled when enter stop mode 1:Reset enabled when enter stop mode
0	IWDG_EN	R	IWDG start control 0:IWDG start by software 1:IWDG automatically start after reset

3.3.7. Information block 4 protection register (SYS_IF4LCK)

Address bias: 0x1C

Reset value: 0x0000 000x

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SYS_IF4LCK	R	4th Information block Erase/Write control 1:4th Information block Erase/Write disable 0:4th Information block Erase/Write enable

Note: When Information block is protected, Any Erase/Write in Information 4 will cause main array data damage.

3.3.8. Information block 5 protection register (SYS_IF5LCK)

Address bias: 0x20

Reset value: 0x0000 000x

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SYS_IF5LCK	R	5th Information block Erase/Write control 1:5th Information block Erase/Write disable 0:5th Information block Erase/Write enable

Note: When Information block is protected, Any Erase/Write in Information block 5 will cause main array data damage.

3.3.9. Information block 6 protection register (SYS_IF6LCK)

Address bias: 0x24

Reset value: 0x0000 000x

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SYS_IF6LCK	R	6th Information block Erase/Write control 1:6th Information block Erase/Write disable 0:6th Information block Erase/Write enable

Note: When Information block is protected, Any Erase/Write in Information block 6 will cause main array data damage.

3.3.10. Information block 7 protection register (SYS_IF7LCK)

Address bias: 0x28

Reset value: 0x0000 000x

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SYS_IF7LCK	R	7th Information block Erase/Write control 1:7th Information block Erase/Write disable 0:7th Information block Erase/Write enable

Note: When Information block is protected, Any Erase/Write in Information block 7 will cause *main array data damage.*

3.3.11. Boot control Register (SYS_BTSR)

Address bias: 0x34

Reset value: 0x0000 0002

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1:0	BOOT_MODE	R	Boot mode 00:Reserved 01:Boot from system memory 10: Boot from user code space 11:Boot from SRAM

4. Cyclic Redundancy Check (CRC)

4.1. CRC introduction

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

This CRC calculation unit can be used to calculate 8 bit, 16 bit or 32 bit CRC code with configurable polynomial.

4.2. CRC main features

- Supports three common polynomials CRC8、CRCCCITT、CRC16 and CRC32.

  CRC-8: $x^8 + x^2 + x + 1$

  CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$

  CRC-16: $x^{16} + x^{15} + x^2 + 1$

  CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1's complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU write or DMA back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

4.3. CRC register description

4.3.1. CRC mode register (CRC_MODE)

Address offset：0x00

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:8	-	-	Reserved
7	SEED_SET	RW	Write 1 to load the seed into the CRC.
6	SEED_OP	RW	CRC Seed setting 0: Use polynomial default value 1: Use the CRC_SEED register as the CRC operation seed
5	CMPL_SUM	RW	CRC sum complement: 1= 1’s complement for CRC_SUM 0=No 1’s complement for CRC_SUM
4	BIT_RVS_SUM	RW	CRC sum bit order: 1= Bit order reverse for CRC_SUM 0= No bit order reverse for CRC_SUM
3	CMPL_WR	RW	Data complement: 1= 1’s complement for CRC_WR_DATA 0= No 1’s complement for CRC_WR_DATA
2	BIT_RVS_WR	RW	Data bit order: 1= Bit order reverse for CRC_WR_DATA (per byte) 0= No bit order reverse for CRC_WR_DATA (per byte)
1:0	CRC_POLY	RW	CRC polynomials: 00= CRC-8 polynomial (default seed value is 0x00) 01= CRC-CCITT polynomial(default seed value is 0xFFFF) 10= CRC-16 polynomial (default seed value is 0x0000) 11= CRC-32 polynomial (default seed value is 0xFFFFFFFF)

4.3.2. CRC seed register (CRC_SEED)

Address offset: 0x04

Reset value: 0x0000 FFFF

Bits	Fields	R/W	Description
31:0	CRC_SEED	RW	CRC seed value

4.3.3. CRC checksum register (CRC_SUM)

Address offset: 0x08

Reset value: 0x0000 FFFF

This register is a Read-only register containing the most recent checksum. The read request to this register is automatically delayed by a finite number of wait states until the results are valid and the checksum computation is complete.

Bits	Fields	R/W	Description
31:0	CRC_SUM	R	CRC checksum

4.3.4. CRC data register (CRC_WDATA)

Address offset: 0x08

Reset value: 0xXXXX XXXX

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Bits	Fields	R/W	Description
31:0	CRC_WDATA	W	Data written to this register will be taken to perform CRC calculation with selected bit order and 1’s complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.

5. Random Number Generator (RNG)

5.1. Introduction

Random Number Generator (RNG) uses a 24-bit LFSR to generate an 8-bit random number. The RNG SFRs can be accessed through APB2.

Note：The APB2 bus clock and RNG clock should be turned on before any read or write access, please refer to APB2PRE, APB2ENR and RNGCLKENR SFRs in RCC macro.

5.2. Random number generator SFR

5.2.1. Random number generator (RNG_RAND)

Address offset: 0x00

Reset value: 0x00000000

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	RNG_RAND	R	Generated 8-bit random number

5.2.2. RNG STOP register (RNG_STOP)

Address offset: 0x04

Reset value: 0x00000000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	RNG_STOP	RW	RNG STOP control, high active

5.2.3. RNG Record register (RNG_REC)

Address offset: 0x08

Reset value: 0x00000000

Bits	Fields	R/W	Description
31:8	-	R	Reserved

7:0	RNG_REC	R	The random data being read last time

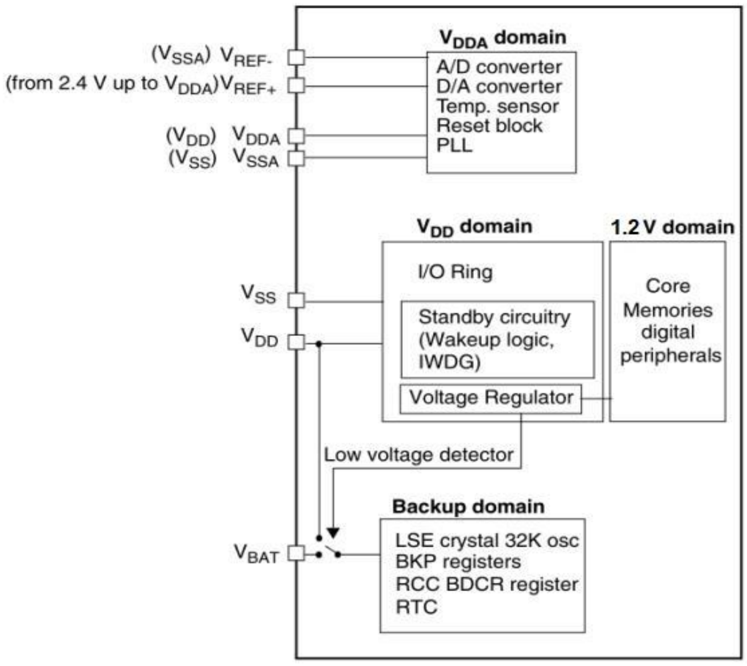
6. Power control (PWR)

6.1. Power supplies

The MG32F10xx requires a 2.0-to-3.6 V operating voltage supply (VDD). An embedded regulator is used to supply the internal 1.2 V digital power.

The real-time clock (RTC) and backup registers can be powered from the VBAT voltage when the main VDD supply is powered off. The VBAT power supply voltage is 1.8V ~ 3.6V.

Table 6-1. Power supply overview



6.1.1. Independent A/D converter supply and reference voltage

To improve conversion accuracy, the ADC have an independent power supply which can be separately filtered and shielded from noise on the PCB.

- The ADC voltage supply input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on pin VSSA

The VREFH is connected to VDDA and the VREFL is connected to VSSA.

These two pins are not packaged.

6.1.2. Battery backup domain

To retain the content of the Backup registers and supply the RTC function when VDD is turned off, VBAT pin can be connected to an optional standby voltage supplied by a battery or by another source.

The VBAT pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 IOs, allowing the RTC to operate even when the main digital supply (VDD) is turned off. The switch to the VBAT supply is controlled by the Power Down Reset embedded in the Reset block.

warning：

- During the rising phase of VDD or after PDR (power down reset) is detected, VBAT and VDD are powered at the same time within a period of time (about 50ns). During this time, current may be injected into the VBAT through an internal diode between the VDD and VBAT.
- If the power supply or battery connected to the VBAT cannot withstand such injection current, it is recommended to connect a low voltage drop diode between the external VBAT and the power supply.If no external battery is used in the application, it is recommended to connect VBAT externally to VDD with a 100 nF external ceramic decoupling capacitor.
- PC14 and PC15 can be used as either GPIO or LSE pins.
- PC13 can be used as GPIO, TAMPER pin, RTC Calibration Clock, RTC Alarm or second output

When the backup domain is supplied by VBAT (analog switch connected to VBAT because VDD is not present), the following functions are available:

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as TAMPER pin, RTC Alarm or Second output (refer to Section 7.4.1: RTC clock calibration register (BKP_RTCCR)).

Note: because analog switch can only pass a small amount of current (3mA), the use of pc13 to pc15 I/O ports in output mode is limited: the speed must be limited below 2MHz, the maximum load is 30pf, and these I/O ports must not be used as current source (such as driving LED).

6.1.3. Voltage regulator

The voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes.

- In Run mode, the regulator supplies full power to the 1.2 V domain (core, memories and digital peripherals).
- In Stop mode the regulator supplies low-power to the 1.2 V domain, preserving contents of registers and SRAM
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

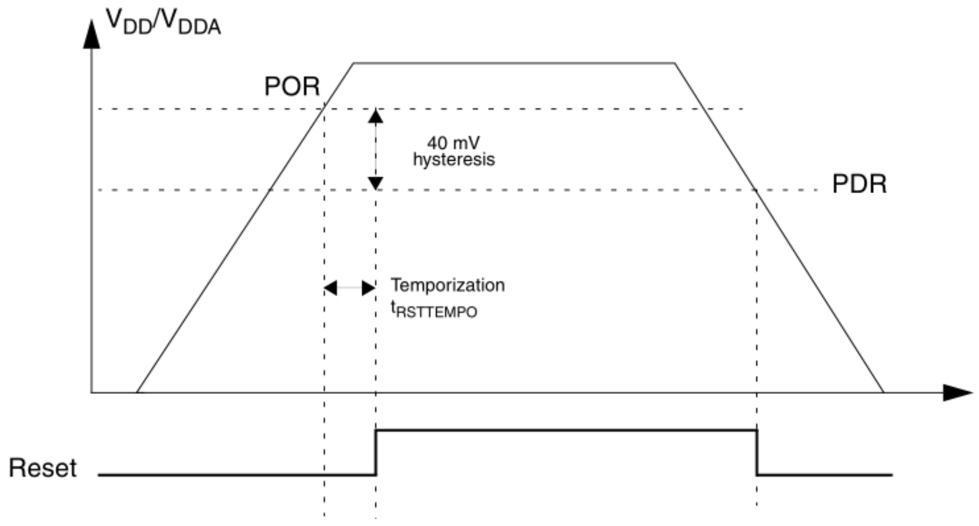
6.2. Power supply supervisor

6.2.1. Power on reset (POR)/power down reset (PDR)

MG32F10xx has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2 V.

The device remains in Reset mode when VDD/VDDA is below a specified threshold, VPOR/PDR, without the need for an external reset circuit. For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.

Figure 6-1. Power on reset/power down reset waveform



6.2.2. Programmable voltage detector

You can use the PVD to monitor the VDD/VDDA power supply by comparing it to a threshold selected by the PLS.

The PVD is enabled by setting the PVDE bit.

PWR_SR0 to indicate if VDD /VDDA is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when VDD /VDDA drops below the PVD threshold and/or when VDD /VDDA rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. For example: the service routine could perform emergency shutdown tasks.

6.3. Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The MG32F10xx devices feature three low-power modes:

- Sleep mode (CPU clock off, all peripherals including Cortex ® -M3 core peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.2V domain powered-off)

In addition, the power consumption in Run mode can be reduce by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

Table 6-2. Low-power mode summary

Mode name	Entry	wakeup	Effect on 1.2V domain clocks	Effect on VDD Domain clocks	Voltage regulator

Mode name	Entry	wakeup	Effect on 1.2V domain clocks	Effect on VDD Domain clocks	Voltage regulator
Sleep	WFI	Any interrupt	CPU clock OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS(0) + SLEEPDEEP BIT + WFI/WFE	Any EXTI line (configured in the EXTI registers)	All 1.2V domain clocks OFF	HSI and HSE oscillators OFF	Normal/low-Power/Ultra-low power mode
Standby	PDDS BIT (1)+SLEEP-DEEP BIT +WFI/WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			Ultra-low power mode

6.3.1. Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK、HCLK、FCLK) can be reduced by programming the prescaler registers. User can also configure the prescalers to slow down the peripherals frequency before entering Sleep mode.

Please refer to RCC macro for more details.

6.3.2. Peripheral clock gating

In Run mode, peripherals and memories can be stopped at any time to reduce power consumption if they are not used.

To further reduce power consumption in Sleep mode, the peripheral clocks can also be gated off prior to executing the WFI or WFE instructions. The SFR AHBENR0, AHBENR1, APBENR1 and APBENR2 can be used for this purpose.

6.3.3. Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait FOR Interrupt) or WFE (Wait for

Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex ® -M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as

WFI or WFE instruction is executed.

- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

In the Sleep mode, all I/O pins keep the same state as in the Run mode.

Refer to Table 6.2 and Table 6.3 for details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

- Enable an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex ® -M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- Configure an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

Refer to Table 6.2 and Table 6.3 for more details on how to exit Sleep mode.

Table 6-3. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex ® -M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to Section 10.1.2: Interrupt and exception vectors If WFE was used for entry Wakeup event: Refer to Section 10.2.3: Wakeup event management
Wakeup latency	None

Table 6-4. Sleep-on-exit

SLEEP-ON-EXIT	Description
Mode entry	WFI (wait for interrupt) while:

SLEEP-ON-EXIT	Description
	– SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex ® -M3 System Control register.
Mode exit	Interrupt: refer to Section 10.1.2: Interrupt and exception vectors.
Wakeup latency	None

Note: In sleep mode, the power consumption can be further reduced by the configuration below:

- Reduce the system clock frequency by half. Please refer to 6.4.1 for details

6.3.4. Stop mode

The Stop mode is based on Cortex ® -M3 deepsleep mode combined with peripheral clock gating. In stop mode, the voltage regulator can operate in normal mode, low power mode or ultra low power mode.In stop mode, all clocks in the 1.2V power supply area are stopped, PLL, MHSI. FHSI and HSE RC oscillators are disabled, and the contents of SRAM and registers are retained.

In stop mode, all I / O pins keep the same state as in the RUN mode.

Enter stop mode

Please refer to table 6.6 for details on how to enter stop mode. If flash programming is in progress, the system will not enter stop mode until the memory access is completed. If the access to APB is in progress, the system will not enter the stop mode until the access to APB is completed.

Four stop modes are supported, and please refer to table 6-5 for detail information.

Table 6-5. Four Stop Modes

Stop mode	Configuration of Analog parts in each mode	CR0 configure	CFGR configure
SP1	– turn off MHSI, FHSI, HSE and PLL – the voltage regulator works normally, but the output voltage decreases (about 100mV) – flash enters sleep mode – bandgap works normally	0x900	0x3F
SP2	– turn off MHSI, FHSI, HSE and PLL – voltage regulator enters low power mode	0x22002	0x3BE

Stop mode	Configuration of Analog parts in each mode	CR0 configure	CFGR configure
	– flash power down – bandgap works normally		
SP3	– turn off MHSI, FHSI, HSE and PLL – the voltage regulator enters the ultra-low power consumption mode – flash power down – bandgap works normally	0x63004	0x3BF
SP4	– turn off MHSI, FHSI, HSE and PLL – the voltage regulator enters the ultra-low power consumption mode – flash power down – bandgap off	0x7B004	0x3B3

Note: the DBP bit in register CR0 is not related to stop mode, please keep the default state during configuration.

By programming the control bits, the following functions can be selected:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except the system reset
- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the RCC_LSICR)
- External 32.768kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Battery domain control register (BKP_BDCR)

In stop mode, if SARADC is not turned off before entering this mode, it still consume current. By clearing ADCON bit of ANCTL_ADCENR register and ADON bit of ADC_CR2 register, the SARADC can be turned off.

Exiting Stop mode

Refer to Table 6.6 for more details on how to exit Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the MHSI RC oscillator is selected as system clock.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop mode.

After exiting from stop mode, all clock configurations return to the initial state, and users need to reconfigure RCC. If the system does not successfully enter the stop mode due to the interruption or event coming ahead, user can read the CKF flag of PWR_SR1 to check whether all the configurations are back to the initial state.

Note: after exiting standby mode, FHSI is turned off. If users need to use FHSI, please open it by yourself.

Table 6-6. Stop mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: –Set SLEEPDEEP bit in Cortex ® -M3 System Control register –Clear PDDS bit in Power Control register –Select the mode of voltage regulator by setting power control register 0 Note: in order to enter the stop mode, all external interrupt request bits and RTC alarm flags must be cleared, otherwise the entry process of stop mode will be skipped and the program will continue to run.
Mode exit	If WFI was used for entry: - Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). 。Refer to Section 10.1.1: If WFE was used for entry:- Any EXTI Line configured in event mode. Refer to Section 10.1.1:
Wakeup latency	MHSI RC wake-up time + voltage regulator wake-up time from low power consumption + bandgap wake-up time (according to configuration).

6.3.5. Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex ® -M3 deepsleep mode, with the voltage regulator disabled. The 1.2V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also turned off. SRAM and the register contents are lost except for registers in the Battery domain and Standby circuitry.

Entering Standby mode

Refer to Table 6.7 for more details on how to enter Standby mode

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See Section 19.3: IWDG functional description
- Real-time clock (RTC): this is configured by the RTCEN bit in the Battery domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status

register (RCC_CSR)

- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin or the rising edge of an RTC alarm occurs.

After waking up from Standby mode, program execution restarts in the same way as after a Reset. Power control status register STA_1 indicates the kernel exits from standby mode.

Refer to Table 6.7 for more details on how to exit Standby mode.

Table 6-7. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex ® -M3 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register
Mode exit	WKUP pin rising edge, RTC alarm event’s rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Start of voltage regulator in reset phase.

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except:

- Reset pad (still available)
- TAMPER pin if configured for tamper or calibration out
- WKUP pin, if enabled

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex ® -M3 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively.

6.3.6. Module status in low power mode

After entering sleep mode, the status of each module remains unchanged, except that the CPU clock is turned off; After entering the stop mode, according to the PWR_CR0 configuration, each module can be turned off or enter the low power consumption mode; After entering the standby mode, each module enters the low power consumption mode or stop mode.

Details are shown in the table below:

Table 6-8. Analog state in different power mode

	MAIN REG	FLASH	MHSI	FHSI	LSI	HSE	PLL	PVD	POR BG	SRAM	Core logic
Active	ON	ON	ON	ON	CFG	CFG	CFG	CFG	ON	ON	ON
Sleep mode	ON	ON	ON	ON	CFG	CFG	CFG	CFG	ON	ON	ON
Stop mode	CFG	CFG	OFF	OFF	CFG	OFF	OFF	CFG	CFG	CFG	ON
Standby mode	ULP	PD	OFF	OFF	CFG	OFF	OFF	OFF	OFF	OFF	OFF

Note: OFF in the table indicates that the macro is turned off; PD in flash indicates power down mode; CFG indicates user configurable; ULP indicates ultra-low power consumption mode.

6.3.7. Auto-wakeup (AWU) from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in BKP_BDCR SFR.

- Low-power 32.768 kHz external crystal oscillator (LSE OSC).
- Low-power internal RC Oscillator (LSI RC)
- This clock source has the advantage of saving the cost of the 32.768 kHz crystal.

To wakeup from Stop mode with an RTC alarm event, it is necessary to:

- Configure the EXTI Line 17 to be sensitive to rising edge

- Configure the RTC to generate the RTC alarm.

To wakeup from Standby mode, there is no need to configure the EXTI Line 17.

6.4. Power control registers

6.4.1. Power control register 0 (PWR_CR0)

Address offset：0x00

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:20	-	R	Reserved
19	S4KMODE	RW	Configure the state of 4K SRAM when it enters stop mode 1：4K SRAM power off 0：4K SRAM power on
18	S32KMODE	RW	Configure the state of 32K SRAM when it enters stop mode 1：32K SRAM power off 0：32K SRAM power on
17	FLASHMODE	RW	Configure the state of the flash memory when it enters stop mode 1：Flash memory enters power down mode 0：Flash memory enters sleep mode
16	PORMODE	RW	Configure the status of the POR BG when it enters stop mode 1：close the POR BG 0：keep the POR BG
15:14	BGMODE	RW	Configure the status of the bandgap when it enters the stop mode. If some analog modules still work normally in stop mode, the bandgap reference module will work normally. The configuration of this bit only works when all analog parts stop working. 00: The bandgap works normally 01: The bandgap works in low power mode 10: The bandgap is turned off 11: Reserved
13:12	FREGMODE	RW	Configure the state of the flash regulator when it enters stop mode 0x：.The flash regulator works normally 10：.The flash regulator works in low power mode 11：.The flash regulator works is turned off

Bits	Fields	R/W	Description
11:9	REGLVL	RW	Configure the step value of the voltage regulator When entering the stop mode, if the voltage regulator is in the normal working mode, its output voltage can be reduced to achieve the purpose of power saving. This bit is used to configure the adjusted step value. 000: 0 step value 001: 1 step value 010: 2 step value 011: 3 step value 100: 4 step value 101: 5 step value 110: 6 step value 111: 7 step value *Note: about 23mv for each step.*
8:7	REGCFG	RW	Configure the voltage change direction of the voltage regulator When entering the stop mode, if the voltage regulator is in the normal working mode, its output voltage can be reduced to achieve the purpose of power saving. This bit is used to configure the direction of voltage regulation. 0x：The output voltage of the voltage regulator does not change 10：Reduce the output voltage of the voltage regulator 11：Increase the output voltage of the voltage regulator *Note: when the output voltage is increased or decreased, the specific step value is configured by REGLVL bits.*
6:5	PDDS	RW	Configure the working state of the device when the Cortex ® -M3 enters deep sleep mode 00：The device enters stop mode 10：The device enters standby mode 01/11：Reserved
4	-	R	Reserved
3	FCLKSD	RW	Reduce the frequency of Cortex ® -M3 FCLK. When entering sleep mode, f_{FCLK} can be reduced to reduce power consumption. This bit is only used in sleep mode. 1: When entering sleep mode, if the frequency division coefficient of FCLK is less than 32 (refer to the register description of RCC mode), f_{FCLK} is reduced to half of the original 0: f_{FCLK} remains unchanged when entering sleep mode
2:1	REGMODE	RW	Configure the state of the voltage regulator when it enters stop mode 00：The voltage regulator works normally 01：The voltage regulator enters the low power mode 10：The voltage regulator enters the ultra low power mode 11：.Reserved

Bits	Fields	R/W	Description
0	DBP	RW	Disable the write protection for backup domain registers .In reset state, RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers. 0：Access to RTC and backup registers is disabled 1：Access to RTC and backup registers is enabled *Note: if the clock of RTC is HSE / 128, this bit must be kept as' 1 '.*

Note:

1) *If the output voltage of the voltage regulator changes in sleep mode, it will automatically return to the normal state when exiting sleep mode.*
2) *If SRAM, flash memory and other analog parts enter power saving mode in stop mode, when they exit stop mode, the hardware will restart them to ensure that they can work normally*
3) *In standby mode, SRAM, flash memory and other analog parts all enter power saving mode. When the stop mode is exited, the system reset will be started, and the CPU instruction starts to work from the initial address.*

6.4.2. Control register 1 (PWR_CR1)

Address offset：0x04

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3	CCKF	W	Clear CKF flag bit This bit is always read as 0. CKF flag will be cleared by hardware when writing 1 to this bit, and writing 0 is invalid
2	CSPF	W	Clear SPF flag bit This bit is always read as 0. SPF flag will be cleared by hardware when writing 1 to this bit, and writing 0 is invalid
1	CSBF	W	Clear SBF flag bit This bit is always read as 0. SBF flag will be cleared by hardware when writing 1 to this bit, and writing 0 is invalid
0	CWUF	W	Clear WUF flag bit

Bits	Fields	R/W	Description
			This bit is always read as 0. WUF flag will be cleared by hardware when writing 1 to this bit, and writing 0 is invalid

6.4.3. Control register 2 (PWR_CR2)

Address offset：0x08

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	EWUP	RW	Enable WKUP pin 0：WKUP pin is used for GPIO. An event on the WKUP pin doesn’t wakeup the device from standby mode 1：WKUP pin is used for wakeup from standby mode and forced in input pull down configuration. (Rising edge on WKUP pin wakeup the system from standby mode) *Note: this bit is reset by a system reset*

6.4.4. Status register 0 (PWR_SR0)

Address offset：0x10

Reset value：0x0000 0000

This register can only be read, but not written.

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	PVDO	R	PVD output 0：VDD is higher than the selected PVD threshold. 1：VDD is below the selected PVD threshold.

Note: PVD is stopped in standby mode. Therefore, after standby mode or reset, until PVDE bit is set, this bit is 0.

6.4.5. Status register 1 (PWR_SR1)

Address offset：0x14

Reset value：0x0000 0000

This register is read only. It is only reset by power on reset, and its contents are retained when system reset occur.

When the system exits stop mode and standby mode, user needs to read the register to check the state of the system.

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3	CKF	R	CKF Flag When reading, if the bit is 1, it means that when entering stop mode or standby mode, the clock configuration of the system returns to the initial state. User needs to do the following operations below: - Write CCKF to 1 to clear the flag bit - Reconfigure clock
2	SPF	R	Stop flag When reading, if the bit is 1, the system exits from stop mode. User needs to write CSPF to clear the flag bit.
1	SBF	R	Standby flag When reading, if the bit is 1, the system exits from standby mode. User needs to write CSBF to clear the flag bit.
0	WUF	R	Wakeup flag 0: No wakup event occured 1: A wakeup event was received from the WKUP pin or from the RTC alarm Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.

6.4.6. General purpose register 0 (PWR_GPREG0)

Address offset：0x18

Reset value：0x0000 0000

The status of this register is reset only by POR/PDR, and the contents remain unchanged when system reset occur.

This register is 32-bit general-purpose register 0.

6.4.7. General purpose register 1 (PWR_GPREG1)

Address offset：0x1C

Reset value：0x0000 0000

The status of this register is reset only by POR/PDR, and the contents remain unchanged when system reset occur.

This register is 32-bit general-purpose register 1.

6.4.8. Configuration register (PWR_CFGR)

Address offset: 0x20

Reset value: 0x0000 03BE

The status of this register is reset only by POR/PDR, and the contents remain unchanged when system reset occur.

Bits	Fields	R/W	Description
31:13	-	R	Reserved
12:0	LPCFG	RW	Configure the time to enter and exit low power mode. 0x3F: Configure the time when entering and exiting stop mode Lp1. 0x3BE: Configure the time when entering and exiting stop mode Lp2. 0x3BF: Configure the time when entering and exiting stop mode Lp3. 0x3B3: Configure the time when entering and exiting stop mode Lp4. 0x13B3: Configure the time to enter and exit standby mode Others: Reserved.

6.4.9. Analog enable register 1 (PWR_ANAKEY 1)

Address offset: 0x28

Reset value: 0x0000 0000

The return value is always 0 When the register is read

It is necessary to write ANAKEY1 as 0x03 and ANAKEY2 as 0x0C to unlock the write protection of registers in ANCTL first.

6.4.10. Analog enable register 2 (PWR_ANAKEY 2)

Address offset: 0x2C

Reset value: 0x0000 0000

The return value is always 0 When the register is read

It is necessary to write ANAKEY1 as 0x03 and ANAKEY2 as 0x0C to unlock the write protection of registers in ANCTL first.

7. Backup registers (BKP)

7.1. Overview

The Backup registers are located in the Backup domain that remains powered-on by VBAT even if VDD power is shut down, they are twenty one 32-bit (84 bytes) registers for data protection of user application data, and the wake-up action from Standby mode or system reset do not affect these registers.

In addition, the BKP registers can be used to implement the tamper detection and RTC calibration function.

After reset, any writing access to the registers in Backup domain is disabled, that is, the Backup registers and RTC cannot be written to access. In order to enable access to the Backup registers and RTC, the Power and Backup interface clocks should be enabled firstly by setting the BDI bit in the RCC_AHBENR2 register, and writing access to the registers in Backup domain should be enabled by setting the DBP bit in the CR0 register.

7.2. Characteristics

- 84 bytes Backup registers which can keep data under power saving mode. If tamper event is detected, Backup registers will be reset.
- The active level of Tamper source (PC13) can be configured.
- RTC Clock Calibration register provides RTC alarm and second output selection, and sets the calibration value.
- Tamper control and status register (BKP_CSR) can control tamper detection with interrupt or event capability.

7.3. Function overview

7.3.1. RTC clock calibration

In order to improve the RTC clock accuracy, the MCU provides the RTC output for calibration function. The RTC clock, or a clock with the frequency is f_{RTCCLK}/64, can be output on the PC13. It is enabled by setting the COO bit in the BKP_RTCCR register.

The calibration value is set by CAL[6:0] in the BKP_RTCCR register, and the calibration function can slow down the RTC clock by steps of 1000000/2^^{20} ppm.

7.3.2. RTC clock calibration

In order to protect the important user data, the MCU provides the tamper detection function, and it can be independently enabled on TAMPER pin by setting corresponding TPE bit in the BKP_CR register. To prevent the tamper event from losing, the edge detection is logically ANDed with the TPE bit, used for tamper detection signal. So the tamper detection configuration should be set before enable TAMPER pin. When the tamper event is detected, the corresponding TEF bit in the BKP_CSR register will be set. Tamper event can generate an interrupt if tamper interrupt is enabled. Any tamper event will reset all Backup data registers.

Note: When TPAL=0/1, if the TAMPER pin is already high/low before it is enabled(by setting TPE bit), an extra tamper event is detected, while there was no rising/falling edge on the TAMPER pin after TPE bit was set.

7.4. Register description

7.4.1. RTC signal output control register (BKP_RTCCR)

Address offset：0x00

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:10	-	R	Reserved
9	ASOS	RW	RTC output selection. 0: RTC alarm pulse is selected as the RTC output 1: RTC second pulse is selected as the RTC output This bit is reset only by a Backup domain reset.
8	ASOE	RW	RTC alarm or second signal output enable 0: Disable RTC alarm or second output 1: Enable RTC alarm or second output When enable, the TAMPER pin will output the RTC output.

Bits	Fields	R/W	Description
			This bit is reset only by a Backup domain reset.
7	COO	RW	RTC clock calibration output enable 0: Disable RTC clock calibration output 1: Enable RTC clock Calibration output When enable, the TAMPER pin will output the RTC clock. ASOE has the priority over COO. When ASOE is set, the TAMPER pin will output the RTC alarm or second signal whether COO is set or not. This bit is reset only by a POR/PDR.
6:0	CAL	RW	RTC clock calibration value The value indicates how many clock pulses are ignored or added every $2^{\wedge 20}$ RTC clock pulses

7.4.2. Tamper pin control register (BKP_CR)

Address offset：0x04

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1	TPAL	RW	TAMPER pin active level 0: The TAMPER pin is active high 1: The TAMPER pin is active low
0	TPE	RW	TAMPER detection enable 0: The TAMPER pin is free for GPIO functions 1: The TAMPER pin is dedicated for the Backup Reset function. The active level on the TAMPER pin resets all data of the BKP_DATAx register.

7.4.3. Tamper control and status register (BKP_CSR)

Address offset：0x08

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:10	-	R	Reserved
9	TIF	R	Tamper interrupt flag 0: No tamper interrupt occurred 1: A tamper interrupt occurred This bit is reset by writing 1 to the CTI bit or the TPIE bit being 0

Bits	Fields	R/W	Description
8	TEF	R	Tamper event flag 0: No tamper event occurred 1: A tamper event occurred This bit is reset by writing 1 to the CTE bit.
7:3	-	R	Reserved
2	TPIE	RW	Tamper interrupt enable 0: Disable the tamper interrupt 1: Enable the tamper interrupt This bit is reset only by a system reset and wake-up from Standby mode.
1	CTI	W	Tamper interrupt reset 0: No effect 1: Reset the TIF bit This bit is always read as 0.
0	CTE	W	Tamper event reset 0: No effect 1: Reset the TEF bit This bit is always read as 0.

7.4.4. Backup data register x (BKP_DATAx) (x= 0..21)

Address offset：0x010 to 0x60

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:0	DATA	RW	Backup data These bits are used for general purpose data storage. The contents of the BKP_DATAx register will remain even if the wake-up action from Standby mode or system reset or power reset.

7.4.5. Backup domain control register (BKP_BDCR)

Address offset：0x100

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15	RTCEN	RW	RTC clock enable control, software set or clear this bit. 0: RTC clock turned off

Bits	Fields	R/W	Description
			1: RTC clock turned on
14:10	-	R	Reserved
9:8	RTCSEL	RW	RTC clock source selection 00: no clock source selected 01: LSE is selected as RTC clock 10: LSI is selected as RTC clock 11: HSE/128 is selected as RTC clock Once RTC clock source is selected, it can not be modified by software until the battery domain reset. User can clear this field by BDRST bit.
7:3	-	R	Reserved
2	LSEBYP	RW	External Low-speed oscillator bypass Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the external 32 kHz oscillator is disabled. 0: LSE oscillator not bypassed 1: LSE oscillator bypassed
1	LSERDY	RW	External low-speed oscillator ready Set and cleared by hardware to indicate when the external 32kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles. 0: External 32 kHz oscillator not ready 1: External 32 kHz oscillator ready
0	LSEON	RW	External low-speed oscillator enable Set and cleared by software. 0: External 32 kHz oscillator OFF 1: External 32 kHz oscillator ON

8. Reset and Clock Control (RCC)

8.1. Reset

8.1.1. System reset

A system reset sets all registers to their default value except the reset flag register (RCC_RSTSTAT) and the registers in battery domain. A system reset is generated when one of the following events occurs:

- A low level on NRST pin (external reset)
- Window watchdog event (WWDG reset)
- Independent watchdog event (IWDG reset)
- A software reset (SW reset)
- Low-power reset (PWR reset)

8.1.1.1. Software reset:

The SYSRESETREQ Bits in Cortex-M3 Application Interrupt and Reset Control Register must be set to force a software reset on the device.

8.1.1.2. Low-power reset:

A low-power reset is generated when the device exits standby mode.

When an external reset occurs, a reset pulse will be generated when NRST pin is on low level.

When an internal reset occurs, the device will generate a low level pulse on NRST pin, this pulse will keep at least 20uS.

8.1.2. Power source reset

A power source reset is generated when one of the following events occurs:

- Power on reset (POR reset)
- Power drop reset (PDR reset)

A power source reset sets all registers to their default value except the registers in battery domain.

8.1.3. Battery domain reset

A battery domain reset sets all RTC core registers, RTC core counters and the registers in battery domain to their default value.

A battery domain reset is generated when one of the following events occurs:

- RCC register RCC_BDRSTR is set to 1 by software.
- VDD or VBAT is powered on after both of them were powered off.

8.2. Clock

8.2.1. System clock source

There are four clock sources can be used to drive the system clock:

- MHSI(8MHz) internal OSC
- FHSI(48MHz) internal OSC
- PLL clock
- HSE external OSC

8.2.2. Sub-clock source

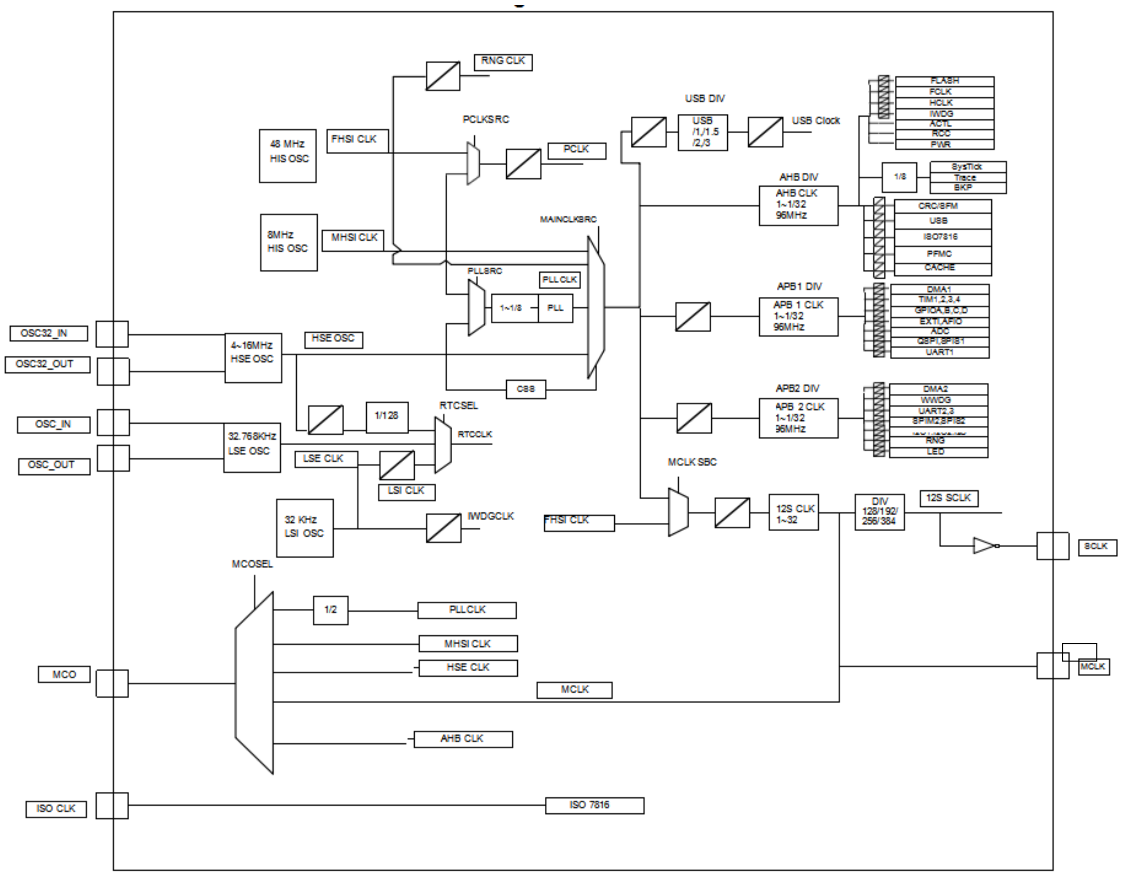
There are two sub-clock sources in device:

- LSI(32KHz) internal low-speed OSC, used to drive IWDG and RTC
- LSE(32.768KHz) external low-speed OSC clock

8.2.3. Clock tree

The clock tree in device is showed below:

Figure 8-1. Clock tree



The system clock “MAINCLK” acts as the clock source of AHB, APB, USB and I2S. The clock source of Cortex SysTick and TRACE is AHB clock divided by 8.

8.2.4. HSE clock

The external high-speed clock (HSE) has two clock sources:

1. External crystal/ceramic resonator

The resonator and load capacitor should be as close as possible to the OSC pins of to minimize output distortion and startup stabilization time.

The loading capacitance value should be adjusted according to the selected oscillator.

2. External clock

OSC_IN pin must be driven by external clock source which has 50% duty cycle.

To use HSE clock, user should set GPIOD[1] and GPIOD[0] to analog mode, then configure the HSE control register in analog control unit (ANCTL). Please refer to the chapters of GPIO and ANCTL.

Note: To use HSE clock, user have to wait until the register "HSESR" is set to 1 by hardware, that means HSE clock is stable. This register is in analog control unit (ANCTL).

8.2.5. HSI clock

The internal high-speed clock (HSI) has two sources:

- 8MHz OSC (MHSI)
- 48MHz OSC (FHSI)

HSI clock can be used as system clock, or clock source for PLL.

MHSI clock can be enabled/disabled through the register "MHSIENR" in ANCTL. FHSI clock can be enabled/disabled through the register "FHSIENR" in ANCTL.

Note: to use MHSI or FHSI clock, user have to wait until the register "MHSISR" or "FHSISR" is set to 1 by hardware, that means MHSI or FHSI clock is stable, these registers are in analog control unit (ANCTL).

8.2.6. PLL clock

PLL is used to generate the system clock

PLL clock source can be the external high-speed clock (HSE) or the internal 8MHz OSC (MHSI). PLL clock source can be divided before input to PLL.

Please refer to the PLL registers in ANCTL for more configurations.

8.2.7. LSE clock

LSE clock can be generated through 32.768KHz external clock, external low-speed crystal or ceramic resonator, it provides a low-power but highly accurate clock to RTC for calendar or other timing functions.

When the external clock is used as LSE clock source, the duty of the external clock must be 50%

8.2.8. LSI clock

LSI clock acts as a low-power clock source which can be kept running in STOP and STANDBY mode for IWDG, the frequency is 32KHz, please refer to the LSI registers in ANCTL.

8.2.9. System clock MAINCLK selection

The system clock is MHSI (8MHz) after reset.

When MHSI is used directly or through PLL as a system clock, it cannot be stopped.

A switch from one clock source to another occurs only when the target source is stable, please refer to the clock source status registers in ANCTL.

8.2.10. Clock Security System CSS

Clock security system can be activated by software, the clock detector is enabled after HSE is stable, and disabled when HSE is stopped.

If a failure is detected on HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to TIM1, and generates an interrupt (CSSI) to MCU.

CSSI is linked to MCU NMI exception vector.

Note: once CSS is enabled and the HSE clock fails interrupt is generated to NMI, in NMI ISR user must clear the CSSI interrupt.

When HSE clock is used directly as the system clock, a detected failure causes a switch of the system clock, MHSI (8MHz) will be used as the system clock automatically and HSE oscillator will be disabled.

When HSE clock is used through PLL as the system clock, a detected failure will also disable PLL

8.2.11. RTC clock

RTC clock source can be HSE, LSE or LSI. Once it has been selected, it can be changed only after battery domain reset.

If HSE is selected as RTC clock, the actual RTC clock frequency is HSE clock divided by

128.

If LSE is selected as RTC clock, RTC continues to work even if VDD supply is off, VBAT supply provides to RTC.

If HSE or LSI is selected as RTC clock, RTC doesn’t work when VDD supply is off.

8.2.12. IWDG clock

If IWDG is enabled, the LSI oscillator is forced on and cannot be disabled. After LSI clock is stable, it is provided to IWDG.

8.2.13. Clock-out capability

Five clocks can be output to MCO pin (PA8) through register configure:

- PLL clock divided by 2
- MHSI clock
- HSE clock
- I2S MCLK clock
- AHB clock

To output a clock on MCO pin, user should set PA[8] to alternate mode.

8.2.14. Bus clock switch and divider

AHB clock comes from the system clock through a divider

APB clock comes from the system clock through a divider, the divide ratio is 1 to 64, and the APB clock can be enable or disable.

User can enable or disable the clock of each peripheral on APB individual.

8.3. RCC registers

8.3.1. PLL pre-divider control register (RCC_PLLPRE)

Address Offset: 0x000

Reset value: 0x0000_0000

This is the PLL input clock pre-divider.

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5	SRCEN	RW	PLL pre-divider input clock enable 0: Disable PLL pre-divider input clock 1: Enable PLL pre-divider input clock
4:1	RATIO	RW	ratio 0000: divided by 2 0001: divided by 3 ...... 1110: divided by 16
0	DIVEN	RW	PLL pre-divider enable 0: PLL pre-divider is disabled 1: PLL pre-divider is enabled

8.3.2. PLL clock source selection register (RCC_PLLSRC)

Address Offset: 0x004

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SRC	RW	PLL clock source selection 0: MHSI(8MHz) as PLL input clock 1: HSE as PLL input clock

8.3.3. System clock source selection register (RCC_MAINCLKSRC)

Address Offset: 0x008

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1:0	SRC	RW	main clock source selection 00: Select MHSI(8MHz) as system clock 01: Select FHSI(48MHz) as system clock 10: Select PLL as system clock 11: Select HSE as system clock

8.3.4. System clock source update enable register (RCC_MAINCLKUEN)

Address Offset: 0x00C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	ENA	W	update main clock source Write 1 to this register causes system clock source switching to a target clock source. This register will be cleared automatically by hardware

8.3.5. USB clock pre-divider control register (RCC_USBPRE)

Address Offset: 0x014

Reset value: 0x0000_0000

Description: Divide the USB input clock

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3	SRCEN	RW	USB pre-divider input clock enable 0: disable USB pre-divider input clock 1: enable USB pre-divider input clock
2:1	RATIO	RW	Ratio 00: divided by 2 10: divided by 1.5 Others: divided by 3
0	DIVEN	RW	USB in clock pre-divider enable 0: USB input clock pre-divider is disabled 1: USB input clock pre-divider is enabled

8.3.6. AHB clock pre-divider register (RCC_AHBPRE)

Address Offset: 0x018

Reset value: 0x0000_0000

Description: Divide AHB input clock

Bits	Fields	R/W	Description
31:7	-	R	Reserved
6:1	RATIO	RW	Ratio 0x00: divided by 2 0x01: divided by 3

			...... 0x3E: divided by 64
0	DIVEN	RW	AHB pre-divider enable 0: AHB clock pre-divider is disabled 1: AHB clock pre-divider is enabled

8.3.7. APB1 clock pre-divider control register (RCC_APB1PRE)

Address Offset: 0x01C

Reset value: 0x0000_0000

Description: Divide APB1 input clock

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	SRCEN	RW	APB1 pre-divider input clock enable 0: disable APB1 pre-divider input clock 1: enable APB1 pre-divider input clock
6:1	RATIO	RW	Ratio 0x00: divided by 2 0x01: divided by 3 ...... 0x3E: divided by 64
0	DIVEN	RW	APB1 pre-divider enable 0: APB1 clock pre-divider is disabled 1: APB1 clock pre-divider is enabled

8.3.8. APB2 clock pre-divider control register (RCC_APB2PRE)

Address Offset: 0x020

Reset value: 0x0000_0000

Description: Divide APB2 input clock

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	SRCEN	RW	APB2 pre-divider input clock enable 0: disable APB2 pre-divider input clock 1: enable APB2 pre-divider input clock
6:1	RATIO	RW	Ratio 0x00: divided by 2

			0x01: divided by 3 ...... 0x3E: divided by 64
0	DIVEN	RW	APB2 pre-divider enable 0: APB2 clock pre-divider is disabled 1: APB2 clock pre-divider is enabled

8.3.9. I2S MCLK clock pre-divider control register (RCC_MCLKPRE)

Address Offset: 0x024

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	SRCEN	RW	I2S MCLK pre-divider input clock enable 0: disable MCLK pre-divider input clock 1: enable MCLK pre-divider input clock
6:1	RATIO	RW	Ratio 0x00: divided by 2 0x01: divided by 3 ...... 0x3E: divided by 64
0	DIVEN	RW	I2S MCLK pre-divider enable (Divider enable) 0: I2S MCLK pre-divider is disabled 1: I2S MCLK pre-divider is enabled

8.3.10. I2S SCLK clock pre-divider control register (RCC_I2SPRE)

Address Offset: 0x028

Reset value: 0x0000_0000

Description: Divide I2S SCLK input clock

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	SRCEN	RW	I2S SCLK pre-divider input clock enable 0: disable SCLK pre-divider input clock 1: enable SCLK pre-divider input clock
9:1	RATIO	RW	Ratio 0x00: divided by 2 0x01: divided by 3

			...... 0x1FE: divided by 512
0	DIVEN	RW	I2S SCLK pre-divider enable 0: I2S SCLK pre-divider is disabled 1: I2S SCLK pre-divider is enabled

8.3.11. I2S MCLK clock source selection register (RCC_MCLKSRC)

Address Offset: 0x02C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SRC	RW	I2S MCLK clock source selection 0: system clock(MAINCLK) as I2S MCLK clock source 1: FHSI(48MHz) clock as I2S MCLK clock source

8.3.12. USB FIFO clock source selection register (RCC_USBFIFOCLKSRC)

Address Offset: 0x034

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SRC	RW	USB FIFO clock source selection 0: AHB clock as USB FIFO clock 1: USB clock as USB FIFO clock

8.3.13. Clock output selection register (RCC_MCOSEL)

Address Offset: 0x038

Reset value: 0x0000_0000

Description: output a clock to MCO pin (PA8), only one Bits is allowed to be set all the time.

For a clock output, must be clear all the Bits in this register first, then write 1 to the Bits corresponding to the clock to output.

Bits	Fields	R/W	Description
31:5	-	R	Reserved

Bits	Fields	R/W	Description
4	MCLK	RW	I2S MCLK output to MCO pin
3	PLLDIV2	RW	PLL clock divided by 2 output to MCO pin
2	MHSI	RW	MHSI(8MHz) output to MCO pin
1	HSE	RW	HSE clock output to MCO pin
0	AHBCLK	RW	AHB clock output to MCO pin

8.3.14. AHB peripherals clock enable register0 (RCC_AHBENR0)

Address Offset: 0x03C

Reset value: 0x0000_007B

Description: enable or disable of the IWDG clock on AHB.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	IWDGEN	RW	IWDG clock enable control 0: IWDG clock is disabled 1: IWDG clock is enabled
1:0	-	R	Reserved

8.3.15. AHB peripherals clock enable register1 (RCC_AHBENR1)

Address Offset: 0x040

Reset value: 0x0000_003D

Description: enable or disable the peripheral’s clock on AHB 1 means enable the peripheral’s clock, 0 means disable

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8	CRCSFMEN	RW	CRC and SFM clock enable
7	DMAC2BREN	RW	DMAC2-APB2 bridge clock enable
6	DMAC1BREN	RW	DMAC1-APB1 bridge clock enable
5	SYSEN	RW	SYS control clock enable
4	CACHEEN	RW	CACHE control clock enable
3	FLASHEN	RW	FLASH control clock enable

2	ISOEN	RW	ISO7816 control clock enable
1	USBEN	RW	USB control clock enable
0	-	R	Reserved

8.3.16. AHB peripherals clock enable register2 (RCC_AHBENR2)

Address Offset: 0x044

Reset value: 0x0000_0003

Description: enable or disable the Battery domain interface clock.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	BDIEN	RW	Battery domain interface clock enable 0: Battery domain interface clock is disabled 1: Battery domain interface clock is enabled
1:0	-	R	Reserved

8.3.17. APB1 peripherals clock enable register (RCC_APB1ENR)

Address Offset: 0x048

Reset value: 0x0000_0000

Description: enable or disable the peripheral’s clock on APB1

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15	BMX1EN	RW	APB1 bus matrix clock enable
14	UART1EN	RW	UART1 control clock enable
13	SPIS1EN	RW	SPIS1 control clock enable
12	QSPIEN	RW	QSPI control clock enable
11	ADCEN	RW	ADC control clock enable
10	AFIOEN	RW	AFIO control clock enable
9	EXTIEN	RW	EXTI control clock enable
8	GPIODEN	RW	GPIOD control clock enable
7	GPIOCEN	RW	GPIOC control clock enable
6	GPIOBEN	RW	GPIOB control clock enable

5	GPIOAEN	RW	GPIOA control clock enable
4	TIM4EN	RW	TIMER4 control clock enable
3	TIM3EN	RW	TIMER3 control clock enable
2	TIM2EN	RW	TIMER2 control clock enable
1	TIM1EN	RW	TIMER1 control clock enable
0	DMAC1EN	RW	DMAC1 control clock enable

Note: BMX1EN Bits must be set for APB1 peripheral's operation.

8.3.18. APB2 peripherals clock enable register (RCC_APB2ENR)

Address Offset: 0x04C

Reset value: 0x0000_0000

Description: enable or disable the peripheral's clock on APB2

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11	BMX2EN	RW	APB2 bus matrix clock enable
10	LEDEN	RW	LED control clock enable
9	RNGEN	RW	RNG control clock enable
8	I2C2EN	RW	I2C2 control clock enable
7	I2C1EN	RW	I2C1 control clock enable
6	I2SEN	RW	I2S control clock enable
5	SPIS2EN	RW	SPIS2 control clock enable
4	SPIM2EN	RW	SPIM2 control clock enable
3	UART3EN	RW	UART3 control clock enable
2	UART2EN	RW	UART2 control clock enable
1	WWDGEN	RW	WWDG clock enable
0	DMAC2EN	RW	DMAC2 control clock enable

Note: BMX2EN Bits must be set for APB2 peripheral's operation.

8.3.19. RNG clock enable register (RCC_RNGCLKENR)

Address Offset: 0x05C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	RNG clock enable 0: RNG clock disabled 1: RNG clock enabled

8.3.20. IWDG clock enable register (RCC_IWDGCLKENR)

Address Offset: 0x064

Reset value: 0x0000_0001

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	DCSSCLKEN	RW	DCSS clock enable 0: DCSS clock disabled 1: DCSS clock enabled
1	-	R	Reserved
0	IWDGCLKEN	RW	IWDG clock enable 0: IWDG clock disabled 1: IWDG clock enabled This Bits has no function once the IWDG is turned on.

8.3.21. USB48MHz clock enable register (RCC_USBCLKENR)

Address Offset: 0x06C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	USB 48MHz clock enable 0: USB 48MHzclock disabled 1: USB 48MHzclock enabled

8.3.22. I2S SCLK clock enable register (RCC_I2SCLKENR)

Address Offset: 0x070

Reset value: 0x0000_0000

Bits	Fields	R/W	Description

31:1	-	R	Reserved
0	CLKEN	RW	I2S SCLK clock enable 0: I2S SCLK clock disabled 1: I2S SCLK clock enabled

8.3.23. SPIS1 clock enable register (RCC_SPIS1CLKENR)

Address Offset: 0x074

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	SPIS1 clock enable 0: SPIS1 clock disabled 1: SPIS1 clock enabled

8.3.24. SPIS2 clock enable register (RCC_SPIS2CLKENR)

Address Offset: 0x078

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	SPIS2 clock enable 0: SPIS2 clock disabled 1: SPIS2 clock enabled

8.3.25. USB FIFO clock enable register (RCC_USBFIFOCLKENR)

Address Offset: 0x07C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	USB FIFO clock enable 0: USB FIFO clock disabled 1: USB FIFO clock enabled

8.3.26. AHB peripheral reset register1 (RCC_AHBRSTR1)

Address Offset: 0x088

Reset value: 0x0000_0000

Description: enable or disable the peripheral’s reset on AHB, ‘1’ means reset the peripheral, ‘0’ means the peripheral’s reset cleared.

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8	CRCSFMRST	RW	CRC and SFM reset control
7:6	-	R	Reserved
5	SYSRST	RW	SYS reset control
4	CACHERST	RW	CACHE reset control
3	FLASHRST	RW	FLASH reset control
2	ISORST	RW	ISO7816 reset control
1	USBRST	RW	USB reset control
0	-	R	Reserved

8.3.27. APB1 peripheral reset register(RCC_APB1RSTR)

Address Offset: 0x090

Reset value: 0x0000_0000

Description: enable or disable the peripheral’s reset on APB1, ‘1’ means reset the peripheral, ‘0’ means the peripheral’s reset cleared.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15	BMX1RST	RW	APB1 bus matrix reset control
14	UART1RST	RW	UART1 reset control
13	SPIS1RST	RW	SPIS1 reset control
12	QSPIRST	RW	QSPI reset control
11	ADCRST	RW	ADC reset control
10	AFIORST	RW	AFIO reset control
9	EXTIRST	RW	EXTI reset control
8	GPIODRST	RW	GPIOD reset control

7	GPIOCRST	RW	GPIOC reset control
6	GPIOBRST	RW	GPIOB reset control
5	GPIOARST	RW	GPIOA reset control
4	TIM4RST	RW	TIMER4 reset control
3	TIM3RST	RW	TIMER3 reset control
2	TIM2RST	RW	TIMER2 reset control
1	TIM1RST	RW	TIMER1 reset control
0	DMAC1RST	RW	DMAC1 reset control

8.3.28. APB2 peripheral reset register (RCC_APB2RSTR)

Address Offset: 0x094

Reset value: 0x0000_0000

Description: enable or disable the peripheral's reset on APB2. '1' means reset the peripheral, '0' means the peripheral's reset cleared.

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11	BMX2RST	RW	APB2 bus matrix reset
10	LEDRST	RW	LED reset control
9	RNGRST	RW	RNG reset control
8	I2C2RST	RW	I2C2 reset control
7	I2C1RST	RW	I2C1 reset control
6	I2SRST	RW	I2S reset control
5	SPIS2RST	RW	SPIS2 reset control
4	SPIM2RST	RW	SPIM2 reset control
3	UART3RST	RW	UART3 reset control
2	UART2RST	RW	UART2 reset control
1	WWDGRST	RW	WWDG reset control
0	DMAC2RST	RW	DMAC2 reset control

8.3.29. I2S SCLK reset register (RCC_I2SCLKRSTR)

Address Offset: 0x0B8

Reset value: 0x0000_0000

Description: write 1 asserts I2S SCLK domain reset, write 0 de-asserts the reset

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SCLKRST	RW	I2S SCLK clock domain reset

8.3.30. Reset flag clear register (RCC_CLRRSTSTAT)

Address Offset: 0x0C8

Reset value: 0x0000_0000

Description: write 1 to this register will clear RCC_RSTSTAT register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR	W	clear RCC_RSTSTAT register

8.3.31. Battery domain reset register (RCC_BDRSTR)

Address Offset: 0x0D4

Reset value: 0x0000_0000

Description: write 1 asserts battery domain reset, write 0 de-asserts the reset

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	BDRST	RW	battery domain reset

8.3.32. LSI battery domain clock enable register (RCC_LSI2RTCENR)

Address Offset: 0x0D8

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLKEN	RW	LSI enable for battery domain 0: disable LSI as battery domain RTC clock 1: enable LSI as battery domain RTC clock

8.3.33. HSE clock 128-divider enable register (RCC_HSE2RTCENR)

Address Offset: 0x0DC

Reset value: 0x0000_0000

Description: When HSE is selected as RTC clock, the divider should be enabled to divide the HSE clock by 128 before input to RTC.

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	DIVEN	RW	128-divider enable 0: disable 128-divider 1: enable 128-divider

8.3.34. Reset flag register (RCC_RSTSTAT)

Address Offset: 0x100

Reset value: 0x0000_0000

Description: reset flag for all the reset occurred, can only cleared by RCC_CLRRSTSTAT SFR.

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5	PINRSTF	R	set by hardware when NRST pin reset occurs, Write 1 to RCC_CLRRSTSTAT clear this Bits.
4	PORRSTF	R	set by hardware when POR/PDR reset occurs, Write 1 to RCC_CLRRSTSTAT clear this Bits.
3	SFTRSTF	R	Set by hardware when software reset occurs, Write 1 to RCC_CLRRSTSTAT clear this Bits.T
2	IWDGRSTF	R	set by hardware when IWDG reset occurs in VDD domain, Write 1 to RCC_CLRRSTSTAT clear this Bits.
1	WWDGRSTF	R	Set by hardware when WWDG reset occurs, Write 1 to RCC_CLRRSTSTAT clear this Bits.
0	LPWRRSTF	R	Set by hardware when low-power reset occurs, Write 1 to RCC_CLRRSTSTAT clear this Bits.

9. General-purpose and alternate-function I/Os (GPIOs and AFIOs)

9.1. Introduction

Each general-purpose I/O port contains five 32-bit configuration registers(GPIOx_MODER, GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_SMIT and GPIOx_PUPDR, two 32-bit data registers（GPIOx_IDR and GPIOx_ODR） and one 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have one 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

9.2. GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Auxiliary configuration mechanism (GPIOx_CFGMSK), which can temporarily freeze I/O port configuration functions
- Alternate function selection registers
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

9.3. GPIO functional description

According to the hardware characteristics of each I/O port listed in the datasheet, each general-purpose I/O (GPIO) port can be individually configured by software in several modes:

- Input pull-up
- Input-pull-down
- Analog
- Schmitt trigger
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.The GPIOx_CFGMSK register assists in register configuration.

Table 9-1. Port bit configuration table

MODE[1:0]	OTYPER	OSPEED[1:0]	PUPD[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP Output	PP
	0		0	1	GP Output	PP + PU
	0		1	0	GP Output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP Output	OD
	1		0	1	GP Output	OD + PU
	1		1	0	GP Output	OD + PD
	1		1	1	Reserved (GP Output OD)	
10	0	SPEED[1:0]	0	0	AF	PP
	0		0	1	AF	PP + PU

MODE[1:0]	OTYPER	OSPEED[1:0]		PUPD[1:0]		I/O configuration	
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
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 (Input Floating)	
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		

9.3.1. General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

The pins listed below are in pull-up/pull-down after reset:

- PA14：pull-up state
- PA13：pull-up state
- PA12：pull-up state
- PA11：pull-down state

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Impedance).The input data register (GPIOx_IDR) captures the data present on the I/O pin every APB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or depending on the value in the GPIOx_PUPDR register.

9.3.2. I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to embedded peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- Debug function: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- System function: MCO pins have to be configured in alternate function mode.
- GPIO: configure the I/O as output, input or analog in the GPIOx_MODER
- Peripheral alternate function：
  - Connect the I/O to the AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
  - Configure the I/O as an alternate function in the GPIOx_MODER register.
- Additional functions:
  - For the ADC, CMPx, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and CMPx

registers.

- For additional functions like RTC_OUT, RTC_TS, RTC_TAMPx, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

9.3.3. I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

9.3.4. I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

9.3.5. I/O data bit handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

There are two control bits in GPIOx_BSRR for each bit in GPIOx_ODR: BS(i) and BR(i).When written to 1, bit BS(i) sets the corresponding ODR(i) bit. When written to 1, bit BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bit-wise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic write access.

9.3.6. GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_ OSPEEDR, GPIOx_PUPDR,GPIOx_CURRENT,GPIOx_SMIT,GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same).

When the LOCK sequence has been applied to the bits, the value of the bits can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bits in the control registers（GPIOx_MODER, GPIOx_OTYPER, GPIOx_ OSPEEDR, GPIOx_PUPDR, GPIOx_CURRENT, GPIOx_SMIT, GPIOx_AFRL and GPIOx_ AFRH）.

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

The configuration auxiliary register GPIOx_CFGMSK function is similar to the GPIOx_LCKR register, which can be used to quickly configure the value of the register. When the specific bit of GPIOx_CFGMSK is set to 1, the corresponding GPIO register bits cannot be written.There are no restrictions on GPIOx_CFGMSK's reading and writing.

9.3.7. I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

Use the GPIOx_AFRL and GPIOx_AFRH multiplexing function registers to multiplex peripheral functions on each GPIO:

Table 9-2. Pin function selection

AFR[3:0]		0	1	2	3	4	5	6	7
PORTA	PA0	WKUP	TIM2_CH1_ETR						UART2_CTS
	PA1		TIM2_CH2						UART2_RTS
	PA2		TIM2_CH3						UART2_TX
	PA3		TIM2_CH4						UART2_RX
	PA4						QSPI_NSS0	SPIS1_NSS	UART2_CK
	PA5						QSPI_SCK	SPIS1_SCK	
	PA6		TIM1_BKIN	TIM3_CH1			QSPI_MI_IO1	SPIS1_SO	
	PA7		TIM1_CH1N	TIM3_CH2			QSPI_MO_IO0	SPIS1_SI	
	PA8	MCO	TIM1_CH1			LED0			UART1_CK
	PA9		TIM1_CH2			LED1			UART1_TX
	PA10		TIM1_CH3			LED2			UART1_RX
	PA11		TIM1_CH4			LED3			UART1_CTS
	PA12		TIM1_ETR						UART1_RTS
	PA13	SWDIO					QSPI_NSS1		
	PA14	SWCLK					QSPI_NSS2		
	PA15		TIM2_CH1_ETR		I2S_WS	I2C1_SMBAI	QSPI_NSS0	SPIS1_NSS	
PORTB	PB0		TIM1_CH2N	TIM3_CH3	I2S_MCLK		QSPI_IO2		
	PB1		TIM1_CH3N	TIM3_CH4			QSPI_IO3		
	PB2	BOOT1							
	PB3	SWO	TIM2_CH2		I2S_SCLK		QSPI_SCK	SPIS1_SCK	
	PB4			TIM3_CH1			QSPI_MI_IO1	SPIS1_SO	

AFR[3:0]		0	1	2	3	4	5	6	7
	PB5			TIM3_CH2	I2S_SD1	I2C1_SMBAI	QSPI_MO_IO0	SPIS1_SI	
	PB6			TIM4_CH1		I2C1_SCL	QSPI_NSS1		UART1_TX
	PB7			TIM4_CH2		I2C1_SDA	SPIM2_NSS1		UART1_RX
	PB8			TIM4_CH3		I2C1_SCL	SPIM2_NSS2		UART1_CTS
	PB9			TIM4_CH4		I2C1_SDA			UART1_RTS
	PB10		TIM2_CH3	TIM4_CH1		I2C2_SCL	QSPI_NSS2		UART3_TX
	PB11		TIM2_CH4			I2C2_SDA	SPIM2_NSS1		UART3_RX
	PB12		TIM1_BKIN		I2S_WS	LED4	SPIM2_NSS0	SPIS2_NSS	UART3_CK
	PB13		TIM1_CH1N		I2S_SCLK	LED5	SPIM2_SCK	SPIS2_SCK	UART3_CTS
	PB14		TIM1_CH2N			LED6	SPIM2_MI	SPIS2_SO	UART3_RTS
	PB15		TIM1_CH3N		I2S_SD0	LED7	SPIM2_MO	SPIS2_SI	
PORTC	PC0				I2S_WS		SPIM2_NSS0	SPIS2_NSS	
	PC1				I2S_SCLK		SPIM2_SCK	SPIS2_SCK	
	PC2				I2S_SD0		SPIM2_MI	SPIS2_SO	
	PC3				I2S_SD1		SPIM2_MO	SPIS2_SI	
	PC4	TRACECK							
	PC5	TRACED0					SPIM2_NSS2		
	PC6			TIM3_CH1	I2S_MCLK	LED0			
	PC7			TIM3_CH2	I2S_MCLK	LED1			
	PC8			TIM3_CH3		LED2			
	PC9	TRACED1		TIM3_CH4		LED3			
	PC10	TRACED2				LED4			UART3_TX
	PC11	TRACED3				LED5			UART3_RX
	PC12			TIM4_ETR		LED6			UART3_CK
	PC13	TAMPER_RTC							
	PC14	OSC32_IN							
	PC15	OSC32_OUT							
PORTD	PD0	OSC_IN							

AFR[3:0]		0	1	2	3	4	5	6	7
	PD1	OSC_OUT							
	PD2			TIM3_ETR		LED7			

9.3.8. External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. Refer to Section 11: Extended interrupts and events controller (EXTI)

9.3.9. Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every APB clock cycle
- A read access to the input data register provides the I/O state

9.3.10. Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Impedance (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
  - The Schmitt trigger input is activated
  - The pull-up and pull-down resistors are activated depending on the value in the

GPIOx_PUPDR register

- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

9.3.11. Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (Output enable and output data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every APB clock cycle
- A read access to the input data register gets the I/O state

9.3.12. Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0)
- The weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

9.3.13. Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the ANCTL or BKP bit in the corresponding register) the oscillator takes control of its associated pins.

When the oscillator is configured in a user external clock mode, only the OSC_IN or OSC32_IN pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

9.3.14. Using the GPIO pins in the backup supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if the GPIO configuration is not bypassed by the RTC configuration, these pins are in input mode.

9.4. GPIO registers

9.4.1. GPIO port mode register（GPIOx_MODER）

Address offset：0x00

Reset value：

0x2A80 0000 (port A)

0x0000 0000 (other ports)

Bits	Fields	R/W	Description
2y+1:2y (y=0..15)	MODERy[1:0]	RW	Port x configuration bits These bits are written by software to configure the I/O mode. 00：Input mode (reset state) 01：General purpose output mode 10：Alternate function mode 11：Analog mode

9.4.2. GPIO port output type register（GPIOx_OTYPER）

Address offset：0x04

Reset value：0x0000 0000

Bits	Fields	R/W	Description

31:16	-	R	Reserved
15:0	OTy	RW	Port x configuration bits These bits are written by software to configure the I/O output type. 0：Output push-pull (reset state) 1：Output open-drain

9.4.3. GPIO port output speed register（GPIOx_OSPEEDR）

Address offset：0x08

Reset value：

0x0F00 0000 (port A)

0x0000 0000 (other ports)

Bits	Fields	R/W	Description
2y+1:2y (y=0..15)	OSPEEDRy	RW	Port x configuration bits These bits are written by software to configure the I/O output speed. 00：High speed（reset state） 01：Low speed Others：Reserved

9.4.4. GPIO port pull-up/pull-down register（GPIOx_PUPDR）

Address offset：0x0C

Reset value：

0x2580 0000 (port A)

0x0000 0000 (other ports)

Bits	Fields	R/W	Description
2y+1:2y (y=0..15)	PUPDRy[1:0]	RW	Port x configuration bits These bits are written by software to configure the I/O pull-up or pull-down 00：No pull-up or pull-down 01：Pull-up 10：Pull-down 11：Reserved

9.4.5. GPIO port input data register（GPIOx_IDR）

Address offset：0x10

Reset value：0xXXXX XXXX

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IDRy	R	Port input data(y = 0..15) These bits are read-only. They contain the input value of the corresponding I/O port.

9.4.6. GPIO port output data register（GPIOx_ODR）

Address offset：0x14

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	ODRy	RW	ALTODR_SEL=0 ,Port output data) (y = 0..15) These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

9.4.7. GPIO port bit set/reset register （GPIOx_BSRR）

Address offset：0x18

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	BRy	W	Port x reset bit y (y= 0..15) These bits are write-only. A read to these bits returns 0x0000. 0：No action on the corresponding ODRx bit 1：Resets the corresponding ODRx bit
15:0	BSy	W	Port x set bit y (y= 0..15) These bits are write-only. A read to these bits returns 0x0000. 0:No action on the corresponding ODRx bit 1:Sets the corresponding ODRx bit

9.4.8. GPIO port configuration lock register（GPIOx_LCKR）

Address offset：0x1C

Reset value：0x0000 0000

Bits	Fields	R/W	Description

31:17	-	R	Reserved
16	LCKK	RW	This bit can be read any time. It can only be modified using the lock key write sequence. 0： Port configuration lock key not active 1：Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset. LOCK key write sequence: WR LCKR[16] = ‘1’ + LCKR[15:0] WR LCKR[16] = ‘0’ + LCKR[15:0] WR LCKR[16] = ‘1’ + LCKR[15:0] RD LCKR RD LCKR[16] = ‘1’（this read operation is optional but it confirms that the lock is active)）
15:0	LCKy	RW	Port x lock bit y (y= 0..15) These bits are read/write but can only be written when the LCKK bit is ‘0. 0：Port configuration not locked 1：Port configuration locked

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.After the first lock sequence on any bit of the port, any read access on the LCKK bit will return ‘1’ until the next MCU reset or peripheral reset.

9.4.9. GPIO alternate function low register（GPIOx_AFRL）

Address offset：0x20

Reset value：0x0000 0000

Bits	Fields	R/W	Description
4y+3:4y (y=0..7)	AFRy[3:0]	RW	Alternate function selection for port x pin y(y =0..7) These bits are written by software to configure alternate function I/Os. AFSELy selection： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8

			1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15

9.4.10. GPIO alternate function high register（GPIOx_AFRH）

Address offset：0x24

Reset value：0x0000 0000

Bits	Fields	R/W	Description
4y- 29 :4y- 32 (y=8..15)	AFRy[3:0]	RW	Alternate function selection for port x pin y(y =8..15) These bits are written by software to configure alternate function I/Os. AFSELy selection： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15

9.4.11. GPIO Port Schmitt Register（GPIOx_SMIT）

Address offset：0x28

Reset value：

0x0000FFFF (port A,B,C)

0x0000 0007(port D)

Bits	Fields	R/W	Description
31:17	-	R	Reserved
15:0	SMITy[1:0]	RW	Port x configuration bits)(y=0..15 These bits are written by software to configure the I/O Schmitt function. 0：Schmidt function closed 1：Schmidt function enable

9.4.12. GPIO Port driver register（GPIOx_CURRENT)

Address offset：0x2C

Reset value：0x0000 0000

Bits	Fields	R/W	Description
2y+1:2y (y=0..15)	CURRENTy [1:0]	RW	Port x configuration bits These bits are written by software to configure the I/O drive current. 00：4mA drive current 01：8mA drive current 10：12mA drive current 11：16mA drive current

9.4.13. GPIO Port configuration auxiliary register（GPIOx_CFGMSK）

Address offset：0x30

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	CFGMSKy	RW	Port x lock bit y(y= 0..15) These bits are read/write bits and can perform write operations. 0：Port configuration is not locked 1：Port configuration is locked

9.5. AFIO registers

9.5.1. External interrupt configuration register 1（AFIO_EXTICR1）

Address offset：0x08

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved

4y+3:4y (y=0..3)	EXTIy[3:0]	RW	EXTI x configuration (x = 0 to 3) These bits are written by software to select the source input for EXTIx external interrupt. 0000：PA[x] pin 0001：PB[x] pin 0010：PC[x] pin 0011：PD[x] pin Others：Reserved

9.5.2. External interrupt configuration register 2（AFIO_EXTICR2）

Address offset：0x0C

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
4y- 13: 4y- 16 (y=4..7)	EXTIy[3:0]	RW	EXTI x configuration (x = 4 to 7) These bits are written by software to select the source input for EXTIx external interrupt. 0000：PA[x] pin 0001：PB[x] pin 0010：PC[x] pin 0011：PD[x] pin Others：Reserved

9.5.3. External interrupt configuration register 3（AFIO_EXTICR3）

Address offset：0x10

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
4y- 29 :4y- 32 (y=8..11)	EXTIy[3:0]	RW	EXTI x configuration (x = 8 to 11) These bits are written by software to select the source input for EXTIx external interrupt. 0000：PA[x] pin 0001：PB[x] pin 0010：PC[x] pin 0011：PD[x] pin Others：Reserved

9.5.4. External interrupt configuration register 4 （AFIO_EXTICR4）

Address offset：0x14

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
4y- 45 :4y- 48 (y=12..15)	EXTIy[3:0]	RW	EEXTI x configuration (x = 12 to 15) These bits are written by software to select the source input for EXTIx external interrupt. 0000：PA[x] pin 0001：PB[x] pin 0010：PC[x] pin 0011：PD[x] pin Others：Reserved

10. Nested vectored interrupt controller (NVIC)

10.1. NVIC Features

Nested vectored interrupt controller contains the following features:

- MG32F10xx has up to 38 mask-able interrupt channels
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

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

10.2. Interrupt and exception vectors

The table below shows the vector table of MG32F10XX devices.

Table 10-1. MG32F10xx vectors

Position	priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-3	fixed	Reset	Reset	0x0000_0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault	All class of fault	0x0000_000C
-	0	settable	MemManage	Memory management	0x0000_0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	-	Reserved	0x0000_001C - 0x0000_002B
-	3	settable	SVCall	System service call via SWI instruction	0x0000_002C

Position	priority	Type of priority	Acronym	Description	Address
-	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
-	-	-	-	Reserved	0x0000_0034
-	5	settable	PendSV	Pendable request for system service	0x0000_0038
-	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FMC	FMC global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMAC1	DMAC1 global interrupt	0x0000_006C
12	19	settable	DMAC2	DMAC2 global interrupt	0x0000_0070
13	20	settable	ADC	ADC global interrupt	0x0000_0074
14	21	settable	USB	USB global interrupt	0x0000_0078
15	22	settable	USB_DMA	USB_DMA global interrupt	0x0000_007C
16	23	settable	EXTI9_5	EXTI Line[9:5] interrupt	0x0000_0080
17	24	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_0084
18	25	settable	TIM1_UP	TIM1 Update interrupt	0x0000_0088
19	26	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_008C
20	27	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_0090
21	28	settable	TIM2	TIM2 global interrupt	0x0000_0094
22	29	settable	TIM3	TIM3 global interrupt	0x0000_0098
23	30	settable	TIM4	TIM4 global interrupt	0x0000_009C
24	31	settable	I2C1	I2C1 event interrupt	0x0000_00A0
25	32	settable	I2C2	I2C2 event interrupt	0x0000_00A4

Position	priority	Type of priority	Acronym	Description	Address
26	33	settable	QSPI	QSPI global interrupt	0x0000_00A8
27	34	settable	SPIM2	SPIM2 global interrupt	0x0000_00AC
28	35	settable	SPIS1	SPIS1 global interrupt	0x0000_00B0
29	36	settable	SPIS2	SPIS2 global interrupt	0x0000_00B4
30	37	settable	UART1	USART1 global interrupt	0x0000_00B8
31	38	settable	UART2	USART2 global interrupt	0x0000_00BC
32	39	settable	UART3	USART3 global interrupt	0x0000_00C0
33	40	settable	EXTI15_10	EXTI Line[15:10] interrupt	0x0000_00C4
34	41	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00C8
35	42	settable	USBP_WKUP	USB PIN wakeup interrupt	0x0000_00CC
36	43	settable	I2S	I2S global interrupt	0x0000_00D0
37	44	settable	ISO	ISO7816 global interrupt	0x0000_00D4

11. Interrupt/event controller (EXTI)

11.1. Overview

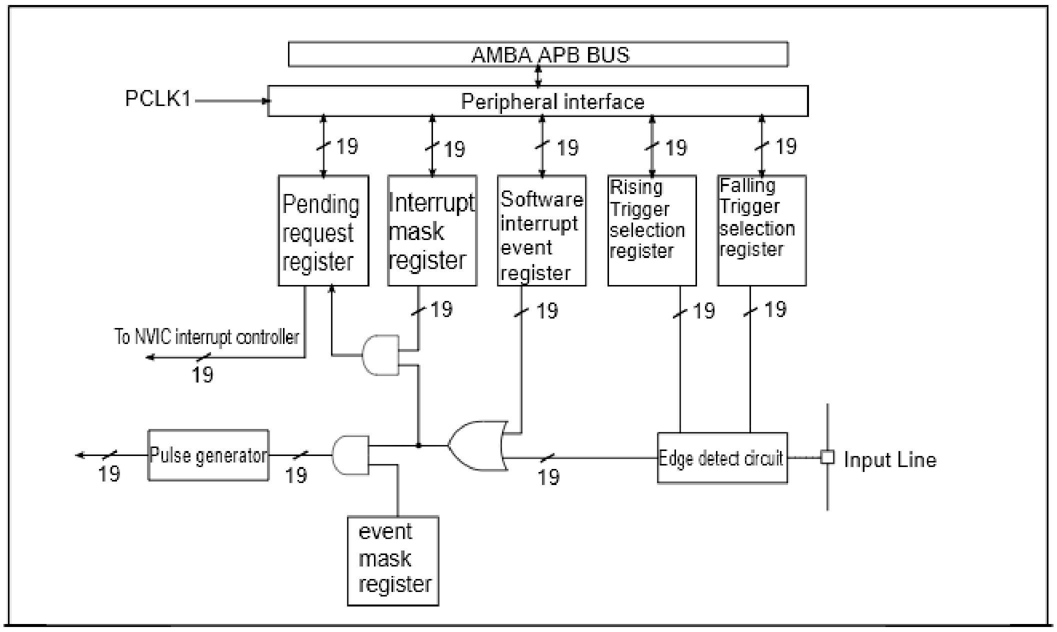
EXTI (interrupt/event controller) contains up to 19 independent edge detectors and generates interrupt requests or events to the processor. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

11.2. Characteristics

- Up to 19 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

11.3. External interrupt and event (EXTI) block diagram

Figure 11-1. Block diagram of EXTI



11.4. External Interrupt and Event function overview

The EXTI contains up to 19 independent edge detectors and generates interrupts request or event to the processor. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

The EXTI trigger source includes 16 external lines from GPIO pins and 4 lines from internal modules (including LVD, RTC Alarm, USB Wakeup).All GPIO pins can be selected as an EXTI trigger source by configuring AFIO_EXTI registers.

EXTI can provide not only interrupts but also event signals to the processor. The Cortex-M3 processor fully implements the Wait FOR Interrupt (WFI), Wait FOR Event (WFE) and the Send Event (SEV) instructions. The Wake-up Interrupt Controller (WIC) enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and event. EXTI can be used to wake up processor and the whole system when some expected event occurs, such as a special GPIO pin toggling or RTC alarm.

Table 11-1. EXTI source

EXTI Line Number	Source
0	PA0/PB0/PC0/PD0
1	PA1/PB1/PC1/PD1
2	PA2/PB2/PC2/PD2
3	PA3/PB3/PC3

4	PA4/PB4/PC4
5	PA5/PB5/PC5
6	PA6/PB6/PC6
7	PA7/PB7/PC7
8	PA8/PB8/PC8
9	PA9/PB9/PC9
10	PA10/PB10/PC10
11	PA11/PB11/PC11
12	PA12/PB12/PC12
13	PA13/PB13/PC13
14	PA14/PB14/PC14
15	PA15/PB15/PC15
16	PVD
17	RTC Alarm
18	USB Wakeup

11.5. Register definition

11.5.1. Interrupt enable register (EXTI_INTEN)

Address offset：0x00

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18:0	MRx	RW	Interrupt enable Bits 0: Interrupt from LINE x is disabled. 1: Interrupt from LINE x is enabled.

11.5.2. Event enable register (EXTI_EMR)

Address offset: 0x04

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:19	-	R	Reserved

18:0	MRx	RW	Event enable Bits 0: Event from LINE x is disabled. 1: Event from LINE x is enabled.

11.5.3. Rising edge trigger enable register (EXTI_RTSR)

Address offset: 0x08

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18:0	TRx	RW	Rising edge trigger enable 0: Rising edge of LINE x is invalid 1: Rising edge of LINE x is valid as an interrupt/event request

11.5.4. Falling edge trigger enable register (EXTI_FTSR)

Address offset: 0x0C

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18:0	TRx	RW	Falling edge trigger enable 0: Falling edge of LINE x is invalid 1: Falling edge of LINE x is valid as an interrupt/event request

11.5.5. Software interrupt event register (EXTI_SWIER)

Address offset: 0x10

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18:0	SWIERx	RW	Interrupt/Event software trigger 0: Deactivate the EXTIx software interrupt/event request 1: Activate the EXTIx software interrupt/event request

11.5.6. Pending register (EXTI_PR)

Address offset: 0x14

Reset value: undefined

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18:0	PRx	RC_W1	Interrupt pending status 0: EXTI LINE x is not triggered 1: EXTI LINE x is triggered. This Bits is cleared to 0 by writing 1 to it.

12. Analog control (ANCTL)

12.1. Analog control introduction

MG32F10xx contains analog macros, such as ADC, CMP, FLASH, PVD, Oscillator, PLL, POR/PDR block. These blocks are all controlled by registers in analog control macro.

12.2. Analog control major features

The major features of analog control are listed below

- Enable/disable analog macros
- Adjust analog macro parameters
- Monitor analog macro status
- Support analog macro debug and testing

12.3. Analog control function description

Because analog macros behavior related to the working performance of chip, analog control registers are all in write protect state by default to prevent mis-operation. A special key value must be written to ANAKEY1 and ANAKEY2 registers in PWR macro to unlock the SFR write protect. After write operation finished, user must clear the value in ANAKEY1 and ANAKEY2 registers to enable the write protect of analog control registers.

Read operation of ANCTL SFR is always allowed.

12.3.1. MHSI control

The internal 8M RC oscillator, MHSI, has a fixed frequency of 8 MHz and is the default clock source selection for the CPU when the device is powered up. The MHSI oscillator provides a lower cost type clock source as no external components are required. The MHSI RC oscillator can be switched on or off using the ANCTL_MHSIENR register.

ANCTL_MHSISR monitor working status of MHSI.

12.3.2. FHSI control

The internal 48M RC oscillator, FHSI, has a fixed frequency of 48 MHz and is the clock source for the Flash Erase/program operation. The FHSI oscillator provides a lower cost type clock source as no external components are required. The FHSI RC oscillator can be switched on or off using the ANCTL_FHSIENR register. ANCTL_FHSISR monitor working status of FHSI.

12.3.3. LSI control

The internal RC oscillator has a frequency of about 32 kHz and is a low power clock source for the Real Time Clock circuit or the IWDG. The LSI offers a low cost clock source as no external components are required. The LSI RC oscillator can be switched on or off by using the ANCTL_LSIENR Register.

ANCTL_LSISR monitor working status of LSI.

12.3.4. HSE control

The high speed external crystal oscillator (HSE), which has a frequency from 4 to 16 MHz, produces a highly accurate clock source for use as the system clock. A crystal with a specific frequency must be connected and located close to the two HSE pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

The external crystal oscillator can be switched on or off using the HSEON bit in the ANCTL_HSECR0 register. The HSERDY flag in ANCTL_HSESR register indicates if the high speed external crystal oscillator is stable.

Select external clock bypass mode by setting the BYPASS bits in the ANCTL_HSECR1 register. The HSE clock is equal to the external clock which drives the OSCIN pin.

The configuration of HSE mode is listed in table 12-1:

Table 12-1. HSE mode configuration

HSE mode	HSEON	BYPASS	PADOEN
external crystal/ceramic resonator	1	0	1
external clock	0	1	0

Note: When HSE is used as system clock source and the clock failure detection function is enabled, When HSE frequency below the threshold frequency, system will generate a NMI and switch to MHSI automatically.

When system clock source is HSE, user should not turn off HSE.

12.3.5. Phase locked loop (PLL)

There is one internal Phase Locked Loop, named PLL.

The PLL can be switched on or off by using the PLLON bit in the ANCTL_PLLENR Register. The PLLRDY flag in the ANCTL_PLLSR Register will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLIE, in the ANCTL_IER Register, is set as the PLL becomes stable.

The PLL is closed by hardware when entering the Deepsleep/Standby mode or XTAL monitor fail when XTAL used as the source clock of the PLL.

The PLL output frequency should not exceed 128MHz, PLL_PRE SFR in RCC macro control the pre-divider of PLL input clock, ANCTL_PLLCR control PLL output frequency multiplier factor.

The typical configurations of PLL are listed in table12-2:

Table 12-2. PLL frequency configuration

HSE Frequency	Pre-divider	Multiplier	Output frequency
8MHz	1	16	128MHz

HSE Frequency	Pre-divider	Multiplier	Output frequency
8MHz	2	12	48MHz
8MHz	2	20	80MHz
8MHz	2	24	96MHz
6MHz	1	16	96MHz
6MHz	3	24	96MHz

12.3.6. PVD control

PVD can monitor the VDD/VDDA power supply by comparing it to a threshold selected by the PLS[2:0] bits in ANCTL_PVDCR register. When PVD event is detected, the event is internally connect to the EXTI line 16 and can generate an interrupt if enabled.

ANCTL_PVDCR is used to configure PVD voltage threshold, refer to table12-3:

Table 12-3. PVD voltage threshold selection

PLS[2:0]	Voltage detect level on falling edge	Voltage detect level on rising edge
3’b000	2.14	2.25
3’b001	2.24	2.35
3’b010	2.34	2.45
3’b011	2.44	2.55
3’b100	2.54	2.65
3’b101	2.64	2.75
3’b110	2.74	2.85
3’b111	2.84	2.95

ANCTL_PVDENR is used to control ON/OFF of PVD.

PVD status can be checked from PWR_SR0 SFR.

12.3.7. SARADC control

The chip contains one 12-bit successive approximation analog-to-digital converter (SARADC) with 16 external input channels. The SARADC converter must be enabled by software before enabling the ADC controller. When ADC is not used, the SARADC should be turned off to save power.

ANCTL_ADCENR register turn ON/OFF the SARADC macro.

12.3.8. Analog Comparator (CMP) control

MG32F10xx embedded two Analog comparators: CMPA and CMPB. Each comparator has a positive input and a negative input, and each comparator input has a four-to-one analog MUX to select input voltage from IOPAD or internal macros.

ANCTL_CMPACR and ANCTL_CMPBCR select the input channel and enable the comparator function. ANCTL_CMPASR and ANCTL_CMPBSR read out the results of comparator.

CMPA input channel selection:

Table 12-4. CMPA negative input selection

NSEL	Channel selection	input pad
0x00	Negative input 0	PA9
0x01	Negative input 1	PB6
0x02	Negative input 2	PB7
0x03	Negative input 3	PA14
Others	Reserved	-

Table 12-5. CMPA positive input selection

PSEL	Channel selection	input pad
0x00	Positive input 0	PA8
0x01	Positive input 1	PB4
0x02	Positive input 2	PB5
0x03	Positive input 3	PA13
Others	Reserved	-

CMPB input channel selection:

Table 12-6. CMPB negative input selection

NSEL	Channel selection	input pad
0x00	Negative input 0	PC12
0x01	Negative input 1	PB8
0x02	Negative input 2	PB9
0x03	Negative input 3	PB3
Others	Reserved	-

Table 12-7. CMPB positive input selection

PSEL	Channel selection	input pad
0x00	Positive input 0	PD2
0x01	Positive input 1	PC10
0x02	Positive input 2	PC11
0x03	Positive input 3	PA15

Others	Reserved	-

12.3.9. CSS control

Clock Security System (CSS) is used to monitor the reliability of the HSE, If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers and an NMI (Non-Maskable Interrupt) is generated to inform the software about the failure. A HSE detected failure causes a switch of the system clock to the MHSI oscillator and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

ANCTL_CSSENR enable the clock detect function. ANCTL_CSSCR must be configured as 0x3.

12.4. Register definition

12.4.1. BG control SFR2 (ANCTL_BGCR2)

Address offset: 0x1C

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1	TEMPOUTEN	RW	Bandgap temperature sensor output enable 0: temperature sensor output disabled 1: temperature sensor output enabled
0	-	R	Reserved

12.4.2. MHSI enable SFR (ANCTL_MHSIENR)

Address offset: 0x2C

Reset Value: 0x0000 0001

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	MHSION	RW	Internal MHSI enable 0: Internal MHSI is turned off 1: Internal MHSI is turned on

12.4.3. MHSI Status SFR (ANCTL_MHSISR)

Address offset: 0x30

Reset Value: 0x0000 0001

Bits	Fields	R/W	Description
31:1	-	R	Reserved

0	MHSIRDY	R	Internal MHSI output status 0: Internal MHSI output not stable 1: Internal MHSI output stable

12.4.4. FHSI enable SFR (ANCTL_FHSIENR)

Address offset: 0x38

Reset Value: 0x0000 0001

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	FHSION	RW	Internal FHSI enable control 0: Internal FHSI disabled 1: Internal FHSI enabled

12.4.5. FHSI Status SFR (ANCTL_FHSISR)

Address offset: 0x3C

Reset Value: 0x0000 0001

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	FHSIRDY	R	Internal FHSI clock output status 0: Internal FHSI clock output not stable 1: Internal FHSI clock output stable

12.4.6. LSI enable SFR (ANCTL_LSIENR)

Address offset: 0x44

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	LSION	RW	Internal LSI enable 0: Internal LSI disabled 1: Internal LSI enabled

12.4.7. LSI Status SFR (ANCTL_LSISR)

Address offset: 0x48

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	LSIRDY	R	Internal LSI clock output Status 0: Internal LSI clock output not stable 1: Internal LSI clock output stable

12.4.8. HSE control SFR 0 (ANCTL_HSECR0)

Address offset: 0x4C

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1	BYPASS	RW	External clock input mode enable 0: External clock input mode disabled 1:External clock input mode enable
0	HSEON	RW	HSE enable 0: HSE disabled 1: HSE enabled

12.4.9. HSE control SFR 1 (ANCTL_HSECR1)

Address offset: 0x50

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1	PADOEN	RW	HSE clock loop back from PAD 0: HSE clock loop back function disabled 1:HSE clock loop back function enabled
0	-	R	Reserved

12.4.10. HSE Status SFR (ANCTL_HSESR)

Address offset: 0x58

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	HSERDY	R	HSE clock output Status 0: HSE clock output not stable 1: HSE clock output stable

12.4.11. PLL control SFR (ANCTL_PLLCR)

Address offset: 0x74

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:6	PLLMUL	RW	PLL multiplier factor 00: PLL input clock x 24

Bits	Fields	R/W	Description
			01: PLL input clock x 20 10: PLL input clock x 16 11: PLL input clock x 12
5:0	-	R	Reserved

Note: The PLL output frequency must not exceed 128MHz.

12.4.12. PLL enable SFR (ANCTL_PLLENR)

Address offset: 0x78

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	PLLON	RW	PLL enable control 0: PLL disabled 1: PLL enabled

12.4.13. PLL Status SFR (ANCTL_PLLSR)

Address offset: 0x7C

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1:0	PLLRDY	R	PLL clock output Status 00: PLL clock output not stable 01: PLL clock output within 90% of target frequency 10: Invalid Status 11: PLL clock output stable

12.4.14. PVD control SFR (ANCTL_PVDCR)

Address offset: 0x80

Reset Value: 0x0000 0004

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	PLS	RW	PVD threshold voltage selection. Please refer to datasheet for detail.

12.4.15. PVD enable SFR (ANCTL_PVDENR)

Address offset: 0x84

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	PVDE	RW	PVD enable control 0: PVD disabled 1: PVD enabled

12.4.16. SARADC enable SFR (ANCTL_SARENR)

Address offset: 0x8C

Reset Value: 0x0000 001E

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SAREN	RW	SARADC enable control 0: SARADC disabled 1: SARADC enabled

12.4.17. POR control SFR (ANCTL_PORCR)

Address offset: 0x94

Reset Value: 0x0000 0841

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11:0	POREN	RW	POR enable control 0x7BE: POR disabled Other value:POR enabled

12.4.18. CMPA control SFR (ANCTL_CMPACR)

Address offset: 0x98

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8	CMPAEN	RW	CMPA enable control 0: CMPA disabled 1: CMPA enabled
7:4	NSEL	RW	CMPA positive input channel selection 0000: select CMPA negative input channel 0 0001: select CMPA negative input channel 1 0010: select CMPA negative input channel 2 0011: select CMPA negative input channel 3 Others: Reserved
3:0	PSEL	RW	CMPA positive input channel selection 0000: select CMPA positive input channel 0 0001: select CMPA positive input channel 1

Bits	Fields	R/W	Description
			0010: select CMPA positive input channel 2 0011: select CMPA positive input channel 3 Others: Reserved

12.4.19. CMPB control SFR (ANCTL_CMPBCR)

Address offset: 0x9C

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8	CMPBEN	RW	CMPB enable control 0: CMPB disabled 1: CMPB enabled
7:4	NSEL	RW	CMPB positive input channel selection 0000: select CMPB negative input channel 0 0001: select CMPB negative input channel 1 0010: select CMPB negative input channel 2 0011: select CMPB negative input channel 3 Others: Reserved
3:0	PSEL	RW	CMPB positive input channel selection 0000: select CMPB positive input channel 0 0001: select CMPB positive input channel 1 0010: select CMPB positive input channel 2 0011: select CMPB positive input channel 3 Others: Reserved

12.4.20. INT Status SFR (ANCTL_ISR)

Address offset: 0xA0

Reset Value: 0x0000 0003

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	CSSIS	R	HSE clock failure detection interrupt flag
6	-	R	Reserved
5	PLLIS	R	PLL clock output stable flag
4	LSEIS	R	LSE clock output stable flag
3	HSEIS	R	HSE clock output stable flag
2	LSIIS	R	LSI clock output stable flag
1	FHSIIS	R	FHSI clock output stable flag
0	MHSIIS	R	MHSI clock output stable flag

12.4.21. INT enable SFR (ANCTL_IER)

Address offset: 0xA4

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5	PLLIE	RW	PLL clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	LSEIE	RW	LSE clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	HSEIE	RW	HSE clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	LSIIE	RW	LSI clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	FHSIIE	RW	FHSI clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	MHSIIE	RW	MHSI clock stable interrupt enable 0: Interrupt disabled 1: Interrupt enabled

Note: HSE clock failure detection (CSS) is non-maskable interrupt.

12.4.22. INT clear SFR (ANCTL_ICR)

Address offset: 0xA8

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	CSSIC	RW	clear HSE clock failure detection interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
6	-	R	Reserved
5	PLLIC	RW	clear PLL clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
4	LSEIC	RW	clear LSE clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
3	HSEIC	RW	clear HSE clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
2	LSIIC	RW	clear LSI clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
1	FHSIIC	RW	clear FHSI clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.
0	MHSIIC	RW	clear MHSI clock output stable interrupt flag Write “1” to the bit clear the flag in ANCTL_ISR SFR.

12.4.23. CMPA Status SFR (ANCTL_CMPASR)

Address offset: 0xAC

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CMPAOUT	R	CMPA comparison result 0: CMP Voltage from positive input is low than that from negative input. 1: CMP Voltage from positive input is higher than that from negative input.

12.4.24. CMPB Status SFR (ANCTL_CMPBSR)

Address offset: 0xB0

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CMPBOUT	R	CMPB comparison result 0: CMP Voltage from positive input is low than that from negative input. 1: CMP Voltage from positive input is higher than that from negative input.

12.4.25. CSS enable SFR (ANCTL_CSSENR)

Address offset: 0xB4

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CSSON	RW	HSE clock failure detection control 0: HSE clock failure detect function disabled 1: HSE clock failure detect function enabled

12.4.26. CSS configuration SFR (ANCTL_CSSCR)

Address offset: 0xB8

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11:0	FREQCNT	RW	Configure CSS count value, must be configured to 0x03

13. Special Function Macro (SFM)

13.1. SFM Introduction

13.1.1. SFM main features

- Count the number of “1” in a WORD (32bit).
- Expand all the bits in a WORD (32bit) by defined rate, the maximum is 8.
- USB port status detection control and interrupt control.

13.2. Function description

13.2.1. Count the number of “1”

This function is used to count the number of “1” in a WORD (32bit). The usage of this function follow the step:

- The EXPEN bit (SFM_CTRL[3]) is set to 0.
- Write the data to be calculated into the SFM_DATA register.
- Read SFM_DOUT0 to get the result.

13.2.2. Expand all the bits in a WORD (32bit)

This function is used to expand all the bits in a WORD (32bit) by defined rate. (Using rate=3 for example):

- The EXPEN bit (SFM_CTRL[3]) is set to 1 and EXPRATE[2:0] is set to 2.
- Write the data to be operated into the SFM_DATA register.
- Read SFM_DOUT0, SFM_DOUT1, SFM_DOUT2 to get the result after expand 3 times operation.

13.2.3. USB port status detection and interrupt control

USB port status detection supports the following functions:

- Detect the USB port status: SE0/SE1/J STATE/K STATE.
- Enable control bit for each USB port status detection function.
- USB port status event is sent to EXTI[18] to wake up the low-power mode.

13.3. Register definition

13.3.1. Control register(SFM_CTRL)

Address offset：0x00

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3	EXPEN	RW	0：Count the number of “1” in the data register. 1：Perform the corresponding expand operation of the data in the data register.
2:0	EXPRATE	RW	When EXPEN=1, this bit field specifies a multiple of the double width. 000：Single width (the result is consistent with the input data) 001：Double width 010：Triple width 011：Four times width 100：Five times width 101：Six times width 110：Seven times width 111：Eight times width

13.3.2. SFM Data register (SFM_DATA)

Address offset：0x04

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:0	DATA	RW	Store the data to be calculated

13.3.3. SFM Result register (SFM_DOUTx, x=0-7)

Address offset：0x08 - 0x24

Reset value：0x0000 0000

Bits	Fields	R/W	Description
31:0	DOUT	R	Store the result when the operation is completed.

13.3.4. USB port status detection control/status register (SFM_USBPSDCSR)

Address offset：0x44

Reset value：0x0000 000

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11	SE1EN	RW	USB port ‘SE1' status detection is enabled. Need to set the DMSTEN and DPSTEN bits in ANCTL_USBPCR.
10	KSTATEN	RW	USB port 'K' status detection is enabled. Need to set the DMSTEN and DPSTEN bits in ANCTL_USBPCR.
9	JSTATEN	RW	USB port 'J' status detection is enabled. Need to set the DMSTEN and DPSTEN bits in ANCTL_USBPCR.
8	SE0EN	RW	USB port 'SE0' status detection is enabled. Need to set the DMSTEN and DPSTEN bits in ANCTL_USBPCR.

Bits	Fields	R/W	Description
7:4	-	R	Received
3	SE1F	RW	When the USB port status is switched to the "SE1" status, it should be set to "1" by the hardware. This bit is cleared to '0' by software.
2	KSTATF	RW	When the status of the USB port is switched to the ‘K’ status (Differential ”0”), it should be set to ‘1’ by the hardware. This bit is cleared to '0' by software.
1	JSTATF	RW	When the USB port status is switched to the ‘J’ status (Differential ”1”), it should be set to ‘1’ by the hardware. This bit is cleared to '0' by software.
0	SE0F	RW	When the state of the USB port is switched to the state of 'SE0', it should be set to '1' by the hardware. This bit is cleared to '0' by software.

13.3.5. USB Port status register (SFM_USBPSTAT)

Address offset：0x48

Reset value：0x0000 0008

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5	FULL_SPEED	R	USB is in full speed mode, high active
4	SUSPEND	R	USB is in SUSPEND state, high active
3	SE1_FLG	R	USB ports are in SE1 state, high active
2	KSTAT_FLG	R	USB ports are in K state, high active
1	JSTAT_FLG	R	USB ports are in J state, high active
0	SE0_FLG	R	USB ports are in SE0 state, high active

14. Direct Memory access Controller (DMAC)

14.1. Overview

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Data can be quickly moved by DMA between peripherals and memory as well as memory and memory without any CPU actions.

There are two DMAC in MG32F10xx: DMAC1 and DMAC2. Each DMAC has 3 hardware channels (six channels totally), and each of them can be configured separately to manage the transfer of 16 DMA requesting sources. All the channels share the AHB bus master bus interface. An arbitration scheme decides which of them is granted by the master bus interface.

The system bus is shared by the DMA controller and the Cortex™-M3 core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system bus bandwidth for the CPU.

14.2. Characteristics

- 2 AHB master interface, 1 AHB slave interface.
- AMBA 2.0-compliant
- 1 AHB slave interface – used to program the EMAC
- 2 AHB master interface(s)
- 6 (3 for each DMAC) independent channels
- 32 (16 for each DMAC) hardware handshake signals
- Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers
- Support for disabling channel without data loss
- Source and destination can be on different AHB layers
- Configurable data bus width for each AHB master interface
- Programmable source and destination for each channel
- Programmable DMA requesting source, priority for hardware handshaking interfaces, DMA handshake interfaces for each channel
- Configurable transfer type, transfer size for each channel
- Support single block DMA transfer
- Support DMAC as the flow controller of DMA transfer
- Support scatter and gather

- Support pseudo fly-by operation
- Support FIFO mode for each channel to improve the bandwidth

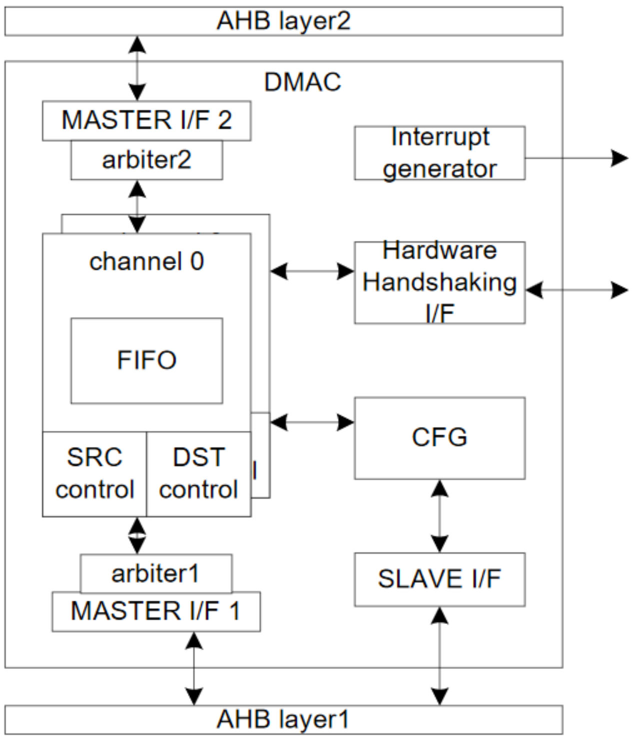
14.3. Pre-Configuration

To use the DMAC, the GPIO and AFIO should be configured correctly if needed. To save the power, user can turn off the peripheral clocks if not need. Before configuring the DMAC1, user should enable the APB1 clock (SFR RCC_APB1PRE) and enable the DMAC clock (bit 15 and bit 0 of SFR RCC_APB1ENR). Before configuring the DMAC2, user should enable the APB2 clock (SFR RCC_APB2PRE) and enable the DMAC clock (bit 15 and bit 0 of SFR RCC_APB2ENR).

14.4. Block Diagram

The figure below shows the block diagram.

Figure 14-1. Block Diagram



The DMAC including six main parts:

- Two AHB master bus interface, to do the DMA data transfer
- One AHB slave bus interface, to configure the DMAC
- One Arbiter for each AHB master interface, to decode which channel is granted
- Three channel controllers including the FIFOs, DMA source control, DMA destination control
- Configuration and interrupt control

- Hardware handshaking interface

14.5. Function Overview

14.5.1. DMA operation

Each DMA transfer consists of two operations, including the loading of data from the source and the storage of the loaded data to the destination. Each DMA transfer includes three operations:

- Read the data from the peripherals or memories based on the SFR SARx
- Write the data to the peripherals or memories based on the SFR DARx
- Update the SFR SARx and DARx when the data transfer finish

User should configure the DMA channels and the handshake interfaces first. If hardware interface is used, the peripherals will send DMA request to DMAC when their DMA events occur. The arbiter decides which channel will be granted. When the channel is granted and the data transfer finish, DMAC will send grant signal to the peripheral. The peripheral will stop requesting the DMA resources when it receive the grant signal, and the DMAC will set the corresponding status and finish the data transfer. The channel will also be disabled automatically. Then the user can configure the DMA channel to other resources.

14.5.2. Arbitration for AHB master interface

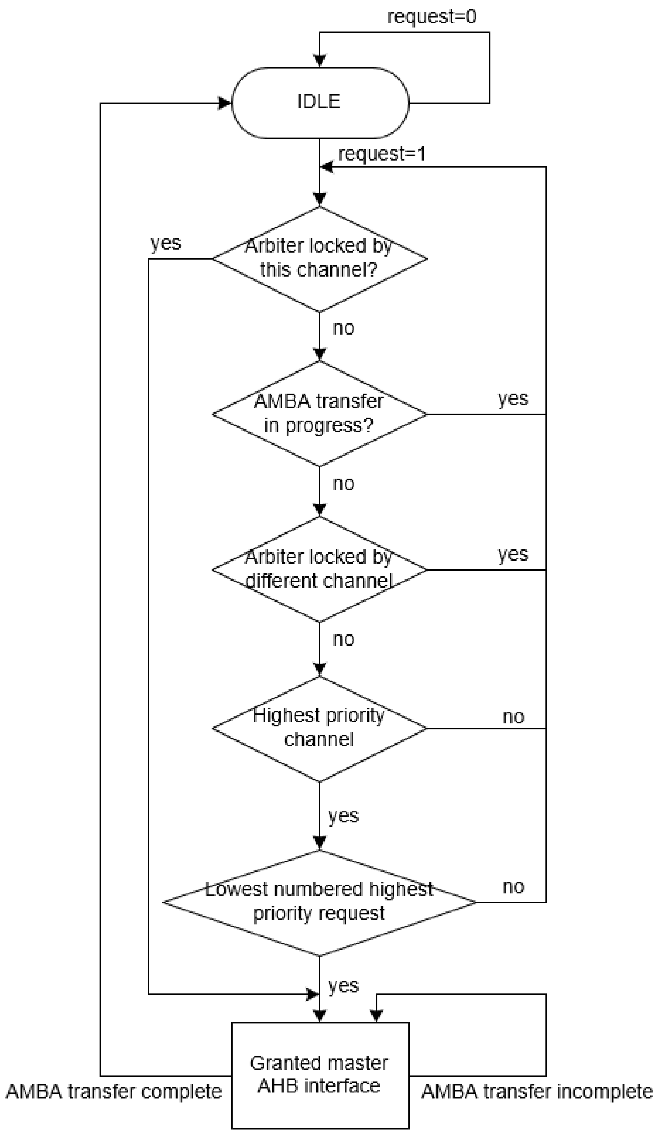
Each DAMC channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines. Source and destination arbitrate separately for the bus.

An arbitration scheme decides which of the request lines is granted the particular master bus interface. Each channel has a programmable priority. A request for the master bus interface can be made at any time, but is granted only after the current AHB transfer (burst or single) has completed. Therefore, if the master interface is transferring data for a lower priority channel and a higher priority channel requests service, then the master interface will complete the current burst for the lower priority channel before switching to transfer data for the higher priority channel.

The following is the interface arbitration scheme employed when no channel has locked (Channel Locking) the arbitration for the master bus interface:

- If only one request line is active at the highest priority level, then the request with the highest priority wins ownership of the AHB master bus interface; it is not necessary for the priority levels to be unique.
- If more than one request is active at the highest requesting priority, then these competing requests proceed to a second tier of arbitration.
- If equal priority requests occur, then the lower-numbered channel is granted. In other words, if a peripheral request attached to Channel 7 and a peripheral request attached to Channel 8 have the same priority, then the peripheral attached to Channel 7 is granted first.

Figure 14-2. Arbitration Flow for Master Bus Interface


request=0
IDLE
request=1
yes
Arbiter locked by this channel?
no
AMBA transfer in progress?
yes
no
Arbiter locked by different channel
yes
no
Highest priority channel
no
yes
Lowest numbered highest priority request
no
yes
Granted master AHB interface
AMBA transfer complete
AMBA transfer incomplete

14.5.3. DMA Handshaking

Handshake interfaces and protocols are needed to control the DMA data transfer starting and finishing. The DMAC supports two kinds of handshake interfaces: software handshake interfaces and hardware handshake interfaces. The user can configure it through the channel SFR CFGLx.HS_SEL_DST and CFGLx.HS_SEL_SRC. Please refer to section 14.5.5 for detail info.

- Software Handshake

When the slave peripheral requires the DMAC to perform a DMA transaction, it communicates this request by sending an interrupt to NVIC. The interrupt service routine then uses the software registers below to initiate and control a DMA transaction. This group of software registers is used to implement the software handshaking interface. The HS_SEL_SRC/HS_SEL_DST bit in the CFGx channel configuration register must be set to enable software handshaking.

The software handshaking registers are:

- ReqSrcReg – source software transaction request
- ReqDstReg – destination software transaction request
- SglReqSrcReg – single source transaction request
- SglReqDstReg – single destination transaction request
- LstSrcReg – last source transaction request
- LstDstReg – last destination transaction request

- Hardware Handshake

Each DMAC has 16 hardware handshake interfaces. Please refer to section 14.5.10 for detail of the interface mapping info. The channel interfaces can be configured through the SFR CFGxL.HS_SEL_SRC, CFGxH.SRC_PER, CFGxL.HS_SEL_DST, CFGxH.DST_PER.

The figure below shows the DMA single transfer waveform and DMA burst transfer waveform.

Figure 14-3. DMA single data transfer

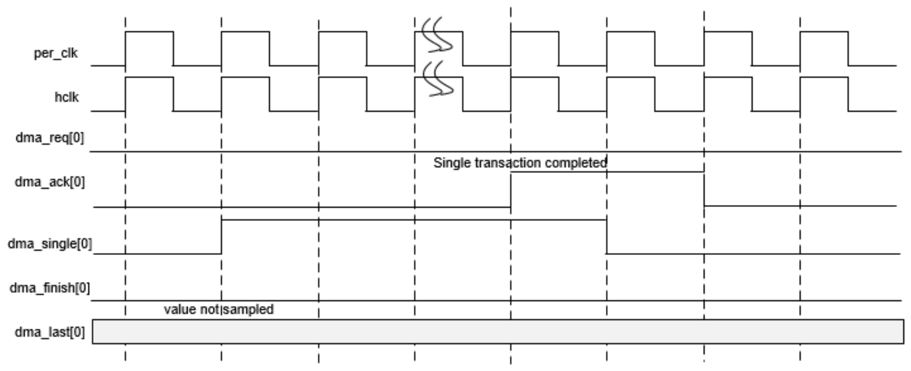
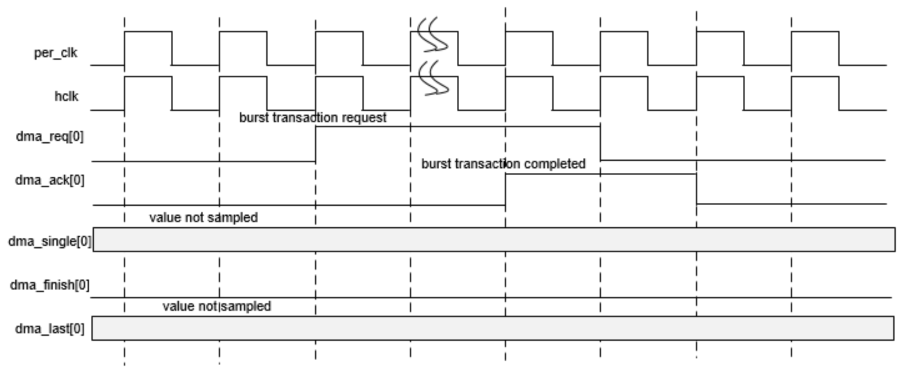


Figure 14-4. DMA burst data transfer



When the peripheral request DMA, the signal “dma_req” will be asserted (assert “dma_single_req” for DMA single transfer). If the channel assigned to this peripheral can be granted by the arbiter, the data transfer starts, and DMAC asserts the signal “dma_ack”. When all the data transfers are finished, the devices will stop requesting by de-assert “dma_req”. Then DMAC will de-assert the “dma_ack” in the next cycle. The DMA transfer size is configured by the channel SFR CTLx.SRC_MSIZE and CTLx.DST_MSIZE.

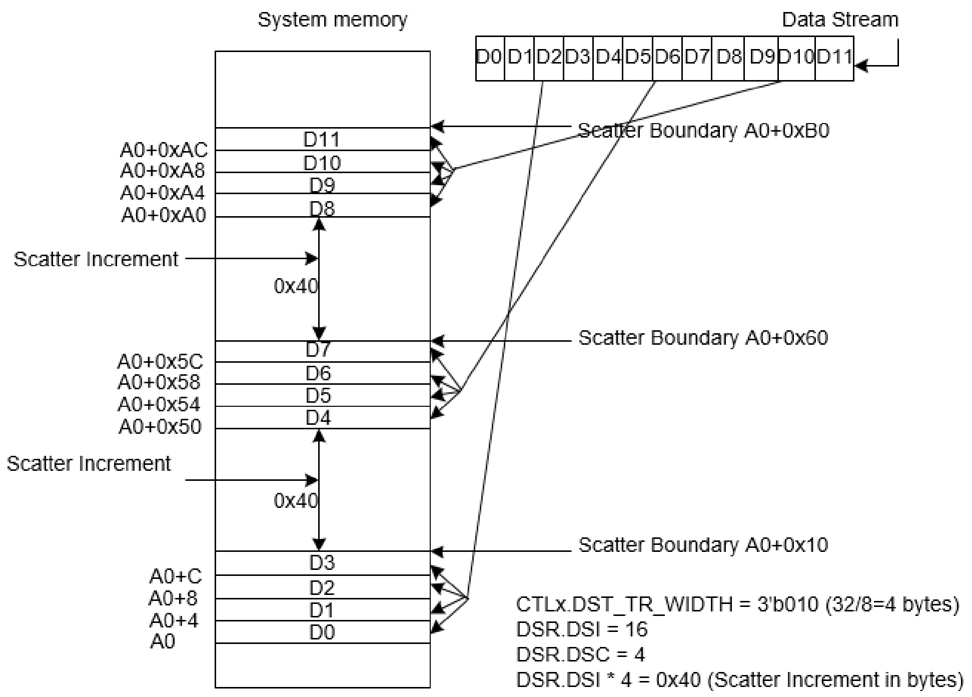
Note: No handshaking interfaces are needed for the data transfer between DMA and SRAM.

14.5.4. Scatter

Scatter is relevant to a destination transfer. The destination address is incremented or decremented by a programmed amount – the scatter increment – when a scatter boundary is reached. Figure 14-5 shows an example destination scatter transfer. The destination address is incremented or decremented by the value stored in the destination scatter increment (DSRx.DSI) field, multiplied by the number of bytes in a single AHB transfer to the destinations (decoded value of CTLx.DST_TR_WIDTH)/8 – when a scatter boundary is reached. The number of destination transfers between successive scatter boundaries is programmed into the Destination Scatter Count (DSC) field of the DSRx register.

Scatter is enabled by writing a 1 to the CTLx.DST_SCATTER_EN field. The CTLx.DINC field determines if the address is incremented, decremented, or remains fixed when a scatter boundary is reached. If the CTLx.DINC field indicates a fixed-address control throughout a DMA transfer, then the CTLx.DST_SCATTER_EN field is ignored, and the scatter feature is automatically disabled.

Figure 14-5. Example of Destination Scatter Transfer

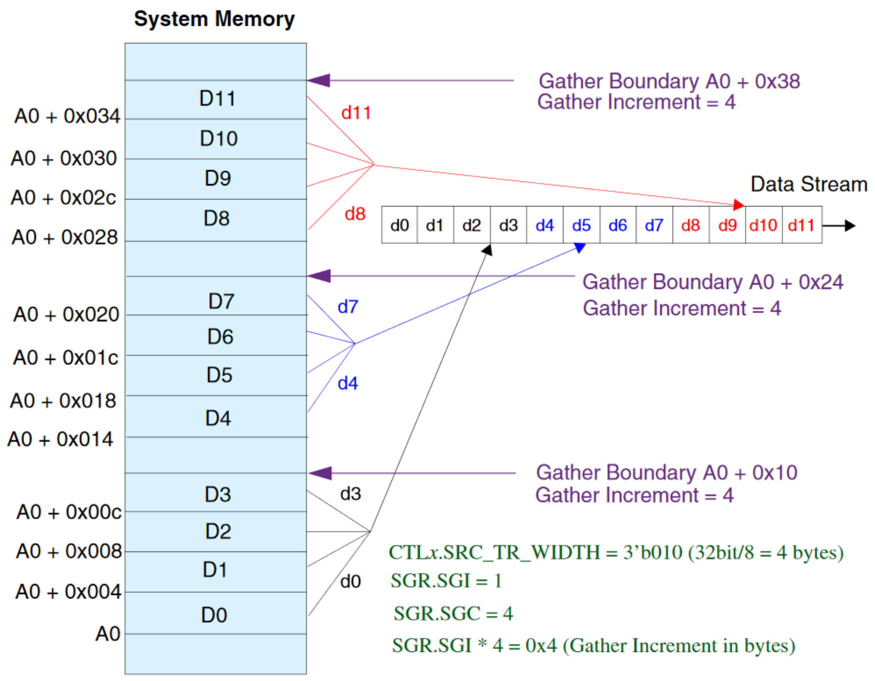


14.5.5. Gather

Gather is relevant to a source transfer. The source address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of source transfers between successive gather boundaries is programmed into the Source Gather Count (SGRx.SGC) field. The source address is incremented or decremented by the value stored in the source gather increment (SGRx.SGI) field, multiplied by the number of bytes in a single AHB transfer from the source – (decoded value of CTLx.SRC_TR_WIDTH)/8 – when a gather boundary is reached.

Gather is enabled by writing a 1 to the CTLx.SRC_GATHER_EN field. The CTLx.SINC field determines if the address is incremented, decremented, or remains fixed when a gather boundary is reached. If the CTLx.SINC field indicates a fixed-address control throughout a DMA transfer, then the CTLx.SRC_GATHER_EN field is ignored, and the gather feature is automatically disabled.

Figure 14-6. Source Gather when SGR.SGI=0x1



In general, if the starting address is A0 and CTLx.SINC = 2'b00 (increment source address control), then the transfer will be:

A0, AO + TWB, A0 + 2*TWB, ... , (A0 + (SGR.SGC-1)*TWB)

<-scatter_increment-> (A0 + (SGR.SGC*TWB) + (SGR.SGI *TWB))

TWB is the transfer width in bytes, decoded value of CTLx.SRC_TR_WIDTH/8 = src_single_size_bytes

14.5.6. AHB Transfer Error Handling

Upon occurrence of an error in an AHB transfer, the following occurs:

- DMA transfer in progress stops immediately
- Relevant channel is disabled
- An interrupt is issued (if not masked)

If multiple channels are enabled, only the one where the AHB error was detected is disabled.

The contents of the FIFO are not cleared, but they become inaccessible and are overwritten once the channel is re-enabled to start a new sequence.

There is no support for automatically resuming the transfer from the point where the error occurred, and the full block transfer has to be re-initiated in order to be successfully completed.

The DMA does not use the hardware handshaking interface to signal the error occurrence in any way, nor does it signal the end of a transfer. In practice, this means that if a request from a peripheral is active when the error occurs—dma_req is high if peripheral is the flow controller; dma_req or dma_single are high if peripheral is not the flow controller—the channel is disabled without the DMA ever asserting dma_ack (or dma_finish).

The hardware handshake interface on the peripheral side has to be re-initiated by the CPU upon detection of the error interrupt. The dma_req signal needs to be brought low before the channel is re-enabled and then brought high when the channel has been enabled.

14.5.7. Disabling a Channel Prior to Transfer Completion

Under normal operation, software enables a channel by writing a1 to the channel enable register, ChEnReg.CH_EN, and hardware disables a channel on transfer completion by clearing the ChEnReg.CH_EN register bit.

The recommended way for software to disable a channel without losing data is to use the CH_SUSP bit in conjunction with the FIFO_EMPTY bit in the Channel Configuration Register (CFGx).

- If software wishes to disable a channel prior to the DMA transfer completion, then it can set the CFGx.CH_SUSP bit to tell the DW_ahb_dmac to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
- Software can now poll the CFGx.FIFO_EMPTY bit until it indicates that the channel FIFO is empty.
- The ChEnReg.CH_EN bit can then be cleared by software once the channel FIFO is empty
- When CTLx.SRC_TR_WIDTH < CTLx.DST_TR_WIDTH and the CFGx.CH_SUSP bit is high, the CFGx.FIFO_EMPTY is asserted once the contents of the FIFO do not permit a single word of CTLx.DST_TR_WIDTH to be formed. However, there may still be data in the channel FIFO, but not enough to form a single transfer of CTLx.DST_TR_WIDTH. In this scenario, once the channel is disabled, the remaining data in the channel FIFO is not transferred to the destination peripheral.

It is permissible to remove the channel from the suspension state by writing a0 to the CFGx.CH_SUSPregister. The DMA transfer completes in the normal manner.

14.5.8. Programming Examples

Below is an example showing how to program DMA.

- Read the Channel Enable register to choose a free (disabled) channel;
- Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.

- Program the following channel registers:
  - Write the starting source address in the SARx register for channel x
  - Write the starting destination address in the DARx register for channel x
  - Write the control information for the DMA transfer in the CTLx register for channel x
    - Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register.
    - Set up the transfer characteristics, such as:
      - Transfer width for the source in the SRC_TR_WIDTH field.
      - Transfer width for the destination in the DST_TR_WIDTH field.
      - Source master layer in the SMS field where the source resides.
      - Destination master layer in the DMS field where the destination resides.
      - Incrementing/decrementing or fixed address for the source in the SINC field.
      - Incrementing/decrementing or fixed address for the destination in the DINC field.
  - Write the channel configuration information into the CFGx register for channel x
    - Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests. Writing a 1 activates the software handshaking interface to handle source and destination requests.
    - If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral; this requires programming the SRC_PER and DEST_PER bits, respectively.
- Ensure that bit 0 of the DmaCfgReg register is enabled before writing to ChEnReg.
- Source and destination request single and burst DMA transactions in order to transfer the block of data (assuming non-memory peripherals). The DW_ahb_dmac acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
- Once the transfer completes, hardware sets the interrupts and disables the channel. At this time, you can respond to either the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled.

14.5.9. Interrupt

Each Channel has five interrupt sources:

- IntBlock – Block Transfer Complete Interrupt
- IntDstTran – Destination Transaction Complete Interrupt
- IntErr – Error Interrupt
- IntSrcTran – Source Transaction Complete Interrupt
- IntTfr – DMA Transfer Complete Interrupt

There are several groups of interrupt-related registers:

- RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr
- StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr
- MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr
- ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr
- StatusInt
- When a channel has been enabled to generate interrupts, the following is true:
- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int_* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.
- The contents of each of the five Status registers is ORed to produce a single bit for each interrupt type in the Combined Status register; that is, StatusInt.

14.5.10. DMA Request Mapping

Both DMAC1 and DMAC2 has 16 hardware handshake interfaces, and each of them are connected to peripherals.

Table 14-1. DMA Request Mapping

Macros	Interface Number	Peripheral DMA requesting source
DMAC1	0	TIM1_CH1 or TIM2_UP or TIM3_CH3
	1	TIM1_CH4 or TIM1_TRIG or TIM1_COM or TIM4_CH2
	2	TIM1_UP or TIM2_CH1 or TIM4_CH3
	3	TIM1_CH3 or TIM3_CH1 or TIM3_TRIG
	4	TIM2_CH3 or TIM_CH1

Macros	Interface Number	Peripheral DMA requesting source
	5	TIM2_CH2 or TIM2_CH4 or TIM4_UP
	6	TIM3_CH4 or TIM3_UP or TIM1_CH2
	7	QSPI RX
	8	QSPI TX
	9	SPIS1 RX
	10	SPIS1 TX
	11	UART1 RX
	12	UART1 TX
	13	ADC regular channel conversion
	14	ADC injected channel conversion
	15	Reserved
DMAC2	0	SPIM2 RX
	1	SPIM2 TX
	2	SPIS2 RX
	3	SPIS2 TX
	4	UART2 RX
	5	UART2 TX
	6	UART3 RX
	7	UART3 TX
	8	I2C1 RX
	9	I2C1 TX
	10	I2C2 RX
	11	I2C2 TX
	12	Reserved
	13	Reserved
	14	Reserved
	15	Reserved

Note 1: Although the peripherals below have DMA requesting signals, but they are not connected to DMAC block. So hardware handshake are not supported. If needed, please use software handshake instead:

- *TIM2_COM and TIM2_TRIG*

- *TIM3_CC2 and TIM3_COM*
- *TIM4_CC4, TIM4_COM and TIM4_TRIG*

Note2: ADC only support single DMA transfer

14.6. DMA Registers

14.6.1. Source Address for Channel x: SARx (x=0,1,2)

Address Offset: 0x0+x*0x58

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:0	SAR	RW	Current Source Address of DMA transfer.

14.6.2. Destination Address for Channel x: DARx (x=0,1,2)

Address Offset: 0x8+x*0x58

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:0	DAR	RW	Current Destination address of DMA transfer.

14.6.3. Channel Control register Low: CTLLx (x=0,1,2)

Address Offset: 0x18+x*0x58

Default Value: 0x00304801

Bits	Fields	R/W	Description
31:27	-	R	Reserved
26:25	SMS	RW	Source Master Select. Identifies the Master Interface layer where the source device (peripheral or memory) resides. Values: 0x0: Source device (peripheral or memory) is accessed from AHB master 1 0x1: Source device (peripheral or memory) is accessed from AHB master 2 0x2,0x3: Reserved
24:23	DMS	RW	Destination Master Select. Identifies the Master Interface layer where the destination device (peripheral or memory) resides. Values: 0x0: Destination device (peripheral or memory) is accessed from AHB master 1 0x1: Destination device (peripheral or memory) is accessed from AHB master 2

Bits	Fields	R/W	Description
			0x2,0x3: Reserved
22:20	TT_FC	RW	Transfer Type and Flow Control. Values: 0x0: Transfer type is Memory to Memory and Flow Controller is DW_ahb_dmac 0x1: Transfer type is Memory to Peripheral and Flow Controller is DW_ahb_dmac 0x2: Transfer type is Peripheral to Memory and Flow Controller is DW_ahb_dmac 0x3: Transfer type is Peripheral to Peripheral and Flow Controller is DW_ahb_dmac 0x4-0x7: Reserved
19	-	R	Reserved
18	DST_SCAT_EN	RW	Destination scatter enable.
17	SRC_GAT_EN	RW	Source gather enable.
16:14	SRC_MSIZE	RW	Source Burst Transaction Length. Number of data items, each of width CTLx.SRC_TR_WIDTH, to be read from the source every time a burst transferred request is made from either the corresponding hardware or software handshaking interface. Values: 0x0: Number of data items to be transferred is 1 0x1: Number of data items to be transferred is 4 0x2: Number of data items to be transferred is 8 0x3: Number of data items to be transferred is 16 0x4: Number of data items to be transferred is 32 0x5: Number of data items to be transferred is 64 0x6: Number of data items to be transferred is 128 0x7: Number of data items to be transferred is 256
13:11	DST_MSIZE	RW	Destination Burst Transaction Length. Number of data items, each of width CTLx.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. Values: 0x0: Number of data items to be transferred is 1 0x1: Number of data items to be transferred is 4 0x2: Number of data items to be transferred is 8 0x3: Number of data items to be transferred is 16 0x4: Number of data items to be transferred is 32 0x5: Number of data items to be transferred is 64 0x6: Number of data items to be transferred is 128 0x7: Number of data items to be transferred is 256

Bits	Fields	R/W	Description
10:9	SINC	RW	Source Address Increment. Values: 0x0: Increments the source address 0x1: Decrements the source address 0x2: No change in the source address 0x3: No change in the source address
8:7	DINC	RW	Destination Address Increment. Values: 0x0: Increments the destination address 0x1: Decrements the destination address 0x2: No change in the destination address 0x3: No change in the destination address
6:4	SRC_TR_WIDTH	RW	Source Transfer Width, Mapped to AHB bus hsize. Values: 0x0: Source transfer width is 8 bits 0x1: Source transfer width is 16 bits 0x2: Source transfer width is 32 bits 0x3: Source transfer width is 64 bits 0x4: Source transfer width is 128 bits 0x5: Source transfer width is 256 bits 0x6: Source transfer width is 256 bits 0x7: Source transfer width is 256 bits
3:1	DST_TR_WIDTH	RW	Destination Transfer Width. Mapped to AHB bus hsize. Values: 0x0: Destination transfer width is 8 bits 0x1: Destination transfer width is 16 bits 0x2: Destination transfer width is 32 bits 0x3: Destination transfer width is 64 bits 0x4: Destination transfer width is 128 bits 0x5: Destination transfer width is 256 bits 0x6: Destination transfer width is 256 bits 0x7: Destination transfer width is 256 bits
0	INT_EN	RW	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; raw* interrupt registers still assert if this bit is 0.

14.6.4. Channel Control register High: CTLHx (x=0,1,2)

Address Offset: 0x1C+x*0x58

Default Value: 0x00000002

Bits	Fields	R/W	Description

Bits	Fields	R/W	Description
31:13	-	R	Reserved
12	DONE	RW	Done bit. Software can poll this bit to see when a block transfer is complete
11:0	BLOCK_TS	RW	Block Transfer Size. When the DMAC is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. Width: The width of single transaction is determined by CTLx.SRC_TR_WIDTH. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller.

14.6.5. Channel Configuration register Low: CFGLx (x=0,1,2)

Address Offset: 0x40+x*0x58

Default Value: 0x00000C00+x*0x20

Bits	Fields	R/W	Description
31:20	-	R	Reserved
19	SRC_HS_POL	RW	Source Handshaking Interface Polarity. Values: 0x0: Source Handshaking Interface Polarity is Active high 0x1: Source Handshaking Interface Polarity is Active low
18	DST_HS_POL	RW	Destination Handshaking Interface Polarity. Values: 0x0: Destination Handshaking Interface Polarity is Active high 0x1: Destination Handshaking Interface Polarity is Active low
17:12	-	R	Reserved
11	HS_SEL_SRC	RW	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces - hardware or software - is active for source requests on this channel. Values: 0x0: Hardware handshaking interface. Software initiated transaction requests are ignored. 0x1: Software handshaking interface. Hardware initiated transaction requests are ignored.
10	HS_SEL_DST	RW	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces - hardware or software - is active for destination requests on this channel.

Bits	Fields	R/W	Description
			Values: 0x0: Hardware handshaking interface. Software initiated transaction requests are ignored. 0x1: Software handshaking interface. Hardware initiated transaction requests are ignored
9	FIFO_EMPTY	RW	Channel FIFO status. Indicates if there is data left in the channel FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel. Values: 0x0: Channel FIFO is not empty 0x1: Channel FIFO is empty
8	CH_SUSP	RW	Channel Suspend. Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data. Values: 0x0: DMA transfer from the source is not suspended 0x1: Suspend DMA transfer from the source
7:5	CH_PRIOR	RW	Channel Priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the range 0 to DMAH_NUM_CHANNELS-1. A programmed value outside this range will cause erroneous behavior. Values: 0x0: Channel priority is 0 0x1: Channel priority is 1 0x2: Channel priority is 2 0x3-0x7: Reserved
4:0	-	R	Reserved

14.6.6. Channel Configuration register High: CFGHx (x=0,1,2)

Address Offset: 0x44+x*0x58

Default Value: 0x00000002

Bits	Fields	R/W	Description
31:15	-	R	Reserved
14:11	DST_PER	RW	Destination hardware interface. The value can be 0-15. NOTE: For correct DMAC operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
10:7	SRC_PER	RW	Source Hardware Interface. The value can be 0-15. NOTE: For correct DMAC operation, only one peripheral (source or

Bits	Fields	R/W	Description
			destination) should be assigned to the same handshaking interface.
6:5	-	R	
4:2	PROTCL	RW	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. Mapping of HPROT bus is as follows: 1'b1 to HPROT[0] CFGx.PROTCTL[1] to HPROT[1] CFGx.PROTCTL[2] to HPROT[2] CFGx.PROTCTL[3] to HPROT[3]
1	FIFO_MODE	RW	FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. Values: 0x0: Space/data available for single AHB transfer of the specified transfer width 0x1: Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer
0	FCMODE	RW	Flow Control Mode. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller. Values: 0x0: Source transaction requests are serviced when they occur. Data pre-fetching is enabled 0x1: Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

14.6.7. Source Gather Register: SGRx (x=0,1,2)

Address Offset: 0x48+x*0x58

Default Value: 0x00000000

Bits	Fields	R/W	Description

31:20	SGC	RW	Source Gather Count. Source contiguous tranfer count between successive gather boundaries.
19:0	SGI	RW	Source Gather Interval.

14.6.8. Destination Scatter Register: DSRx (x=0,1,2)

Address Offset: 0x50+x*0x58

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:20	DSC	RW	Destination Scatter Count. Destination contiguous transfer count between successive scatter boundaries.
19:0	DSI	RW	Destination Scatter Interval.

14.6.9. Interrupt Raw Status Register:

(RawTfr, RawBlock,RawSrcTran,RawDstTran,RawErr)

Address Offset: 0x2C0, 0x2C8, 0x2D0, 0x2D8, 0x2E0

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	RAW2	R	Raw Status for Interrupt, channel 2 Values: 0x0 (INACTIVE): Inactive Raw Interrupt Status 0x1 (ACTIVE): Active Raw Interrupt Status
1	RAW1	R	Raw Status for Interrupt, channel 1 Values: 0x0 (INACTIVE): Inactive Raw Interrupt Status 0x1 (ACTIVE): Active Raw Interrupt Status
0	RAW0	R	Raw Status for Interrupt, channel 0 Values: 0x0 (INACTIVE): Inactive Raw Interrupt Status 0x1 (ACTIVE): Active Raw Interrupt Status

14.6.10. Interrupt Status Register

(StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr)

Address Offset: 0x2E8, 0x2F0, 0x2F8, 0x300, 0x308

Default Value: 0x00000000

Bits	Fields	R/W	Description

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	STAT2	R	Status for Interrupt, channel 2 Values: 0x0 (INACTIVE): Inactive Interrupt Status 0x1 (ACTIVE): Active Interrupt Status
1	STAT1	R	Status for Interrupt, channel 1 Values: 0x0 (INACTIVE): Inactive Interrupt Status 0x1 (ACTIVE): Active Interrupt Status
0	STAT0	R	Status for Interrupt, channel 0 Values: 0x0 (INACTIVE): Inactive Interrupt Status 0x1 (ACTIVE): Active Interrupt Status

14.6.11. Interrupt Mask Register

(MaskTfr, MaskBlock, MaskSrcTran, MaskDstTran, MaskErr)

Address Offset: 0x310, 0x318, 0x320, 0x328, 0x330

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	MSK2_WE	RW	Interrupt Mask Write Enable for channel 2 Values: 0x0 (DISABLED): Interrupt mask write disable 0x1 (ENABLED): Interrupt mask write enable
9	MSK1_WE	RW	Interrupt Mask Write Enable for channel 1 Values: 0x0 (DISABLED): Interrupt mask write disable 0x1 (ENABLED): Interrupt mask write enable
8	MSK0_WE	RW	Interrupt Mask Write Enable for channel 0 Values: 0x0 (DISABLED): Interrupt mask write disable 0x1 (ENABLED): Interrupt mask write enable
7:3	-	R	
2	MSK2	RW	Mask for Interrupt for channel 2 Values: 0x0 (MASK): Mask the interrupts 0x1 (UNMASK): Unmask the interrupts

1	MSK1	RW	Mask for Interrupt for channel 1 Values: 0x0 (MASK): Mask the interrupts 0x1 (UNMASK): Unmask the interrupts
0	MSK0	RW	Mask for Interrupt for channel 0 Values: 0x0 (MASK): Mask the interrupts 0x1 (UNMASK): Unmask the interrupts

14.6.12. Interrupt Clear Register

(ClrTfr, ClrBlock, ClrSrcTran, ClrDstTran, ClrErr)

Address Offset: 0x338, 0x340, 0x348, 0x350, 0x358

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	CLR2	WO	Clear the Interrupt status, channel 2 Values: 0x0 (INACTIVE): No effect 0x1 (ACTIVE): Clears interrupts
1	CLR1	WO	Clear the Interrupt status, channel 1 Values: 0x0 (INACTIVE): No effect 0x1 (ACTIVE): Clears interrupts
0	CLR0	WO	Clear the Interrupt status, channel 0 Values: 0x0 (INACTIVE): No effect 0x1 (ACTIVE): Clears interrupts

14.6.13. Status for each Interrupt type: StatusInt

Address Offset: 0x360

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:5	-	R	Reserved
4	ERR	R	OR of the contents of StatusErr Values: 0x0 (INACTIVE): OR of the contents of StatusErr register is 0 0x1 (ACTIVE): OR of the contents of StatusErr register is 1
3	DSTT	R	OR of the contents of StatusDstTran Values:

			0x0 (INACTIVE): OR of the contents of StatusDstTran register is 0 0x1 (ACTIVE): OR of the contents of StatusDstTran register is 1
2	SRCT	R	OR of the contents of StatusSrcTran Values: 0x0 (INACTIVE): OR of the contents of StatusSrcTran register is 0 0x1 (ACTIVE): OR of the contents of StatusSrcTran register is 1
1	BLOCK	R	OR of the contents of StatusBlock register Values: 0x0 (INACTIVE): OR of the contents of StatusBlock register is 0 0x1 (ACTIVE): OR of the contents of StatusBlock register is 1
0	TFR	R	OR of the contents of StatusTfr register Values: 0x0 (INACTIVE): OR of the contents of StatusTfr register is 0 0x1 (ACTIVE): OR of the contents of StatusTfr register is 1

14.6.14. Source Software Transaction Request register: ReqSrcReg

Address Offset: 0x368

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Source Software Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Source request write Disable 0x1 (ENABLED): Source request write Enable
9	REQ_WE1	RW	Source Software Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Source request write Disable 0x1 (ENABLED): Source request write Enable
8	REQ_WE0	RW	Source Software Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Source request write Disable 0x1 (ENABLED): Source request write Enable
7:3	-	R	
2	SRC_REQ2	RW	Source Software Transaction Request for channel 2 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Source request is active
1	SRC_REQ1	RW	Source Software Transaction Request for channel 1 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Source request is active

Bits	Fields	R/W	Description
0	SRC_REQ0	RW	Source Software Transaction Request for channel 0 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Source request is active

14.6.15. Destination Software Transaction Request register: ReqDstReg

Address Offset: 0x370

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Destination Software Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Destination request write Disable 0x1 (ENABLED): Destination request write Enable
9	REQ_WE1	RW	Destination Software Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Destination request write Disable 0x1 (ENABLED): Destination request write Enable
8	REQ_WE0	RW	Destination Software Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Destination request write Disable 0x1 (ENABLED): Destination request write Enable
7:3	-	R	
2	DST_REQ2	RW	Destination Software Transaction Request for channel 2 Values: 0x0 (INACTIVE): Destination request is not active 0x1 (ACTIVE): Destination request is active
1	DST_REQ1	RW	Destination Software Transaction Request for channel 1 Values: 0x0 (INACTIVE): Destination request is not active 0x1 (ACTIVE): Destination request is active
0	DST_REQ0	RW	Destination Software Transaction Request for channel 0 Values: 0x0 (INACTIVE): Destination request is not active 0x1 (ACTIVE): Destination request is active

14.6.16. Source Single Transaction Request register: SglRqSrcReg

Address Offset: 0x378

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Source Single Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Single write Disable 0x1 (ENABLED): Single write Enable
9	REQ_WE1	RW	Source Single Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Single write Disable 0x1 (ENABLED): Single write Enable
8	REQ_WE0	RW	Source Single Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Single write Disable 0x1 (ENABLED): Single write Enable
7:3	-	R	
2	SRC_SGLREQ2	RW	Source Single Transaction Request for channel 2 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Source request is active
1	SRC_SGLREQ1	RW	Source Single Transaction Request for channel 1 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Single is active
0	SRC_SGLREQ0	RW	Source Single Transaction Request for channel 0 Values: 0x0 (INACTIVE): Source request is not active 0x1 (ACTIVE): Source request is active

14.6.17. Destination Single Transaction Request register: SglRqDstReg

Address Offset: 0x380

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Destination Single Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Destination write Disable 0x1 (ENABLED): Destination write Enable
9	REQ_WE1	RW	Destination Single Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Destination write Disable 0x1 (ENABLED): Destination write Enable
8	REQ_WE0	RW	Destination Single Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Destination write Disable

Bits	Fields	R/W	Description
			0x1 (ENABLED): Destination write Enable
7:3	-	R	
2	DST_SGLREQ2	RW	Destination Single Transaction Request for channel 2 Values: 0x0 (INACTIVE): Destination single or burst request is not active 0x1 (ACTIVE): Destination single or burst request is active
1	DST_SGLREQ1	RW	Destination Single Transaction Request for channel 1 Values: 0x0 (INACTIVE): Destination single or burst request is not active 0x1 (ACTIVE): Destination single or burst request is active
0	DST_SGLREQ0	RW	Destination Single Transaction Request for channel 0 Values: 0x0 (INACTIVE): Destination single or burst request is not active 0x1 (ACTIVE): Destination single or burst request is active

14.6.18. Source Last Transaction Request register: LstSrcReg

Address Offset: 0x388

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Source Last Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Source last transaction request write disable 0x1 (ENABLED): Source last transaction request write enable
9	REQ_WE1	RW	Source Last Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Source last transaction request write disable 0x1 (ENABLED): Source last transaction request write enable
8	REQ_WE0	RW	Source Last Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Source last transaction request write disable 0x1 (ENABLED): Source last transaction request write enable
7:3	-	R	
2	LSTSRC2	RW	Source Last Transaction Request for channel 2 Values: 0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block
1	LSTSRC1	RW	Source Last Transaction Request for channel 1 Values:

Bits	Fields	R/W	Description
			0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block
0	LSTSRC0	RW	Source Last Transaction Request for channel 0 Values: 0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block

14.6.19. Destination Last Transaction Request register: LstDstReg

Address Offset: 0x390

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	REQ_WE2	RW	Destination Last Transaction Request write enable, for channel 2 Values: 0x0 (DISABLED): Destination last transaction request write disable 0x1 (ENABLED): Destination last transaction request write enable
9	REQ_WE1	RW	Destination Last Transaction Request write enable, for channel 1 Values: 0x0 (DISABLED): Destination last transaction request write disable 0x1 (ENABLED): Destination last transaction request write enable
8	REQ_WE0	RW	Destination Last Transaction Request write enable, for channel 0 Values: 0x0 (DISABLED): Destination last transaction request write disable 0x1 (ENABLED): Destination last transaction request write enable
7:3	-	R	
2	LSTDST2	RW	Destination Last Transaction Request for channel 2 Values: 0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block
1	LSTDST1	RW	Destination Last Transaction Request for channel 1 Values: 0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block
0	LSTDST0	RW	Destination Last Transaction Request for channel 0 Values: 0x0 (INACTIVE): Not last transaction in current block 0x1 (ACTIVE): Last transaction in current block

14.6.20. DMA Configuration Register: DmaCfgReg

Address Offset: 0x398

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	DMA_EN	RW	DW_ahb_dmac Enable bit. Values: 0x0 (DISABLED): DMAC Disabled 0x1 (ENABLED): DMAC Enabled

14.6.21. Channel Enable register: ChEnReg

Address Offset: 0x3A0

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:11	-	R	Reserved
10	CHEN_WE2	RW	Channel enable register write enable for channel 2
9	CHEN_WE1	RW	Channel enable register write enable for channel 1
8	CHEN_WE0	RW	Channel enable register write enable for channel 0
7:3	-	R	
2	CH_EN2	RW	Similar to CH_EN0
1	CH_EN1	RW	Similar to CH_EN0
0	CH_EN0	RW	Channel Enable for channel 0. The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. Values: 0x0 (DISABLED): Disable the channel 0x1 (ENABLED): Enable the channel

14.6.22. DMA ID register0: DmaCompsID0

Address Offset: 0x3F8

Default Value: 0x44571110

Bits	Fields	R/W	Description
31:0	COMP_TYPE	R	DMA Component Type Number = `h44571110.

14.6.23. DMA ID register1: DmaCompsID1

Address Offset: 0x3FC

Default Value: 0x00000000

Bits	Fields	R/W	Description
31:0	COMP_VER	R	DMA Component Version

15. Analog to digital conversion (ADC)

15.1. ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 channels and can measure 16 external and 2 internal signal sources. The A/D conversion of various channels can be performed in single, continuous, scan or discontinuous mode. The results of ADC can be stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage exceeds a user-defined high/low threshold.

The ADC input is generated from APB1_CLK divided by a prescaler and it must not exeed 16MHz.

15.2. ADC main features

- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Regular conversion end, injection conversion end, analog watchdog event, FIFO overflow and FIFO empty interrupt
- Single and continuous conversion mode
- Scan mode for automatic conversion of channel 0 to channel N
- Self-calibration
- Data alignment with in-built data coherency
- Channel-by-channel programmable sample time.
- External trigger option for both regular and injected conversion
- Discontinuous mode
- ADC conversion time:
  - The fastest conversion time of 12 bit ADC is 17 ADC clock cycles
- ADC power supply requirements: 2.4V to 3.6V
- ADC measurement range: 0V ≤ VIN ≤ VDDA
- DMA is supported for regular channel conversion.

15.3. ADC functional description

Figure 15.1 is the block diagram of ADC module, and table 15.1 gives the ADC pin description.

Figure 15-1. ADC block diagram

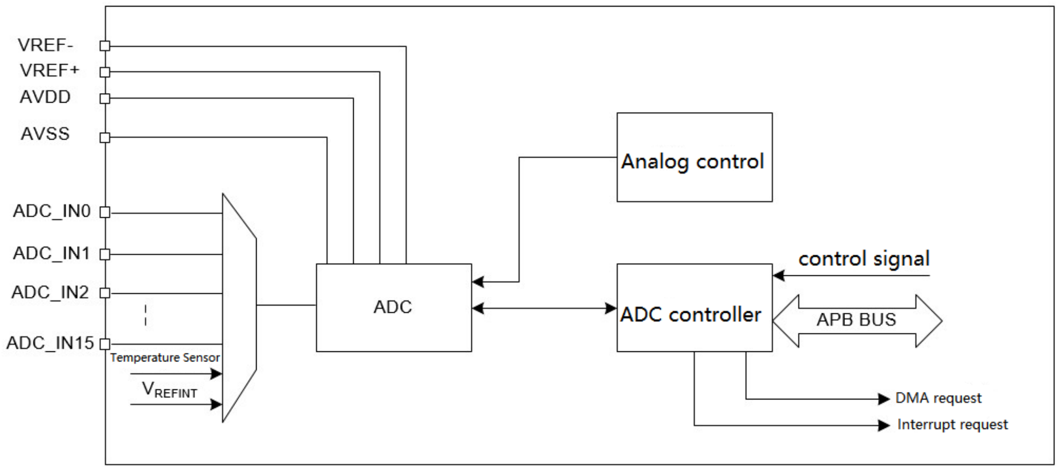


Table 15-1. ADC PINS

Name	Signal type	Remarks
VREF+	Input, analog reference positive	The high / positive reference voltage used by ADC is connected to AVDD
AVDD	Input, analog supply	Analog power supply, 2.4V ≤AVDD≤ 3.6V
VREF-	Input, analog reference negative	The low / negative reference voltage used by ADC is connected to AVSS
AVSS	Input, analog supply ground	Ground for analog power supply equal to VSS
ADC_IN[17:0]	Analog signals	Up to 18 analog channels
ADON	Input, digital signal	ADC enable signal. High active
NPOR	Input, digital signal	ADC power reset signal, low active

Note: The ADC works normally when both ADON=1 and NPOR=1

15.3.1. ADC on-off control

There are two steps to enable ADC conversion

- Enable ADC converter: Setting ADC_CTL0.ADON bit in ANCTL module enables ADC macro. If the ADC is not used, clear the bit to reduce power consumption.
- Enable ADC controller: after ADC is turned on, set ADC_CR2.ADON bit enables the ADC control module. Once the ADC controller is enabled, another write to the ADC_CR2.ADON bit start a new conversion.

15.3.2. ADC clock

There is a programmable prescaler in the ADC controller to generate ADC input clock. The ADC input clock is synchronized with the clock of the ADC controller.

15.3.3. ADC resolution

ADC resolution is fixed to 12-bit.

15.3.4. Channel selection

There are 16 multiplexed channels. The conversions can be organized in two groups: regular group and injected group. A group consists of a sequence of conversions which can be done on any channel and in any order. For instance, it is possible to do the conversion in the following order: Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.

- The regular group is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be configured in the ADC_SQRx registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- The injected group is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be configured in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are modified during a conversion, the current conversion is reset and a new start pulse is sent to the ADC to convert the new chosen group.

Temperature sensor / VREFINT internal channels

The temperature sensor is connected to SARADC channel ADC_IN16, and the internal reference voltage VREFINT is connected to SARADC channel ADC_IN17. These two internal channels can be converted by injection or regular channel.

15.3.5. Single conversion mode

In single conversion mode, ADC only performs one conversion. This mode can be started either by setting the ADON bit of ADC_CR2 register (only applicable to regular channel) or by external trigger (applicable to regular channel or injection channel), when the CONT bit is 0. Once the conversion of the selected channel is completed:

If a regular channel was converted:

- The converted data is stored in the 16-bit ADC_DR register
- The EOC (End Of Conversion) flag is set
- Interrupt is generated if the EOCIE is set.
- The ADC conversion is stopped.

If an injected channel was converted:

- The converted data is stored in the 16-bit ADC_DRJ1 register
- The JEOC (End Of Conversion Injected) flag is set

- Interrupt is generated if the JEOCIE bit is set.
- The ADC conversion is stopped.

15.3.6. Continuous conversion mode

In continuous conversion mode ADC starts another conversion as soon as it finishes one.

This mode is started either by external trigger or by setting the ADON bit in the ADC_CR2, while the CONT bit is 1.

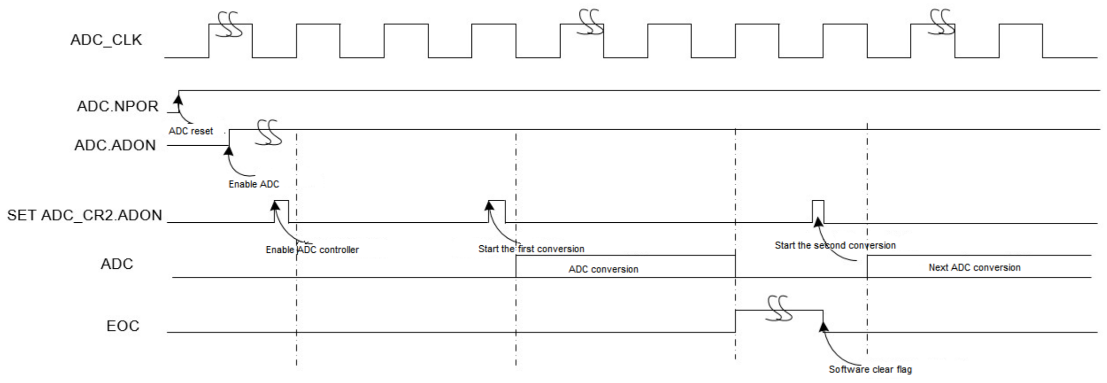
After each conversion:

- If a regular channel was converted:
  - The converted data is stored in the 16-bit ADC_DR register
  - The EOC (End Of Conversion) flag is set
  - An interrupt is generated if the EOCIE is set.
- If an injected channel was converted:
  - The converted data is stored in the 16-bit ADC_DRJ1 register
  - The JEOC (End Of Conversion Injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set.

15.3.7. Timing diagram

As shown in Figure 15.2, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately. After the start of ADC conversion and after 14 clock cycles, the EOC flag is set and the 16-bit ADC Data register contains the result of the conversion.

Figure 15-2. ADC sequence diagram



15.3.8. Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The comparison is done before the alignment.

The analog watchdog can be enabled on one or more channels by configuring the ADC_CR1 register as shown in Table 15.2.

Figure 15-3. ADC simulated watchdog alert area

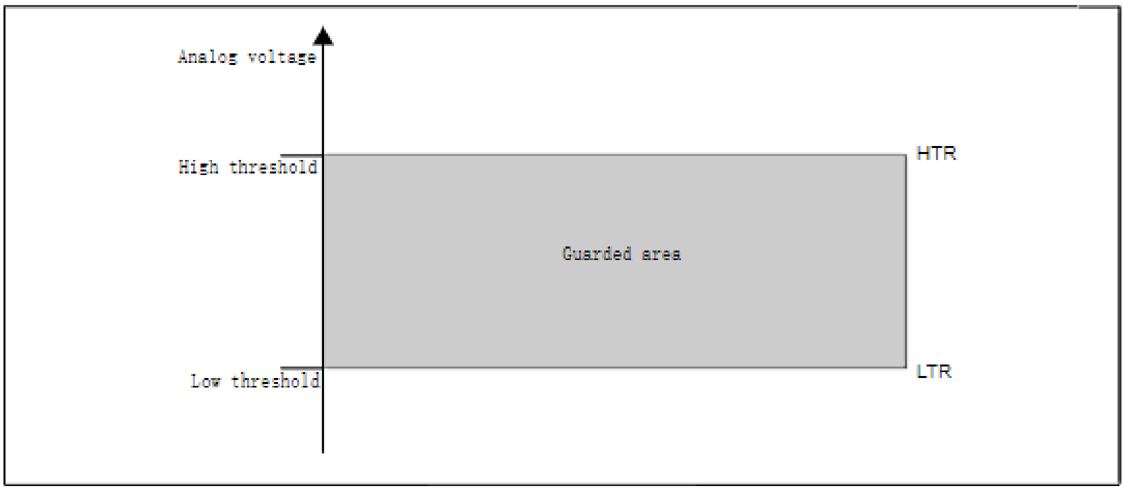


Table 15-2. ADC analog watchdog channel selection

Simulated guard dog alert channel	AWDSGL	AWDEN	JAWDEN
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single injected channel(1)	1	0	1
Singleregular channel(1)	1	1	0
Single regular or injected channel(1)	1	1	1

Note: the single channel is selected by AWDCH [4:0] bits

15.3.9. Scan mode

This mode is used to scan a group of analog channels.

Scan mode can be selected by setting the SCAN bit in the ADC_CR1 register. Once this bit is set, ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR (for injected channels). A single conversion is performed for each channel of the group. After each end of conversion the next channel of the group is converted automatically. If the CONT bit is set, conversion does not stop at the last selected group channel but continues again from the first selected group channel.

When using scan mode, DMA bit must be set and the direct memory access controller is used to transfer the converted data of regular group channels to SRAM after each update of the ADC_DR register.

The injected channel converted data is always stored in the ADC_JDRx registers.

15.3.10. Injected channel management

Triggered injection

To use triggered injection, the JAUTO bit must be cleared and SCAN bit must be set in the ADC_CR1 register.

- Start conversion for a group of regular channels either by external trigger or by setting the ADON bit in the ADC_CR2 register.
- If an external injected trigger occurs during the regular group channel conversion, the current conversion is reset and the injected channel sequence is converted in Scan once mode.
- Then, the regular group channel conversion is resumed from the last interrupted regular conversion. If a regular event occurs during an injected conversion, it doesn’t interrupt it but the regular sequence is executed at the end of the injected sequence.

Note: when using triggered injection conversion, the interval between triggering events must be longer than the injection sequence. For example, the injection sequence length is 28 ADC clock cycles, and the minimum interval between each trigger is 29 ADC clock cycles.

Automatic injection

If the JAUTO bit is set, the injection group channel is automatically converted after the rule group channel. This can be used to convert up to 20 conversion sequences set in the ADC_SQRx and ADC_JSQR registers.

In this mode, external triggering of the injection channel must be disabled.

If the CONT bit is set in addition to the JAUTO bit, the conversion sequence from the regular channel to the injection channel is executed continuously.

Note: automatic injection and discontinuous mode cannot be used at the same time.

15.3.11. Discontinuous mode

Rule group

This mode is activated by setting the DISCEN bit on the ADC_CR1 register. It can be used to perform n conversions of a short sequence (n ≤ 8), which is a part of the conversion sequence selected by ADC_SQRx register. The value n is given by the DISCNUM [2:0] bit of ADC_CR1 register.

An external trigger signal can start the next round of N conversions described in ADC_SQRx register until all conversions of this sequence are completed. The total sequence length is defined by L [3:0] of adcsqr1 register

For example:

N = 3, converted channel = 0, 1, 2, 3, 6, 7, 9, 10

First trigger: converted sequence is 0, 1, 2

Second trigger: converted sequence is 3, 6, 7

Third trigger: the sequence of conversion is 9 and 10, and the EOC event is generated.

The fourth trigger: the sequence of conversion is 0, 1 and 2

Note:

- When converting a rule group in discontinuous mode, it does not automatically start from the beginning after the end of the conversion sequence.
- When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the above example, the fourth trigger re-transforms channels 0, 1, and 2 of the first subgroup.

Injection group

This mode is activated by setting the JDISCEN bit of ADC_CR1 register. After an external trigger event, the mode converts the sequence selected in the ADC_JSQR register one by one in channel order.

An external trigger signal can start the conversion of the next channel sequence selected by ADC_JSQR register until all the conversions in the sequence are completed. The total sequence length is defined by JL [1:0] bit of ADC_JSQR register.

For example:

N = 1, converted channel = 1, 2, 3

First trigger: Channel 1 is converted

Second trigger: Channel 2 is converted

Third trigger: Channel 3 is converted, and EOC and JECO events are generated.

Fourth trigger: Channel 1 is converted.

Note:

- When all injection channels are converted, the next trigger starts the conversion of the first injection channel. In the above example, the fourth trigger re switches the first injection channel 1.
- Automatic injection and discontinuous mode cannot be used at the same time.
- It is necessary to avoid setting up break mode for rule group and injection group at the same time. Discontinuous mode can only be used for one set of transitions.

15.4. Calibration

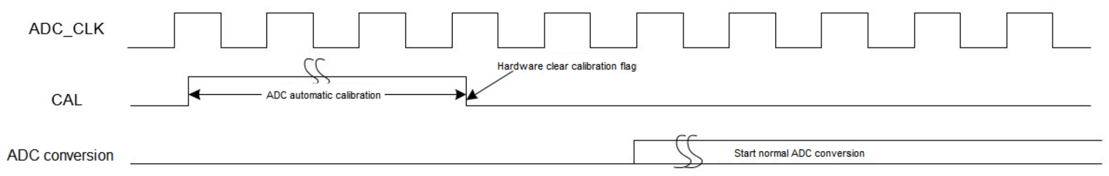
ADC has a built-in self calibration mode. Calibration can greatly reduce the accuracy error caused by the change of internal capacitor bank. During calibration, an error-correction code (digital word) is calculated on each capacitor, which is used to eliminate the error generated on each capacitor in subsequent conversion.

Calibration is started by setting CAL bit of ADC_CR2 register. Once the calibration is finished, the CAL bit is reset by the hardware and normal conversion can be started. It is recommended to perform an ADC calibration at power up. After the calibration phase, the calibration code is stored in ADCCALL and ADCCALH.

Note: it is recommended to perform a calibration after each power up.

Before starting a calibration, the SARADC must been enabled and it must have been in power-on state (ADON bit = '1') for at least two ADC clock cycles.

Figure 15-4. ADC calibration sequence diagram


ADC_CLK
CAL
ADC conversion
ADC automatic calibration
Hardware clear calibration flag
Start normal ADC conversion

15.5. Data alignment

ALIGN bit in the ADC_CR2 register selects the alignment of data stored after conversion.

Data can be left or right aligned as shown in Figure 15-6 and Figure 15-7.

The injected group channels converted data value is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit is the extended sign value.

For regular group channels no offset is subtracted so only twelve bits are significant.

Figure 15-5. Left alignment of data

Injection group

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

Regular group

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

Figure 15-6. Right alignment of data

Injection group

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

Regular group

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

15.6. Channel-by-channel programmable sample time

ADC samples the input voltage for a number of ADC_CLK cycles which can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sample time.

The total conversion time is calculated as follows:

$$T_{conv} = \text{Sampling time} + 12.5 \text{ cycles}$$

Example:

With an ADCCLK = 14 MHz and a sampling time of 1.5 cycles:

$$T_{conv} = 1.5 + 12.5 = 14 \text{ cycles} = 1\ \mu s$$

15.7. Conversion on external trigger

Conversion can be triggered by external events (such as timer capture, EXTI line). If the EXTTRIG control bit is set, the conversion can be triggered by external events. EXTSEL [2:0] and JEXTSEL[2:0] control bits allow the application to select one of the eight possible events to trigger the sampling of rules and injection groups.

Note: when an external trigger signal is selected as ADC rule or injection conversion, only its rising edge can start the conversion.

Table 15-3. External trigger for regular channels

Source	type	EXTSEL[2:0]
TIM1_CC1 event	TIM1 channel 1 interrupt signal	000
TIM1_CC2 event	TIM1 channel 2 interrupt signal	001
TIM1_CC3 event	TIM1 channel 3 interrupt signal	010
TIM2_CC2 event	TIM2 channel 2 interrupt signal	011
TIM3_TRGO event	TIM3 TRGO signal	100
TIM4_CC4 event	TIM4 channel 4 interrupt signal	101
EXTI11	External pin	110
SWSTART	Software control bit	111

Table 15-4. External trigger for injected channels

Source	type	JEXTSEL[2:0]
TIM1_TRGO event	TIM1 TRGO signal	000
TIM1_CC4 event	TIM1 channel 4 interrupt signal	001
TIM2_TRGO event	TIM2 TRGO signal	010
TIM2_CC1 event	TIM2 channel 1 interrupt signal	011
TIM3_CC4 event	TIM3 channel 4 interrupt signal	100
TIM4_TRGO event	TIM4 TRGO signal	101

EXTI15	External pin	110
JSWSTART	Software control bit	111

15.8. DMA request

Regular group DMA request

Because the value of regular channel conversion is stored in a FIFO with a depth of 4, when converting multiple regular channels (the sequence length exceeds 4), DMA is needed to avoid data loss. The register at DMA read is ADC_DR.

DMA requests are generated at the end of a conversion of the rule channel (FIFO is not empty) and the converted data is transferred from the ADC_CR register to the target address specified by the user.

Injection group DMA request

After all the injection group conversions are completed, DMA requests are generated. DMA reads the register ADCJMAR to get the data.

15.9. Temperature sensor

Temperature sensors can be used to measure the temperature (TA) around the device. The temperature sensor is internally connected to the ADC_IN16 input channel, which converts the voltage output of the sensor into a digital value. The recommended sampling time of temperature sensor analog input is 17.1 μs (TBD).

BGCR2.TEMPOUTEN bit of the analog register module must be set to 1 to turn on the temperature sensor. The output voltage of temperature sensor varies linearly with temperature. Due to the change of production process, the deviation of temperature change curve will be different on different chips (the maximum difference is 45 ° C TBD).

The internal temperature sensor is more suitable for detecting the temperature change than measuring the absolute temperature. If accurate temperature measurement is required, an external temperature sensor should be used.

The procedure to use the temperature sensor is as follows:

- select ADC_IN16 input channel
- select the sampling time of 17.1us (TBD)
- set TSVREFE bit of ADC control register 2 (ADC_CR2) to wake up temperature sensor from power down mode
- set the analog register module BGCR2.TEMPOUT to 1
- the temperature is obtained by using the following formula:

$$\text{Temperature } (\cdot\ C) = \{(v25 - VSENSE) / Avg\_Slope\} + 25$$

V25 = VSENSE value at 25 ° C.

Avg_Slope = average slope of temperature vs. VSENSE curve in MV / · C or μ V / · C

Note: there is a build time for the sensor from wake-up in off mode to VSENSE that can *output the correct level. ADC also has a set-up time after power on. Therefore, in order to shorten the delay, it is recommended to turn on both ADC and temperature sensor.*

15.10. ADC interrupts

Two flags are present in the ADC_SR register, but there is no interrupt associated with them:

JSTRT (Start of conversion for injected group channels)

STRT (Start of conversion for regular group channels)

Table 15-5. ADC interrupt

Interrupt event	Event flag	Enable Control bit
Overflow of conversion regular group	OVF	OVFIE
empty of conversion regular group	EMPF	EMPIE
End of conversion regular group	EOC	EOCIE
End of conversion injected group	JEOC	JEOCIE
Analog watchdog status bit is set	AWD	AWDIE

15.11. ADC registers

15.11.1. ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:15	-	R	Reserved, must be kept 0
6	OVF	RC_W0	Regular channel FIFO overflow flag It is set by hardware when FIFO of regular channel overflows and cleared by hardware or software. 0: FIFO of regular channel does not overflow; 1: Regular channel FIFO overflow
5	EMPF	RC_W0	Regular channel FIFO empty flag It is set by hardware when FIFO of regular channel is empty and cleared by hardware or software. 0: there is unread data in FIFO of regular channel; 1: Regular channel FIFO null
4	STRT	RC_W0	Regular channel Start flag This bit is set by hardware when regular channel conversion starts. It is cleared by software. 0: No regular channel conversion started 1: Regular channel conversion has started
3	JSTRT	RC_W0	Injected channel Start flag This bit is set by hardware when injected channel group conversion starts. It is cleared by software. 0: No injected group conversion started 1: Injected group conversion has started
2	JEOC	RC_W0	Injected channel end of conversion This bit is set by hardware at the end of all injected group channel conversion. It is cleared by software. 0: Conversion is not complete

Bits	Fields	R/W	Description
			1: Conversion complete
1	EOC	RC_W0	End of conversion This bit is set by hardware at the end of a group channel conversion (regular or injected). It is cleared by software or by reading the ADC_DR. 0: Conversion is not complete 1: Conversion complete
0	AWD	RC_W0	Analog watchdog flag This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software. 0: No Analog watchdog event occurred 1: Analog watchdog event occurred

15.11.2. ADC control register 1(ADC_CR1)

Address offset: 0x04

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:24	-	R	Reserved, must be kept 0
23	AWDEN	RW	Analog watchdog enable on regular channels This bit is set/reset by software. 0: Analog watchdog disabled on regular channels 1: Analog watchdog enabled on regular channels
22	JAWDEN	RW	Analog watchdog enable on injected channels This bit is set/reset by software. 0: Analog watchdog disabled on injected channels 1: Analog watchdog enabled on injected channels
21:16	-	R	Reserved, must be kept 0.
15:13	DISCNUM[2:0]	RW	Discontinuous mode channel count These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger. 000: 1 channel 001: 2 channels .... 111: 8 channels
12	JDISCEN	RW	Discontinuous mode on injected channels This bit set and cleared by software to enable/disable discontinuous mode on injected group channels

Bits	Fields	R/W	Description
			0: Discontinuous mode on injected channels disabled 1: Discontinuous mode on injected channels enabled
11	DISCEN	RW	Discontinuous mode on regular channels This bit set and cleared by software to enable/disable Discontinuous mode on regular channels. 0: Discontinuous mode on regular channels disabled 1: Discontinuous mode on regular channels enabled
10	JAUTO	RW	Automatic Injected Group conversion This bit set and cleared by software to enable/disable automatic injected group conversion after regular group conversion. 0: Automatic injected group conversion disabled 1: Automatic injected group conversion enabled
9	AWDSGL	RW	Enable the watchdog on a single channel in scan mode This bit set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits. 0: Analog watchdog enabled on all channels 1: Analog watchdog enabled on a single channel
8	SCAN	RW	Scan mode This bit is set and cleared by software to enable/disable Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted. 0: Scan mode disabled 1: Scan mode enabled Note: An EOC or JEOC interrupt is generated only on the end of conversion of the last channel if the corresponding EOCIE or JEOCIE bit is set
7	JEOCIE	RW	Interrupt enable for injected channels This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels. 0: JEOC interrupt disabled 1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.
6	AWDIE	RW	Analog watchdog interrupt enable This bit is set and cleared by software to enable/disable the analog watchdog interrupt. 0: Analog watchdog interrupt disabled 1: Analog watchdog interrupt enabled
5	EOCIE	RW	Interrupt enable for EOC This bit is set and cleared by software to enable/disable the End of Conversion interrupt. 0: EOC interrupt disabled

Bits	Fields	R/W	Description
			1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set
4:0	AWDCH[4:0]	RW	Analog watchdog channel select bits These bits are set and cleared by software. They select the input channel to be guarded by the Analog watchdog. 00000: ADC analog Channel0 00001: ADC analog Channel1 .... 01111: ADC analog Channel15 10000: ADC analog Channel16 10001: ADC analog Channel17 Other values Reserved

15.11.3. ADC control register 2(ADC_CR2)

Address offset: 0x08

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:24	-	R	Reserved, must be kept 0
23	TSVREFE	RW	Temperature sensor and VREFINT enable This bit is set and cleared by software to enable/disable the temperature sensor and VREFINT channel. 0: Temperature sensor and VREFINT channel disabled 1: Temperature sensor and VREFINT channel enabled
22	SWSTART	RW	Start conversion of regular channels This bit is set by software to start conversion and cleared by hardware as soon as conversion starts. It starts a conversion of a group of regular channels if SWSTART is selected as trigger event by the EXTSEL[2:0] bits. 0: Reset state 1: Starts conversion of regular channels
21	JSWSTART	RW	Start conversion of injected channels This bit is set by software and cleared by software or by hardware as soon as the conversion starts. It starts a conversion of a group of injected channels (if JSWSTART is selected as trigger event by the JEXTSEL[2:0] bits. 0: Reset state 1: Starts conversion of injected channels
20	EXTTRIG	RW	External trigger conversion mode for regular channels This bit is set and cleared by software to enable/disable the external trigger used to start conversion of a regular channel group.

Bits	Fields	R/W	Description
			0: Conversion on external event disabled 1: Conversion on external event enabled
19:17	EXTSEL[2:0]	RW	External event select for regular group These bits select the external event used to trigger the start of conversion of a regular group: the assigned triggers are: 000: Timer 1 CC1 event 001: Timer 1 CC2 event 010: Timer 1 CC3 event 011: Timer 2 CC2 event 100: Timer 3 TRGO event 101: Timer 4 CC4 event 110: EXTI line 1 event 111: SWSTART
16	-	R	Reserved, must be kept 0
15	JEXTTRIG	RW	External trigger conversion mode for injected channels This bit is set and cleared by software to enable/disable the external trigger used to start conversion of an injected channel group. 0: Conversion on external event disabled 1: Conversion on external event enabled
14:12	JEXTSEL[2:0]	RW	External event select for injected group These bits select the external event used to trigger the start of conversion of an injected group: the assigned triggers are: 000: Timer 1 TRGO event 001: Timer 1 CC4 event 010: Timer 2 TRGO event 011: Timer 2 CC1 event 100: Timer 3 CC4 event 101: Timer 4 TRGO event 110: EXTI line15 111: JSWSTART
11	ALIGN	RW	Data alignment This bit is set and cleared by software. Refer to Figure 15.5 15.6 15.7 15.8 0: Right Alignment 1: Left Alignment
10	-	R	Reserved, must be kept 0
9	JDMAEN	RW	DMA: Direct memory access mode This bit is set and cleared by software. Refer to the DMA controller

Bits	Fields	R/W	Description
			chapter for more details. 0: DMA mode disabled 1: DMA mode enabled
8	DMAEN	RW	Regular group DMA mode This bit is set and cleared by software. See DMA controller for details. 0: do not use DMA mode; 1: Use DMA mode.
7:4	-	R	Reserved, must be kept 0
3	RSTCAL	RW	Reset calibration This bit is set by software and cleared by hardware. It is cleared after the calibration registers are initialized. 0: Calibration register initialized. 1: Initialize calibration register. Note: If RSTCAL is set when conversion is ongoing, additional cycles are required to clear the calibration registers.
2	CAL	RW	CAL: A/D Calibration This bit is set by software to start the calibration. It is reset by hardware after calibration is complete. 0: Calibration completed 1: Enable calibration
1	CONT	RW	CONT: Continuous conversion This bit is set and cleared by software. If set conversion takes place continuously till this bit is reset. 0: Single conversion mode 1: Continuous conversion mode
0	ADON	RW	ADON: A/D converter ON / OFF This bit is set and cleared by software. If this bit holds a value of zero and a 1 is written to it then it wakes up the ADC from Power Down state. Conversion starts when this bit holds a value of 1 and a 1 is written to it. The application should allow a delay of t_{STAB} between power up and start of conversion. Refer to Figure 23. 0: Disable ADC conversion/calibration and go to power down mode. 1: Enable ADC and to start conversion Note: If any other bit in this register apart from ADON is changed at the same time, then conversion is not triggered. This is to prevent triggering an erroneous conversion.

15.11.4. ADC sample time register 1(ADC_SMPR1)

Address offset: 0x0C

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:24	-	R	Reserved, must be kept 0
23:0	SMPx[2:0]	RW	Channel x Sample time selection These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged. 000: 1.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles 100: 41.5 cycles 101: 55.5 cycles 110: 71.5 cycles 111: 239.5 cycles

15.11.5. ADC sample time register2(ADC_SMPR2)

Address offset: 0x10

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:30	-	R	Reserved, must be kept 0
29:0	SMPx[2:0]	RW	Channel x Sample time selection These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged. 000: 1.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles 100: 41.5 cycles 101: 55.5 cycles 110: 71.5 cycles 111: 239.5 cycles

15.11.6. ADC injected channel data offset register x (ADC_JOFRx）(x =1..4)

Address offset: 0x14 - 0x20

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:12	-	R	Reserved, must be kept 0

Bits	Fields	R/W	Description
11:0	JOFFSETx[11:0]	RW	Data offset for injected channel x These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDRx registers.

15.11.7. ADC watchdog high threshold register (ADC_HTR)

Address offset: 0x24

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:12	-	R	Reserved, must be kept 0
11:0	HT[10:0]	RW	Analog watchdog high threshold These bits are written by software to define the high threshold for the analog watchdog.

15.11.8. ADC watchdog low threshold register (ADC_LTR)

Address offset: 0x28

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:12	-	R	Reserved, must be kept 0
11:0	LT[10:0]	RW	Analog watchdog low threshold These bits are written by software to define the low threshold for the analog watchdog.

15.11.9. ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:24	-	R	Reserved, must be kept 0
23:20	L[3:0]	RW	Regular channel sequence length These bits are written by software to define the total number of conversions in the regular channel conversion sequence. 0000: 1 conversion 0001: 2 conversions ..... 1111: 16 conversions
19:15	SQ16[4:0]	RW	16th conversion in regular sequence These bits are written by software with the channel number (0..17)

Bits	Fields	R/W	Description
			assigned as the 16th in the conversion sequence.
14:10	SQ15[4:0]	RW	15th conversion in regular sequence
9:5	SQ14[4:0]	RW	14th conversion in regular sequence
4:0	SQ13[4:0]	RW	13th conversion in regular sequence

15.11.10. ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value：0x0000_0000

Bits	Fields	R/W	Description
31:30	-	R	Reserved, must be kept 0
29:25	SQ12[4:0]	RW	12th conversion in regular sequence These bits are written by software with the channel number (0..17) assigned as the 12th in the sequence to be converted.
24:20	SQ11[4:0]	RW	11th conversion in regular sequence
19:15	SQ10[4:0]	RW	10th conversion in regular sequence
14:10	SQ9[4:0]	RW	9th conversion in regular sequence
9:5	SQ8[4:0]	RW	8th conversion in regular sequence
4:0	SQ7[4:0]	RW	7th conversion in regular sequence

15.11.11. ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value：0x0000_0000

Bits	Fields	R/W	Description
31:30	-	R	Reserved, must be kept at reset value.
29:25	SQ6[4:0]	RW	6th conversion in regular sequence These bits are written by software with the channel number (0~17) assigned as the 6th in the sequence to be converted.
24:20	SQ5[4:0]	RW	5th conversion in regular sequence
19:15	SQ4[4:0]	RW	4th conversion in regular sequence
14:10	SQ3[4:0]	RW	3th conversion in regular sequence
9:5	SQ2[4:0]	RW	2th conversion in regular sequence
4:0	SQ1[4:0]	RW	1th conversion in regular sequence

15.11.12. ADC injected sequence register (0ADC_JSQR)

Address offset: 0x38

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:22	-	R	Reserved, must be kept 0
21:20	JL[1:0]	RW	Injected sequence length These bits are written by software to define the total number of conversions in the injected channel conversion sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
19:15	JSQ4[4:0]	RW	4th conversion in injected sequence (when JL[1:0] = 3)(1) These bits are written by software with the channel number (0..17) assigned as the 4th in the sequence to be converted. Note: Unlike a regular conversion sequence, if JL[1:0] length is less than four, the channels are converted in a sequence starting from (4-JL). Example: ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that a scan conversion will convert the following channel sequence: 7, 3, 3. (not 2, 7, 3)
14:10	JSQ3[4:0]	RW	3rd conversion in injected sequence
9:5	JSQ2[4:0]	RW	2nd conversion in injected sequence
4:0	JSQ1[4:0]	RW	1st conversion in injected sequence

15.11.13. ADC injected data register x (ADC_JDRx) (x=1..4)

Address offset: 0x3C - 0x48

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved, must be kept 0
15:0	JDATA[15:0]	R	Injected data These bits are read only. They contain the conversion result from injected channel x. The data is left or right-aligned as shown in Figure 15-5 15-6 15-7 15-8

15.11.14. ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved, must be kept 0
15:0	DATA[15:0]	R	Regular data These bits are read only. They contain the conversion result from the regular channels. The data is left or right-aligned as shown in Figure 15-5 15-6 15-7 15-8

15.11.15. ADC control register 3(ADC_CR3)

Address offset: 0x54

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:18	-	R	Reserved, must be kept 0
17	EMPIE	RW	FIFO empty interrupt enable bit. This bit is set and cleared by software. 0: disable EMPF interrupt; 1: Allow EMPF interrupt.
16	OVFIE	RW	FIFO overflow interrupt enable bit. This bit is set and cleared by software. 0: disable OVF interrupt; 1: OVF interrupt is allowed.
15:8	ADC_PRS[7:0]	RW	ADC clock prescaler configuration: 8'h00 : Tad = 2 ∗ Tapb 8'h01 : Tad = 4 ∗ Tapb 8'h02 : Tad = 6 ∗ Tapb ...... 8'hFF : Tad = 512 ∗ Tapb
7	-	R	Reserved, must be kept 0
6	12BIT	RW	Configure ADC accuracy 0: Reserved 1: ADC is 12 bits
5:4	-	R	Reserved, must be kept 0
3:2	-	R	Reserved
1:0	-	R	Reserved

15.11.16. ADC injected convertion DMA register (ADC_JDMAR)

Address offset: 0x58

Reset value: 0x0000_0000

When the injected conversion finish, DMA can read this address to get the conversion result.

16. Advanced timer (TIM1)

16.1. Overview

The advanced timer module (TIM1) is a four-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The advanced timer has a 20-bit counter that can be used as an unsigned counter.

In addition, the advanced timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which is suitable for motor control applications.

Advanced Timer (TIM1) and General Timer (TIMx) are completely independent with each other, but they may be synchronized to provide a larger timer with their counters increment in unison.

16.2. Characteristics

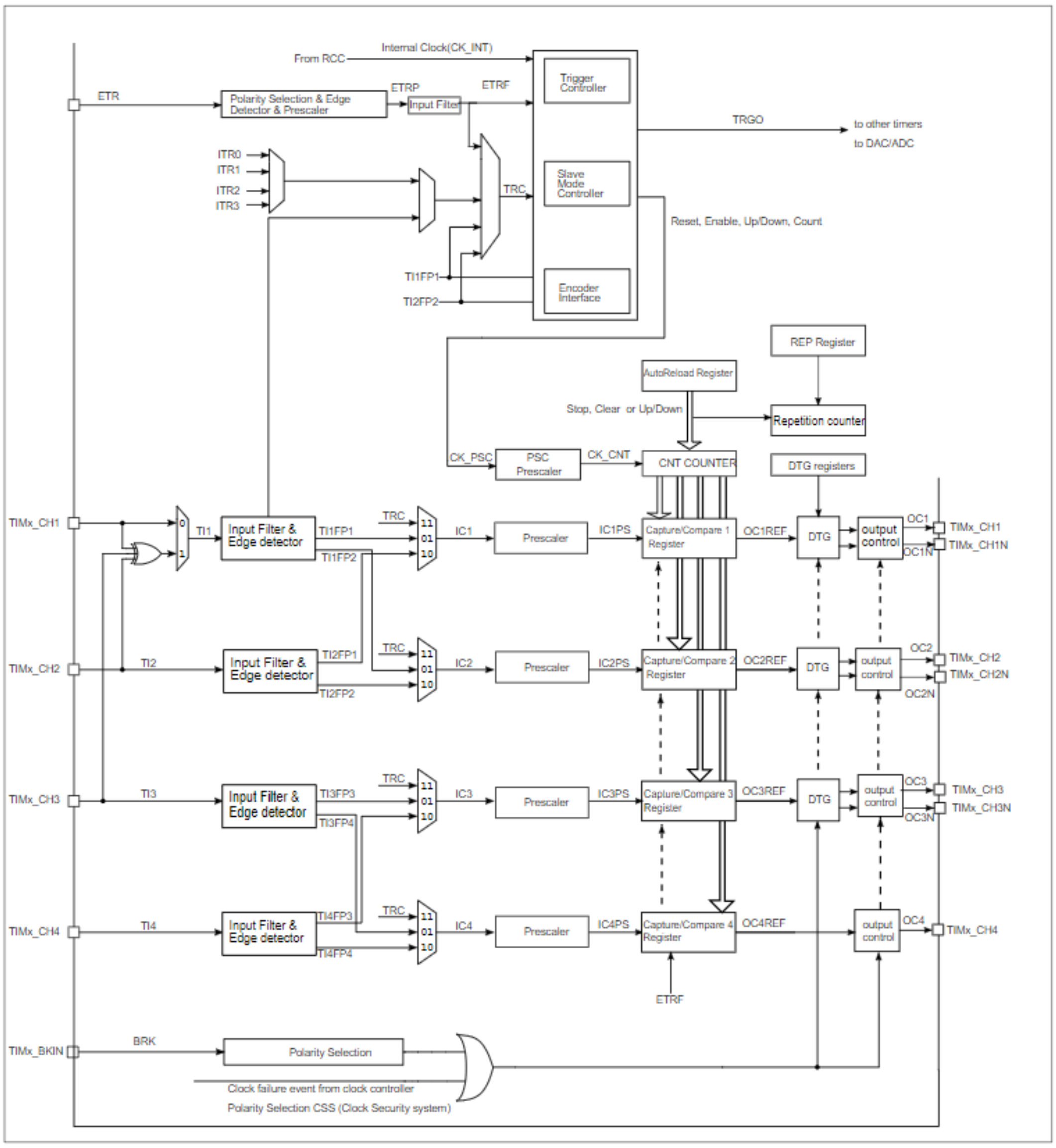
- Total channel number: 4.
- Counter width: 20bit.
- Source of counter clock is select-able:
    - internal clock
    - internal trigger
    - external input
    - external trigger
- Multiple counter modes: count up, count down, count up/down.
- Quadrature Decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit. The factor can be changed on the go.
- Each channel is user-configurable:
    - input capture mode
    - output compare mode
    - programmable PWM mode
    - single pulse mode
- Complementary outputs with programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.

- Break input .
- Interrupt output or DMA request on:
  - update
  - trigger event
  - compare/capture event
  - break input
- Daisy chaining of timer modules allows a single timer to initiate multiple timers .
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

16.3. Block diagram

Figure 16-1. Advanced timer block diagram provides the details of the internal configuration of the advanced timer.

Figure 16-1. Advanced timer block diagram



16.4. Function overview

16.4.1. Clock selection

The advanced timer has the capability of being clocked by either the TIMER_CK or an alternate clock source controlled by SMS (TIMx_SMCR bit [2:0]).

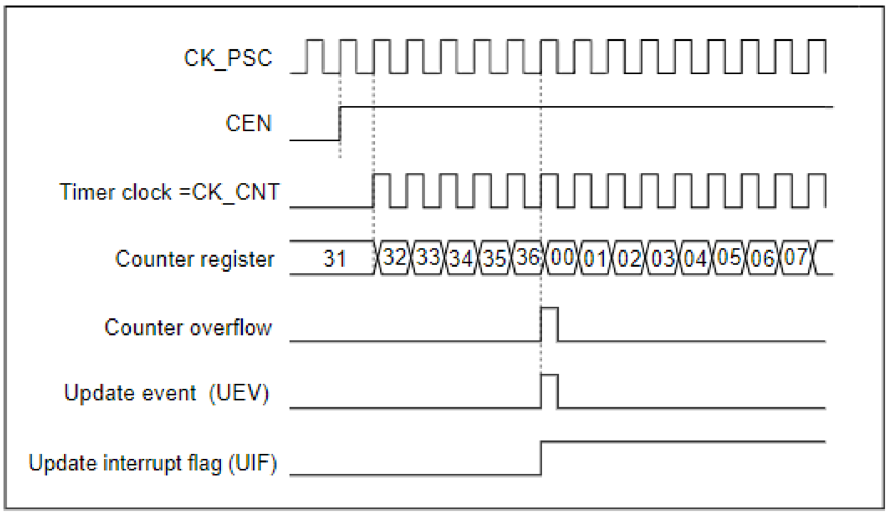
- SMS [2:0] == 3’b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCC .

The default clock source is the CK_TIMER for driving the counter prescaler when the slave mode is disabled (SMS [2:0] == 3’b000). When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK .

In this mode, the TIMER_CK, which drives counter’s prescaler to count, is equal to CK_TIMER which is from RCC.

If the slave mode controller is enabled by setting SMS [2:0] in the TIMx_SMCR register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TS[2:0] in the TIMx_SMCR register , details as follows. When the slave mode selection bits SMS[2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER_CK is the counter prescaler driving clock source.

Figure 16-2. Normal mode, internal clock divided by 1



- SMS[2:0] == 3'b111 (external clock mode 1). External input pin is selected as timer clock Source.

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMx_CH1/TIMx_CH2. This mode can be selected by setting SMS[2:0] to 0x7 and the TS[2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITR0/1/2/3. This mode can be selected by setting SMS [2:0] to 0x7 and the TS [2:0] to 0x0, 0x1, 0x2 or 0x3.

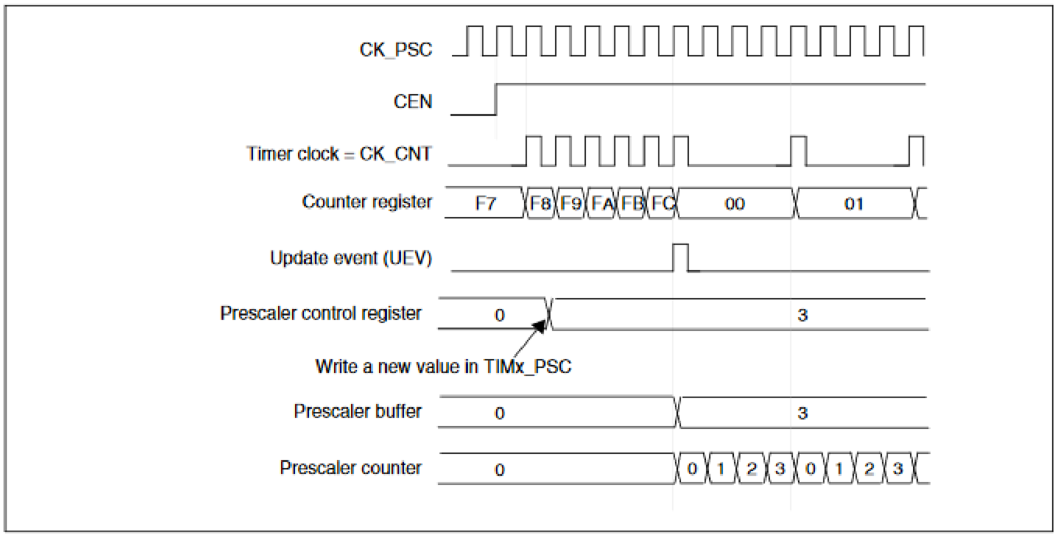
- ECE== 1'b1 (external clock mode 2). External input is selected as timer clock source (ETR).

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETR . This mode can be selected by setting the ECE bit in the TIMx_SMCR register to 1. The other way to select the ETR signal as the clock source is to set the SMS [2:0] to 0x7 and the TS [2:0] to 0x7 respectively. Note that the ETR signal is derived from the ETR pin sampled by a digital filter. When the ETR signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETR signal rising edge to clock the counter prescaler.

16.4.2. Prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMx_PSC) which can be changed on the go but is taken into account at the next update event.

Figure 16-3. Counter timing diagram with prescaler division change from 1 to 4



16.4.3. Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMx_ARR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after (TIMx_RCR+1) times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit DIR in the TIMx_CR1 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UG bit in the TIMx_EGR register, the counter value will be initialized to 0 and generates an update event.

If set the UDIS bit in TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMx_ARR=0x36.

Figure 16-4. Up-counter timechart, PSC=0

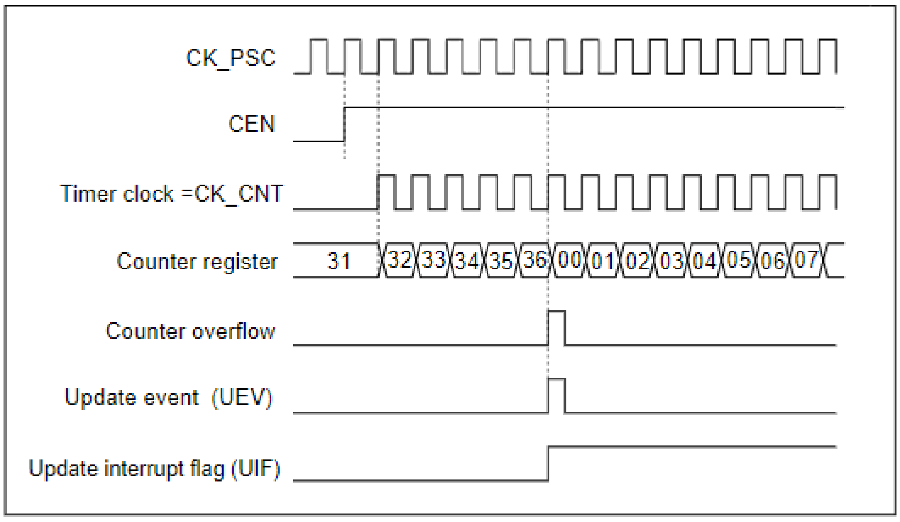


Figure 16-5. Up-counter timechart, PSC=1

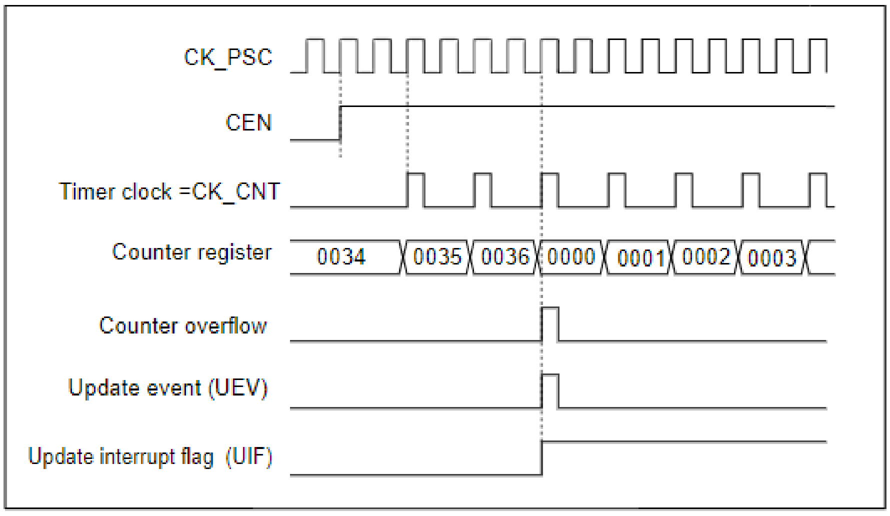


Figure 16-6. Up-counter timechart, PSC=3

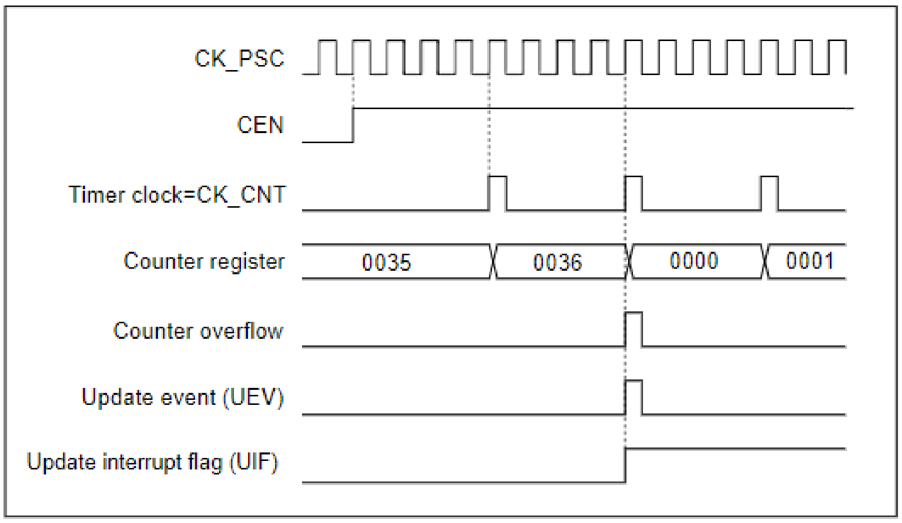


Figure 16-7. Up-counter timechart, PSC=N

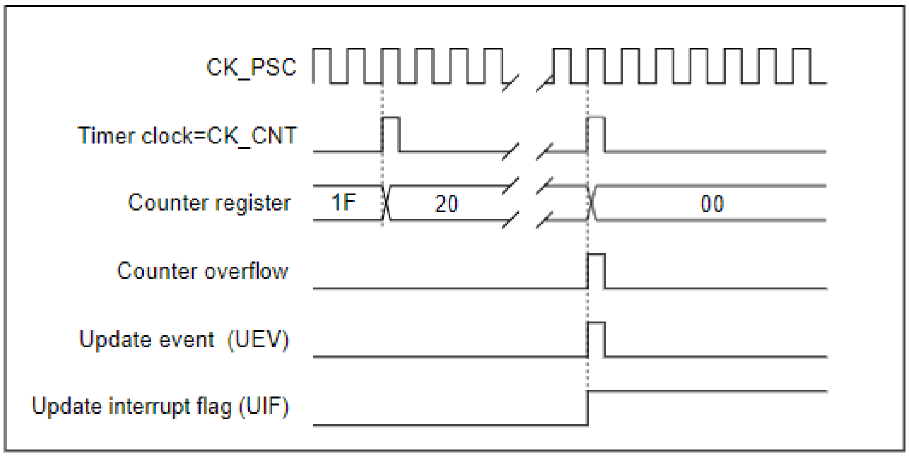
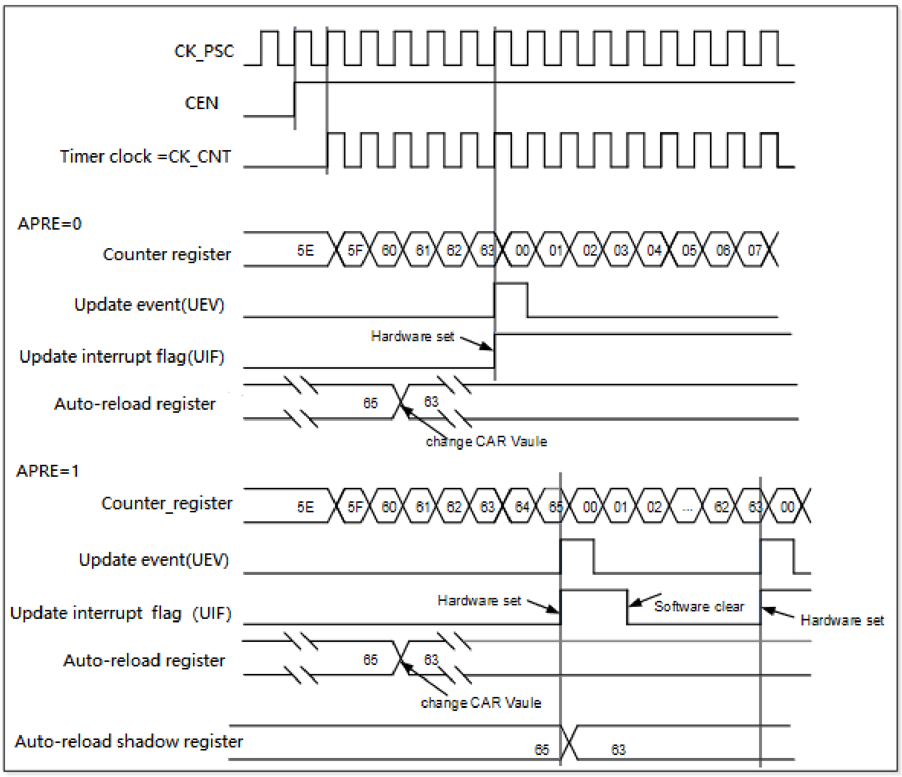


Figure 16-8. Up-counter timechart, change TIMx_ARR on the go



16.4.4. Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIM1_ARR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter restarts to count again from the counter-reload value. If the repetition counter is set, the update event will be generated after (TIMx_RCR+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMx_CR1 register should be set to 1 for the down-counting mode.

When the update event is set by the UG bit in the TIMx_EGR register, the counter value will be initialized to the counter-reload value and generates an update event.

If set the UDIS bit in TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior in different clock frequencies when TIMx_ARR=0x36.

Figure 16-9. Down-counter timechart, PSC=0

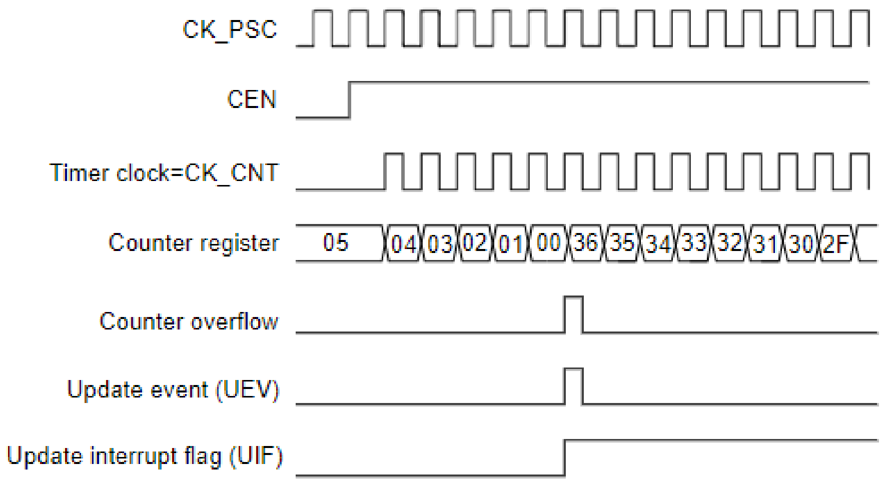


Figure 16-10. Down-counter timechart, PSC=1

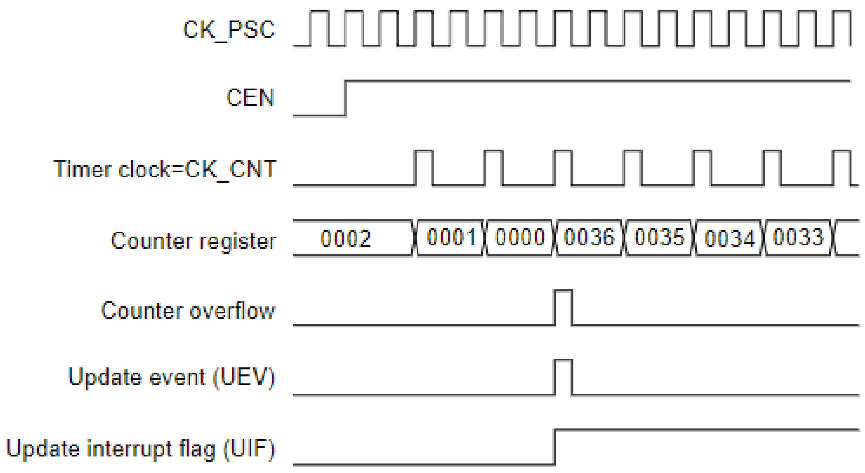


Figure 16-11. Down-counter timechart, PSC=3

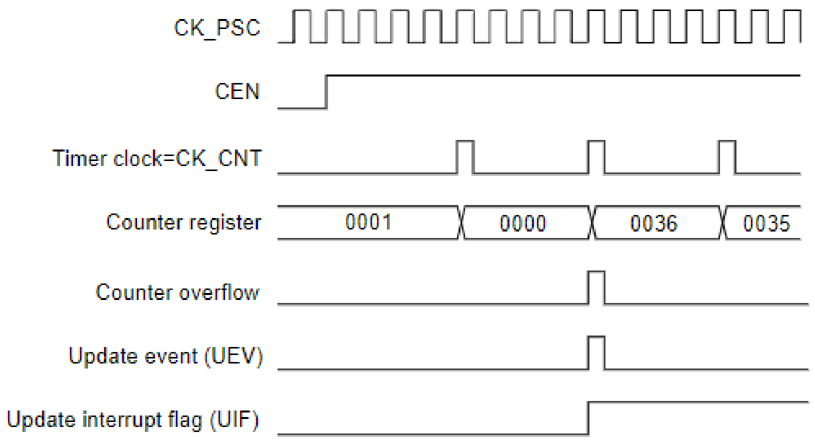


Figure 16-12. Down-counter timechart, PSC=N

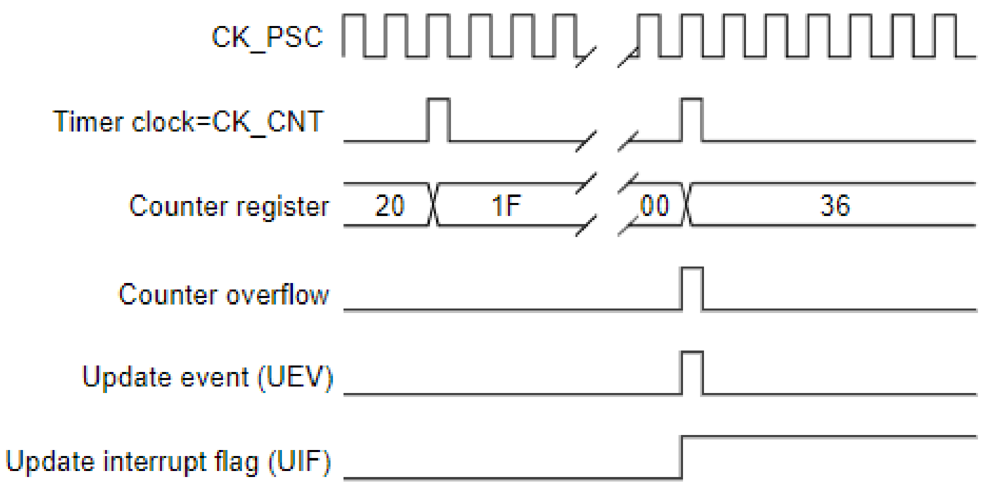
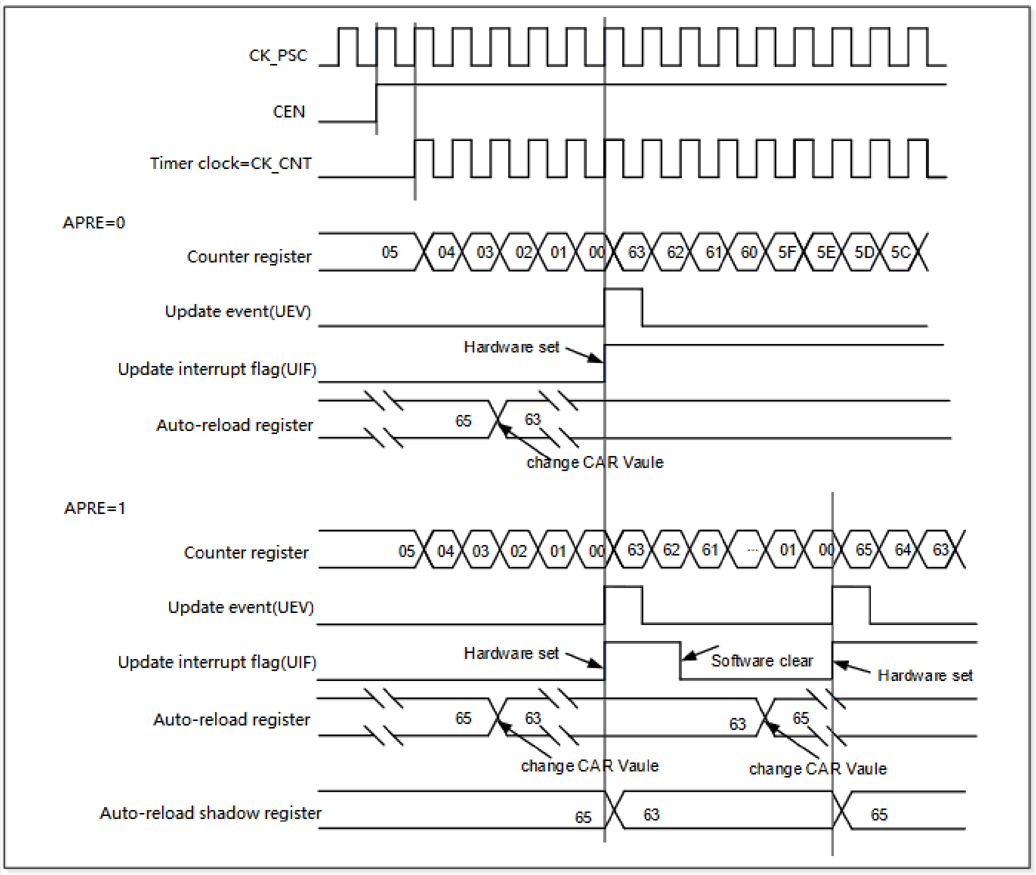


Figure 16-13. Down-counter timechart, change TIMx_ARR on the go (APRE=0)



16.4.5. Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The Timer module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMx_CR1 register is read-only and indicates the counting direction when in the center-aligned mode. The counting direction is updated by hardware automatically.

Setting the UG bit in the TIMx_EGR register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

The UIF bit in the TIMx_SR register can be set to 1 either when an underflow event or an overflow event occurs. While the CCxIF bit is associated with the value of CMS in TIMx_CR1. Refer to Figure 16-14 for the details.

If set the UDIS bit in the TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register, prescaler register) are updated.

The figures below show some examples of the counter behavior for different clock frequencies.

Figure 16-14. Center-aligned counter timechart, PSC=0

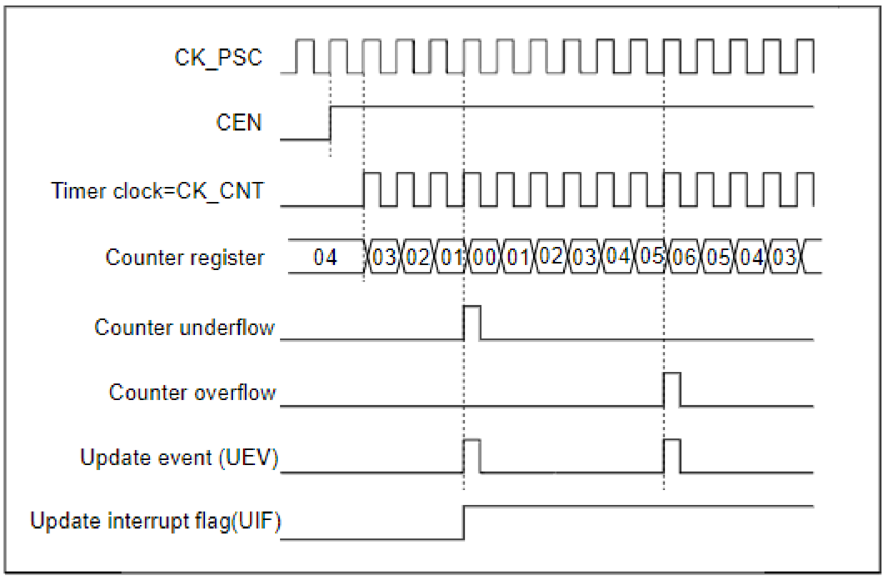


Figure 16-15. Center-aligned counter timechart, PSC=1

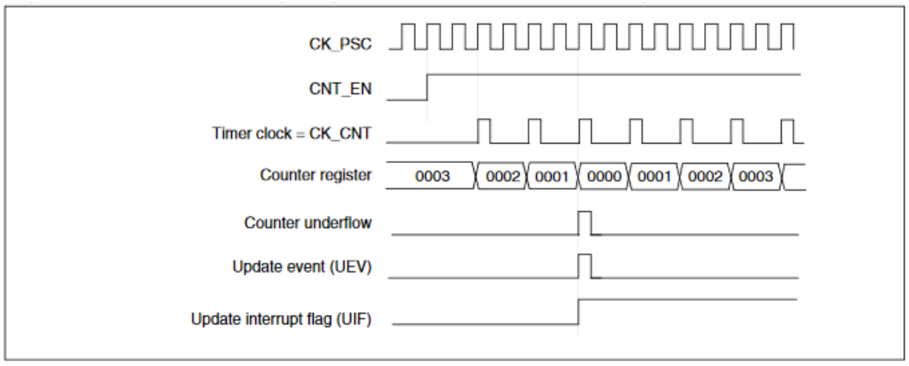


Figure 16-16. Center-aligned counter timechart, PSC=3

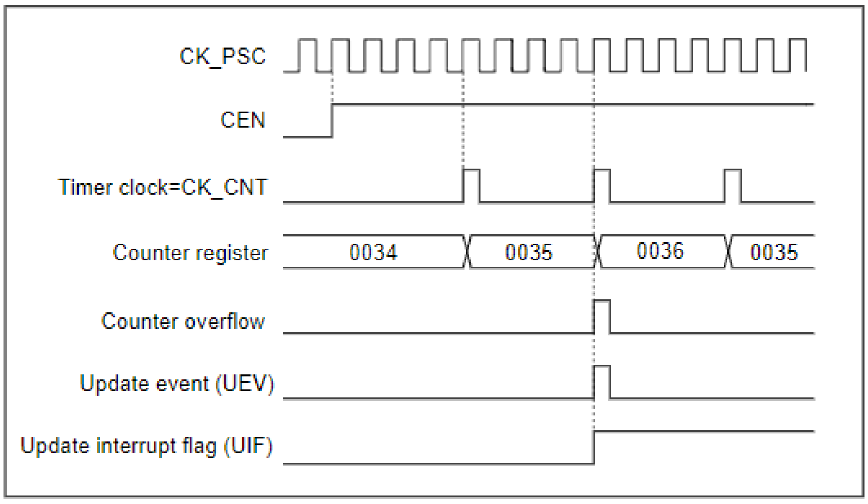


Figure 16-17. Center-aligned counter timechart, PSC=N

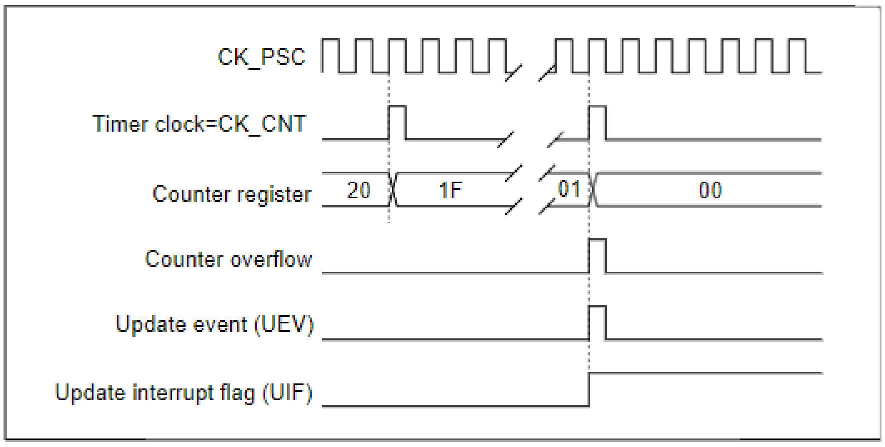


Figure 16-18. Center-aligned counter timechart, update event with ARPE=1 (counter underflow)

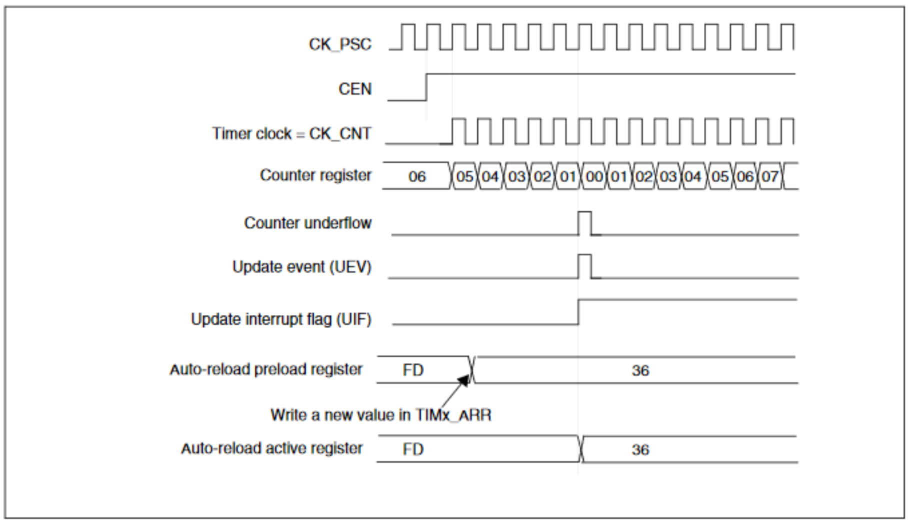
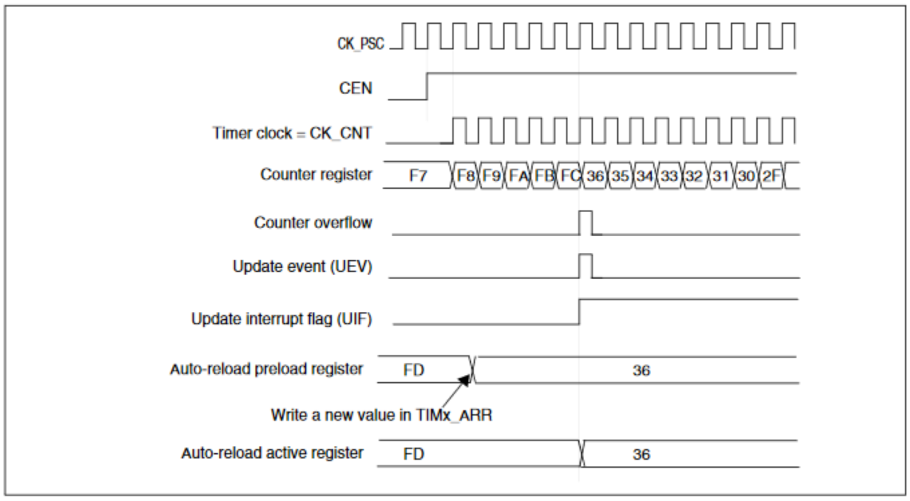


Figure 16-19. Center-aligned counter timechart, update event with ARPE=1 (counter overflow)



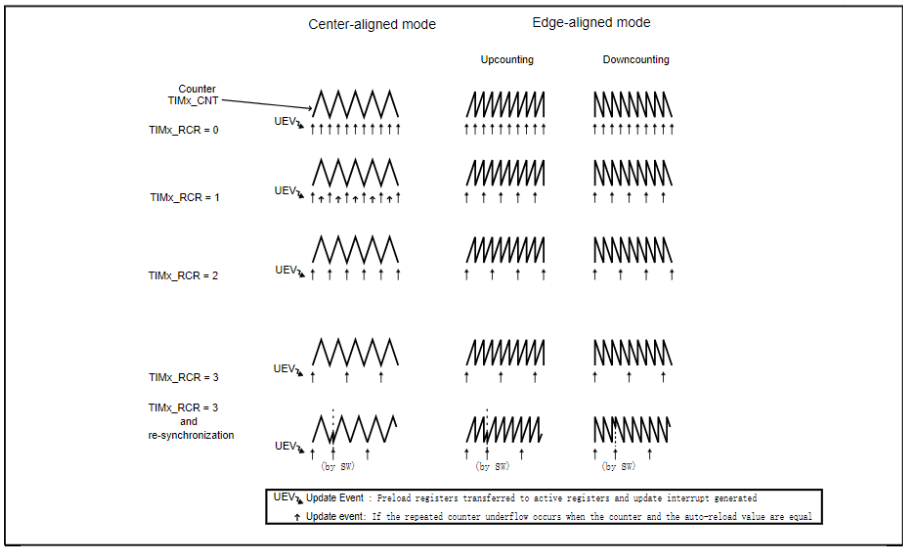
16.4.6. Counter repetition

Counter Repetition is used to generate update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is REP in TIMx_RCR register. The repetition counter is decremented at each counter overflow in up-counting mode, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned mode.

Setting the UG bit in the TIMx_EGR register will reload the content of REP in TIMx_RCR register and generate an update event.

For odd values of REP in center-aligned mode, the update event occurs either on the overflow or on the underflow depending on when the REP register was written and when the counter was started. The update event is generated at overflow when the REP was written before starting the counter, and generated at underflow when the REP was written after starting the counter.

Figure 16-20. Repetition time chart for center-aligned counter


Center-aligned mode
Edge-aligned mode
Upcounting
Downcounting
Counter TIMx_CNT
TIMx_RCR = 0
TIMx_RCR = 1
TIMx_RCR = 2
TIMx_RCR = 3
TIMx_RCR = 3 and re-synchronization
UEV
(by SW)
UEV Update Event : Preload registers transferred to active registers and update interrupt generated
Update event: If the repeated counter underflow occurs when the counter and the auto-reload value are equal

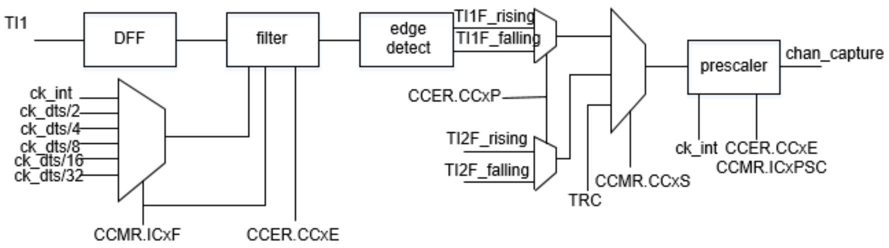
16.4.7. Capture/compare channels

The advanced timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- Input capture mode

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMx_CCRx register, at the same time the CCxIF bit is set and the channel interrupt is generated if enabled by CCxIE = 1.

Figure 16-21. Input capture logic



One of channels' input signals (TIx) can be chosen from the TIMx_CHx signal or the Excusive-OR function of the TIMx_CH1, TIMx_CH2 and TIMx_CH3 signals. First, the channel input signal (TIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CCxP. One more selector is for the other channel and trig, controlled by CCxS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CCRx will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter configuration. (ICxF in TIMx_CCMR1)

Based on the input signal and requested signal quality, configure compatible ICxF.

Step2: Edge selection. (CCxP/CCxNP in TIMx_CCER)

Rising or falling edge, choose one by CCxP/CCxNP .

Step3: Capture source selection. (CCxS in TIMx_CCMR1)

As soon as you select one input capture source by CCxS, you have set the channel to input mode (CCxS != 0x0) and TIMx_CCRx cannot be written any more.

Step4: Interrupt enable. (CCxIE and CCxDE in TIMx_DIER)

Enable the related interrupt enable; you can got the interrupt and DMA request.

Step5: Capture enables. (CCxE in TIMx_CCER)

Result: when your wanted input signal is got, TIMx_CCRx will be set by counter's value. And CCxIF is also asserted. If the CCxIF has been high, the CCxOF will also be asserted. The interrupt and DMA request will be asserted based on the configuration of CCxIE and CCxDE in TIMx_DIER.

Direct generation: if you want to generate a DMA request or Interrupt, you can set CCxG by software directly.

The input capture mode can also be used for pulse width measurement from signals on the TIMx_CHx pins. For example, PWM signal connect to TI1 input. Select Channel 1 capture signals to TI1 by setting CC1S to 2'b01 in the channel control register (TIMx_CCMR1) and set capture on rising edge. Select Channel 2 capture signal to TI1 by setting CC2S to 2'b10 in the channel control register (TIMx_CCMR1) and set capture on falling edge. The counter set to restart mode and restart on Channel 1 rising edge. Then the TIMx_CCR1 can measure the PWM period and the TIMx_CCR2 can measure the PWM duty.

- Output compare mode

In output compare mode, the TIMx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CCRx register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on OCxM . When the counter reaches the value in the CCRx register, the CCxIF bit is set and the channel (n) interrupt is generated if CCxIE = 1. And the DMA request will be asserted, if CCxDE = 1.

So the process can be divided to several steps as below:

Step1: Clock Configuration. Such as clock source, clock prescaler and so on.

Step2: Compare mode configuration.

* Set the shadow enable mode by OCxPE

* Set the output mode (Set/Clear/Toggle) by OCxM .

* Select the active high polarity by CCxP/CCxNP

* Enable the output by CCxE

Step3: Interrupt/DMA-request enables configuration by CCxIE/CCxDE.

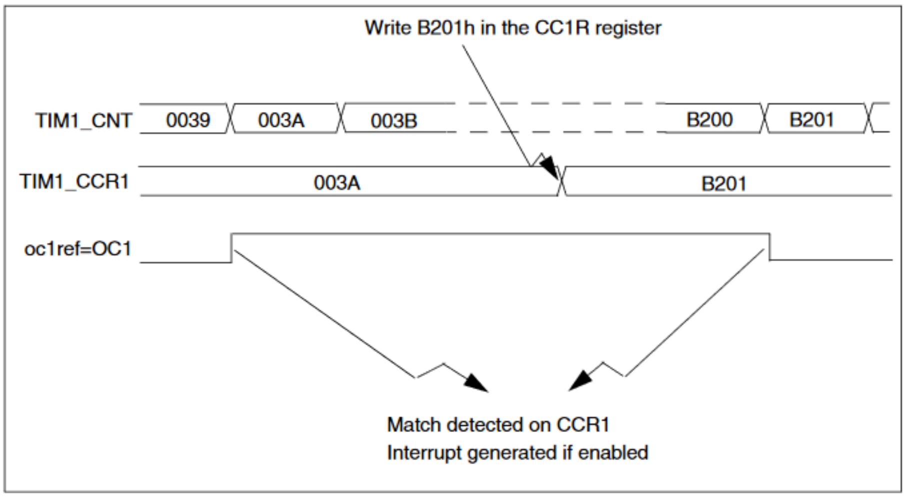
Step4: Compare output timing configuration by TIMx_ARR and TIMx_CCRx

About the CCRx; you can change it on the go to meet the waveform you expected.

Step5: Start the counter by setting CEN.

The timechart below shows the output compare mode in detail.

Figure 16-22. Output-compare mode, toggle on OC1.



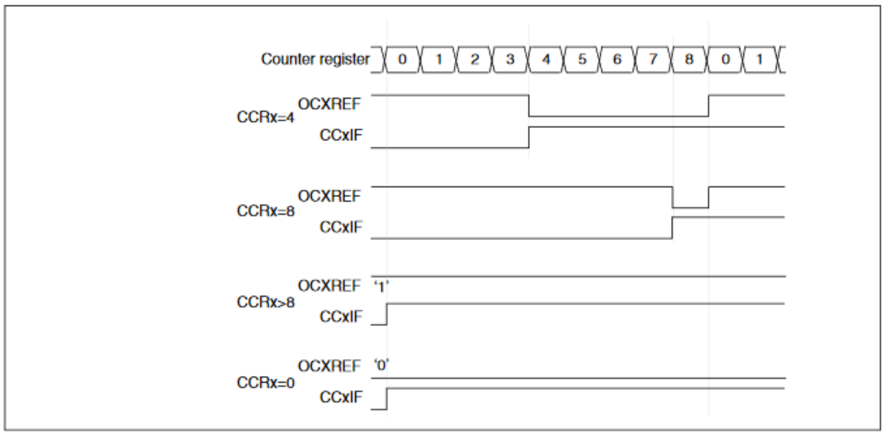
16.4.8. PWM mode

In the output PWM mode (by setting the OCxM bits to 3'b110 (PWM mode1) or to 3'b 111(PWM mode2), the channel can generate PWM waveform according to the TIMx_ARR registers and TIMx_CCRx registers.

Based on the counter mode, we can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

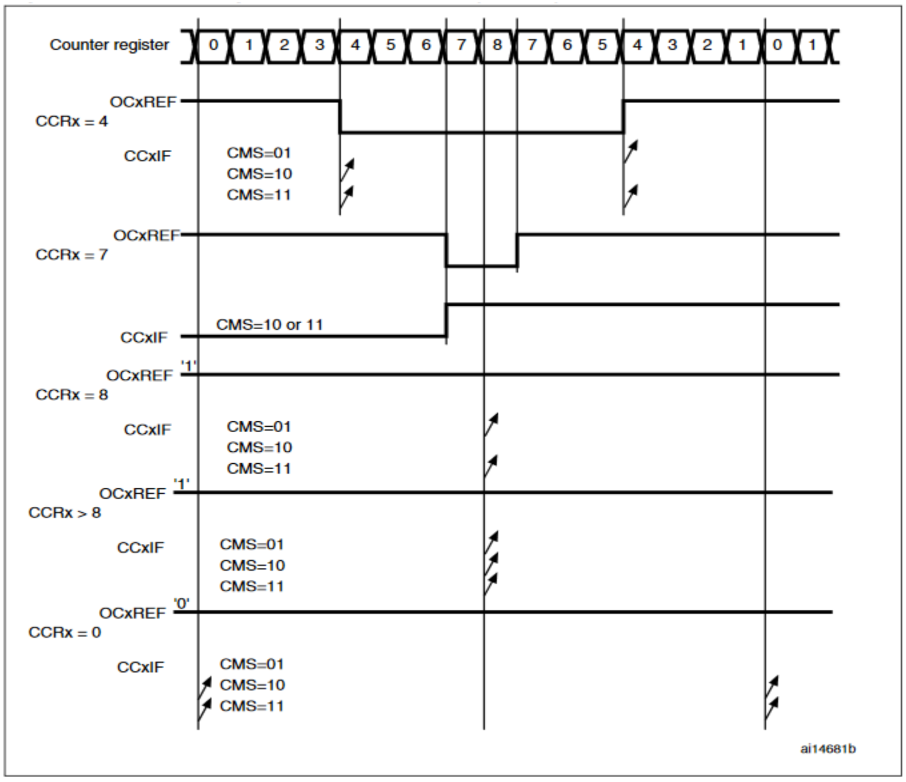
The EAPWM period is determined by TIMx_ARR and duty cycle is determined by TIMx_CCRx. The Figure below shows the EAPWM output and interrupts waveform.

Figure 16-23. EAPWM waveforms (ARR=8)



The CAPWM period is determined by 2*TIMx_ARR, and duty cycle is by 2*TIMx_CCRx. The figure below shows the CAPWM output and interrupts waveform.

Figure 16-24. CAPWM waveforms (ARR=8)



If TIMx_CCRx is greater than TIMx_ARR , the output will be always active under PWM mode1 (OCxM = 3'b110).

And if TIMx_CCRx is equal to zero, the output will be always inactive under PWM mode1 (OCxM = 3'b110)

16.4.9. Channel output reference signal

When the TIMx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the OCxM filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the OCxM field to 0x00, set to 1 by setting the OCxM field to 0x01, set to 0 by setting the OCxM field to 0x02 or signal toggle by setting the OCxM field to 0x03 when the counter value matches the content of the TIMx_CCRx register.

The PWM mode 1 and PWM mode 2 outputs are also another kind of OxCPRE output which

is setup by setting the OCxM field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMx_CCRx content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the OCxM field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMx_CCRx values.

The OxCPRE signal can be forced to 0 when the ETRF signal is derived from the external ETR pin and when it is set to a high level by setting the OCxCE bit to 1 in the TIMx_CCMR1 register. The OxCPRE signal will not return to its active level until the next update event occurs.

16.4.10. Outputs complementary

Function of complementary is for a pair of CHx_O and CHx_ON. Those two output signals cannot be active at the same time. The TIMx has 4 channels, but only the first three channels have this function. The complementary signals CHx_O and CHx_ON are controlled by a group of parameters: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OSSR, OSSI, ISOx and ISOxN bits in the TIMx_BDTR and TIMx_CR2 registers. The outputs polarity is determined by CCxP and CCxNP bits in the TIMx_CCER register.

Table 16-1. Complementary outputs controlled by parameters

Complementary Parameters					Output Status	
MOE	**OSSR**	**OSSI**	**CCxE**	**CCxNE**	**CHx_O_**	**CHx ON_**
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable.	
				1	CHx_O = CCxP, CHx_ON = CCxNP CHx_O/CHx_ON output disable. If clock is enable: CHx_O = ISOx, CHx_ON = ISOxN	
			1	0		
				1		
		1	0	0	CHx_O = CCx, CHx_ON = CCxNP CHx_O/CHx_ON output disable.	
				1	CHx_O = CCxP CHx_ON = CCxNP CHx_O/CHx_ON output enable. If clock is enable: CHx_O = ISOx, CHx_ON = ISOxN	
			1	0		
				1		
1	0	0/1	0	0	CHx_O/CHx_ON = LOW CHx_O/CHx_ON output disable.	
				1	CHx_O = LOW CHx_O output disable.	CHx_ON=OxCPRE⊕ CCxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕ CCxP CHx_O output enable	CHx_ON = LOW CHx_ON output disable.
				1	CHx_O=OxCPRE⊕ CCxP CHx_O output enable	CHx_ON=OxCPRE⊕ CCxNP CHx_ON output enable
	1		0	0	CHx_O = CCxP CHx_O output disable.	CHx_ON = CCxNP CHx_ON output disable.
				1	CHx_O = CCxP CHx_O output enable	CHx_ON=OxCPRE⊕ CCxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕ CCxP CHx_O output enable	CHx_ON = CCxNP CHx_ON output enable.
				1	CHx_O=OxCPRE⊕ CCxP CHx_O output enable	CHx_ON=OxCPRE⊕ CCxNP CHx_ON output enable.

16.4.11. Dead time insertion

The dead time insertion is enabled when both CCxE and CCxNE are 1’b1, and set MOE is also necessary. The field named DTG defines the dead time delay that can be used for all channels expect for Channel 4. The detail about the delay time, refer to the register TIMx_BDTR

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

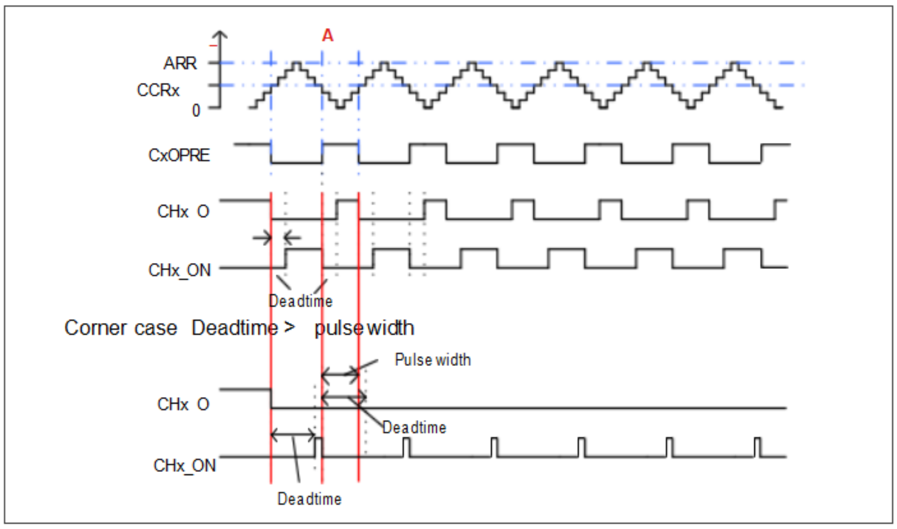
When the channel (x) match (TIMx counter = CCRx) occurs, OxCPRE will be toggled because under PWM1 mode. At point A in the figure 16-25, CHx_O signal remains at the low value until the end of the deadtime delay, while CHx_ON will be cleared at once.

Similarly, at point B when counter match (counter = CCRx) occurs again, OxCPRE is cleared, CHx_O signal will be cleared at once, while CHx_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx_O duty cycle, then the CHx_O signal is always the inactive value (as show in the figure 16-25).

Figure 16-25. Complementary output with dead-time insertion



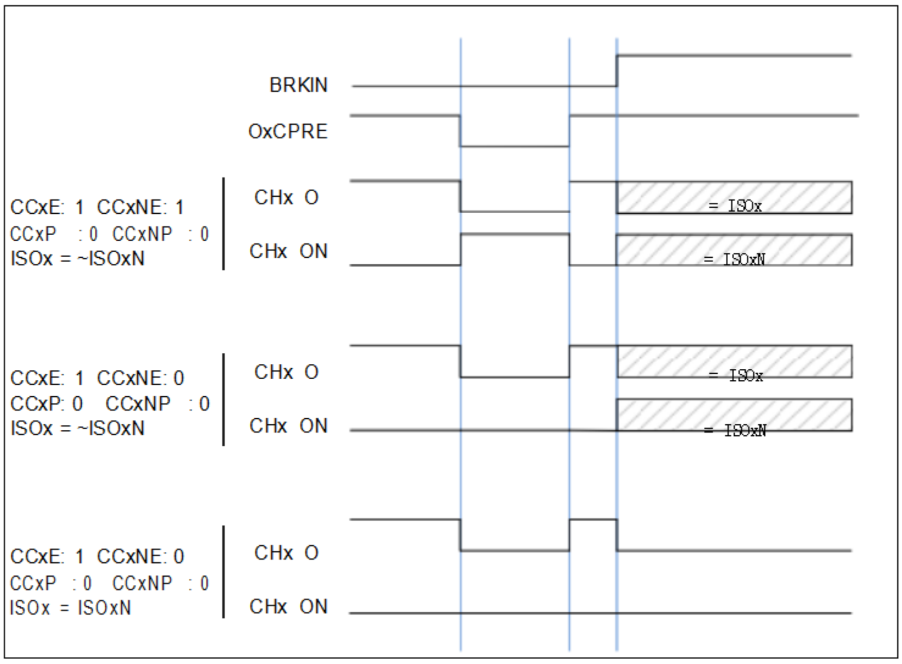
16.4.12. Break function

In this function, the output CHx_O and CHx_ON are controlled by the MOE, OSSI and OSSR bits in the TIMx_BDTR register, ISOx and ISOxN bits in the TIMx_CR2 register and cannot be set both to active level when break occurs. The break sources are input break pin and HSE stuck event by Clock Monitor (CCS) in RCC. The break function enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity is setting by the BKP bit in TIMx_BDTR.

When a break occurs, the MOE bit is cleared asynchronously, the output CHx_O and CHx_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMx_CR2 register as soon as MOE is 0. If OSSI is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BIF bit in the TIMx_SR register is set. If BIE is 1, an interrupt generated.

Figure 16-26. Output behavior in response to a break (The break high active)



16.4.13. Quadrature decoder

The quadrature decoder function uses two quadrature inputs TI1 and TI2 derived from the TIMx_CH1 and TIMx_CH2 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either TI1 only, TI2 only or both TI1 and TI2, the selection mode by setting the SMS [2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-reload value. Therefore, users must configure the TIMx_ARR register before the counter starts to count.

Table 16-2. Counting direction versus encoder signals

Counting mode	Level	TI1FP1		TI2FP2	
		Rising	Falling	Rising	Falling
TI1 only counting	TI2FP2=High	Down	Up	-	-
	TI2FP2=Low	Up	Down	-	-
TI2 only counting	TI1FP1=High	-	-	Up	Down
	TI1FP1=Low	-	-	Down	Up
TI1 and TI2 counting	TI2FP2=High	Down	Up	X	X
	TI2FP2=Low	Up	Down	X	X
	TI1FP1=High	X	X	Up	Down
	TI1FP1=Low	X	X	Down	Up

Note:"-" means "no counting"; "X" means impossible.

Figure 16-27. Example of counter operation in encoder interface mode

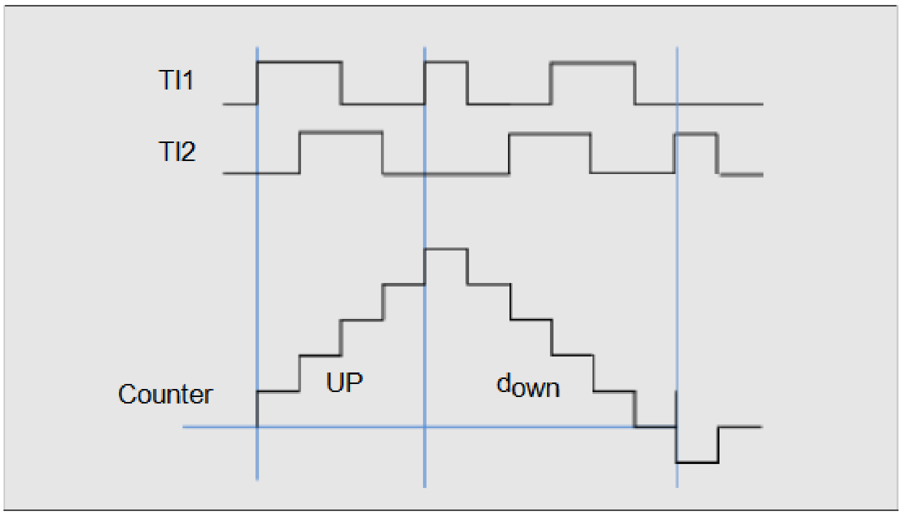
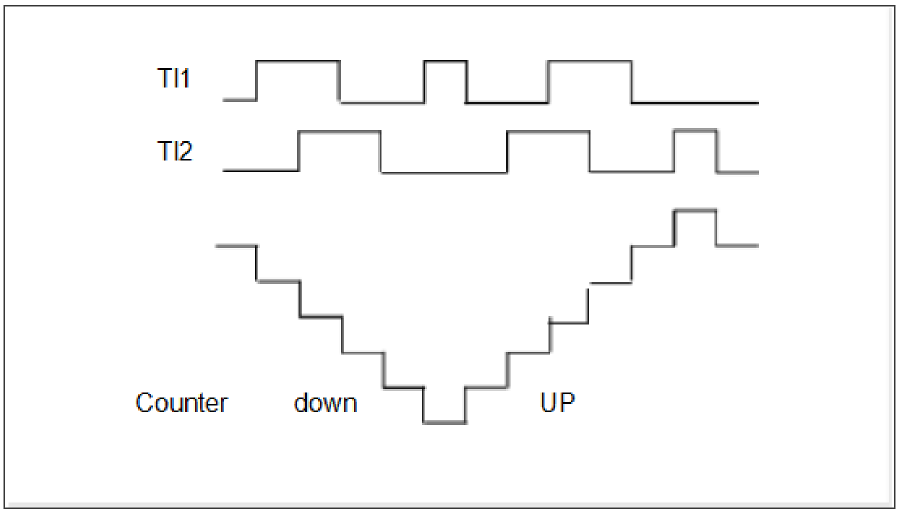


Figure 16-28. Example of encoder interface mode with CI1FE0 polarity inverted



16.4.14. Hall sensor function

Hall sensor is generally used to control BLDC Motor; advanced timer can support this function.

Figure 16-29 shows how to connect. And we can see we need two timers. First TIMER in (General purpose TIMER) should accept three Rotor Position signals from Motor.

Each of the 3 sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

By the internal connection such as TRGO- ITIx, TIMER_in and TIMER_out can be connected. TIMER_out will generate PWM signal to control BLDC motor’s speed based on the ITRx. Then, the feedback circuit is finished, also you change configuration to fit your request.

About the TIMER_in, it need have input XOR function, so you can choose from Advanced TIMER.

And TIMER_out need have functions of complementary and Dead-time, so only advanced timer can be chosen. Else, based on the timers' internal connection relationship, pair's timers can be selected. For example:

TIMER_in (TIMER2) -> TIMER_out (Timer1 ITR1)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI1S, then, each of input signal change will make the TI1 toggle. CCR1 will record the value of counter at that moment.
- Enable ITRx connected to commutation function directly by setting CCUS and CCPC.
- Configuration PWM parameter based on your request.

Figure 16-29. Hall sensor is used to BLDC motor

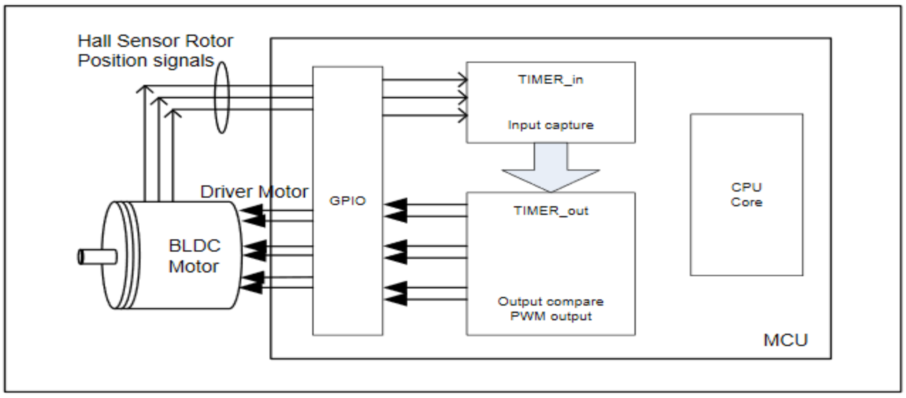
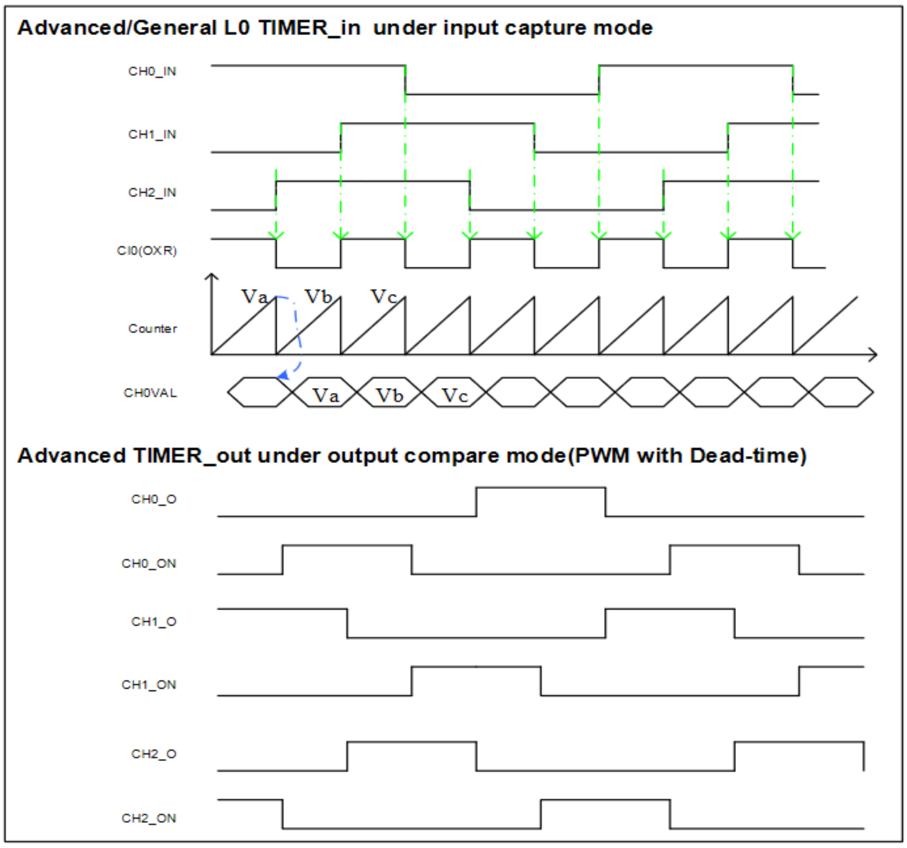


Figure 16-30. Hall sensor timing between two timers


Advanced/General L0 TIMER_in under input capture mode
CH0_IN
CH1_IN
CH2_IN
CI0(OXR)
Counter
Va
Vb
Vc
CH0VAL
Advanced TIMER_out under output compare mode(PWM with Dead-time)
CH0_O
CH0_ON
CH1_O
CH1_ON
CH2_O
CH2_ON

Slave controller

The TIMx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMS [2:0] in the TIMx_SMCR register. The trigger input of these modes can be selected by the TS [2:0] in the TIMx_SMCR register.

16.4.15. Single pulse mode

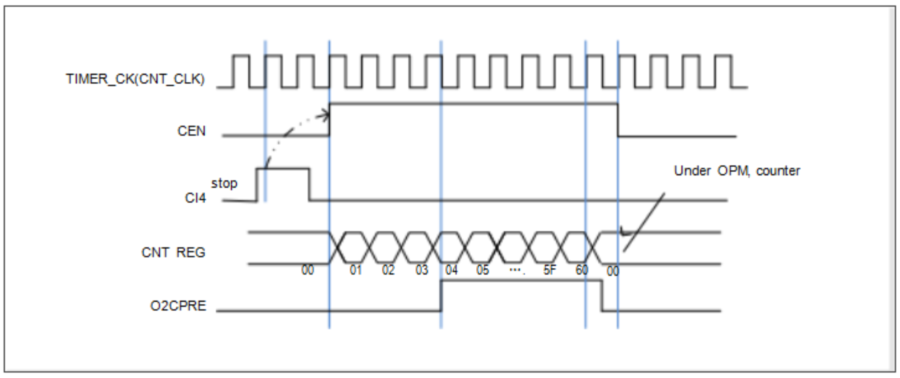
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting OPM in TIMx_CR1. When you set OPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMx to PWM mode or compare by OCxM .

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMx_CR1 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN

bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMx_CCRx value. In order to reduce the delay to a minimum value, the user can set the OCxFE bit in each TIMx_CCMR1/1 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The OCxFE bit is available only when the output channel is configured to operate in the PWM1 or PWM2 output mode and the trigger source is derived from the trigger signal.

Figure 16-31. Single pulse mode, TIMx_CCRx = 0x04, TIMx_ARR=0x60

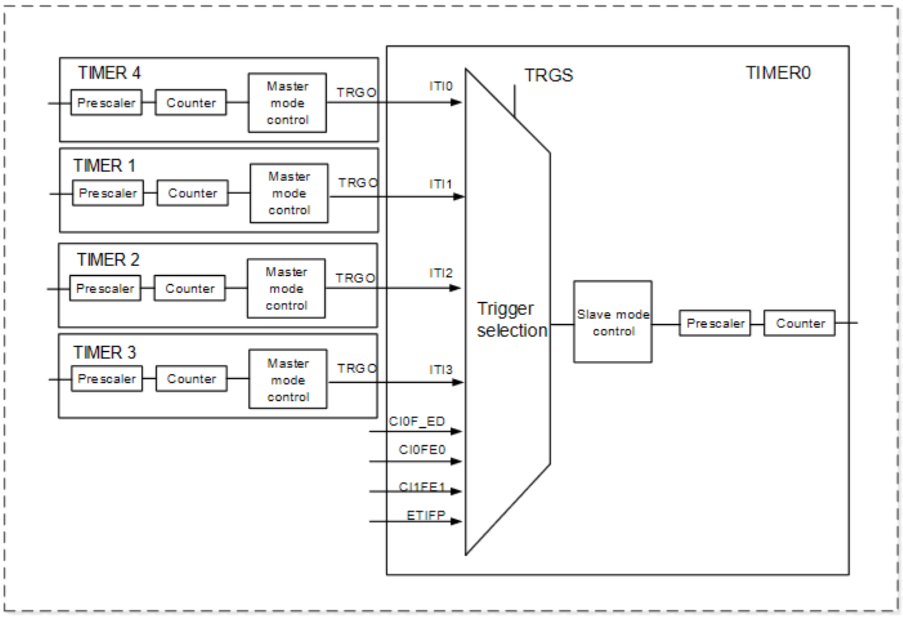


16.4.16. Timers interconnection

The timers can be internally connected together for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the master mode while configuring another timer to be in the slave mode. The following figures present several examples of trigger selection for the master and slave modes.

Figure 16-32. Timer1 master/slave mode timer example shows the timer1 trigger selection when it is configured in slave mode.

Figure 16-32. Timer1 master/slave mode timer example



- Other interconnection examples:
  - Timer 2 as prescaler for timer 1

We configure Timer2 as a prescaler for Timer 1. Refer to ***Figure 16-32. Timer1 master/ slave mode timer example*** for connections. Do as bellow:

- ◆ Configure Timer2 in master mode and select its update event (UPE) as trigger output (MMS=3'b010 in the TIMER2_CR2 register). Then timer2 drives a periodic signal on each counter overflow.
- ◆ Configure the Timer2 period (TIMER2_ARR registers).
- ◆ Select the Timer1 input trigger source from Timer2 (TS=3'b010 in the TIMx_SMCR register).
- ◆ Configure Timer1 in external clock mode 1 (SMS=3'b111 in TIMx_SMCR register).
- ◆ Start Timer1 by writing '1 in the CEN bit (TIM1 _CR1 register).
- ◆ Start Timer2 by writing '1 in the CEN bit (TIMER2_CR1 register).

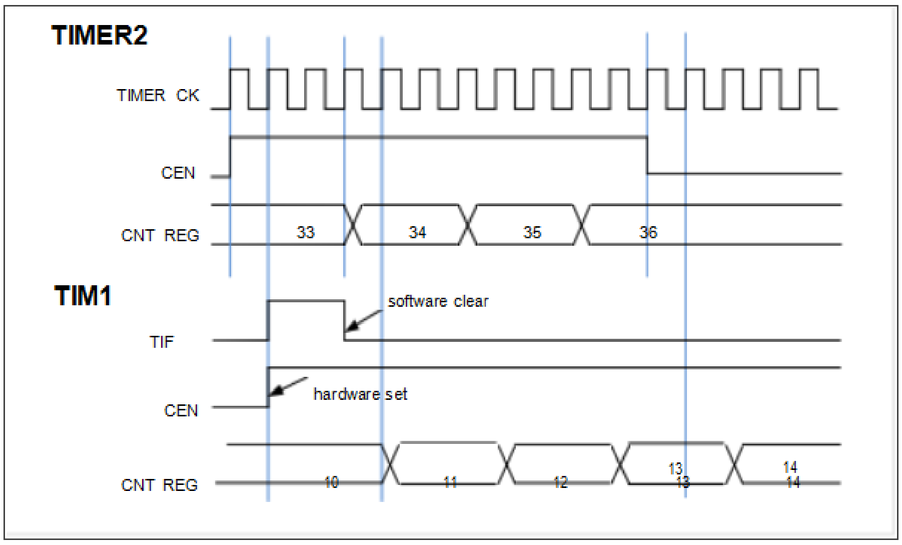
■ Start Timer1 with Timer2's Enable/Update signal

In this example , we enable Timer1 with the enable output of Timer2. Refer to ***Figure 16-33. Triggering Timer1 with enable signal of TIMER2***. Timer1 starts counting from its current value on the divided internal clock after trigger by Timer2 enable output.

When Timer1 receives the trigger signal, its CEN bit is set automatically and the counter counts until we disable TIM1 . In this example, both counter clock frequencies are divided by 3 by the prescaler compared to TIMER_CK ($f_{CNT\_CLK} = f_{TIMER\_CK}/3$) . TIM1 's SMS is set as event mode, so Timer1 can not be disabled by Timer2's disable signal . Do as follow:

- ◆ Configure Timer2 master mode to send its enable signal as trigger output(MMS=3'b001 in the TIMER2_CR2 register)
- ◆ Configure Timer1 to select the input trigger from Timer2 (TS=3'b010 in the TIMx_SMCR register).
- ◆ Configure Timer1 in event mode (SMS=3'b110 in TIMx_SMCR register).
- ◆ Start Timer2 by writing 1 in the CEN bit (TIMER2_CR1 register).

Figure 16-33. Triggering Timer1 with enable signal of TIMER2

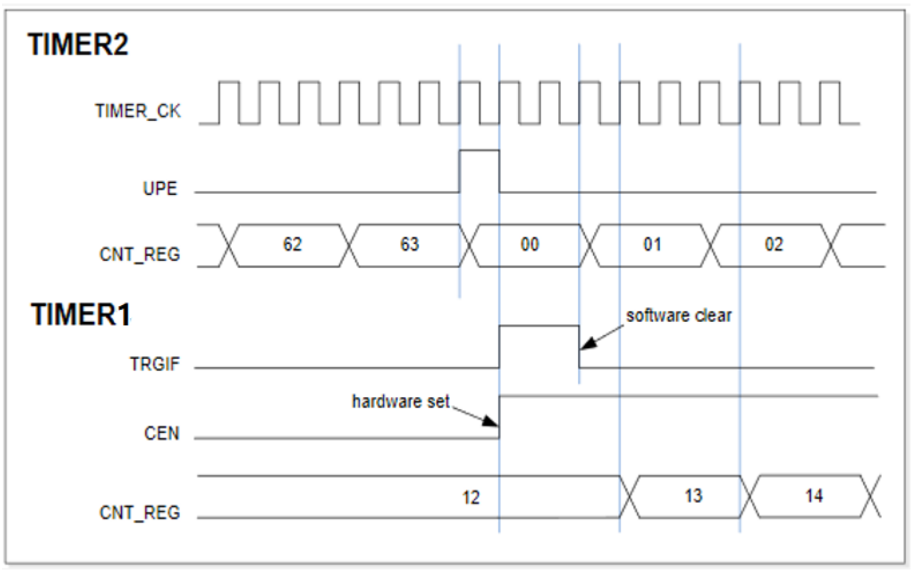


In this example, we also can use update Event as trigger source instead of enable signal.

Refer to ***Figure 16-34. Triggering Timer1 with update signal of TIMER2***. Do as follow:

- Configure Timer2 in master mode and send its update event (UPE) as trigger output (MMS=3'b010 in the TIMER2_CR2 register).
- Configure the Timer2 period (TIMER2_ARR registers).
- Configure Timer1 to get the input trigger from Timer2 (TS=3'b010 in the TIMx_SMCR register).
- Configure Timer1 in event mode (SMS=3'b110 in TIMx_SMCR register).
- Start Timer2 by writing '1 in the CEN bit (TIMER2_CR1 register).

Figure 16-34. Triggering Timer1 with update signal of TIMER2

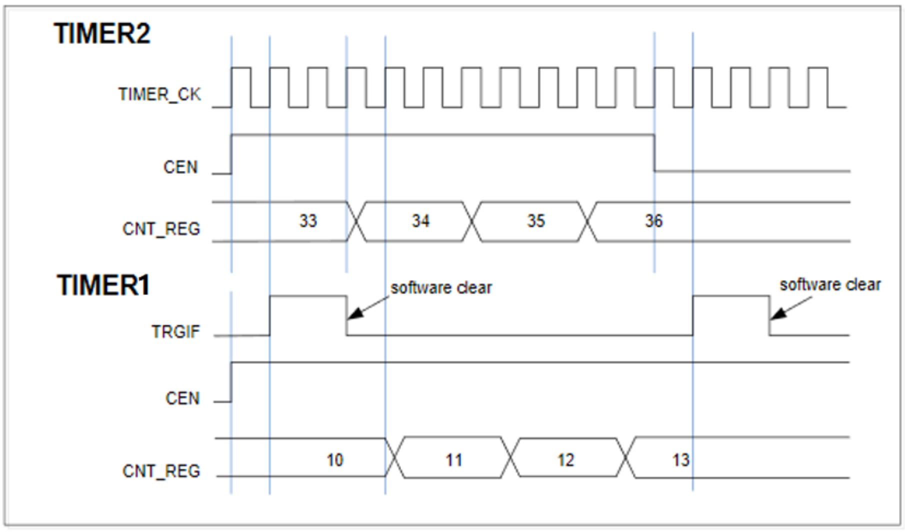


- ■ Enable Timer1 count with Timer2's enable/O1CPRE signal

In this example, we control the enable of Timer1 with the enable output of Timer2 .Refer to ***Figure 16-35. Pause Timer1 with enable signal of TIMER2***. Timer1 counts on the divided internal clock only when Timer 2 is enable. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_TIMER (f_{CNT_CLK} = f_{PCLK} /3). TIM1 's SMS is set as pause mode, so Timer1 can be enabled/disabled by Timer2's enable/disable signal. Do as follow:

- Configure Timer2 input master mode and output enable signal as trigger output (MMS=3'b001 in the TIMER2_CR2 register).
- Configure Timer1 to get the input trigger from Timer2 (TS=3'b010 in the TIMx_SMCR register).
- Configure Timer1 in pause mode (SMS=3'b101 in TIMx_SMCR register).
- Enable Timer1 by writing '1 in the CEN bit (TIM1 _CR1 register)
- Start Timer2 by writing '1 in the CEN bit (TIMER2_CR1 register).
- Stop Timer2 by writing '0 in the CEN bit (TIMER2_CR1 register).

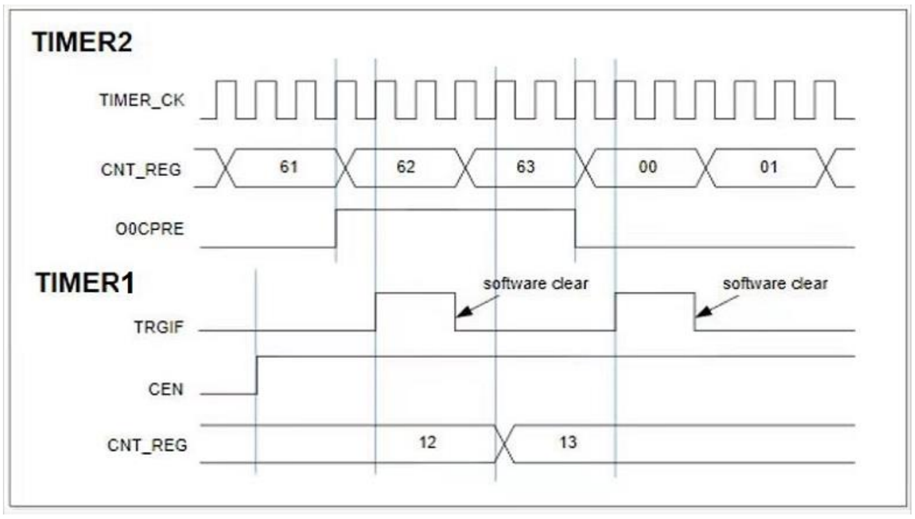
Figure 16-35. Pause Timer1 with enable signal of TIMER2



In this example, we also can use O1CPRE as trigger source instead of enable signal output. Do as follow:

- Configure Timer2 in master mode and output 0 Compare Prepare signal (O1CPRE) as trigger output (MMS=3'b100 in the TIMER2_CR2 register).
- Configure the Timer2 O1CPRE waveform (TIMER2_CCMR1 register).
- Configure Timer1 to get the input trigger from Timer2 (TS=3'b010 in the TIMx_SMCR register).
- Configure Timer1 in pause mode (SMS=3'b101 in TIMx_SMCR register).
- Enable Timer1 by writing '1 in the CEN bit (TIM1 _CR1 register).
- Start Timer2 by writing '1 in the CEN bit (TIMER2_CR1 register).

Figure 16-36. Pause Timer1 with O1CPREF signal of Timer2



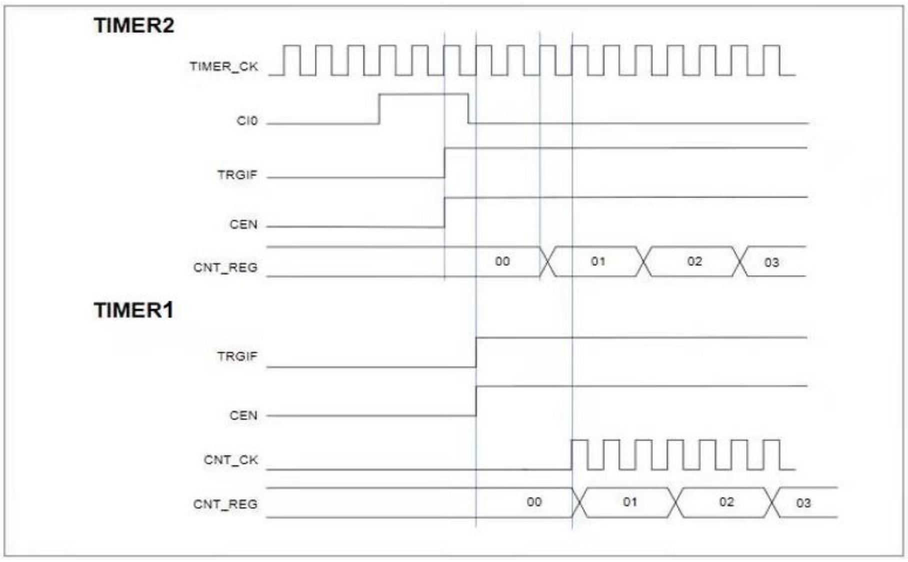
- Using an external trigger to start 2 timers synchronously

We configure the start of Timer1 triggered by the enable signal of Timer2, and Timer2 is triggered by its TI1 input rises edge. To ensure 2 timers start synchronously, Timer2 must be configured in Master/Slave mode. Do as follow:

- Configure Timer2 in slave mode to get the input trigger from TI1 (TS=3’b100 in the TIMER2_SMCR register).
- Configure Timer2 in event mode (SMS=3’b110 in the TIMER2_SMCR register).
- Configure the Timer2 in Master/Slave mode by writing MSM=1 (TIMER2_ SMCR register).
- Configure Timer1 to get the input trigger from Timer2 (TS=3’b010 in the TIMx_SMCR register).
- Configure Timer1 in event mode (SMS=3’b110 in the TIM1 _SMCR register).

When a rising edge occurs on Timer2’s TI1, two timer’s counters start counting synchronously on the internal clock and both TIF flags are set.

Figure 16-37. Triggering Timer1 and TIMER2 with TIMER2’s TI1 input



16.4.17. Timer DMA mode

If one of the DMA request event coming, TIMx will send DMA requests to DMAC. The user can configure DMAC properly and read/write TIMx SFRs under the control of DMA. Please refer to DMAC related chapters for details.

16.4.18. Timer debug mode

When the Cortex™-M3 is halted, and the DBG_TIMx_STOP configuration bit in DBGMCU_CR register is set to 1, the TIMx counter stops.

16.5. Register definition

16.5.1. Control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

Bits	Fields	R/W	Description
31:10	Reserved	R	Must be kept at reset value
9:8	CKD[1:0]	RW	Clock division The CKD bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. 00: $f_{DTS}=f_{TIMER\_CK}$ 01: $f_{DTS}=f_{TIMER\_CK}$ /2 10: $f_{DTS}=f_{TIMER\_CK}$ /4 11: Reserved
7	ARPE	RW	Auto-reload shadow enable 0: The shadow register for TIMx_ARR register is disabled 1: The shadow register for TIMx_ARR register is enabled
6:5	CMS[1:0]	RW	Counter aligns mode selection

Bits	Fields	R/W	Description
			00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit. 01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Only when the counter is counting down, compare interrupt flag of channels can be set. 10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Only when the counter is counting up, compare interrupt flag of channels can be set. 11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set. After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	RW	Direction 0: Count up 1: Count down This bit is read only when the timer is configured in center-aligned mode or encoder mode.
3	OPM	RW	Single pulse mode. 0: Single pulse mode disable. Counter continues after update event. 1: Single pulse mode enable. The CEN is cleared by hardware and the counter stops at next update event.
2	URS	RW	Update source This bit is used to select the update event sources by software. 0: Any of the following events generate an update interrupt or DMA request: – The UG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: Only counter overflow/underflow generates an update interrupt or DMA request.
1	UDIS	RW	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: – The UG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.
0	CEN	RW	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

16.5.2. Control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31-15	Reserved	R	Must be kept at reset value
14	OIS4	RW	Idle state of Channel 4 output Refer to OIS1 bit
13	OIS3N	RW	Idle state of Channel 3 complementary output Refer to OIS1N bit
12	OIS3	RW	Idle state of Channel 3 output Refer to OIS1 bit
11	OIS2N	RW	Idle state of Channel 2 complementary output Refer to OIS1N bit
10	OIS2	RW	Idle state of Channel 2 output Refer to OIS1 bit
9	OIS1N	RW	Idle state of Channel 1 complementary output 0: When MOE bit is reset, CH1_ON is set low. 1: When MOE bit is reset, CH1_ON is set high This bit can be modified only when LOCK [1:0] bits in TIMx_BDTR register is 00.
8	OIS1	RW	Idle state of Channel 1 output 0: When MOE bit is reset, CH1_O is set low. 1: When MOE bit is reset, CH1_O is set high The CH1_O output changes after a dead-time if CH1_ON is implemented. This bit can be modified only when LOCK [1:0] bits in TIMx_BDTR register is 00.
7	TI1S	RW	Channel 1 trigger input selection 0: The TIMx_CH1 pin input is selected as Channel 1 trigger input. 1: The result of combinational XOR of TIMx_CH1, CH2 and CH3 pins is selected as Channel 1 trigger input.
6:4	MMS[2:0]	RW	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UG bit in the TIMx_EGR register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected. 010: Update. In this mode the master mode controller selects the update event as TRGO. 011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channal0. 100: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O4CPRE signal is used as TRGO
3	CCDS	RW	DMA request source selection

Bits	Fields	RW	Descriptions
			0: DMA request of channel x is sent when capture/compare event occurs. 1: DMA request of channel x is sent when update event occurs.
2	CCUS	RW	Commutation control shadow register update control When the commutation control shadow enable (for CCxE , CCxNE and OCxM bits) are set (CCPC=1), these shadow registers update are controlled as below: 0: The shadow registers update by when COMG bit is set. 1: The shadow registers update by when COMG bit is set or a rising edge of TRGI occurs. When a channel does not have a complementary output, this bit has no effect.
1	Reserved	R	Must be kept at reset value.
0	CCPC	RW	Commutation control shadow enable 0: The shadow registers for CCxE, CCxNE and OCxM bits are disabled. 1: The shadow registers for CCxE, CCxNE and OCxM bits are enabled. After these bits have been written, they are updated based when commutation event coming. When a channel does not have a complementary output, this bit has no effect.

16.5.3. Slave mode configuration register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31-16	-	R	Reserved bits
15	ETP	RW	External trigger polarity This bit specifies the polarity of ETR signal 0: ETR is active at high level or rising edge. 1: ETR is active at low level or falling edge.
14	ECE	RW	Part of SMS for enable External clock mode 2. In external clock mode 2, the counter is clocked by any active edge on the ETRF signal. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. Note: It is possible to simultaneously use external clock mode 2 with the restart mode, pause mode or event mode(TS bits must not be 3'b111). The external clock input will be ETRF if external clock mode 1 and external clock mode 2 are enabled at the same time. External clock mode 1 enable is in this register's SMS bit-filed.
13:12	ETPS[1:0]	RW	External trigger prescaler The frequency of external trigger signal ETRP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETRP frequency. 00: Prescaler disable 01: ETRF frequency will be divided by 2 10: ETRF frequency will be divided by 4 11: ETRF frequency will be divided by 8
11:8	ETF[3:0]	RW	External trigger filter control An event counter is used in the digital filter, in which a transition on the output

Bits	Fields	RW	Descriptions
			occurs after N input events. This bit-field specifies the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. 0000: Filter disabled. f_{SAMP}= f_{DTS}, N=1. 0001: f_{SAMP}= f_{TIMER_CK}, N=2. 0010: f_{SAMP}= f_{TIMER_CK}, N=4. 0011: f_{SAMP}= f_{TIMER_CK}, N=8. 0100: f_{SAMP}=f_{DTS}/2, N=6. 0101: f_{SAMP}=f_{DTS}/2, N=8. 0110: f_{SAMP}=f_{DTS}/4, N=6. 0111: f_{SAMP}=f_{DTS}/4, N=8. 1000: f_{SAMP}=f_{DTS}/8, N=6. 1001: f_{SAMP}=f_{DTS}/8, N=8. 1010: f_{SAMP}=f_{DTS}/16, N=5. 1011: f_{SAMP}=f_{DTS}/16, N=6. 1100: f_{SAMP}=f_{DTS}/16, N=8. 1101: f_{SAMP}=f_{DTS}/32, N=5. 1110: f_{SAMP}=f_{DTS}/32, N=6. 1111: f_{SAMP}=f_{DTS}/32, N=8.
7	MSM	RW	Master-slave mode This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together. 0: Master-slave mode disable 1: Master-slave mode enable
6:4	TS[2:0]	RW	Trigger selection This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter. 000: Internal trigger input 0 (ITR0) 001: Internal trigger input 1 (ITR1) 010: Internal trigger input 2 (ITR2) 011: Internal trigger input 3 (ITR3) 100: TI1 edge flag (TI1F_ED) 101: Channel 1 input Filtered output (TI1FP1) 110: Channel 2 input Filtered output (TI2FP2) 111: External trigger input filter output(ETRF) These bits must not be changed when slave mode is enabled.
3	Reserved	R	Must be kept at reset value.
2:0	SMS[2:0]	RW	Slave mode control 000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Quadrature decoder mode 0.The counter counts on TI2FP2 edge, while the direction depends on TI1FP1 level. 010: Quadrature decoder mode 1.The counter counts on TI1FP1 edge, while the direction depends on TI2FP2 level. 011: Quadrature decoder mode 2.The counter counts on both TI1FP1 and TI2FP2 edge, while the direction depends on each other. 100: Restart mode. The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input. 101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter when it is low. 110: Event mode. A rising edge of the trigger input enables the counter. The counter

Bits	Fields	RW	Descriptions
			cannot be disabled by the slave mode controller. 111: External clock mode 1. The counter counts on the rising edges of the selected trigger.

16.5.4. DMA and interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31-15	Reserved	R	Must be kept at reset value.
14	TDE	RW	Trigger DMA request enable 0: disabled 1: enabled
13	COMDE	RW	Commutation DMA request enable 0: disabled 1: enabled
12	CC4DE	RW	Channel 4 capture/compare DMA request enable 0: disabled 1: enabled
11	CC3DE	RW	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
10	CC2DE	RW	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
9	CC1DE	RW	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
8	UDE	RW	Update DMA request enable 0: disabled 1: enabled
7	BIE	RW	Break interrupt enable 0: disabled 1: enabled
6	TE	RW	Trigger interrupt enable 0: disabled 1: enabled
5	COMIE	RW	commutation interrupt enable 0: disabled 1: enabled
4	CC4IE	RW	Channel 4 capture/compare interrupt enable 0: disabled 1: enabled
3	CC3IE	RW	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
2	CC2IE	RW	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled

Bits	Fields	RW	Descriptions
1	CC1IE	RW	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
0	UIE	RW	Update interrupt enable 0: disabled 1: enabled

16.5.5. Interrupt flag register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:13	Reserved	R	Must be kept at reset value.
12	CC4OF	RC_W0	Channel 4 over capture flag Refer to CC1OF description
11	CC3OF	RC_W0	Channel 3 over capture flag Refer to CC1OF description
10	CC2OF	RC_W0	Channel 2 over capture flag Refer to CC1OF description
9	CC1OF	RC_W0	Channel 1 over capture flag When Channel 1 is configured in input mode, this flag is set by hardware when a capture event occurs while CC1IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8	Reserved	R	Must be kept at reset value.
7	BIF	RC_W0	Break interrupt flag This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active. 0: No active level break has been detected. 1: An active level has been detected.
6	TIF	RC_W0	Trigger interrupt flag This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on trigger input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on trigger input generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	COMIF	RC_W0	Channel commutation interrupt flag This flag is set by hardware when channel’s commutation event occurs, and cleared by software 0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred
4	CC4IF	RC_W0	Channel 4‘s capture/compare interrupt flag Refer to CC1IF description
3	CC3IF	RC_W0	Channel 3‘s capture/compare interrupt flag Refer to CC1IF description
2	CC2IF	RC_W0	Channel 2‘s capture/compare interrupt flag

Bits	Fields	RW	Descriptions
			Refer to CC1IF description
1	CC1IF	RC_W0	Channel 1's capture/compare interrupt flag This flag is set by hardware and cleared by software. When Channel 1 is in input mode, this flag is set when a capture event occurs. When Channel 1 is in output mode, this flag is set when a compare event occurs. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UIF	RC_W0	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred

16.5.6. Software event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:8	Reserved	R	Must be kept at reset value.
7	BG	W	Break event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the MOE bit is cleared and BIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event
6	TG	W	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TIF flag in TIMx_SR register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	COMG	W	Channel commutation event generation This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CCxE, CCxNE and OCxM bits) are updated based on the value of CCPC (in the TIMx_CR2). 0: No affect 1: Generate channel's c/c control update event
4	CC4G	W	Channel 4's capture or compare event generation Refer to CC1G description
3	CC3G	W	Channel 3's capture or compare event generation Refer to CC1G description
2	CC2G	W	Channel 2's capture or compare event generation Refer to CC1G description
1	CC1G	W	Channel 1's capture or compare event generation This bit is set by software in order to generate a capture or compare event in Channel 1, it is automatically cleared by hardware. When this bit is set, the CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if Channel 2 is configured in input mode, the current value of the counter is captured in TIMx_CCR1 register, and the CC1OF flag is set if the CC1IF flag was already high.

Bits	Fields	RW	Descriptions
			0: No generate a Channel 2 capture or compare event 1: Generate a Channel 2 capture or compare event
0	UG	W	Update event generation This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

16.5.7. Channel control register 0 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

Output compare mode:

Bits	Fields	RW	Descriptions
31-16	-	R	Reserved bits
15	OC2CE	RW	Output Compare 2 clear enable Refer to OC1CE description
14:12	OC2M[2:0]	RW	Output Compare 2 mode Refer to OC1M description
11	OC2PE	RW	Output Compare 2 preload enable Refer to OC1PE description
10	OC2FE	RW	Output Compare 2 fast enable Refer to OC1PE description
9:8	CC2S[1:0]	RW	Capture/Compare 2 selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC2E bit in TIMx_CCER register is reset). 00: Channel 2 is configured as output 01: Channel 2 is configured as input, IC2 is mapped on TI2 10: Channel 2 is configured as input, IC2 is mapped on TI2 11: Channel 2 is configured as input, IC2 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.
7	OC1CE	RW	Output Compare 1 clear enable When this bit is set, the O1CPRE signal is cleared when High level is detected on ETRF input. 0: Channel 1 output compare clear disable 1: Channel 1 output compare clear enable
6:4	OC1M[2:0]	RW	Output Compare 1 mode This bit-field controls the behavior of the output reference signal O1CPRE which drives CH1_O and CH1_ON . O1CPRE is active high, while CH1_O and CH1_ON active level depends on CC1P and CC1NP bits. 000: Frozen. The O1CPRE signal keeps stable, independent of the comparison between the register TIMx_CCR1 and the counter TIMx_CNT. 001: Set the channel output. O1CPRE signal is forced high when the counter matches the output compare register TIMx_CCR1.

Bits	Fields	RW	Descriptions
			010: Clear the channel output. O1CPRE signal is forced low when the counter matches the output compare register TIMx_CCR1. 011: Toggle on match. O1CPRE toggles when the counter matches the output compare register TIMx_CCR1. 100: Force low. O1CPRE is forced low level. 101: Force high. O1CPRE is forced high level. 110: PWM mode1. When counting up, O1CPRE is high as long as the counter is smaller than TIMx_CCR1 else low. When counting down, O1CPRE is low as long as the counter is larger than TIMx_CCR1 else high. 111: PWM mode2. When counting up, O1CPRE is low as long as the counter is smaller than TIMx_CCR1 else high. When counting down, O1CPRE is high as long as the counter is larger than TIMx_CCR1 else low. Note: When configured in PWM mode, the O1CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes. These bits cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC1S bit-filed is 00(COMPARE MODE).
3	OC1PE	RW	Output Compare 1 preload enable When this bit is set, the shadow register of TIMx_CCR1 register, which updates at each update event, will be enabled. 0: Channel 1 output compare shadow disable 1: Channel 1 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (OPM bit in TIMx_CR1 register is set). This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC1S bit-filed is 00.
2	OC1FE	RW	Output Compare 1 fast enable When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH1_O is set to the compare level independently from the result of the comparison. 0: Channel 1 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH1_O output is 5 clock cycles. 1: Channel 1 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH1_O output is 3 clock cycles.
1:0	CC1S[1:0]	RW	Capture/Compare 1 selection This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC1E bit in TIMx_CCER register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IC1 is connected to TI1FP1 10: Channel 1 is configured as input, IC1 is connected to TI2FP1 11: Channel 1 is configured as input, IC1 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.

Input capture mode:

Bits	Fields	RW	Descriptions

Bits	Fields	RW	Descriptions
15:12	IC2F[3:0]	RW	Input capture 2 filter Refer to IC1F description
11:10	IC2PSC[1:0]	RW	Channel 2 input capture prescaler Refer to IC1PSC description
9:8	CC2S[1:0]	RW	Capture/Compare 2 selection Same as Output compare mode
7:4	IC1F[3:0]	RW	Input capture 1 filter An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample TI1 input signal and the length of the digital filter applied to TI1. 0000: Filter disabled, f_{SAMP}=f_{DTS}, N=1 0001: f_{SAMP}=f_{TIMER_CK}, N=2 0010: f_{SAMP}=f_{TIMER_CK}, N=4 0011: f_{SAMP}=f_{TIMER_CK}, N=8 0100: f_{SAMP} =f_{DTS}/2, N=6 0101: f_{SAMP} =f_{DTS}/2, N=8 0110: f_{SAMP} =f_{DTS}/4, N=6 0111: f_{SAMP} =f_{DTS}/4, N=8 1000: f_{SAMP} =f_{DTS}/8, N=6 1001: f_{SAMP} =f_{DTS}/8, N=8 1010: f_{SAMP} =f_{DTS}/16, N=5 1011: f_{SAMP} =f_{DTS}/16, N=6 1100: f_{SAMP} =f_{DTS}/16, N=8 1101: f_{SAMP} =f_{DTS}/32, N=5 1110: f_{SAMP} =f_{DTS}/32, N=6 1111: f_{SAMP} =f_{DTS}/32, N=8
3:2	IC1PSC[1:0]	RW	Input capture 1 prescaler This bit-field specifies the factor of the prescaler on Channel 1 input. The prescaler is reset when CC1E bit in TIMx_CCER register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4channel input edges 11: Capture is done every 8 channel input edges
1:0	CC1S[1:0]	RW	Capture/Compare 1 Selection Same as Output compare mode

16.5.8. Channel control register 1 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Output compare mode:

Bits	Fields	RW	Descriptions
31-16	-	R	Reserved bits
15	OC4CE	RW	Output compare 4 clear enable Refer to OC1CE description
14:12	OC4M[2:0]	RW	Channel 4 compare output control Refer to OC1M description
11	OC4PE	RW	Output compare 4 preload enable Refer to OC1PE description

Bits	Fields	RW	Descriptions
10	OC4FE	RW	Output compare 4 fast enable Refer to OC1PE description
9:8	CC4S[1:0]	RW	Capture/Compare 4 selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC4E bit in TIMx_CCER register is reset). 00: Channel 4 is configured as output 01: Channel 4 is configured as input, IC4 is mapped on TI4 10: Channel 4 is configured as input, IC4 is mapped on TI3 11: Channel 4 is configured as input, IC4 is connected to TRC, This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.
7	OC3CE	RW	Output compare 3 clear enable When this bit is set, the O3CPRE signal is cleared when High level is detected on ETRF input. 0: Channel 3 output compare clear disable 1: Channel 3 output compare clear enable
6:4	OC3M[2:0]	RW	Output compare 3 mode This bit-field controls the behavior of the output reference signal O3CPRE which drives CH3_O and CH3_ON . O3CPRE is active high, while CH3_O and CH3_ON active level depends on CC3P and CC3NP bits. 000: Frozen. The O3CPRE signal keeps stable, independent of the comparison between the output compare register TIMx_CCR3 and the counter TIMx CNT . 001: Set high on match. O3CPRE signal is forced high when the counter matches the output compare register TIMx_CCR3. 010: Set low on match. O3CPRE signal is forced low when the counter matches the output compare register TIMx_CCR3. 011: Toggle on match. O3CPRE toggles when the counter matches the output compare register TIMx_CCR3. 100: Force low. O3CPRE is forced low level. 101: Force high. O3CPRE is forced high level. 110: PWM mode 1. When counting up, O3CPRE is high as long as the counter is smaller than TIMx_CCR3 else low. When counting down, O3CPRE is low as long as the counter is larger than TIMx_CCR3 else high. 111: PWM mode 2. When counting up, O3CPRE is low as long as the counter is smaller than TIMx_CCR3 else high. When counting down, O3CPRE is high as long as the counter is larger than TIMx_CCR3 else low. Note: When configured in PWM mode, the O3CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC3S bit-filed is 00(COMPARE MODE).
3	OC3PE	RW	Output compare 3 preload enable When this bit is set, the shadow register of TIMx_CCR3 register, which updates at each update event will be enabled. 0: Channel 3 output compare shadow disable 1: Channel 3 output compare shadow enable Note: The PWM mode can be used without validating the shadow register only in single pulse mode (OPM bit in TIMx_CR1 register is set) .

Bits	Fields	RW	Descriptions
			This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC1S bit-filed is 00.
2	OC3FE	RW	Output compare 3 fast enable When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH3_O is set to the compare level independently from the result of the comparison. 0: Channel 3 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH3_O output is 5 clock cycles. 1: Channel 3 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH3_O output is 3 clock cycles.
1:0	CC3S[1:0]	RW	Capture/Compare 3 selection This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC3E bit in TIMx_CCER register is reset).). 00: Channel 3 is configured as output 01: Channel 3 is configured as input, IC3 is mapped on TI3 10: Channel 3 is configured as input, IC3 is mapped on TI4 11: Channel 3 is configured as input, IC3 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.

Input capture mode:

Bits	Fields	RW	Descriptions
15:12	IC4F[3:0]	RW	Input capture 4 filter Refer to IC3F description
11:10	IC4PSC[1:0]	RW	Input capture 4 prescaler Refer to IC3PSC description
9:8	CC4S[1:0]	RW	Capture/Compare 4 selection Same as Output compare mode
7:4	IC3F[3:0]	RW	Input capture 3 filter An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample TI3 input signal and the length of the digital filter applied to TI3. 0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1 0001: $f_{SAMP}=f_{TIMER\_CK}$, N=2 0010: $f_{SAMP}=f_{TIMER\_CK}$, N=4 0011: $f_{SAMP}=f_{TIMER\_CK}$, N=8 0100: $f_{SAMP}=f_{DTS}/2$, N=6 0101: $f_{SAMP}=f_{DTS}/2$, N=8 0110: $f_{SAMP}=f_{DTS}/4$, N=6 0111: $f_{SAMP}=f_{DTS}/4$, N=8 1000: $f_{SAMP}=f_{DTS}/8$, N=6 1001: $f_{SAMP}=f_{DTS}/8$, N=8 1010: $f_{SAMP}=f_{DTS}/16$, N=5 1011: $f_{SAMP}=f_{DTS}/16$, N=6 1100: $f_{SAMP}=f_{DTS}/16$, N=8 1101: $f_{SAMP}=f_{DTS}/32$, N=5 1110: $f_{SAMP}=f_{DTS}/32$, N=6 1111: $f_{SAMP}=f_{DTS}/32$, N=8
3:2	IC3PSC[1:0]	RW	Input capture 3 prescaler

Bits	Fields	RW	Descriptions
			This bit-field specifies the factor of the prescaler on Channel 3 input. The prescaler is reset when CC3E bit in TIMx_CCER register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4 channel input edges 11: Capture is done every 8 channel input edges
1:0	CC3S[1:0]	RW	Capture/compare 3 selection Same as Output compare mode

16.5.9. Channel control register 2 (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:14	Reserved	R	Must be kept at reset value
13	CC4P	RW	Channel 4 capture/compare function polarity Refer to CC1P description
12	CC4E	RW	Channel 4 capture/compare function enable Refer to CC1E description
11	CC3NP	RW	Channel 3 complementary output polarity Refer to CC1NP description
10	CC3NE	RW	Channel 3 complementary output enable Refer to CC1NE description
9	CC3P	RW	Channel 3 capture/compare function polarity Refer to CC1P description
8	CC3E	RW	Channel 3 capture/compare function enable Refer to CC1E description
7	CC2NP	RW	Channel 2 complementary output polarity Refer to CC1NP description
6	CC2NE	RW	Channel 2 complementary output enable Refer to CC1NE description
5	CC2P	RW	Channel 2 capture/compare function polarity Refer to CC1P description
4	CC2E	RW	Channel 2 capture/compare function enable Refer to CC1E description
3	CC1NP	RW	Channel 1 complementary output polarity When Channel 1 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 1 active high 1: Channel 1 active low Note: This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 or 10 and CC1S [1:0] bit-filed in TIMx_CCMR1 register is 00 .
2	CC1NE	RW	Channel 1 complementary output enable When Channel 1 is configured in output mode, setting this bit enables the complementary output in channel0. 0: Channel 1 complementary output disabled 1: Channel 1 complementary output enabled
1	CC1P	RW	Channel 1 capture/compare function polarity When Channel 1 is configured in output mode, this bit specifies the output signal polarity.

Bits	Fields	RW	Descriptions
			0: Channel 1 active high 1: Channel 1 active low When Channel 1 is configured in input mode, this bit specifies the TI1 signal polarity. 0: TI1 is non-inverted. Input capture is done on a rising edge of TI1. When used as extern trigger, TI1 is non-inverted. 1: TI1 is inverted. Input capture is done on a falling edge of TI1. When used as extern trigger, TI1 is inverted. Note: This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 or 10.
0	CC1E	RW	Channel 1 capture/compare function enable When Channel 1 is configured in output mode, setting this bit enables CH1_O signal in active state. When Channel 1 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 1 disabled 1: Channel 1 enabled

16.5.10. Counter register (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31-20	-	R	Reserved bits
19:0	CNT[19:0]	RW	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

16.5.11. Prescaler register (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15:0	PSC[15:0]	RW	Prescaler value of the counter clock The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

16.5.12. Counter auto reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	ARR[19:0]	RW	Counter auto reload value This bit-filed specifies the auto reload value of the counter. Note: When the timer is configured in input capture mode, this register must be

			configured a non-zero value (such as 0xFFFF) which is larger than user expected value.

16.5.13. Counter repetition register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:8	Reserved	R	Must be kept at reset value.
7:0	REP[7:0]	RW	Counter repetition value This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled.

16.5.14. Channel 1 capture/compare value register (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR1[19:0]	RW	Capture or compare value of channel0 When Channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

16.5.15. Channel 2 capture/compare value register (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR2[19:0]	RW	Capture or compare value of channel1 When Channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

16.5.16. Channel 3 capture/compare value register (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR3[19:0]	RW	Capture or compare value of Channel 3 When Channel 3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

16.5.17. Channel 4 capture/compare value register (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR4[19:0]	RW	Capture or compare value of Channel 4 When channel4 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 4 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

16.5.18. Complementary channel protection register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15	MOE	RW	Main output enable This bit s set by software or automatically by hardware depending on the AOE bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CCxE, CCxNE in TIMx_CCER register) have been set. 0: Channel outputs are disabled or forced to idle state. 1: Channel outputs are enabled.
14	AOE	RW	Automatic output enable This bit specifies whether the MOE bit can be set automatically by hardware. 0: MOE can be not set by hardware. 1: MOE can be set by hardware automatically at the next update event, if the break input is not active. Note: This bit can be modified only when LOCK [1:0] bit-filed in TIMx_BDTR register is 00.
13	BKP	RW	Break polarity

Bits	Fields	RW	Descriptions
			This bit specifies the polarity of the BRKIN input signal. 0: BRKIN input active low 1: BRKIN input active high
12	BKE	RW	Break enable This bit can be set to enable the BRKIN and CKM clock failure event inputs. 0: Break inputs disabled 1: Break inputs enabled Note: This bit can be modified only when LOCK [1:0] bit-filed in TIMx_BDTR register is 00.
11	OSSR	RW	Off-state selection for Run mode When MOE bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode. 0: When MOE bit is set, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When MOE bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CCxE/CCxNE bits in TIMx_CCER register. Note: This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 10 or 11.
10	OSSI	RW	Off-state selection for Idle mode When MOE bit is reset, this bit specifies the output state for the channels which has been configured in output mode. 0: When MOE bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When MOE bit is reset, he channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CCxE/CCxNE bits in TIMx_CCER register. Note: This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 10 or 11.
9:8	LOCK[1:0]	RW	Lock configuration This bit-filed specifies the write protection property of registers. 00: protect disable. No write protection. 01: LOCK mode 0.The ISOx/ISOxN bits in TIMx_CR2 register and the BKE/BKP/AOE/DTG bits in TIMx_BDTR register are writing protected. 10: LOCK mode 1. In addition of the registers in LOCK mode 0, the CCxP/CCxNP bits in TIMx_CCER register (if related channel is configured in output mode) and the OSSR/OSSI bits in TIMx_BDTR register are writing protected. 11: LOCK mode 2. In addition of the registers in LOCK mode 1, the OCxM/ OCxPE bits in TIMx_CCMR1/1 registers (if the related channel is configured in output) are writing protected. Note: This bit-field can be written only once after the reset. Once the TIMx_BDTR register has been written, this bit-field will be writing protected.
7:0	DTG[7:0]	RW	Dead time configure This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTG value and the duration of dead-time is as follow:

Bits	Fields	RW	Descriptions
			DTG [7:5] =3’b0xx: DTvalue = DTG [7:0] * tDT, tDT = tDTS. DTG [7:5] =3’b10x: DTvalue = (64+DTG [5:0]) * tDT, tDT = tDTS * 2. DTG [7:5] =3’b110: DTvalue = (32+DTG [4:0]) * tDT, tDT = tDTS * 8. DTG [7:5] =3’b111: DTvalue = (32+DTG [4:0]) * tDT, tDT = tDTS * 16. Note: This bit can be modified only when LOCK [1:0] bit-filed in TIMx_BDTR register is 00.

17. General-Purpose Timers (TIM2-TIM4)

17.1. Overview

The general-purpose timer modules (TIM2-TIM4) is a four-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general-purpose timer has a 20-bit counter that can be used as an unsigned counter.

In addition, the general-purpose timers can be programmed and be used for counting, their external events can be used to drive other timers.

The general-purpose timers (TIM2-TIM4) are completely independent with each other, but they may be synchronized to provide a larger timer with their counters increment in unison.

17.2. Characteristics

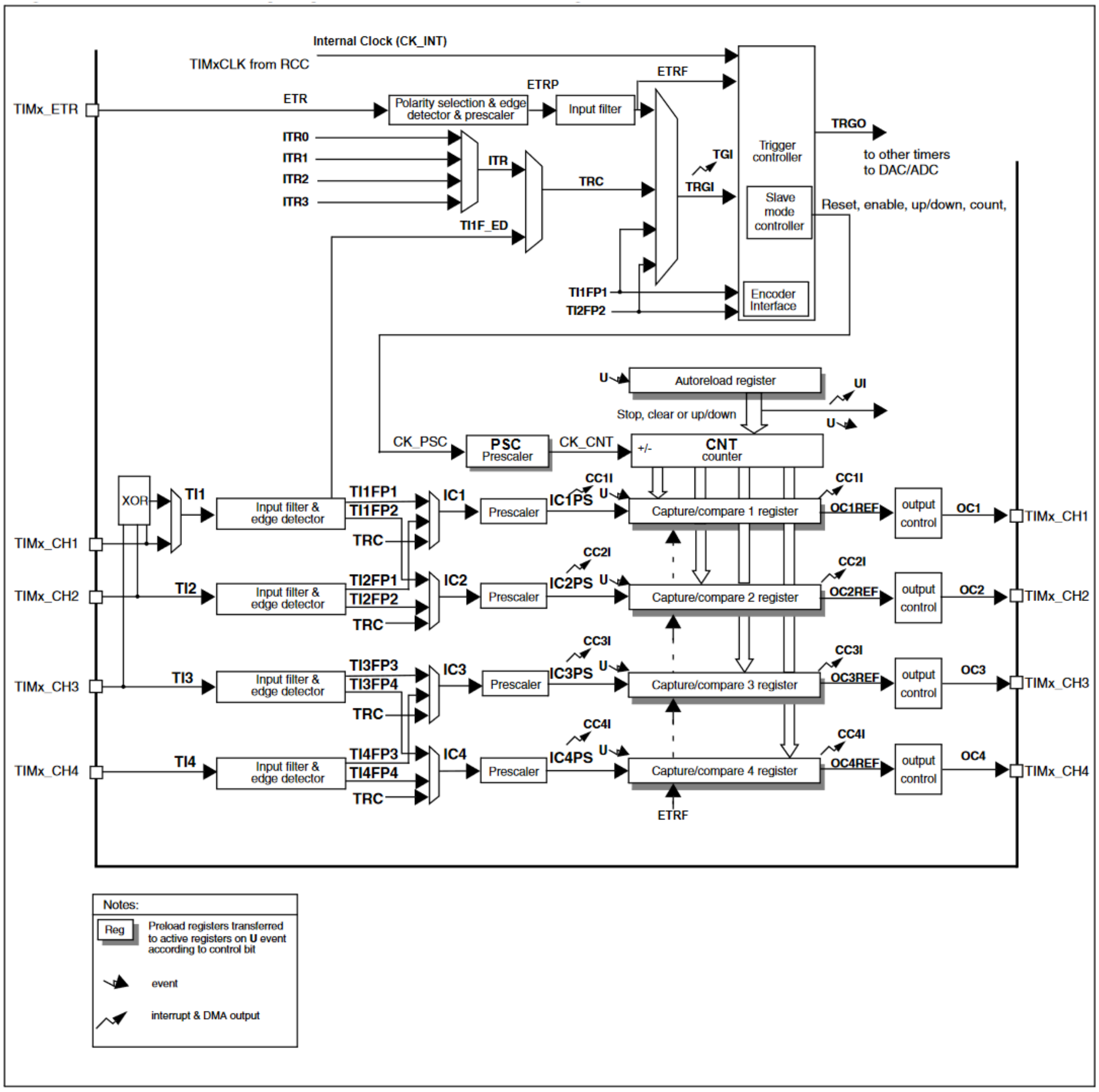
- Total channel number: 4.
- Counter width: 20bit.
- Source of counter clock is select-able:
  - internal clock
  - internal trigger
  - external input
  - external trigger
- Multiple counter modes: count up, count down, count up/down.
- Quadrature Decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit. The factor can be changed on the go.
- Each channel is user-configurable:
  - input capture mode
  - output compare mode
  - programmable PWM mode
  - single pulse mode
- Auto reload function.
- Interrupt output or DMA request on:
  - update
  - trigger event
  - compare/capture event

- Daisy chaining of timer modules allows a single timer to initiate multiple timers .
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

17.3. Block diagram

The figure 17-1 provides the details of the internal configuration of the general-purpose timer.

Figure 17-1. General-purpose timer block diagram



17.4. Function overview

17.4.1. Clock selection

The general-purpose timer has the capability of being clocked by either the TIMER_CK or an alternate clock source controlled by SMS (TIMx_SMCR bit [2:0]).

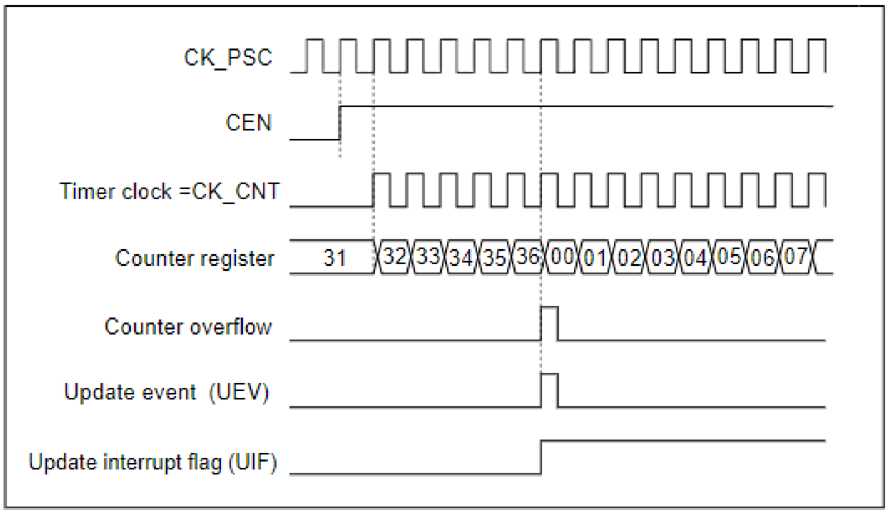
- SMS [2:0] == 3'b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCC.

The default clock source is the CK_TIMER for driving the counter prescaler when the slave mode is disabled (SMS [2:0] == 3'b000). When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, which drives counter's prescaler to count, is equal to CK_TIMER which is from RCC.

If the slave mode controller is enabled by setting SMS [2:0] in the TIMx_SMCR register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TS[2:0] in the TIMx_SMCR register , details as follows. When the slave mode selection bits SMS[2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER_CK is the counter prescaler driving clock source.

Figure 17-2. Normal mode, internal clock divided by 1



- SMS[2:0] == 3'b111 (external clock mode 1). External input pin is selected as timer clock Source.

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMx_CH1/TIMx_CH2. This mode can be selected by setting SMS[2:0] to 0x7 and the TS[2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITR0/1/2/3. This mode can be selected by setting SMS [2:0] to 0x7 and the TS [2:0] to 0x0, 0x1, 0x2 or 0x3.

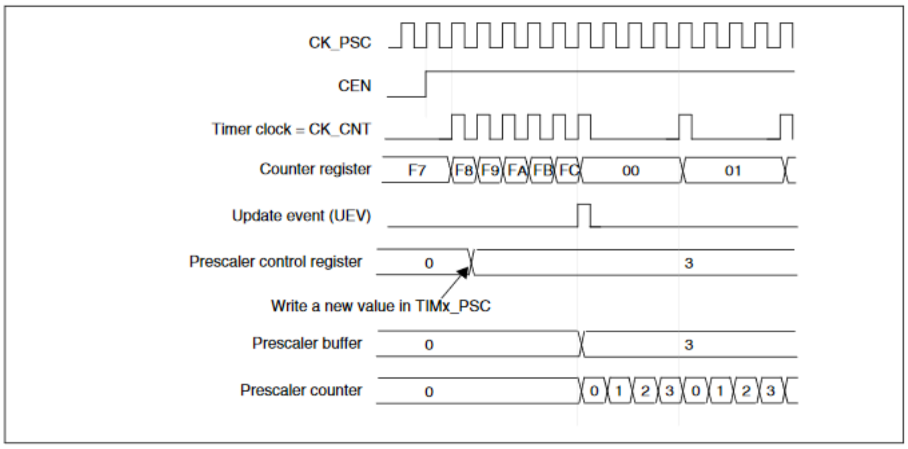
- ECE== 1'b1 (external clock mode 2). External input is selected as timer clock source (ETR).

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETR. This mode can be selected by setting the ECE bit in the TIMx_SMCR register to 1. The other way to select the ETR signal as the clock source is to set the SMS [2:0] to 0x7 and the TS [2:0] to 0x7 respectively. Note that the ETR signal is derived from the ETR pin sampled by a digital filter. When the ETR signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETR signal rising edge to clock the counter prescaler.

17.4.2. Prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMx_PSC) which can be changed on the go but is taken into account at the next update event.

Figure 17-3. Counter timing diagram with prescaler division change from 1 to 4



17.4.3. Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMx_ARR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after (TIMx_RCR+1) times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit DIR in the TIMx_CR1 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UG bit in the TIMx_EGR register, the counter value will be initialized to 0 and generates an update event.

If set the UDIS bit in TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMx_ARR=0x36.

Figure 17-4. Up-counter timechart, PSC=0

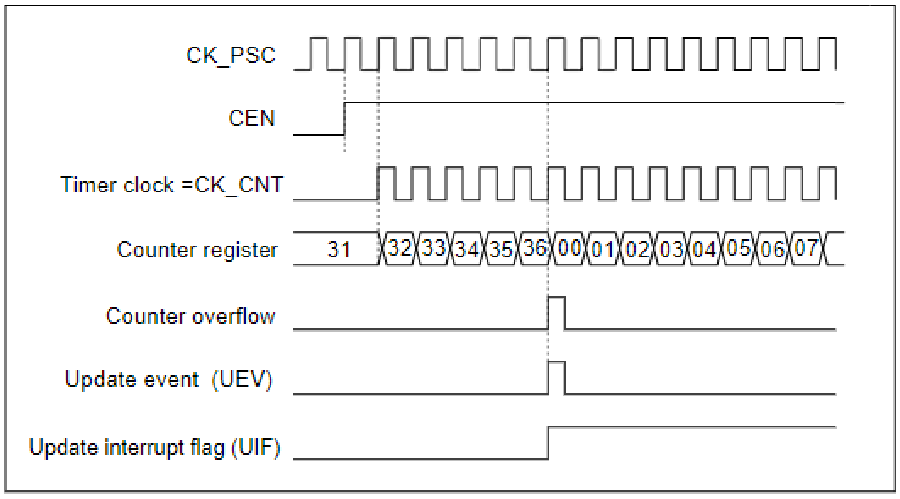


Figure 17-5. Up-counter timechart, PSC=1

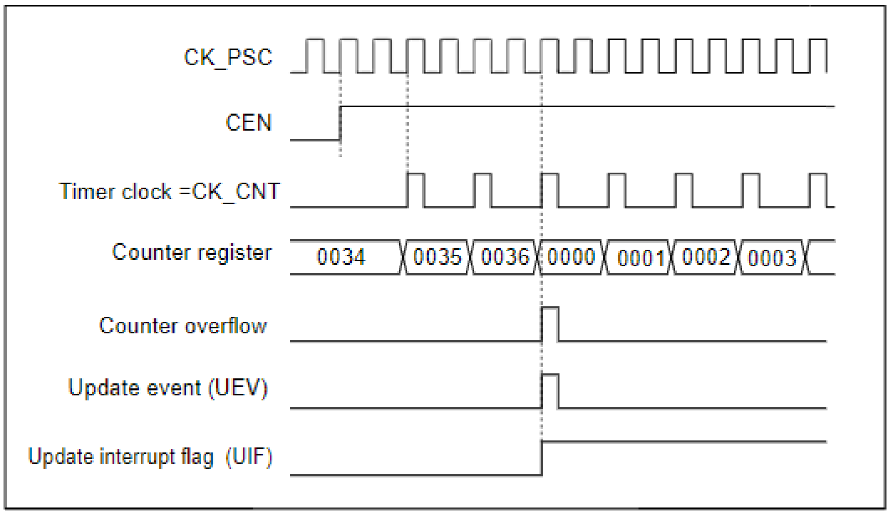


Figure 17-6. Up-counter timechart, PSC=3

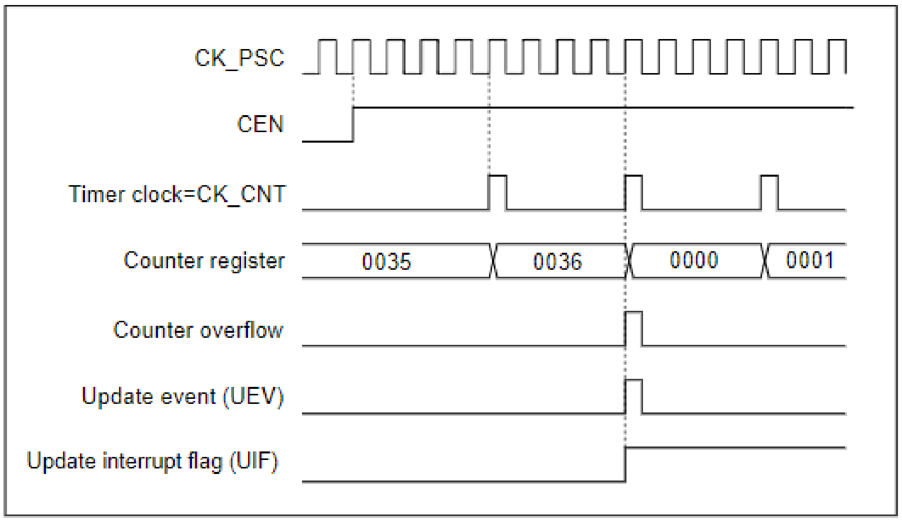


Figure 17-7. Up-counter timechart, PSC=N

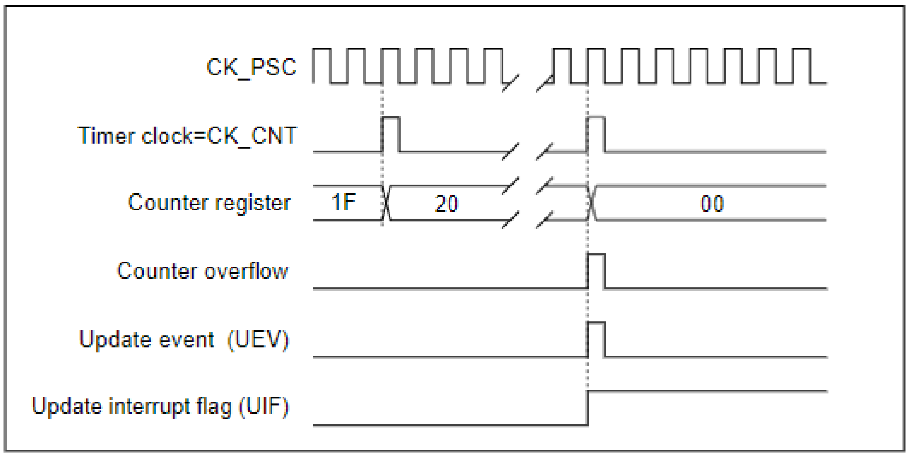
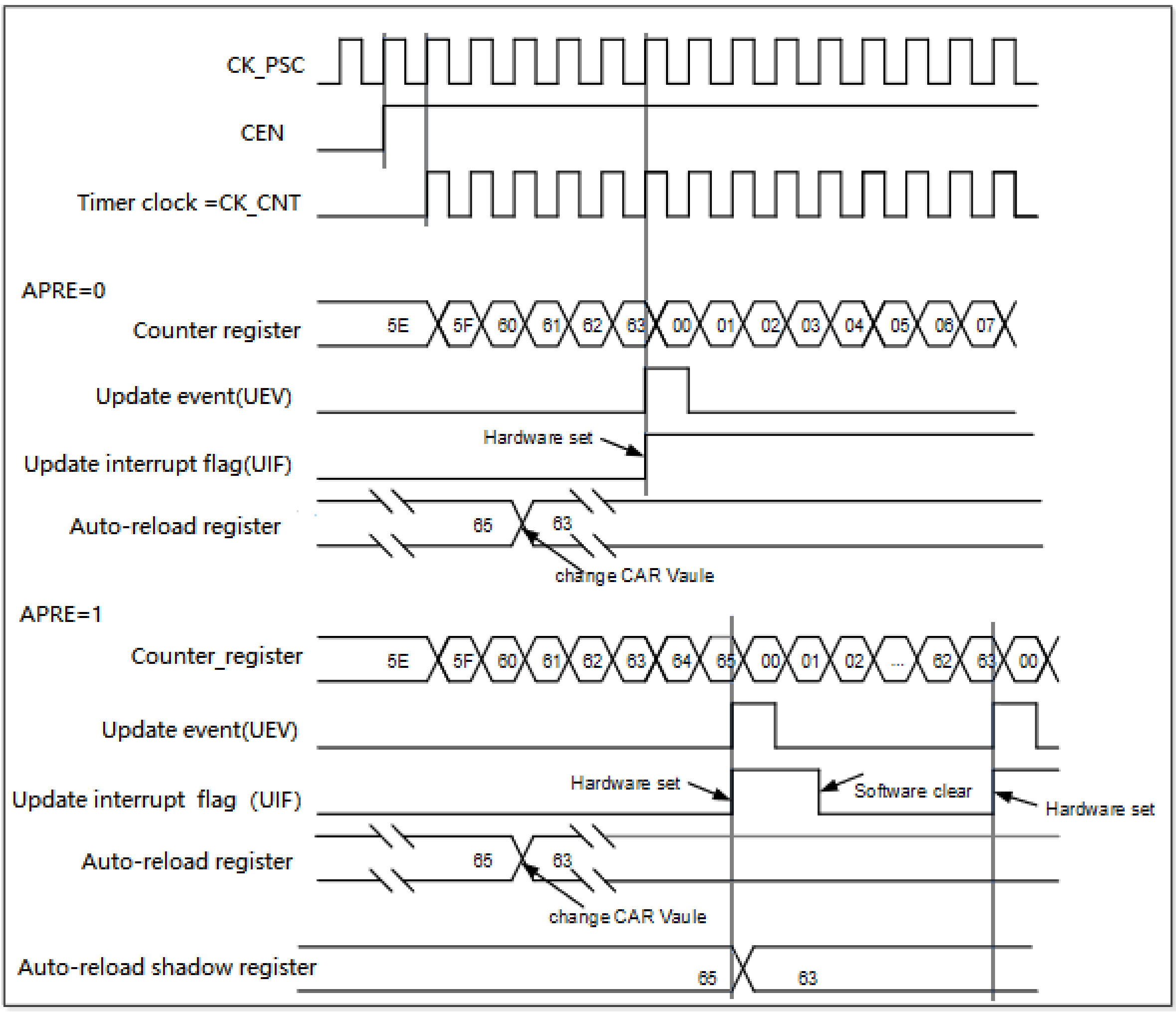


Figure 17-8. Up-counter timechart, change TIMx_ARR on the go



17.4.4. Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIM1_ARR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter restarts to count again from the counter-reload value. If the repetition counter is set, the update event will be generated after (TIMx_RCR+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMx_CR1 register should be set to 1 for the down-counting mode.

When the update event is set by the UG bit in the TIMx_EGR register, the counter value will be initialized to the counter-reload value and generates an update event.

If set the UDIS bit in TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior in different clock frequencies when TIMx_ARR=0x36.

Figure 17-9. Down-counter timechart, PSC=0

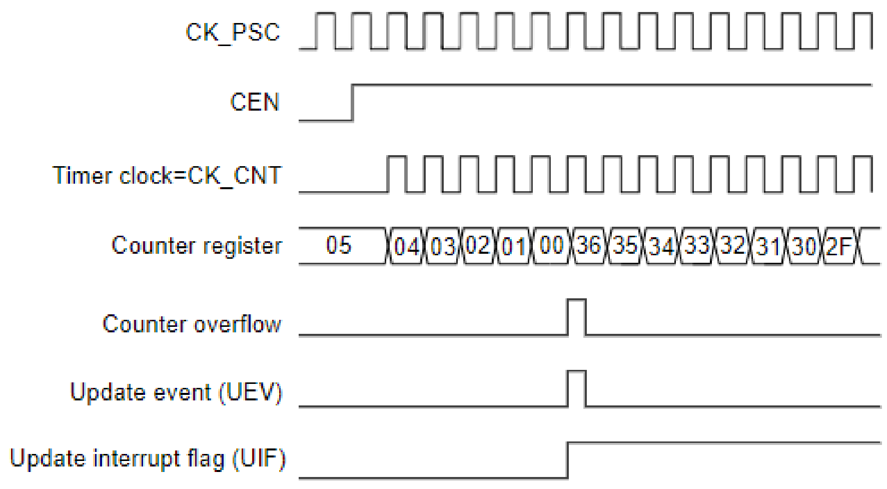


Figure 17-10. Down-counter timechart, PSC=1

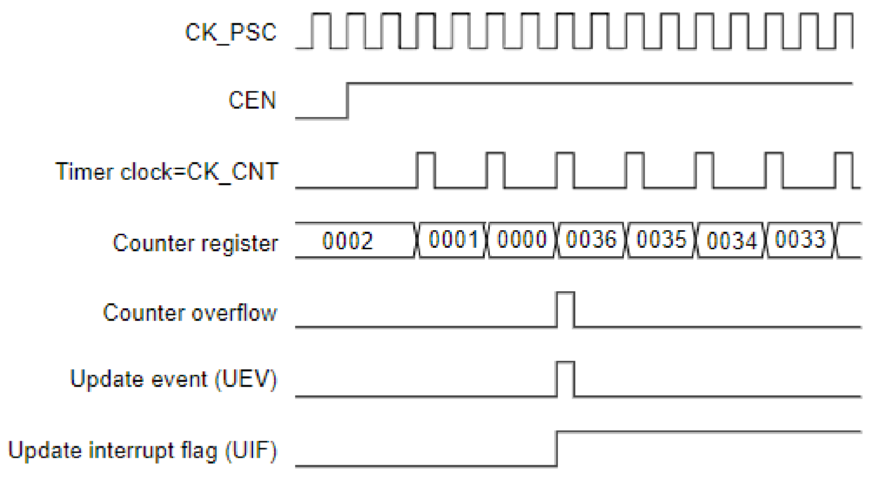


Figure 17-11. Down-counter timechart, PSC=3

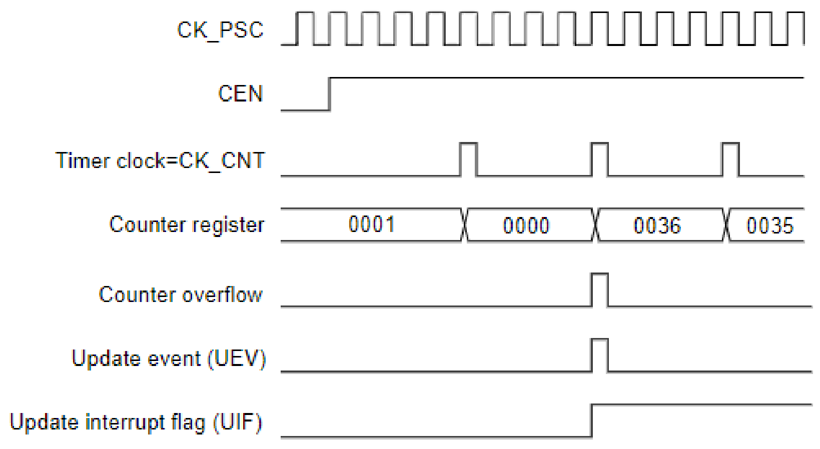


Figure 17-12. Down-counter timechart, PSC=N

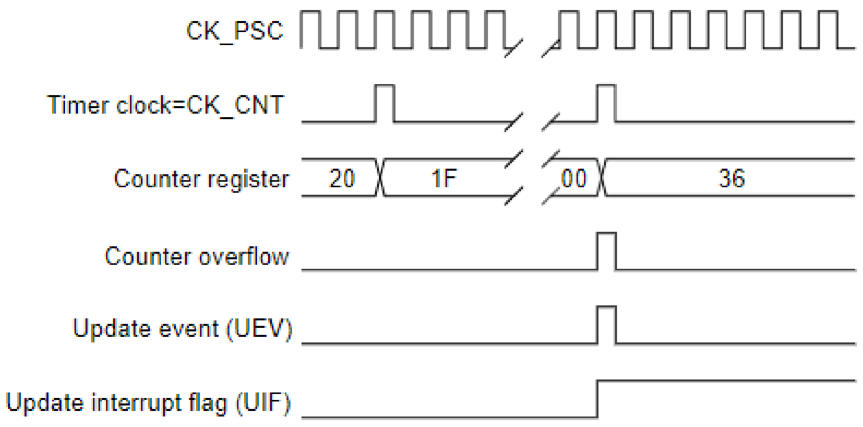
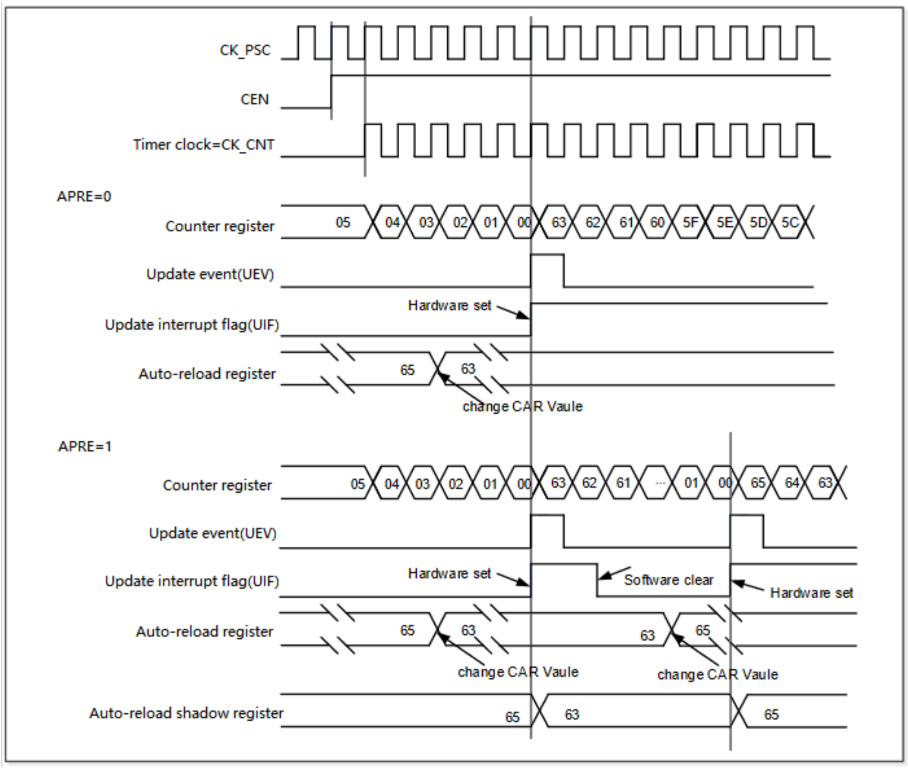


Figure 17-13. Down-counter timechart, change TIMx_ARR on the go (APRE=0)



17.4.5. Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The Timer module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMx_CR1 register is read-only and indicates the counting direction when in the center-aligned mode. The counting direction is updated by hardware automatically.

Setting the UG bit in the TIMx_EGR register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

The UIF bit in the TIMx_SR register can be set to 1 either when an underflow event or an overflow event occurs. While the CCxIF bit is associated with the value of CMS in TIMx_CR1. Refer to Figure 17-14 for the details.

If set the UDIS bit in the TIMx_CR1 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register, prescaler register) are updated.

The figures below show some examples of the counter behavior for different clock frequencies.

Figure 17-14. Center-aligned counter timechart, PSC=0

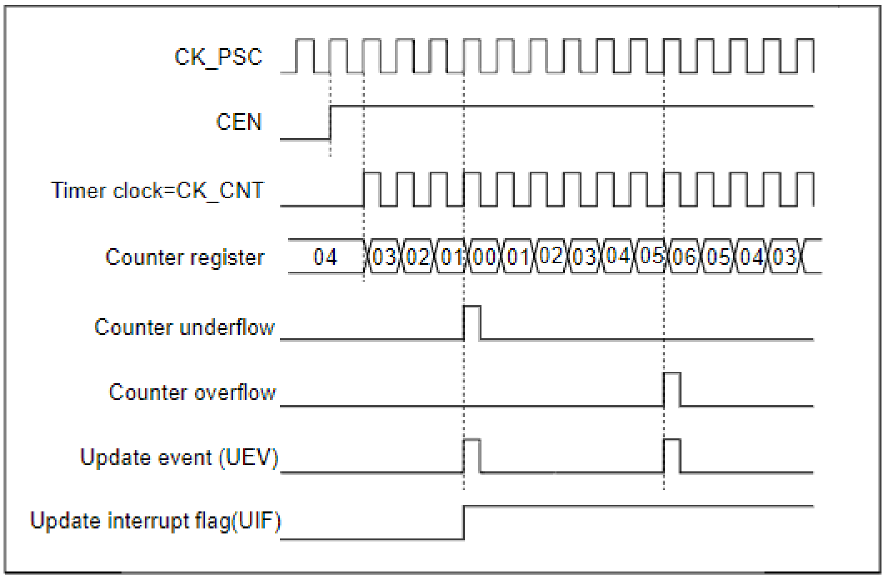


Figure 17-15. Center-aligned counter timechart, PSC=1

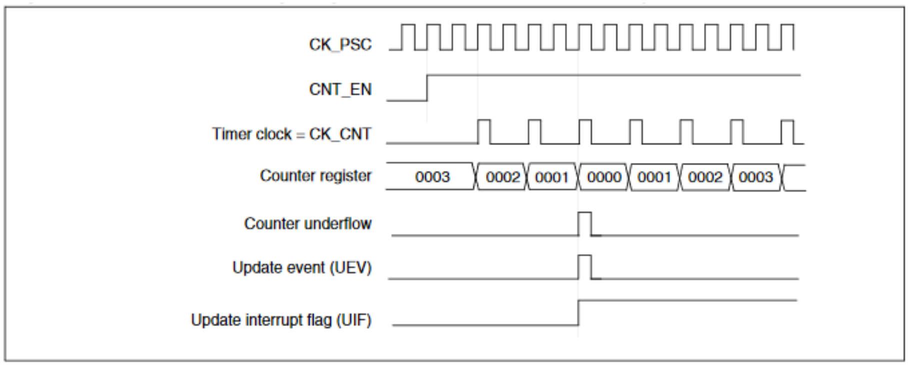


Figure 17-16. Center-aligned counter timechart, PSC=3

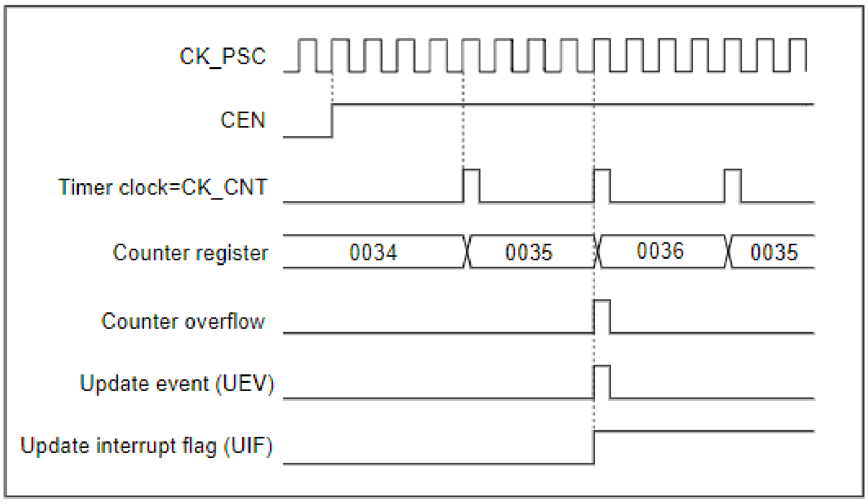


Figure 17-17. Center-aligned counter timechart, PSC=N

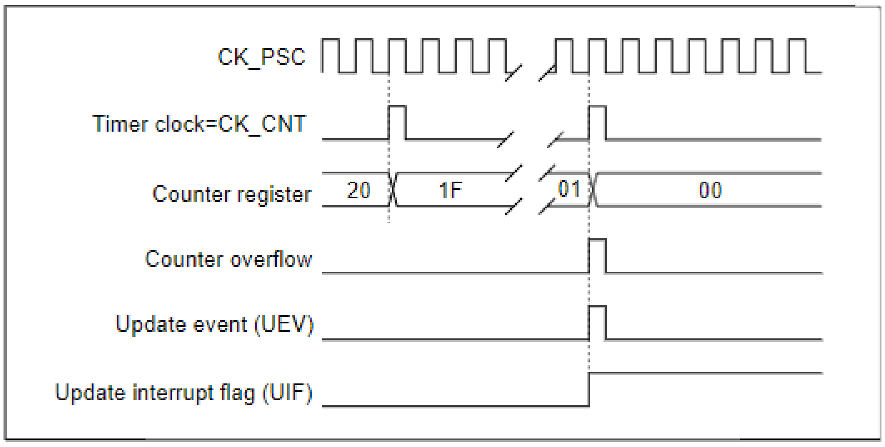


Figure 17-18. Center-aligned counter timechart, update event with ARPE=1 (counter underflow)

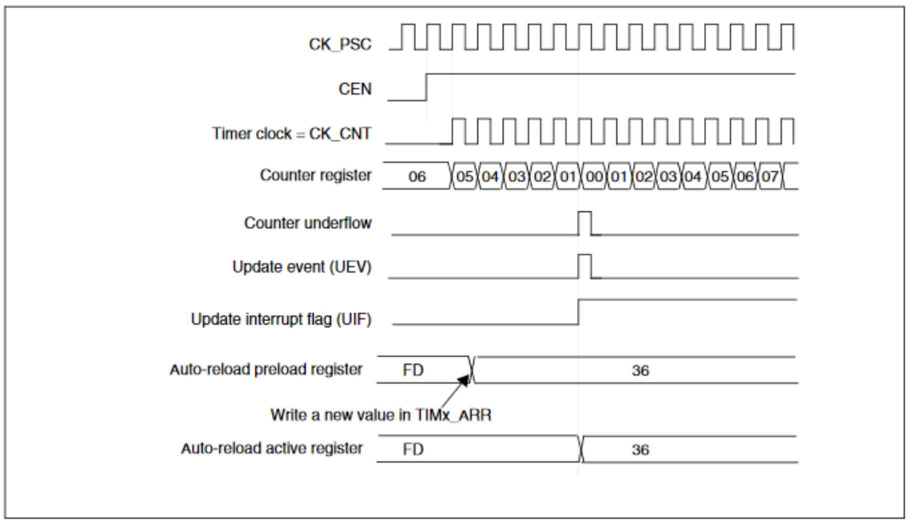
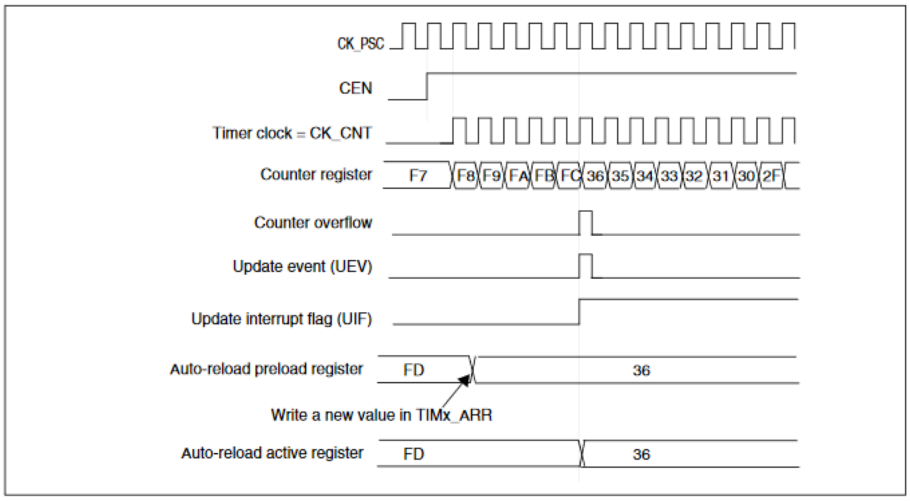


Figure 17-19. Center-aligned counter timechart, update event with ARPE=1 (counter overflow)



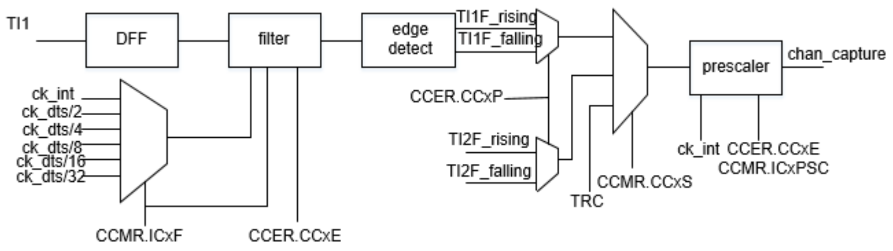
17.4.6. Capture/compare channels

The general-purpose timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Input capture mode**

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMx_CCRx register, at the same time the CCxIF bit is set and the channel interrupt is generated if enabled by CCxIE = 1.

Figure 17-20. Input capture logic



One of channels' input signals (TIx) can be chosen from the TIMx_CHx signal or the Excusive-OR function of the TIMx_CH1, TIMx_CH2 and TIMx_CH3 signals. First, the channel input signal (TIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CCxP. One more selector is for the other channel and trig, controlled by CCxS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CCRx will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter configuration. (ICxF in TIMx_CCMR1)

Based on the input signal and requested signal quality, configure compatible ICxF.

Step2: Edge selection. (CCxP/CCxNP in TIMx_CCER)

Rising or falling edge, choose one by CCxP/CCxNP.

Step3: Capture source selection. (CCxS in TIMx_CCMR1)

As soon as you select one input capture source by CCxS, you have set the channel to input mode (CCxS!=0x0) and TIMx_CCRx cannot be written any more.

Step4: Interrupt enable. (CCxIE and CCxDE in TIMx_DIER)

Enable the related interrupt enable; you can got the interrupt and DMA request.

Step5: Capture enables. (CCxE in TIMx_CCER)

Result: when your wanted input signal is got, TIMx_CCRx will be set by counter's value. And CCxIF is also asserted. If the CCxIF has been high, the CCxOF will also be asserted. The interrupt and DMA request will be asserted based on the configuration of CCxIE and CCxDE in TIMx_DIER.

Direct generation: if you want to generate a DMA request or Interrupt, you can set CCxG by software directly.

The input capture mode can also be used for pulse width measurement from signals on the TIMx_CHx pins. For example, PWM signal connect to TI1 input. Select Channel 1 capture signals to TI1 by setting CC1S to 2'b01 in the channel control register (TIMx_CCMR1) and set capture on rising edge. Select Channel 2 capture signal to TI1 by setting CC2S to 2'b10 in the channel control register (TIMx_CCMR1) and set capture on falling edge. The counter set to restart mode and restart on Channel 1 rising edge. Then the TIMx_CCR1 can measure the PWM period and the TIMx_CCR2 can measure the PWM duty.

- **Output compare mode**

In output compare mode, the TIMx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CCRx register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on OCxM. When the counter reaches the value in the CCRx register, the CCxIF bit is set and the channel (n) interrupt is generated if CCxIE = 1. And the DMA request will be asserted, if CCxDE = 1.

So the process can be divided to several steps as below:

Step1: Clock Configuration. Such as clock source, clock prescaler and so on.

Step2: Compare mode configuration.

* Set the shadow enable mode by OCxPE

* Set the output mode (Set/Clear/Toggle) by OCxM.

* Select the active high polarity by CCxP/CCxNP

* Enable the output by CCxE

Step3: Interrupt/DMA-request enables configuration by CCxIE/CCxDE.

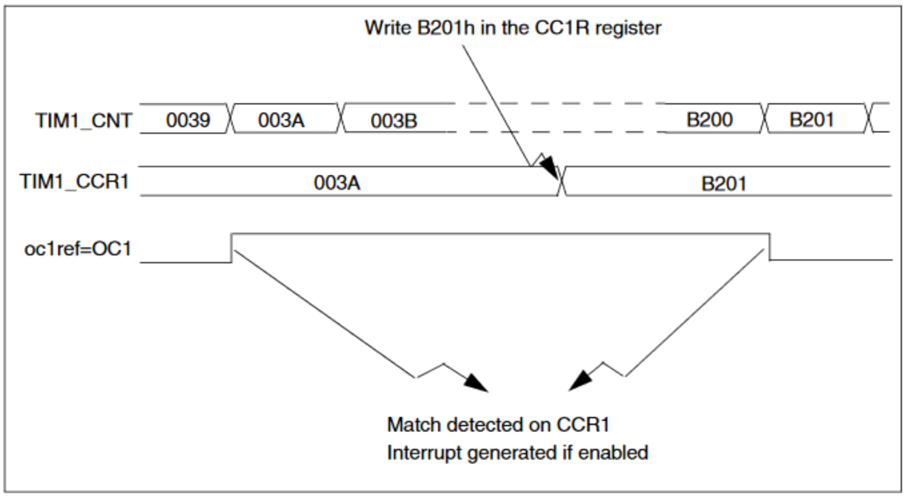
Step4: Compare output timing configuration by TIMx_ARR and TIMx_CCRx

About the CCRx; you can change it on the go to meet the waveform you expected.

Step5: Start the counter by setting CEN.

The timechart below shows the output compare mode in detail.

Figure 17-21. Output-compare mode, toggle on OC1



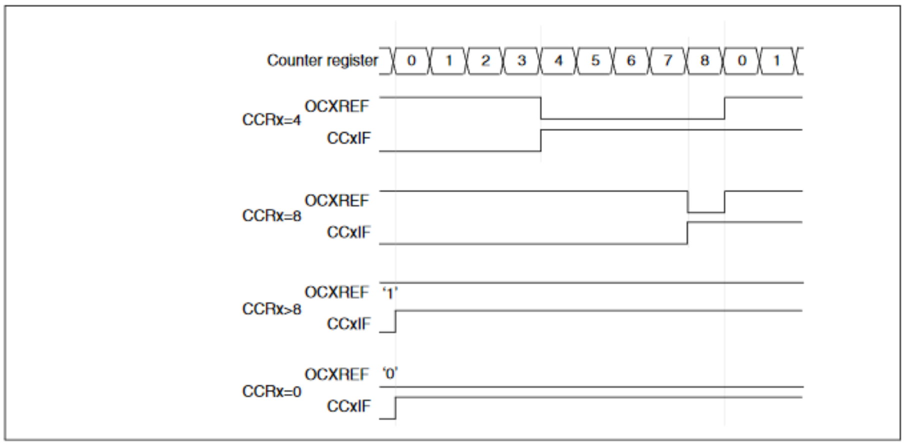
17.4.7. PWM mode

In the output PWM mode (by setting the OCxM bits to 3'b110 (PWM mode1) or to 3'b 111(PWM mode2), the channel can generate PWM waveform according to the TIMx_ARR registers and TIMx_CCRx registers.

Based on the counter mode, we can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

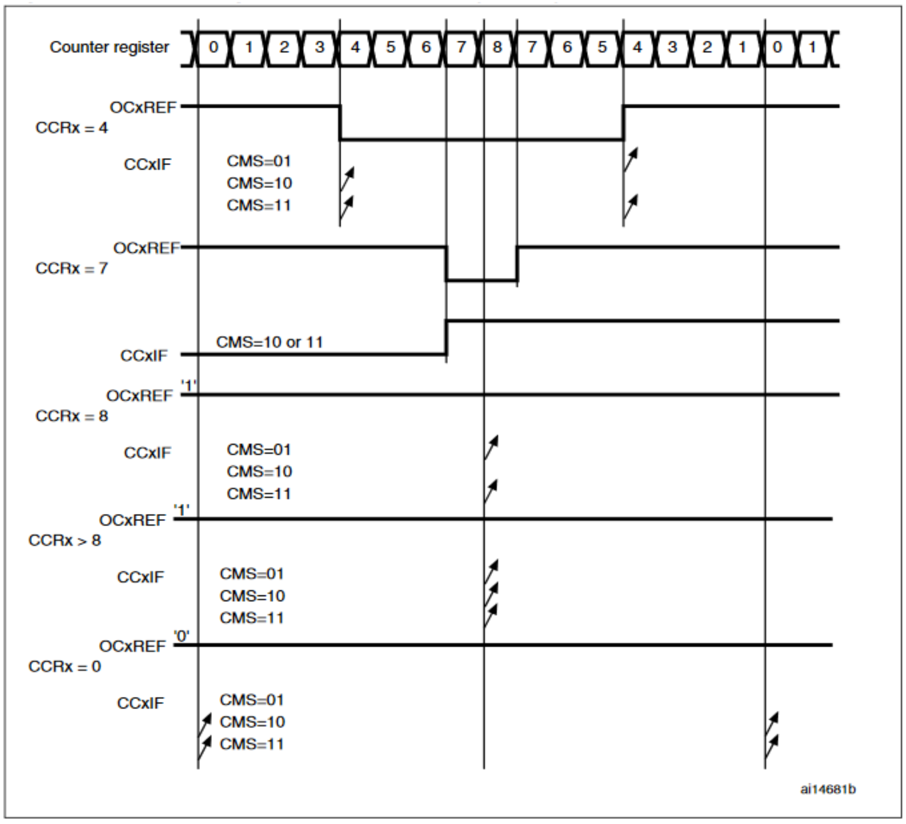
The EAPWM period is determined by TIMx_ARR and duty cycle is determined by TIMx_CCRx. The Figure below shows the EAPWM output and interrupts waveform.

Figure 17-22. EAPWM waveforms (ARR=8)



The CAPWM period is determined by 2*TIMx_ARR, and duty cycle is by 2*TIMx_CCRx. The figure below shows the CAPWM output and interrupts waveform.

Figure 17-23. CAPWM waveforms (ARR=8)



If TIMx_CCRx is greater than TIMx_ARR, the output will be always active under PWM mode1 (OCxM = 3'b110).

And if TIMx_CCRx is equal to zero, the output will be always inactive under PWM mode1 (OCxM = 3'b110)

17.4.8. Channel output reference signal

When the TIMx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the OCxM filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the OCxM field to 0x00, set to 1 by setting the OCxM field to 0x01, set to 0 by setting the OCxM field to 0x02 or signal toggle by setting the OCxM field to 0x03 when the counter value matches the content of the TIMx_CCRx register.

The PWM mode 1 and PWM mode 2 outputs are also another kind of OxCPRE output which is setup by setting the OCxM field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMx_CCRx content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the OCxM field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMx_CCRx values.

The OxCPRE signal can be forced to 0 when the ETRF signal is derived from the external ETR pin and when it is set to a high level by setting the OCxCE bit to 1 in the TIMx_CCMR1 register. The OxCPRE signal will not return to its active level until the next update event occurs.

17.4.9. Quadrature decoder

The quadrature decoder function uses two quadrature inputs TI1 and TI2 derived from the TIMx_CH1 and TIMx_CH2 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either TI1 only, TI2 only or both TI1 and TI2, the selection mode by setting the SMS[2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-reload value. Therefore, users must configure the TIMx_ARR register before the counter starts to count.

Table 17-1. Counting direction versus encoder signals

Counting mode	**Level**	**TI1FP1**		**TI2FP2**	
		Rising	**Falling**	**Rising**	**Falling**
TI1 only counting	TI2FP2=High	Down	Up	-	-
	TI2FP2=Low	Up	Down	-	-
TI2 only counting	TI1FP1=High	-	-	Up	Down
	TI1FP1=Low	-	-	Down	Up
TI1 and TI2 counting	TI2FP2=High	Down	Up	X	X
	TI2FP2=Low	Up	Down	X	X
	TI1FP1=High	X	X	Up	Down
	TI1FP1=Low	X	X	Down	Up

Note:"-" means "no counting"; "X" means impossible.

Figure 17-24. Example of counter operation in encoder interface mode

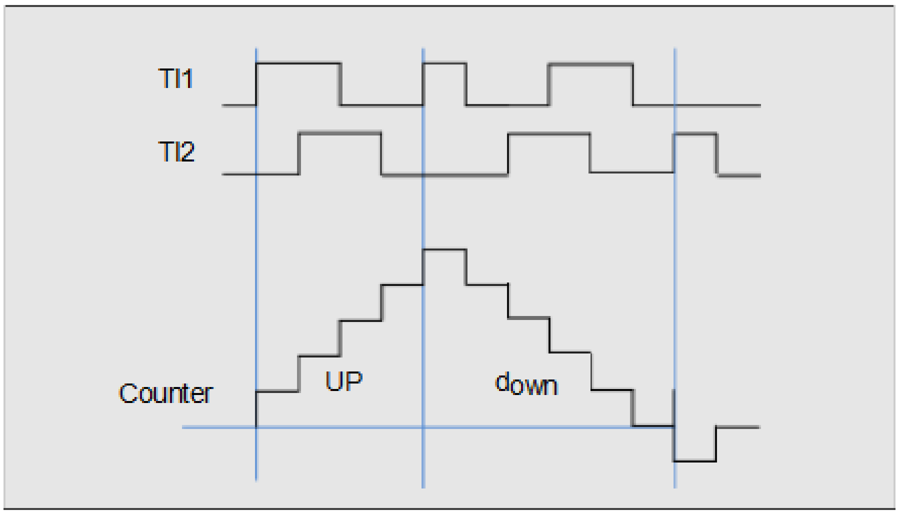
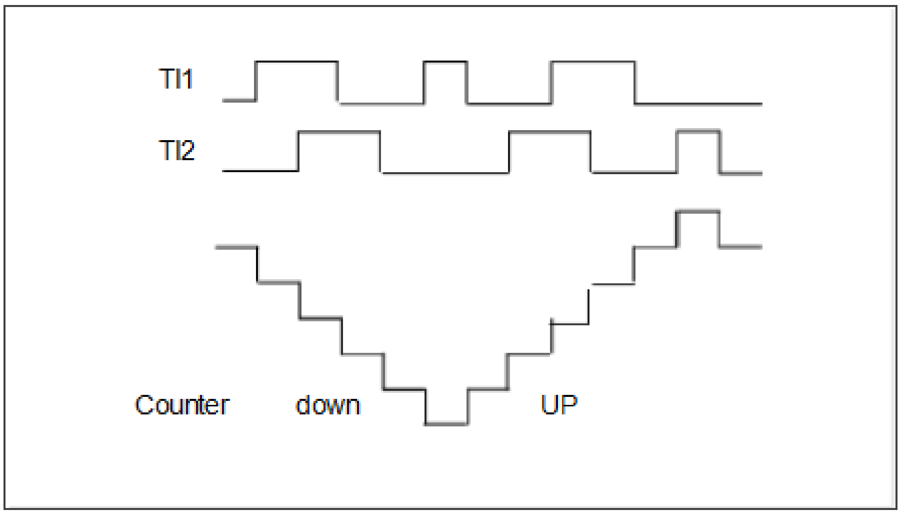


Figure 17-25. Example of encoder interface mode with CI1FE0 polarity inverted



17.4.10. Hall sensor function

Hall sensor is generally used to control BLDC Motor; general-purpose timer can support this function.

Figure 17-27 shows how to connect. And we can see we need two timers. First TIMER in （General purpose TIMER） should accept three Rotor Position signals from Motor.

Each of the 3 sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

By the internal connection such as TRGO- ITIx, TIMER_in and TIMER_out can be connected. TIMER_out will generate PWM signal to control BLDC motor’s speed based on the ITRx. Then, the feedback circuit is finished, also you change configuration to fit your request.

About the TIMER_in, it need have input XOR function, so you can choose from Advanced TIMER.

And TIMER_out need have functions of complementary and Dead-time, so only general-purpose timer can be chosen. Else, based on the timers’ internal connection relationship, pair's timers can be selected. For example:

TIMER_in (TIMER2) -> TIMER_out (Timer1 ITR1)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI1S, then, each of input signal change will make the TI1 toggle. CCR1 will record the value of counter at that moment.
- Enable ITRx connected to commutation function directly by setting CCUS and CCPC.
- Configuration PWM parameter based on your request.

Figure 17-26. Hall sensor is used to BLDC motor

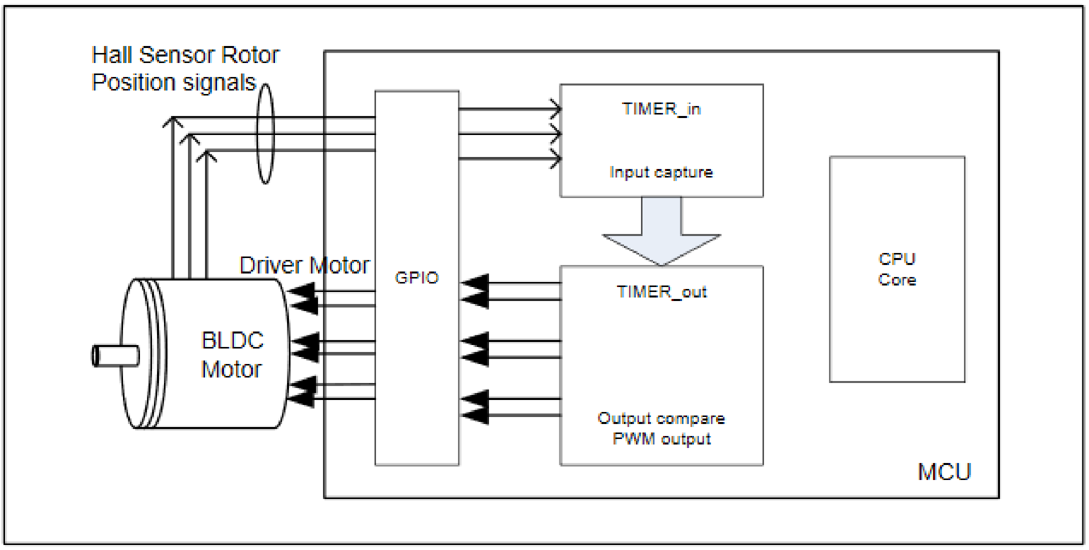
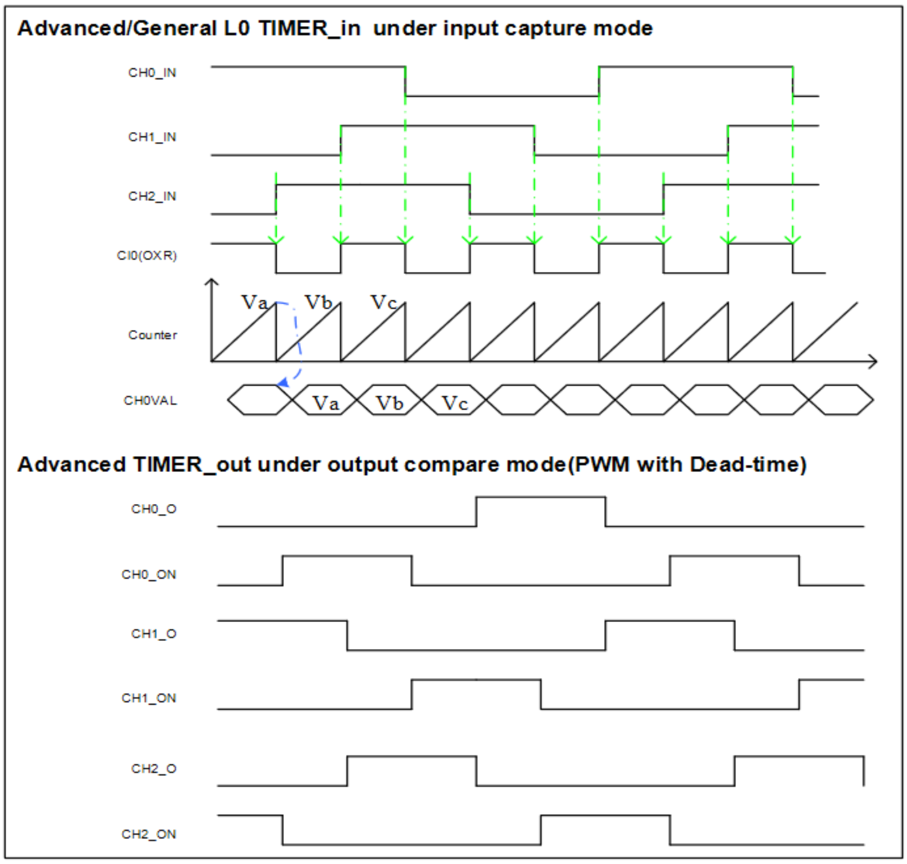


Figure 17-27. Hall sensor timing between two timers


Advanced/General L0 TIMER_in under input capture mode
CH0_IN
CH1_IN
CH2_IN
CI0(OXR)
Va
Vb
Vc
Counter
CH0VAL
Va
Vb
Vc
Advanced TIMER_out under output compare mode(PWM with Dead-time)
CH0_O
CH0_ON
CH1_O
CH1_ON
CH2_O
CH2_ON

Slave controller

The TIMx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMS [2:0] in the TIMx_SMCR register. The trigger input of these modes can be selected by the TS [2:0] in the TIMx_SMCR register.

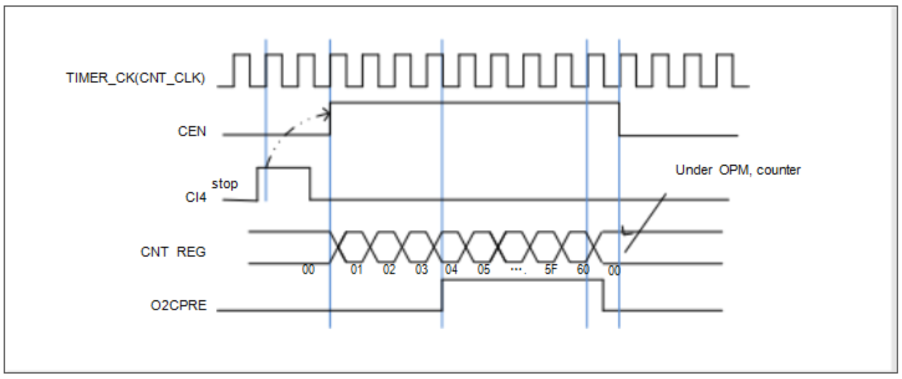
17.4.11. Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting OPM in TIMx_CR1. When you set OPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMx to PWM mode or compare by OCxM.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMx_CR1 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMx_CCRx value. In order to reduce the delay to a minimum value, the user can set the OCxFE bit in each TIMx_CCMR1/1 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The OCxFE bit is available only when the output channel is configured to operate in the PWM1 or PWM2 output mode and the trigger source is derived from the trigger signal.

Figure 17-28. Single pulse mode, TIMx_CCRx = 0x04, TIMx_ARR=0x60



17.4.12. Timers interconnection

The timers can be internally connected together for timer chaining or synchronization. Please refer to section 16.4.14 for detail info.

17.4.13. Timer DMA mode

If one of the DMA request event coming, TIMx will send DMA requests to DMAC. The user can configure DMAC properly and read/write TIMx SFRs under the control of DMA. Please refer to DMAC related chapters for details.

17.4.14. Timer debug mode

When the Cortex™-M3 is halted, and the DBG_TIMx_STOP configuration bit in DBGMCU_CR register is set to 1, the TIMx counter stops.

17.5. Register definition

17.5.1. Control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:10	Reserved	R	Must be kept at reset value
9:8	CKD[1:0]	RW	Clock division The CKD bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. 00: $f_{DTS}=f_{TIMER\_CK}$ 01: $f_{DTS}=f_{TIMER\_CK}$ /2 10: $f_{DTS}=f_{TIMER\_CK}$ /4 11: Reserved
7	ARPE	RW	Auto-reload shadow enable 0: The shadow register for TIMx_ARR register is disabled 1: The shadow register for TIMx_ARR register is enabled
6:5	CMS[1:0]	RW	Counter aligns mode selection 00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit. 01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Only when the counter is counting down, compare interrupt flag of channels can be set. 10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Only when the counter is counting up, compare interrupt flag of channels can be set. 11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CCxS=00 in TIMx_CCMR1 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set. After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	RW	Direction 0: Count up 1: Count down This bit is read only when the timer is configured in center-aligned mode or encoder mode.
3	OPM	Rw	Single pulse mode. 0: Single pulse mode disable. Counter continues after update event.

Bits	Fields	RW	Descriptions
			1: Single pulse mode enable. The CEN is cleared by hardware and the counter stops at next update event.
2	URS	RW	Update source This bit is used to select the update event sources by software. 0: Any of the following events generate an update interrupt or DMA request: – The UG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: Only counter overflow/underflow generates an update interrupt or DMA request.
1	UDIS	RW	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: – The UG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.
0	CEN	RW	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

17.5.2. Control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31-8	Reserved	R	Must be kept at reset value
7	TI1S	RW	Channel 1 trigger input selection 0: The TIMx_CH1 pin input is selected as Channel 1 trigger input. 1: The result of combinational XOR of TIMx_CH1, CH2 and CH3 pins is selected as Channel 1 trigger input.
6:4	MMS[2:0]	RW	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UG bit in the TIMx_EGR register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.

Bits	Fields	RW	Descriptions
			010: Update. In this mode the master mode controller selects the update event as TRGO. 011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channal0 . 100: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O4CPRE signal is used as TRGO
3	CCDS	RW	DMA request source selection 0: DMA request of channel x is sent when capture/compare event occurs. 1: DMA request of channel x is sent when update event occurs.
2:0	-		Reserved bits

17.5.3. Slave mode configuration register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15	ETP	RW	External trigger polarity This bit specifies the polarity of ETR signal 0: ETR is active at high level or rising edge. 1: ETR is active at low level or falling edge.
14	ECE	RW	Part of SMS for enable External clock mode 2. In external clock mode 2, the counter is clocked by any active edge on the ETRF signal. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. Note: It is possible to simultaneously use external clock mode 2 with the restart mode, pause mode or event mode(TS bits must not be 3'b111). The external clock input will be ETRF if external clock mode 1 and external clock mode 2 are enabled at the same time. External clock mode 1 enable is in this register's SMS bit-filed.
13:12	ETPS[1:0]	RW	External trigger prescaler The frequency of external trigger signal ETRP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETRP frequency. 00: Prescaler disable 01: ETRF frequency will be divided by 2 10: ETRF frequency will be divided by 4 11: ETRF frequency will be divided by 8

Bits	Fields	RW	Descriptions
11:8	ETF[3:0]	RW	External trigger filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. 0000: Filter disabled, f_{SAMP}=f_{DTS}, N=1 0001: f_{SAMP}=f_{TIMER_CK}, N=2 0010: f_{SAMP}=f_{TIMER_CK}, N=4 0011: f_{SAMP}=f_{TIMER_CK}, N=8 0100: f_{SAMP} =f_{DTS}/2, N=6 0101: f_{SAMP} =f_{DTS}/2, N=8 0110: f_{SAMP} =f_{DTS}/4, N=6 0111: f_{SAMP} =f_{DTS}/4, N=8 1000: f_{SAMP} =f_{DTS}/8, N=6 1001: f_{SAMP} =f_{DTS}/8, N=8 1010: f_{SAMP} =f_{DTS}/16, N=5 1011: f_{SAMP} =f_{DTS}/16, N=6 1100: f_{SAMP} =f_{DTS}/16, N=8 1101: f_{SAMP} =f_{DTS}/32, N=5 1110: f_{SAMP} =f_{DTS}/32, N=6 1111: f_{SAMP} =f_{DTS}/32, N=8
7	MSM	RW	Master-slave mode This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together. 0: Master-slave mode disable 1: Master-slave mode enable
6:4	TS[2:0]	RW	Trigger selection This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter. 000: Internal trigger input 0 (ITR0) 001: Internal trigger input 1 (ITR1) 010: Internal trigger input 2 (ITR2) 011: Internal trigger input 3 (ITR3) 100: TI1 edge flag (TI1F_ED) 101: Channel 1 input Filtered output (TI1FP1) 110: Channel 2 input Filtered output (TI2FP2) 111: External trigger input filter output(ETRF) These bits must not be changed when slave mode is enabled.
3	Reserved	R	Must be kept at reset value.
2:0	SMS[2:0]	RW	Slave mode control 000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Quadrature decoder mode 0.The counter counts on TI2FP2 edge, while the direction depends on TI1FP1 level. 010: Quadrature decoder mode 1.The counter counts on TI1FP1 edge, while the direction depends on TI2FP2 level. 011: Quadrature decoder mode 2.The counter counts on both TI1FP1 and TI2FP2 edge, while the direction depends on each other. 100: Restart mode. The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input. 101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter when it is low.

Bits	Fields	RW	Descriptions
			110: Event mode. A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller. 111: External clock mode 1. The counter counts on the rising edges of the selected trigger.

17.5.4. DMA and interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:15	-	R	Reserved bits
14	TDE	RW	Trigger DMA request enable 0: disabled 1: enabled
13	-	R	Reserved bit
12	CC4DE	RW	Channel 4 capture/compare DMA request enable 0: disabled 1: enabled
11	CC3DE	RW	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
10	CC2DE	RW	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
9	CC1DE	RW	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
8	UDE	RW	Update DMA request enable 0: disabled 1: enabled
7	-	R	Reserved bit
6	TE	RW	Trigger interrupt enable 0: disabled 1: enabled
5	-	R	Reserved bit
4	CC4IE	RW	Channel 4 capture/compare interrupt enable 0: disabled 1: enabled
3	CC3IE	RW	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
2	CC2IE	RW	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
1	CC1IE	RW	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled

Bits	Fields	RW	Descriptions
0	UIE	RW	Update interrupt enable 0: disabled 1: enabled

17.5.5. Interrupt flag register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:13	Reserved	R	Must be kept at reset value.
12	CC4OF	RC_W0	Channel 4 over capture flag Refer to CC1OF description
11	CC3OF	RC_W0	Channel 3 over capture flag Refer to CC1OF description
10	CC2OF	RC_W0	Channel 2 over capture flag Refer to CC1OF description
9	CC1OF	RC_W0	Channel 1 over capture flag When Channel 1 is configured in input mode, this flag is set by hardware when a capture event occurs while CC1IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	-	R	Reserved bits
6	TIF	RC_W0	Trigger interrupt flag This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on trigger input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on trigger input generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	-	R	Reserved bit
4	CC4IF	RC_W0	Channel 4's capture/compare interrupt flag Refer to CC1IF description
3	CC3IF	RC_W0	Channel 3's capture/compare interrupt flag Refer to CC1IF description
2	CC2IF	RC_W0	Channel 2's capture/compare interrupt flag Refer to CC1IF description
1	CC1IF	RC_W0	Channel 1's capture/compare interrupt flag This flag is set by hardware and cleared by software. When Channel 1 is in input mode, this flag is set when a capture event occurs. When Channel 1 is in output mode, this flag is set when a compare event occurs. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UIF	RC_W0	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred

17.5.6. Software event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:7	Reserved	R	Must be kept at reset value.
6	TG	W	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TIF flag in TIMx_SR register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	-	R	Reserved bit
4	CC4G	W	Channel 4's capture or compare event generation Refer to CC1G description
3	CC3G	W	Channel 3's capture or compare event generation Refer to CC1G description
2	CC2G	W	Channel 2's capture or compare event generation Refer to CC1G description
1	CC1G	W	Channel 1's capture or compare event generation This bit is set by software in order to generate a capture or compare event in Channel 1, it is automatically cleared by hardware. When this bit is set, the CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if Channel 2 is configured in input mode, the current value of the counter is captured in TIMx_CCR1 register, and the CC1OF flag is set if the CC1IF flag was already high. 0: No generate a Channel 2 capture or compare event 1: Generate a Channel 2 capture or compare event
0	UG	W	Update event generation This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

17.5.7. Channel control register 0 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

Output compare mode:

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15	OC2CE	RW	Output Compare 2 clear enable Refer to OC1CE description
14:12	OC2M[2:0]	RW	Output Compare 2 mode Refer to OC1M description
11	OC2PE	RW	Output Compare 2 preload enable Refer to OC1PE description

Bits	Fields	RW	Descriptions
10	OC2FE	RW	Output Compare 2 fast enable Refer to OC1PE description
9:8	CC2S[1:0]	RW	Capture/Compare 2 selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC2E bit in TIMx_CCER register is reset). 00: Channel 2 is configured as output 01: Channel 2 is configured as input, IC2 is mapped on TI2 10: Channel 2 is configured as input, IC2 is mapped on TI2 11: Channel 2 is configured as input, IC2 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.
7	OC1CE	RW	Output Compare 1 clear enable When this bit is set, the O1CPRE signal is cleared when High level is detected on ETRF input. 0: Channel 1 output compare clear disable 1: Channel 1 output compare clear enable
6:4	OC1M[2:0]	RW	Output Compare 1 mode This bit-field controls the behavior of the output reference signal O1CPRE which drives CH1_O and CH1_ON . O1CPRE is active high, while CH1_O and CH1_ON active level depends on CC1P and CC1NP bits. 000: Frozen. The O1CPRE signal keeps stable, independent of the comparison between the register TIMx_CCR1 and the counter TIMx_CNT. 001: Set the channel output. O1CPRE signal is forced high when the counter matches the output compare register TIMx_CCR1. 010: Clear the channel output. O1CPRE signal is forced low when the counter matches the output compare register TIMx_CCR1. 011: Toggle on match. O1CPRE toggles when the counter matches the output compare register TIMx_CCR1. 100: Force low. O1CPRE is forced low level. 101: Force high. O1CPRE is forced high level. 110: PWM mode1. When counting up, O1CPRE is high as long as the counter is smaller than TIMx_CCR1 else low. When counting down, O1CPRE is low as long as the counter is larger than TIMx_CCR1 else high. 111: PWM mode2. When counting up, O1CPRE is low as long as the counter is smaller than TIMx_CCR1 else high. When counting down, O1CPRE is high as long as the counter is larger than TIMx_CCR1 else low. Note: When configured in PWM mode, the O1CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes. These bits cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC1S bit-filed is 00(COMPARE MODE).
3	OC1PE	RW	Output Compare 1 preload enable When this bit is set, the shadow register of TIMx_CCR1 register, which updates at each update event, will be enabled. 0: Channel 1 output compare shadow disable 1: Channel 1 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (OPM bit in TIMx_CR1 register is set). This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11

Bits	Fields	RW	Descriptions
			and CC1S bit-filed is 00.
2	OC1FE	RW	Output Compare 1 fast enable When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH1_O is set to the compare level independently from the result of the comparison. 0: Channel 1 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH1_O output is 5 clock cycles. 1: Channel 1 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH1_O output is 3 clock cycles.
1:0	CC1S[1:0]	RW	Capture/Compare 1 selection This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writa ble only when the channel is not active. (CC1E bit in TIMx_CCER register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IC1 is connected to TI1FP1 10: Channel 1 is configured as input, IC1 is connected to TI2FP1 11: Channel 1 is configured as input, IC1 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.

Input capture mode:

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15:12	IC2F[3:0]	RW	Input capture 2 filter Refer to IC1F description
11:10	IC2PSC[1:0]	RW	Channel 2 input capture prescaler Refer to IC1PSC description
9:8	CC2S[1:0]	RW	Capture/Compare 2 selection Same as Output compare mode
7:4	IC1F[3:0]	RW	Input capture 1 filter An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample TI1 input signal and the length of the digital filter applied to TI1. 0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1 0001: $f_{SAMP}=f_{TIMER\_CK}$, N=2 0010: $f_{SAMP}=f_{TIMER\_CK}$, N=4 0011: $f_{SAMP}=f_{TIMER\_CK}$, N=8 0100: $f_{SAMP}=f_{DTS}/2$, N=6 0101: $f_{SAMP}=f_{DTS}/2$, N=8 0110: $f_{SAMP}=f_{DTS}/4$, N=6 0111: $f_{SAMP}=f_{DTS}/4$, N=8 1000: $f_{SAMP}=f_{DTS}/8$, N=6 1001: $f_{SAMP}=f_{DTS}/8$, N=8 1010: $f_{SAMP}=f_{DTS}/16$, N=5 1011: $f_{SAMP}=f_{DTS}/16$, N=6 1100: $f_{SAMP}=f_{DTS}/16$, N=8

Bits	Fields	RW	Descriptions
			1101: f_{SAMP} =f_{DTS}/32, N=5 1110: f_{SAMP} =f_{DTS}/32, N=6 1111: f_{SAMP} =f_{DTS}/32, N=8
3:2	IC1PSC[1:0]	RW	Input capture 1 prescaler This bit-field specifies the factor of the prescaler on Channel 1 input. The prescaler is reset when CC1E bit in TIMx_CCER register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4channel input edges 11: Capture is done every 8 channel input edges
1:0	CC1S[1:0]	RW	Capture/Compare 1 Selection Same as Output compare mode

17.5.8. Channel control register 1 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Output compare mode:

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15	OC4CE	RW	Output compare 4 clear enable Refer to OC1CE description
14:12	OC4M[2:0]	RW	Channel 4 compare output control Refer to OC1M description
11	OC4PE	RW	Output compare 4 preload enable Refer to OC1PE description
10	OC4FE	RW	Output compare 4 fast enable Refer to OC1PE description
9:8	CC4S[1:0]	RW	Capture/Compare 4 selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC4E bit in TIMx_CCER register is reset). 00: Channel 4 is configured as output 01: Channel 4 is configured as input, IC4 is mapped on TI4 10: Channel 4 is configured as input, IC4 is mapped on TI3 11: Channel 4 is configured as input, IC4 is connected to TRC, This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.
7	OC3CE	RW	Output compare 3 clear enable When this bit is set, the O3CPRE signal is cleared when High level is detected on ETRF input. 0: Channel 3 output compare clear disable 1: Channel 3 output compare clear enable
6:4	OC3M[2:0]	RW	Output compare 3 mode This bit-field controls the behavior of the output reference signal O3CPRE which drives CH3_O and CH3_ON . O3CPRE is active high, while CH3_O and CH3_ON active level depends on CC3P and CC3NP bits.

Bits	Fields	RW	Descriptions
			000: Frozen. The O3CPRE signal keeps stable, independent of the comparison between the output compare register TIMx_CCR3 and the counter TIMx CNT . 001: Set high on match. O3CPRE signal is forced high when the counter matches the output compare register TIMx_CCR3. 010: Set low on match. O3CPRE signal is forced low when the counter matches the output compare register TIMx_CCR3. 011: Toggle on match. O3CPRE toggles when the counter matches the output compare register TIMx_CCR3. 100: Force low. O3CPRE is forced low level. 101: Force high. O3CPRE is forced high level. 110: PWM mode 1. When counting up, O3CPRE is high as long as the counter is smaller than TIMx_CCR3 else low. When counting down, O3CPRE is low as long as the counter is larger than TIMx_CCR3 else high. 111: PWM mode 2. When counting up, O3CPRE is low as long as the counter is smaller than TIMx_CCR3 else high. When counting down, O3CPRE is high as long as the counter is larger than TIMx_CCR3 else low. Note: When configured in PWM mode, the O3CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC3S bit-filed is 00(COMPARE MODE).
3	OC3PE	RW	Output compare 3 preload enable When this bit is set, the shadow register of TIMx_CCR3 register, which updates at each update event will be enabled. 0: Channel 3 output compare shadow disable 1: Channel 3 output compare shadow enable Note: The PWM mode can be used without validating the shadow register only in single pulse mode (OPM bit in TIMx_CR1 register is set) . This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 and CC1S bit-filed is 00.
2	OC3FE	RW	Output compare 3 fast enable When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH3_O is set to the compare level independently from the result of the comparison. 0: Channel 3 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH3_O output is 5 clock cycles. 1: Channel 3 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH3_O output is 3 clock cycles.
1:0	CC3S[1:0]	RW	Capture/Compare 3 selection This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CC3E bit in TIMx_CCER register is reset).). 00: Channel 3 is configured as output 01: Channel 3 is configured as input, IC3 is mapped on TI3 10: Channel 3 is configured as input, IC3 is mapped on TI4 11: Channel 3 is configured as input, IC3 is connected to TRC. This mode is working only if an internal trigger input is selected through TS bits in TIMx_SMCR register.

Input capture mode:

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits
15:12	IC4F[3:0]	RW	Input capture 4 filter Refer to IC3F description
11:10	IC4PSC[1:0]	RW	Input capture 4 prescaler Refer to IC3PSC description
9:8	CC4S[1:0]	RW	Capture/Compare 4 selection Same as Output compare mode
7:4	IC3F[3:0]	RW	Input capture 3 filter An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample TI3 input signal and the length of the digital filter applied to TI3. 0000: Filter disabled, f_{SAMP}=f_{DTS}, N=1 0001: f_{SAMP}=f_{TIMER_CK}, N=2 0010: f_{SAMP}=f_{TIMER_CK}, N=4 0011: f_{SAMP}=f_{TIMER_CK}, N=8 0100: f_{SAMP} =f_{DTS}/2, N=6 0101: f_{SAMP} =f_{DTS}/2, N=8 0110: f_{SAMP} =f_{DTS}/4, N=6 0111: f_{SAMP} =f_{DTS}/4, N=8 1000: f_{SAMP} =f_{DTS}/8, N=6 1001: f_{SAMP} =f_{DTS}/8, N=8 1010: f_{SAMP} =f_{DTS}/16, N=5 1011: f_{SAMP} =f_{DTS}/16, N=6 1100: f_{SAMP} =f_{DTS}/16, N=8 1101: f_{SAMP} =f_{DTS}/32, N=5 1110: f_{SAMP} =f_{DTS}/32, N=6 1111: f_{SAMP} =f_{DTS}/32, N=8
3:2	IC3PSC[1:0]	RW	Input capture 3 prescaler This bit-field specifies the factor of the prescaler on Channel 3 input. The prescaler is reset when CC3E bit in TIMx_CCER register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4 channel input edges 11: Capture is done every 8 channel input edges
1:0	CC3S[1:0]	RW	Capture/compare 3 selection Same as Output compare mode

17.5.9. Channel control register 2 (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:14	Reserved	R	Must be kept at reset value
13	CC4P	RW	Channel 4 capture/compare function polarity Refer to CC1P description
12	CC4E	RW	Channel 4 capture/compare function enable Refer to CC1E description
11:10	-	R	Reserved bits

Bits	Fields	RW	Descriptions
9	CC3P	RW	Channel 3 capture/compare function polarity Refer to CC1P description
8	CC3E	RW	Channel 3 capture/compare function enable Refer to CC1E description
7:6	-	R	Reserved bits
5	CC2P	RW	Channel 2 capture/compare function polarity Refer to CC1P description
4	CC2E	RW	Channel 2 capture/compare function enable Refer to CC1E description
3:2	-	R	Reserved bits
1	CC1P	RW	Channel 1 capture/compare function polarity When Channel 1 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 1 active high 1: Channel 1 active low When Channel 1 is configured in input mode, this bit specifies the TI1 signal polarity. 0: TI1 is non-inverted. Input capture is done on a rising edge of TI1. When used as extern trigger, TI1 is non-inverted. 1: TI1 is inverted. Input capture is done on a falling edge of TI1. When used as extern trigger, TI1 is inverted. Note: This bit cannot be modified when LOCK [1:0] bit-filed in TIMx_BDTR register is 11 or 10.
0	CC1E	RW	Channel 1 capture/compare function enable When Channel 1 is configured in output mode, setting this bit enables CH1_O signal in active state. When Channel 1 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 1 disabled 1: Channel 1 enabled

17.5.10. Counter register (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CNT[19:0]	RW	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

17.5.11. Prescaler register (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:16	-	R	Reserved bits

15:0	PSC[15:0]	RW	Prescaler value of the counter clock The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

17.5.12. Counter auto reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	ARR[19:0]	RW	Counter auto reload value This bit-filed specifies the auto reload value of the counter. Note: When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value.

17.5.13. Channel 1 capture/compare value register (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR1[19:0]	RW	Capture or compare value of channel0 When Channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

17.5.14. Channel 2 capture/compare value register (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR2[19:0]	RW	Capture or compare value of channel1 When Channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

17.5.15. Channel 3 capture/compare value register (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR3[19:0]	RW	Capture or compare value of Channel 3 When Channel 3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

17.5.16. Channel 4 capture/compare value register (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

Bits	Fields	RW	Descriptions
31:20	-	R	Reserved bits
19:0	CCR4[19:0]	RW	Capture or compare value of Channel 4 When channel4 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When Channel 4 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

18. Real-time Clock (RTC)

18.1. Overview

The RTC is usually used as a clock-calendar. The RTC circuits are located in two power supply domains. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. That means the RTC settings and time are kept when the device resets or wakes up from Standby mode. While the circuits in the VDD domain only include the APB interface and a control register. In the following sections, the details of the RTC function will be described.

18.2. Characteristics

- 32-bit programmable counter for counting elapsed time

  Programmable prescaler: Max division factor is up to 2^{20}
- Separate clock domains:

  A) PCLK1 clock domain

  B) RTC clock domain (this clock must be at least 4 times slower than the PCLK1 clock)
- RTC clock source:

  A) HSE clock divided by 128

  B) LSE oscillator clock

  C) LSI oscillator clock
- Mask-able interrupt source:

  A) Alarm interrupt

  B) Second interrupt

  C) Overflow interrupt

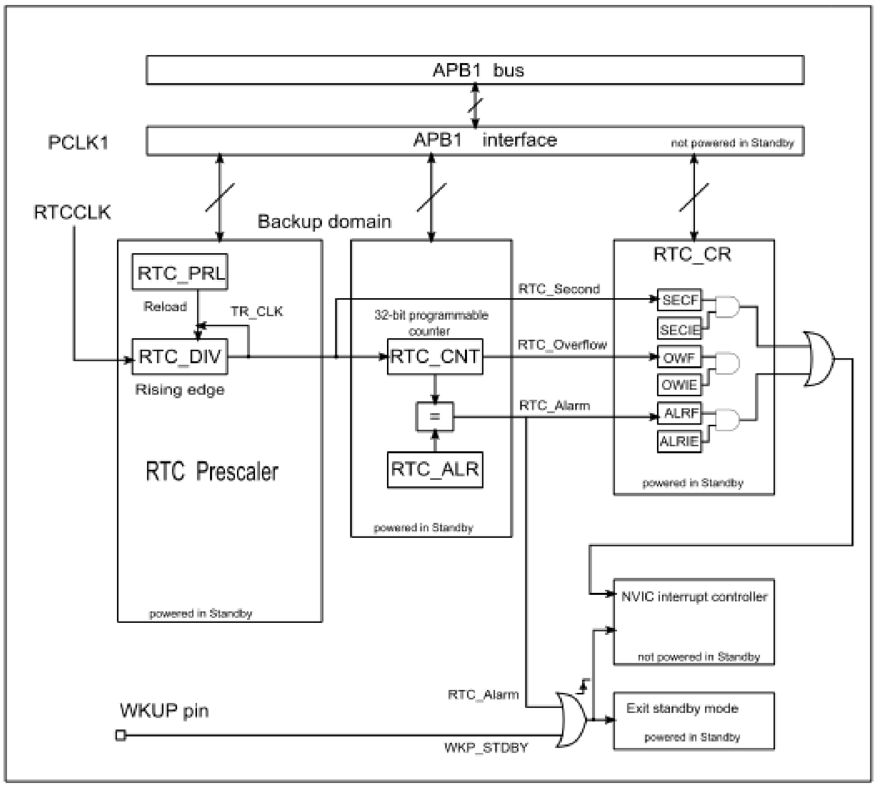
18.3. Function overview

The RTC circuits consist of two major units: APB interface located in PCLK1 clock domain and RTC core located in RTC clock domain.

APB Interface is connected with the APB1 bus. It includes a set of registers, can be accessed by APB1 bus.

RTC core includes two major blocks. One is the RTC prescaler block, which generates the RTC time base clock SC_CLK. RTC prescaler block includes a 20-bit programmable divider (RTC prescaler) which can make SC_CLK is divided from RTC source clock. If second interrupt is enabled in the RTC_CRH register, the RTC will generate an interrupt at every SC_CLK rising edge. Another block is a 32-bit programmable counter, which can be initialized with the value of current system time. If alarm interrupt is enabled in the RTC_CRH register, the RTC will generate an alarm interrupt when the system time equals to the alarm time (stored in the RTC_ALRH/L register),

Figure 18-1. Block diagram of RTC



18.3.1. RTC reset

The APB interface and the RTC_CRH register are reset by system reset. The RTC core (prescaler, divider, counter and alarm) is reset only by a backup domain reset.

Steps to enable access to the backup registers and the RTC after reset are as follows:

- Set the BDI bit in the AHBENR register to enable the power and backup interface clocks.
- Enable access to the backup registers and RTC by setting the DBP bit in the (PWR_CR0).

18.3.2. RTC reading

The APB interface and RTC core are located in two different power supply domains.

In the RTC core, only counter and divider registers are readable registers. And the values in the two registers and the RTC flags are internally updated at each rising edge of the RTC clock, which is resynchronized by the APB1 clock.

When the APB interface is immediately enabled from a disable state, the read operation is not recommended because the first internal update of the registers has not finished. That means, when a system reset, power reset, waking up from Standby mode or Deep-sleep mode occurs, the APB interface was in disabled state, but the RTC core has been kept running. In these cases, the correct read operation should first clear the RSF bit in the RTC

_CTL register and wait for it to be set by hardware. While WFI and WFE have no effects on the RTC APB interface.

18.3.3. RTC configuration

The RTC_PSC, RTC_CNT and RTC_ALRM registers in the RTC core are writable. These registers' value can be set only when the peripheral enter configuration mode. And the CNF bit in the RTC_CRL register is used to indicate the configuration mode status. The write operation executes when the peripheral exit configuration mode, and it takes at least three RTCCLK cycles to complete. The value of the RTOFF bit in the RTC_CRL register sets to '1', if the write operation finished. The new write operation should wait for the previous one finished.

The configuration steps are as follows:

A) Wait until the value of RTOFF bit in the RTC_CRL register sets to '1';

B) Enter Configuration mode by setting the CNF bit in the RTC_CRL register;

C) Write to the RTC registers;

D) Exit Configuration mode by clearing the CNF bit in the RTC_CRL register;

E) Wait until the value of RTOFF bit in the RTC_CRL register sets to '1'.

18.3.4. RTC flag assertion

Before the update of the RTC Counter, the RTC second interrupt flag (SECF) is asserted on the last RTCCLK cycle.

Before the counter equal to the RTC Alarm value which stored in the Alarm register increases by one, the RTC Alarm interrupt flag (ALRF) is asserted on the last RTCCLK cycle.

Before the counter equals to 0x0, the RTC Overflow interrupt flag (OWF) is asserted on the last RTCCLK cycle.

The RTC Alarm write operation and Second interrupt flag must be synchronized by using either of the following sequences:

- Use the RTC alarm interrupt and update the RTC Alarm and/or RTC Counter registers inside the RTC interrupt routine;
- Update the RTC Alarm and/or the RTC Counter registers after the SECF bit to be set in the RTC Control register.

Figure 18-2. RTC second and alarm waveform example (RTC_PSC = 3, RTC_ALRM = 4)

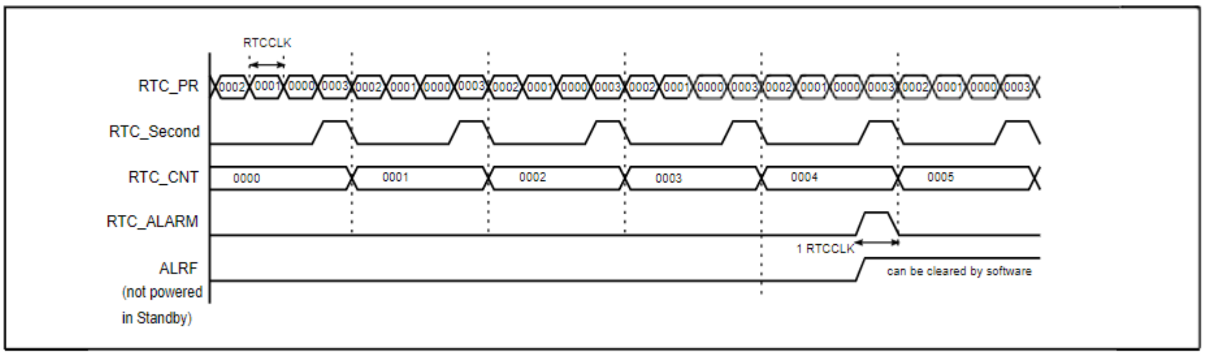
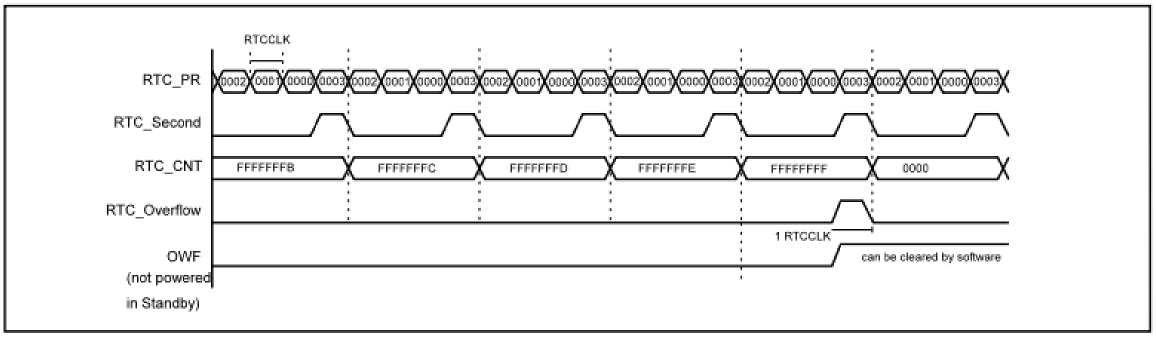

RTCCLK
RTC_PR
0002
0001
0000
0003
RTC_Second
RTC_CNT
0000
0001
0002
0003
0004
0005
RTC_ALARM
1 RTCCLK
ALRF
(not powered
in Standby)
can be cleared by software

Figure 18-3. RTC second and overflow waveform example (RTC_PSC= 3)



18.4. Register definition

18.4.1. RTC interrupt enable register (RTC_CRH)

Address offset: 0x00

Reset value: 0x0000

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	OWIE	RW	Overflow interrupt enable 0: Disable overflow interrupt 1: Enable overflow interrupt
1	ALRIE	RW	Alarm interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
0	SECIE	RW	Second interrupt enable 0: Disable second interrupt. 1: Enable second interrupt

18.4.2. RTC control register (RTC_CRL)

Address offset: 0x04

Reset value: 0x0020

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5	RTOFF	R	Last write operation finished flag 0: Last write operation on RTC registers did not finished 1: Last write operation on RTC registers finished.
4	CNF	RW	Configuration mode flag 0: Exit configuration mode. 1: Enter configuration mode.
3	RSF	RC_W0	Registers synchronized flag 0: Registers not yet synchronized with the APB1 clock. 1: Registers synchronized with the APB1 clock.
2	OWF	RC_W0	Overflow interrupt flag

Bits	Fields	R/W	Description
			0: Overflow event not detected 1: Overflow event detected. An interrupt will occur if the OWIE bit is set.
1	ALRF	RC_W0	Alarm interrupt flag 0: Alarm event not detected 1: Alarm event detected. An interrupt named RTC global interrupt will occur if the ALRIE bit is set in RTC_CRH. And another interrupt named the RTC Alarm interrupt will occur if the EXTI 17 is enabled in interrupt mode.
0	SECF	RC_W0	Second interrupt flag 0: Second event not detected. 1: Second event detected An interrupt will occur if the SECIE bit is set. Set by hardware when the divider reloads the value in RTC_PRLH/L, thus increment the RTC counter

18.4.3. RTC prescaler high register (RTC_PRLH)

Address offset: 0x08

Reset value: 0x0000

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3:0	PSC[19:16]	W	RTC prescaler value high

18.4.4. RTC prescaler low register (RTC_PRLL)

Address offset: 0x0C

Reset value: 0x0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	PSC[15:0]	W	RTC prescaler value low The frequency of TR_CLK is the RTCCLK frequency divided by (PSC[19:0]+1)

18.4.5. RTC divider high register (RTC_DIVH)

Address offset: 0x10

Reset value: 0x0000

Bits	Fields	R/W	Description
31:4	-	R	Reserved
3:0	DIV[19:16]	R	RTC divider value high

18.4.6. RTC divider low register (RTC_DIVL)

Address offset: 0x14

Reset value: 0x0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	DIV[15:0]	R	RTC divider value low The RTC divider register is reloaded by hardware when the RTC prescaler or RTC counter register updated.

18.4.7. RTC counter high register (RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	CNT[31:16]	RW	RTC counter value high

18.4.8. RTC counter low register (RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	CNT[15:0]	RW	RTC counter value low

18.4.9. RTC alarm high register (RTC_ALRH)

Address offset: 0x20

Reset value: 0xFFFF

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	ALRM[31:16]	RW	RTC alarm value high

18.4.10. RTC alarm low register (RTC_ALRL)

Address offset: 0x24

Reset value: 0xFFFF

Bits	Fields	R/W	Description
31:16	-	R	Reserved

Bits	Fields	R/W	Description
15:0	ALRM[15:0]	RW	RTC alarm value low

19. Watchdog timer (IWDG)

19.1. Overview

The free watchdog timer (IWDG) has free clock source. There upon the IWDG can operate even if the main clock fails. It's suitable for the situation that requires an independent environment and lower timing accuracy.

The free watchdog timer causes a reset when the internal down counter reaches 0. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

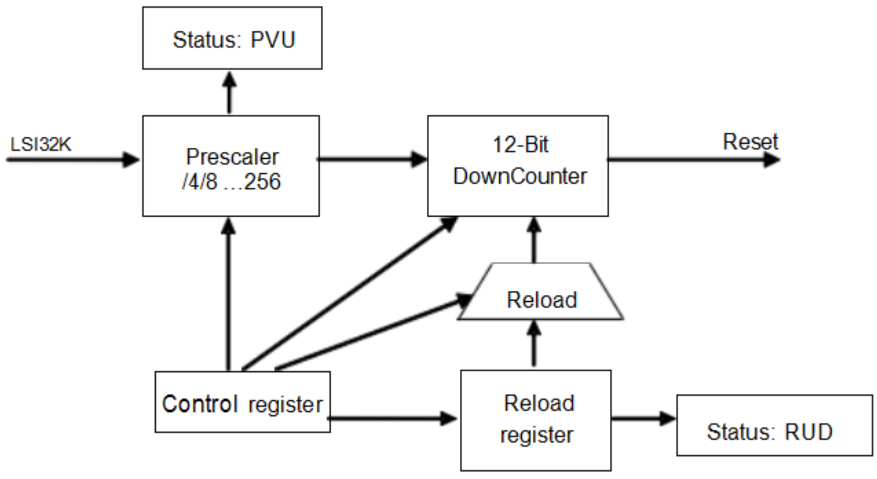
19.2. Characteristics

- Free-running 12-bit downcounter.
- Reset occur when the downcounter reaches 0, if the watchdog is enabled.
- Free clock source, IWDG can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog bit, automatically start the IWDG at power on.
- IWDG debug mode, the IWDG can stop or continue to work in debug mode.

19.3. Function overview

The free watchdog consists of an 8-stage pre-scaler and a 12-bit down-counter. Refer to the figure below for the functional block of the free watchdog module.

Figure 19-1. Free watchdog block diagram



The free watchdog is enabled by writing the value 0xCCCC in the control register (IWDG_KR), and the counter starts counting down. When the counter reaches the value 0x000, a reset is generated.

The counter can be reloaded by writing the value 0xAAAA to the IWDG_KR register at any time. The reload value comes from the IWDG_RLR register. The software can prevent the watchdog reset by reloading the counter before the counter reaches the value 0x000.

The free watchdog can automatically start at power on when the hardware free watchdog bit in the device option bytes is set. To avoid reset, the software should reload the counter before the counter reaches 0x000.

The IWDG_PR register and the IWDG_RLR register are write-protected. Before writing these registers, the software should write the value 0x5555 to the IWDG_KR register. These registers will be protected again by writing any other value to the IWDG_KR register. When an update operation of the pre-scaler register (IWDG_PR) or the reload value register (IWDG_RLR) is on going, the status bits in the IWDG_SR register are set.

If the IWDG_HOLD bit in DBG module is cleared, the IWDG continues to work even the Cortex™-M3 core halted (Debug mode). While the IWDG stops in Debug mode if the IWDG HOLD bit is set.

Table 19-1. Min/max IWDG timeout period at 32 KHz (LSI 32K)

Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]=0x000	Max timeout (ms) RLR[11:0]=0xFFF
1/4	000	0.125	512
1/8	001	0.25	1024
1/16	010	0.5	2048
1/32	011	1.0	4096
1/64	100	2.0	8192
1/128	101	4.0	16384
1/256	110 or 111	8.0	32768

The IWDG timeout can be more accurate by calibrating the LSI32K.

19.4. Register definition

19.4.1. Control register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	KEY	W	Write only. Several different functions are realized by writing these bits with different values: 0x5555: Disable the IWDG　PSC and IWDG_RLR write protection 0xCCCC: Start the free watchdog counter. When the counter reduces to 0, the free watchdog generates a reset 0xAAAA: Reload the counter

19.4.2. Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	PR	RW	Free watchdog timer prescaler selection. Write 0x5555 in the IWDG_KR register before writing these bits. During a write operation to this register, the PVU bit in the IWDG_SR register is set and the value read from this register is invalid. 000: 1/4 100: 1/64 001: 1/8 101: 1/128 010: 1/16 110: 1/256 011: 1/32 111: 1/256

If several prescaler values are used by the application, it is mandatory to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PVU is reset before continuing code execution except in case of low-power mode entry .

19.4.3. Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11:0	RL	RW	Free watchdog timer counter reload value. Write 0xAAAA in the IWDG_KR register will reload the IWDG counter with the RL value. These bits are write-protected. Write 0x5555 in the IWDG_KR register before writing these bits. During a write operation to this register, the RVU bit in the IWDG_SR register is set and the value read from this register is invalid. If several reload values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value. However, after updating the reload value it is not necessary to wait until RVU is reset before continuing code execution except in case of low-power mode entry .

19.4.4. Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:2	-	R	Reserved

1	RVU	R	Free watchdog timer counter reload value update During a write operation to IWDG_RLR register, this bit is set and the value read from IWDG_RLR register is invalid. This bit is reset by hardware after the update operation of IWDG_RLR register.
0	PVU	R	Free watchdog timer prescaler value update During a write operation to IWDG_PR register, this bit is set and the value read from IWDG_PR register is invalid. This bit is reset by hardware after the update operation of IWDG_PR register

20. Window watchdog timer (WWDG)

20.1. Overview

The window watchdog timer (WWDG) is used to detect system failures due to software malfunctions. After the window watchdog timer starts, the value of downcounter reduces progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared). The watchdog timer also causes a reset if the counter is refreshed before the counter reached the window register value. So the software should refresh the counter in a limited window. The window watchdog timer generates an early wakeup status flag when the counter reaches 0x40 or refreshes before the counter reaches the window value. Interrupt occurs if it is enabled.

The window watchdog timer clock is prescaled from the APB2 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

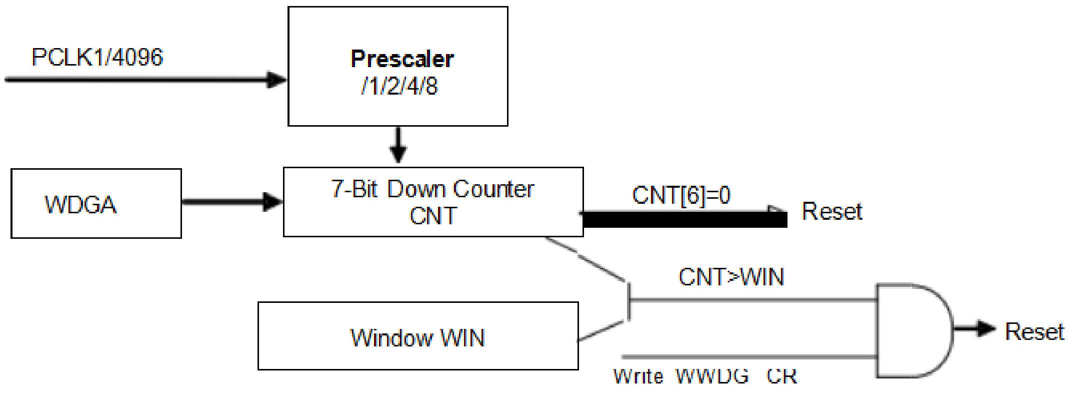
20.2. Characteristics

- Programmable free-running 7-bit downcounter.
- Generate reset in two conditions when WWDG is enabled:
  - Reset when the counter reached 0x3F.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Early wakeup interrupt (EWI): if the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40 or refreshes before it reaches the window value.
- WWDG debug mode, the WWDG can stop or continue to work in debug mode.

20.3. Function overview

If the window watchdog timer is enabled (set the WDGA bit in the WWDG_CR), the watchdog timer cause a reset when the counter reaches 0x3F (the CNT[6] bit becomes cleared), or when the counter is refreshed before the counter reaches the window register value.

Figure 20-1. Window watchdog timer block diagram



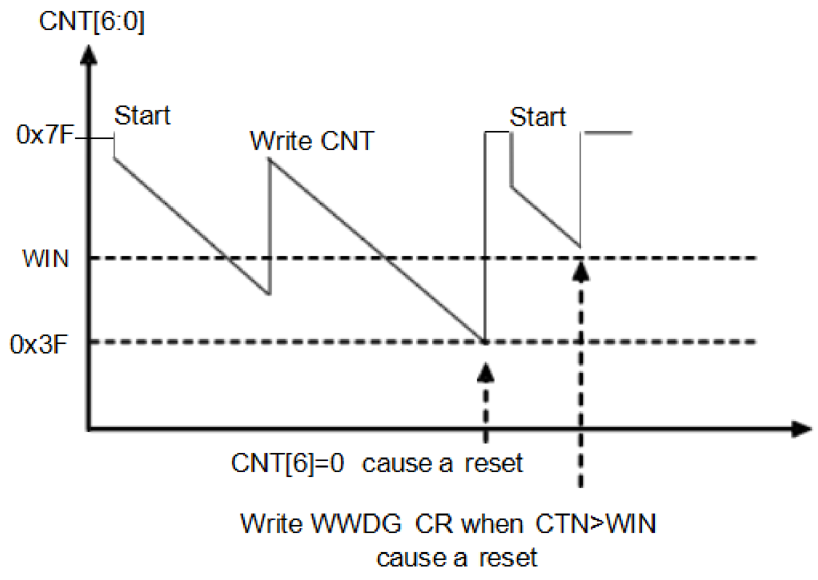
The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDGA bit in the WWDG_CR register. Whenever window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, it implies that the CNT[6] bit should be set. The CNT[5:0] determine the maximum time interval of two reloading. The countdown speed depends on the APB2 clock and the prescaler (PSC[1:0] bits in the WWDG_CFR register).

The WIN[6:0] bits in the configuration register (WWDG_CFR) specifies the window value. The software can prevent the reset event by reloading the downcounter when counter value is less than the window value and greater than 0x3F, otherwise the watchdog causes a reset.

The early wakeup interrupt (EWI) is enabled by setting the EWIE bit in the WWDG_CFR register, and the interrupt is generated when the counter reaches 0x40 or the counter is refreshed before it reaches the window value. The software can do something such as communication or data logging in the interrupt service routine (ISR) in order to analyse the reason of software malfunctions or save the important data before resetting the device. Moreover the software can reload the counter in ISR to manage a software system check and so on. In this case, the WWDG will never generate a WWDG reset but can be used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Figure 20-2. Window watchdog timing diagram



Calculate the WWDG timeout by using the formula below.

$$t_{WWDG}=t_{PCLK2} \times 4096 \times 2^{PSC} \times (CNT[5:0] + 1) \quad \text{(ms)} \qquad (20\text{-}1)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK2}: APB2 clock period measured in ms

If the DBG_WWDG_STOP bit in DBG module is cleared, the WWDG continues to work even the Cortex™-M3 core halted (Debug mode). While the DBG_WWDG_STOP bit is set, the WWDG stops in Debug mode.

20.4. Register definition

20.4.1. Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x0000 007F

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7	WDGA	RS	Start the window watchdog timer. Cleared by a hardware reset. Writing 0 has no effect. 0: Window watchdog timer disabled 1: Window watchdog timer enabled
6:0	CNT	RW	The value of the watchdog timer counter. A reset occurs when the value of this counter decreases from 0x40 to 0x3F. When the value of this counter is greater than the window value, writing this counter also causes a reset

20.4.2. Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

Bits	Fields	R/W	Description
31:10	-	R	Reserved
9	EWI	RS	Early wakeup interrupt enable. An interrupt occurs when the counter reaches 0x40 or the counter is refreshed before it reaches the window value if the bit is set. It can be cleared by a hardware reset or by a RCC WWDG software reset. A write operation of ‘0’ has no effect.
8:7	WDGTB	RW	Prescaler. The time base of the watchdog timer counter 00: (PCLK2 / 4096) / 1 01: (PCLK2 / 4096) / 2 10: (PCLK2 / 4096) / 4 11: (PCLK2 / 4096) / 8
6:0	WIN	RW	The Window value. A reset occurs if the watchdog counter (CNT bits in WWDG_CR) is written when the value of the watchdog counter is greater than the Window value.

20.4.3. Status register (WWDG_SR)

Address offset: 0x08

Reset value: 0x0000 0000

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	EWIF	RC_W0	Early wakeup interrupt flag. When the counter reaches 0x40 or refreshes before it reaches the window value, this bit is set by hardware even the interrupt is not enabled (EWI in WWDG_CFR is cleared). This bit is cleared by writing 0 to it. There is no effect when writing 1 to it.

21. Flash controller (FMC)

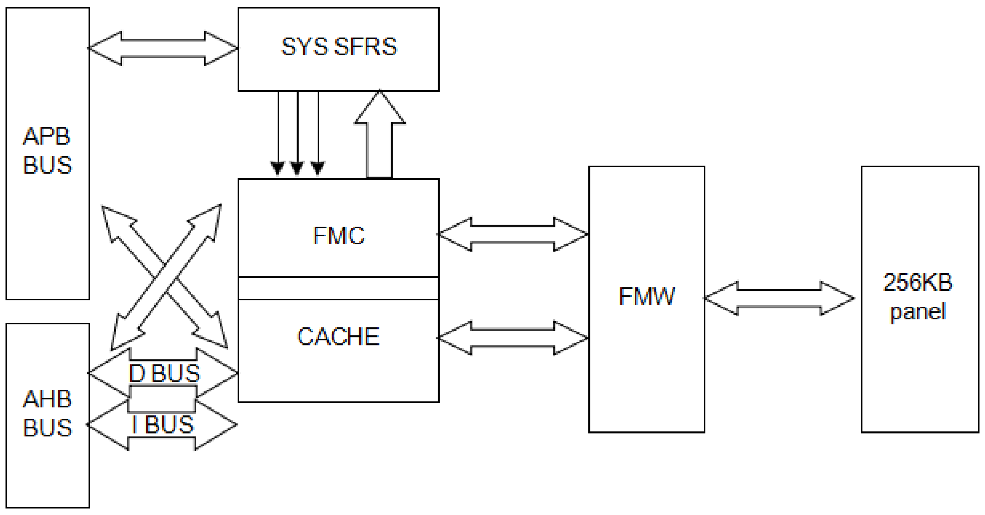
21.1. Flash controller overview

Flash controller is the interface between flash macro and the AHB bus in the chip. The whole flash macro can be configured, erased, programmed or read through flash controller respectively. Beyond that, flash data integrity CRC check also can be done through controller. To speed up the code execution, a cache implemented in the flash controller.

21.2. Major features of flash controller

- Support page program, page erase, block erase, chip erase and page write latch operation.
- Support block protection
- Support flash data integration check CRC: $X^{16} + X^{15} + X^{2} + 1$
- Self calibration for flash controller when power on. The configure data can be read out and updated each time power up.
- Operation code fetch and data fetch will be suspend when programming or erase the flash.

Figure 21-1. Flash connection Figure



21.3. Function description

21.3.1. Write Operation

- Write operation flow
  - ■ Erase operation
    - ◆ Main array support page erase, block erase and chip erase
    - ◆ Information block only support page erase

- Program Operation
  - Support page program
  - Erase operation should be done before program operation
  - Program Operation:
    - Erase the target page
    - Write Data to the page write latch
    - Start page program

21.3.2. CRC calculation

- Support flash data integrity CRC check.
- CRC calculation, please reference to CRCON register.
- Flash macro read cycle is 15-20ns cycle/Period should be inserted according to system frequency and should be configured in CRCON
- After CRC calculation finish, an interrupt will be triggered.

Table 21-1. CRC Calculation configure

System frequency	CRC Calculation (CRCON.PERIOD)
36MHz	0
48MHz	1
72MHz	2
96MHz	3
128MHz	4

21.4. Register definition

21.4.1. CRC control register (FMC_CRCON)

Address offset: 0x4

Reset Value: 0x0000 F000

Bits	Fields	R/W	Description
31-26	-	R	Reserved
25:16	CRCLEN	RW	CRC data depth for calculation 0:CRC base on 1 page 1:CRC base on 2 page ......
15:12	PERIOD	RW	read cycles while doing CRC check 0: Single cycle for flash read 1: Two cycles for flash read ...... 15: Sixteen cycles for flash read

Bits	Fields	R/W	Description
11:9	-	R	Reserved
8	CRCFIE	RW	CRC calculation interrupt enable, interrupt generate when CRC calculation finish
7:4	-	R	Reserved
3	SLOWRD	RW	control the CRC calculation period 1: Insert 16 clock cycles between 2 adjacent CRC calculation 0: No wait cycle between 2 adjacent CRC calculation
2	PAUSE	RW	suspend CRC calculation
1	CRCF	RW	CRC calculation finish symbol 1: CRC accomplished 0: CRCEN=1
0	CRCEN	RW	set 1 to start CRC

Note: Writing to CRCON is not effective during flash erase/write operation.

21.4.2. Address register (FMC_ADDR)

Address offset: 0x10

Reset value: 0x0000 0000

Only reset when power on reset occur

● Chip erase:	Do not need be configured
● Page erase, Page program:	Page address
● Block erase:	Block address
● CRC check:	Start address

21.4.3. Data register (FMC_DATA1)

Address offset: 0x1C

Reset value: 0x0000 0000

Only reset when power on reset occur

- This register includes the CRC check result after CRC operation done

21.4.4. Cache register (FMC_BUFx) x=0...63

Address offset: 0x100~0x1FF

Reset value: 0x0000 0000

During page program operation, the data should be written into BUF0~BUF63. Such data will be stored in page latch and used when write operation kicked off.

This register is write only.

No write operation effective while flash in write or erase operation.

22. USB FULL SPEED DEVICE (USB)

22.1. USB introduction

The USB peripheral implements an interface between a full-speed USB2.0 bus and the AHB bus.

USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

22.2. USB main features

- USB specification version 2.0 full-speed compliant
- The USB modules is configurable for 3 IN endpoints and for 3 OUT endpoints in addition to Endpoint 0.
- Support Bulk/Interrupt/Isochronous transfers
- USB Suspend/Resume operations

22.3. USB FIFO

The USB FIFO size is 1024B and is a dedicated RAM. The FIFO size configuration for endpoints 0~3 is shown in Table 22.1

Table 22-1. Dedicated RAM allocation table

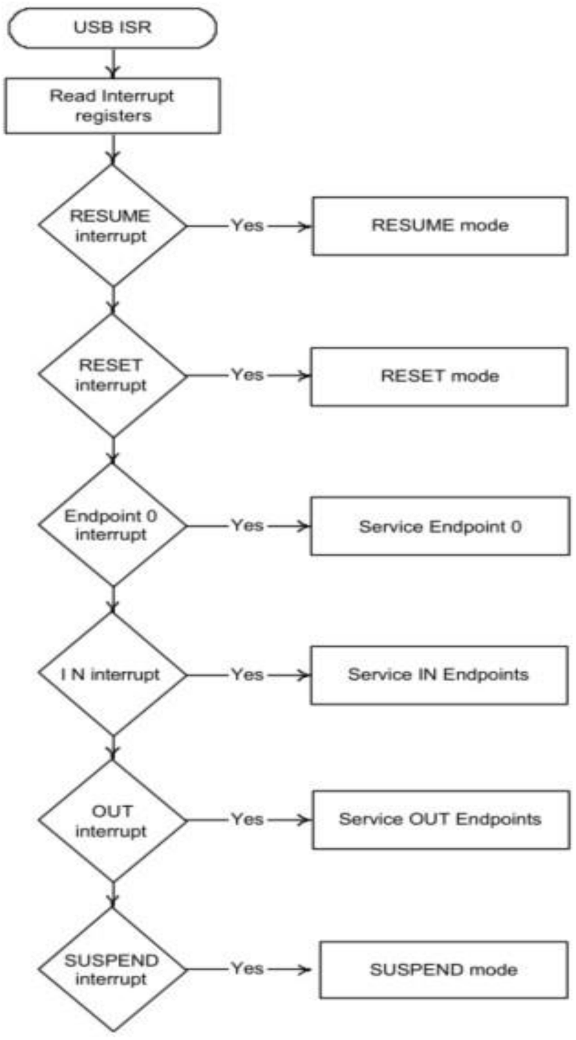
Base addresses	Endpoint No.	Size
0x0000	Endpoint 0	64B
0x0040	Endpoint 1 IN	128B
0x00C0	Endpoint 1 OUT	128B
0x0140	Endpoint 2 IN	128B
0x01C0	Endpoint 2 OUT	128B
0x0240	Endpoint 3 IN	128B
0x02C0	Endpoint 3 OUT	128B
0x0340	unused	192B

22.4. Programming Scheme

22.4.1. USB Interrupt Handling

When the CPU is interrupted with a USB interrupt, it needs to read the interrupt status register to determine which endpoint(s) have caused the interrupt and jump to the appropriate routine. If multiple endpoints have caused the interrupt, Endpoint 0 should be serviced first, followed by the other endpoints. The Suspend interrupt should be serviced last.

Figure 22-1. USB Interrupt handling



22.5. USB Reset

When a reset condition is detected on the USB, the USB modules performs the following actions:

- Sets FAddr to 0.
- Sets Index to 0.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all interrupts, except Suspend.
- Generates a Reset interrupt.

When the software receives a Reset interrupt, it should close any open pipes and wait for USB enumeration to begin.

22.6. Suspend/Resume

When the USB modules sees no activity on the USB for 3 ms, it will generate a Suspend interrupt (if enabled).

It is up to the software to decide what, if anything, to disable when the USB is in Suspend mode.

22.6.1. USB modules active during suspend

The USB may exit Suspend mode by sending Resume signaling on the bus.

If the USB modules is left active while the USB is in Suspend mode, the USB modules will monitor the USB for Resume signaling. When Resume signaling appears on the bus, the USB modules will generate a Resume interrupt.

22.6.2. USB modules inactive during suspend

When the Suspend interrupt described above is received, the software may disable the USB modules by stopping its clock (this must be done by some external means). However, because the USB modules is disabled, it will not then be able to detect Resume signaling on the USB. As a result, some external hardware will be needed to detect Resume signaling (by monitoring the DIM and DIP signals), so that the clock to the USB modules can be restarted.

22.6.3. Remote wake up

If the USB modules is in Suspend mode and the software wants to initiate a remote wake up, it should write to the Power register to set the Resume bit (D2) to 1. (If the clock to the USB modules has been stopped, it will need to be restarted before this write can occur.)

The software should leave this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) then reset it to 0. By this time the hub should have taken over driving Resume signaling on the USB.

Note: No Resume interrupt will be generated when the software initiates a remote wake up.

22.7. Endpoint 0 Handling

Endpoint 0 is the main control endpoint of the core. As such, the routines required to service Endpoint 0 are more complicated than those required to service other endpoints.

The software is required to handle all the Standard Device Requests that may be received via Endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transaction per transfer. To accommodate this, the CPU needs to take a state machine approach to command decoding and handling.

The Standard Device Requests can be divided into three categories: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of events that the software must perform to process the different types of device request.

Note: The Setup packet associated with any Standard Device Request should include an 8-byte command. Any Setup packet containing a command field of anything other than 8 bytes will be automatically rejected by the USB modules core.

22.7.1. Zero Data Requests

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred.Examples of zero data Standard Device Requests are: SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit (CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO, decoded and the appropriate action taken. For example if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FAddr register.

The CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has been read from the FIFO) and to set the DataEnd bit (D3) (indicating that no further data is expected for this request). When the host moves to the status stage of the request, a second Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the second interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6) and to set the SendStall bit (D5). When the host moves to the status stage of the request, the USB modules will send a STALL to tell the host that the request was not executed. A second Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

If the host sends more data after the DataEnd bit has been set, then the USB modules will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

22.7.2. Write Requests

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a write Standard Device Request is: SET_ DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit (CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has been read from the FIFO) but in this case the DataEnd bit (D3) should not be set (indicating that more data is expected).

When a second Endpoint 0 interrupt is received, the CSR0 register should be read to check the endpoint status. The OutPktRdy bit (CSR0.D0) should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the Endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for Endpoint 0, further data packets will be sent. In this case, CSR0 should be written to set the ServicedOutPktRdy bit, but the DataEnd bit should not be set.

When all the expected data packets have been received, the CSR0 register should be written to set the ServicedOutPktRdy bit and to set the DataEnd bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6) and to set the SendStall bit (D5). When the host sends more data, the USB modules will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

If the host sends more data after the DataEnd has been set, then the USB modules will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

22.7.3. Read Requests

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of read Standard Device Requests are: GET_ CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_ FRAME.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit (CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded. The CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the Endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for Endpoint 0, only the maximum packet size should be written to the FIFO. The CSR0 register should then be written to set the InPktRdy bit (D1) (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another Endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the CSR0 register should be written to set the InPktRdy bit and to set the DataEnd bit (D3) (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6) and to set the SendStall bit (D5). When the host requests data, the USB modules will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

If the host requests more data after DataEnd (D3) has been set, then the USB modules will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

22.7.4. Endpoint 0 states

The Endpoint 0 control needs three modes – IDLE, TX and RX – corresponding to the different phases of the control transfer and the states Endpoint 0 enters for the different phases of the transfer.

The default mode on power-up or reset should be IDLE.

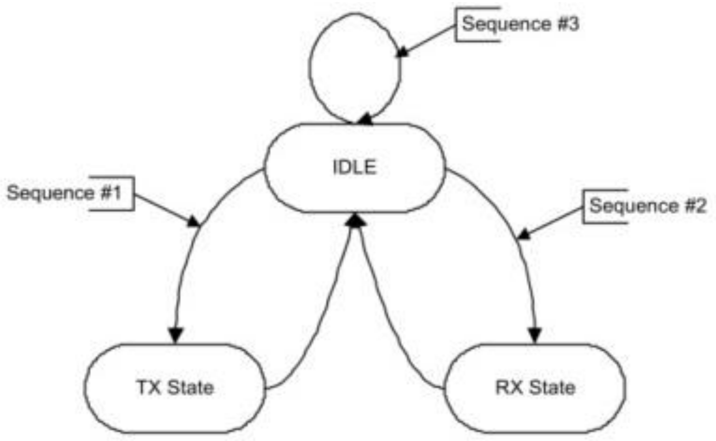
OutPktRdy (CSR0.D0) becoming set when Endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the USB modules decodes the descriptor to find whether there is a Data phase and, if so, the direction of the Data phase for the control transfer (in order to set the FIFO direction).

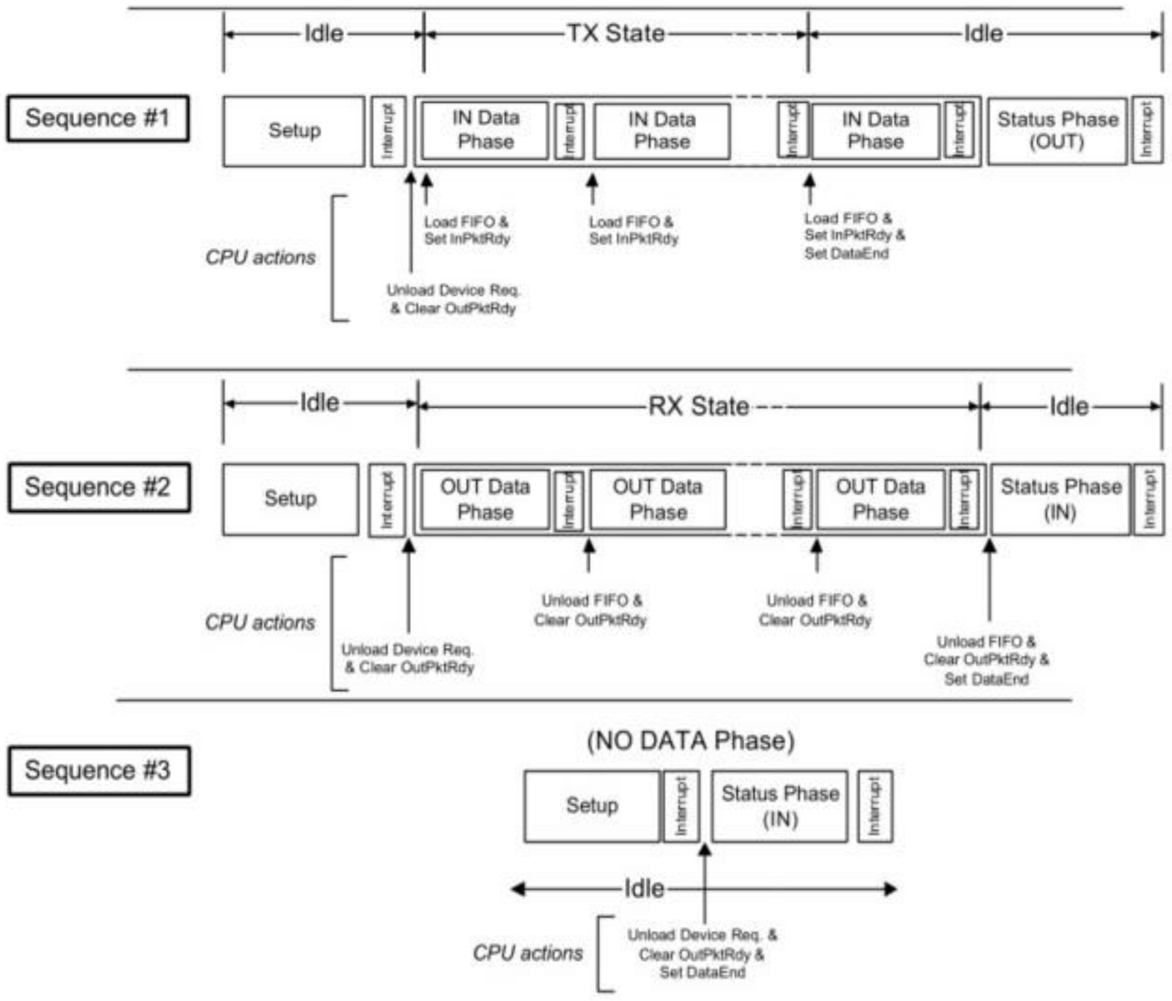
Depending on the direction of the Data phase, Endpoint 0 goes into either TX state or RX state. If there is no Data phase, Endpoint 0 remains in IDLE state to accept the next device request.

The actions that the CPU needs to take at the different phases of the possible transfers (e.g. Loading the FIFO, Setting InPktRdy) (see Figure 22-2).

Note that the USB modules changes the FIFO direction depending on the direction of the Data phase independently of the CPU.

Figure 22-2. ENDPOINT 0 STATES





22.7.5. Endpoint 0 Service Routine

An Endpoint 0 interrupt is generated:

- When the core sets the OutPktRdy bit (CSR0.D0) after a valid token has been received and data has been written to the FIFO.

- When the core clears the InPktRdy bit (CSR0.D1) after the packet of data in the FIFO has been successfully transmitted to the host.
- When the core sets the SentStall bit (CSR0.D2) after a control transaction is ended due to a protocol violation.
- When the core sets the SetupEnd bit (CSR0.D4) because a control transfer has ended before DataEnd (CSR0.D3) is set.

Whenever the Endpoint 0 service routine is entered, the firmware must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SentStall bit would be set. If the control transfer ends due to a premature end of control transfer, the SetupEnd bit would be set. In either case, the firmware should abort processing the current control transfer and set the state to IDLE.

Once the firmware has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the Endpoint state.

If Endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the core receiving data from the USB bus. The service routine must check for this by testing the OutPktRdy (CSR0.D0) bit. If this bit is set, then the core has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the core must take.Depending on the command contained within the SETUP packet, Endpoint 0 will enter one of three states:

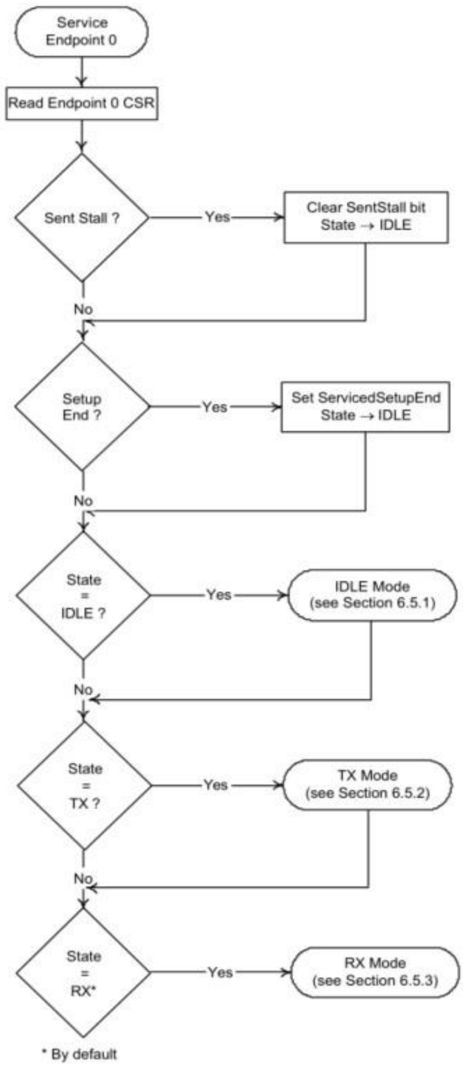
- If the command is a single packet transaction (SET_ADDRESS, SET_INTERFACE etc.) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET_DESCRIPTOR etc.), the endpoint will enter RX state.
- If the command has an IN data phase (GET_DESCRIPTOR etc.), the endpoint will enter TX state.

If the endpoint is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent.

The firmware must respond to this either by placing more data in the FIFO if the host is still expecting more data 2 or by setting the DataEnd bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, Endpoint 0 should be returned to IDLE state to await the next control transaction.

If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The firmware must respond by unloading the received data from the FIFO. The firmware must then determine whether it has received all of the expected data 2. If it has, the firmware should set the DataEnd bit and return Endpoint 0 to IDLE state. If more data is expected, the firmware should set the ServicedOutPktRdy bit (CSR0.D6) to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

Figure 22-3. ENDPOINT 0 SERVICE ROUTINE

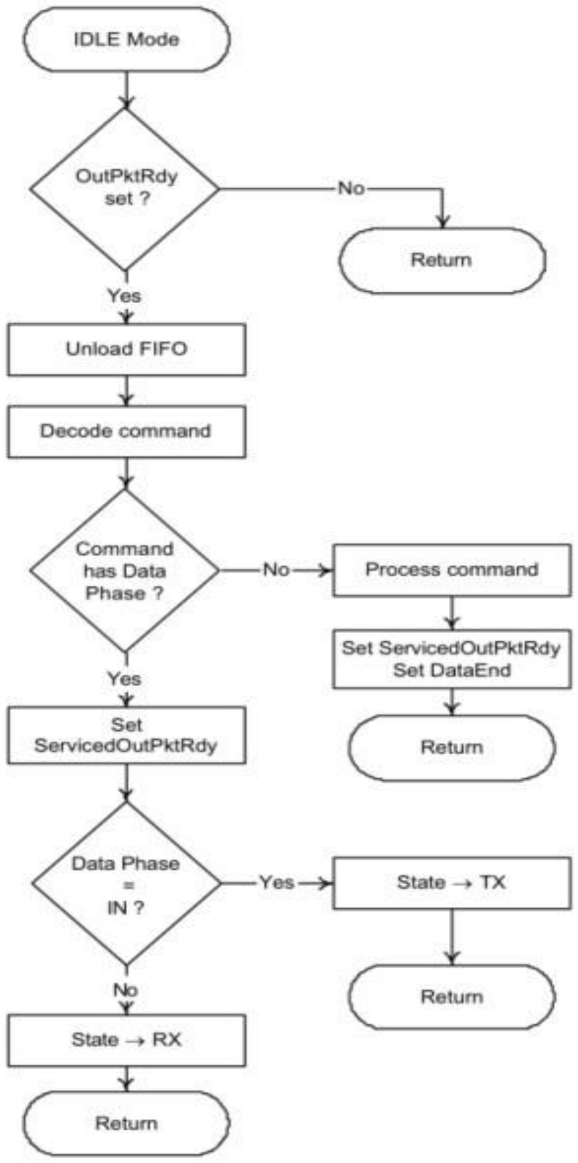

Service Endpoint 0
Read Endpoint 0 CSR
Sent Stall ?
Yes
Clear SentStall bit State → IDLE
No
Setup End ?
Yes
Set ServicedSetupEnd State → IDLE
No
State = IDLE ?
Yes
IDLE Mode (see Section 6.5.1)
No
State = TX ?
Yes
TX Mode (see Section 6.5.2)
No
State = RX*
Yes
RX Mode (see Section 6.5.3)
* By default

IDLE mode

IDLE mode is the mode the Endpoint 0 control needs to select at power-on or reset and is the mode to which the Endpoint 0 control should return when the RX and TX modes are terminated.

It is also the mode in which the SETUP phase of control transfer

Figure 22-4. SETUP Phase of Control Transfer



TX mode

When the endpoint is in TX state, all arriving IN tokens need to be treated as part of a Data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received whilst the endpoint is in the TX state, this will cause a SetupEnd condition to occur as the core expects only IN tokens.

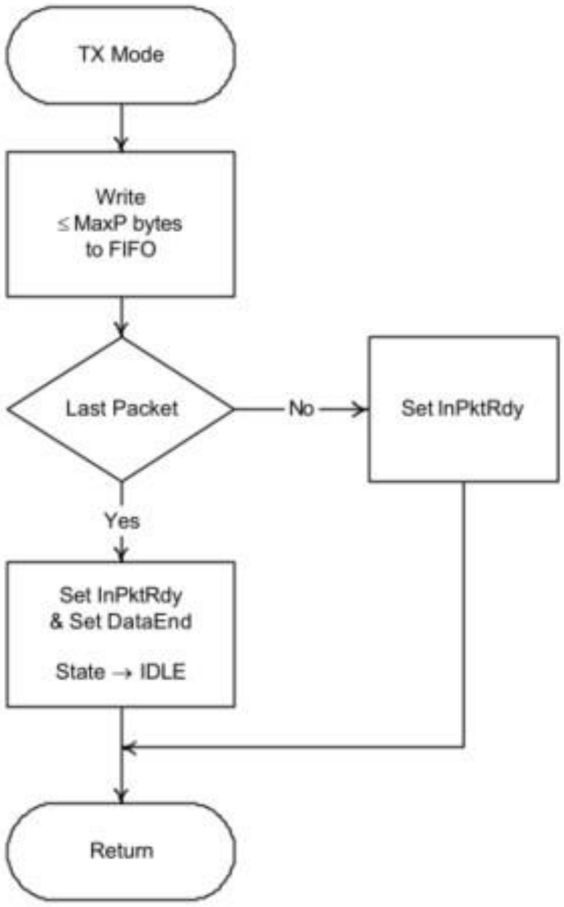
Three events can cause TX mode to be terminated before the expected amount of data has been sent:

- The host sends an invalid token causing a SetupEnd condition (CSR0.D4 set)
- The firmware sends a packet containing less than the maximum packet size for Endpoint 0 (MaxP)
- The firmware sends an empty data packet

Until the transaction is terminated, the firmware simply needs to load the FIFO when it receives an interrupt which indicates that a packet has been sent from the FIFO. (An interrupt is generated when InPktRdy is cleared.)

When the firmware forces the termination of a transfer (by sending a short or empty data packet), it should set the DataEnd bit(CSR0.D3) to indicate to the core that the Data phase is complete and that the core should next receive an acknowledge packet.

Figure 22-5. IN Data Phase for Control transfer



RX mode

In RX mode, all arriving data should be treated as part of a Data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, this will cause a SetupEnd condition to occur as the core expects only OUT tokens.

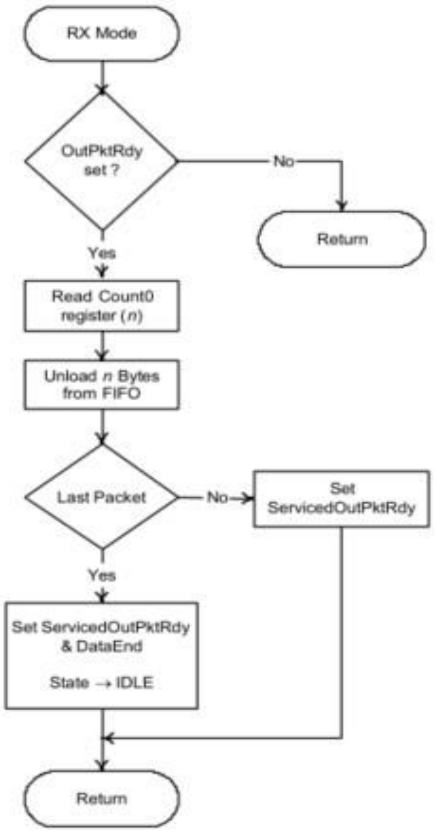
Three events can cause RX mode to be terminated before the expected amount of data has been received:

- The host sends an invalid token causing a SetupEnd condition (CSR0.D4 set)
- The host sends a packet which contains less than the maximum packet size for Endpoint 0
- The host sends an empty data packet

Until the transaction is terminated, the firmware simply needs to unload the FIFO when it receives an interrupt which indicates that new data has arrived (OutPktRdy (CSR0.D0) set) and to clear OutPktRdy by setting the ServicedOutPktRdy bit (CSR0.D6).

When the firmware detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DataEnd bit (CSR0.D3) to indicate to the core that the Data phase is complete and that the core should receive an acknowledge packet next.

Figure 22-6. OUT Data Phase for Control Transfer



22.7.6. Error Handling

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the function controller software wishes to abort the transfer (e.g. because it cannot process the command).

The USB modules will automatically detect protocol errors and send a STALL packet to the host under the following conditions:

- The host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DataEnd bit (CSR0.D3) has been set.
- The host request more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DataEnd bit in the CSR0 register has been set.
- The host sends more than MaxP data bytes in an OUT data packet.

- The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the USB modules has sent the STALL packet, it sets the SentStall bit (CSR0.D2) and generates an interrupt. When the software receives an Endpoint 0 interrupt with the SentStall bit set, it should abort the current transfer, clear the SentStall bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SetupEnd bit (CSR0.D4) will be set and an Endpoint 0 interrupt generated. When the software receives an Endpoint 0 interrupt with the SetupEnd bit set, it should abort the current transfer, set the ServicedSetupEnd bit (CSR0.D7), and return to the IDLE state. If the OutPktRdy bit (CSR0.D0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SendStall bit (CSR0.D5). The USB modules will then send a STALL packet to the host, set the SentStall bit (CSR0.D2) and generate an Endpoint 0 interrupt.

22.8. Bulk IN Endpoint

A Bulk IN endpoint is used to transfer non-periodic data from the function controller to the host.

Three optional features are available for use with a Bulk IN endpoint:

- Double packet buffering

If the value written to the InMaxP register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host.

- AutoSet

When the AutoSet feature is enabled, the INPKTRDY bit (INCSR1.D0) will be automatically set when a packet of InMaxP bytes has been loaded into the FIFO.

22.8.1. Setup Enpoint

Before using a Bulk IN endpoint, the INMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRINE register should be set to 1 (if an interrupt is required for this endpoint) and the INCSR2 register should be set as shown below：

7	AUTOSET	0/1	Set to 1 if the AutoSet feature is required.
6	ISO	0	Set to 0 to enable Bulk protocol.
3	FRCDATATOG	0	Set to 0 to allow normal data toggle operation.

When a Bulk IN endpoint is first configured, following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0, the INCSR1 register should be written to set the CLRDATATOG bit. This will ensure that the data toggle (which is handled automatically by the USB modules) starts in the correct state. Also, if there are any data packets in the FIFO (indicated by the FIFONE bit being set), they should be flushed by setting the FLUSHFIFO bit(INCSR1.D3).

Note: It may be necessary to set this bit twice in succession if double buffering is enabled.

22.8.2. Enpoint Operation

When data is to be transferred over a Bulk IN pipe, a data packet needs to be loaded into the FIFO and the InCSR1 register written to set the INPKTRDY bit. When the packet has been sent, the INPKTRDY bit will be cleared by the USB modules and an interrupt generated so that the next packet can be loaded into the FIFO. If double packet buffering is enabled, then after the first packet has been loaded and the InPktRdy bit set, the InPktRdy bit will immediately be cleared by the USB modules and an interrupt generated so that a second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

The packet size must not exceed the size specified in the InMaxP register. When a block of data larger than InMaxP is to be transferred, it must be sent as multiple packets. These packets should be InMaxP in size, except the last packet which holds the residue. The host may determine that all the data for a transfer has been sent by knowing the total amount of data that is expected. Alternatively it may infer that all the data have been sent when it receives a packet which is less than InMaxP in size. In the latter case, if the total size of the data block is a multiple of InMaxP, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting InPktRdy when the next interrupt is received, without loading any data into the FIFO

22.8.3. Error Handling

If the software wants to shut down the Bulk IN pipe, it should set the SendStall bit (INCSR1.D4). When the USB modules receives the next IN token, it will send a STALL to the host, set the SentStall bit (INCSR1.D5) and generate an interrupt.

When the software receives an interrupt with the SentStall bit (INCSR1.D5) set, it should clear the SentStall bit. It should leave the SendStall bit (INCSR1.D4) set until it is ready to re-enable the Bulk IN pipe. Note: If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SendStall bit set until the software is ready to re-enable the Bulk IN pipe. When a pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOGb it in the INCSR1 register.

22.9. Bulk OUT Endpoint

A Bulk OUT endpoint is used to transfer non-periodic data from the host to the function controller.

Three optional features are available for use with a Bulk OUT endpoint:

- Double packet buffering

If the value written to the OutMaxP register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO.

- AutoClear

When the AutoClear feature is enabled, the OutPktRdy bit (OutCSR1.D0) will be automatically cleared when a packet of OutMaxP bytes has been unloaded from the FIFO.

22.9.1. Setup Enpoint

Before using a Bulk OUT endpoint, the OutMaxP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRINE register should be set to 1 (if an interrupt is required for this endpoint) and the OUTCSR2 register should be set as shown below:

7	AUTOCLR	0/1	Set to 1 if the AutoClear feature is required.
6	ISO	0	Set to 0 to enable Bulk protocol.

When a Bulk OUT endpoint is first configured, following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0, OUTCSR1 should be written to set the CLRDATATOG bit. This will ensure that the data toggle (which is handled automatically by the USB modules) starts in the correct state. Also, if there are any data packets in the FIFO (indicated by the OUTPKTRDY bit being set), they should be flushed by setting the FLUSHFIFO bit.

Note: It may be necessary to set this bit twice in succession if double buffering is enabled.

22.9.2. Enpoint Operation

When a data packet is received by a Bulk OUT endpoint, the OUTPKTRDY bit (OutCSR1.D0) is set and an interrupt is generated.The software should read the two OutCount registers for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, then the OUTPKTRDY bit should be cleared.

The packet sizes should not exceed the size specified in the OUTMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than OutMaxP is to be sent to the function, it will be sent as multiple packets. All the packets will be OUTMAXP in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than OUTMAXP in size. (If the total size of the data block is a multiple of OUTMAXP, a null data packet will be sent after the data to signify that the transfer is complete.)

22.9.3. Error Handling

If the software wants to shut down the Bulk OUT pipe, it should set the SendStall bit (OUTCSR1.D5). When the USB modules receives the next packet it will send a STALL to the host, set the SentStall bit(OUTCSR1.D6) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (OUTCSR1.D6) set, it should clear the SENTSTALL bit. It should leave the SENTSTALL bit (OUTCSR1.D5) set until it is ready to re-enable the Bulk OUT pipe.

Note: If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SENTSTALL bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit in the OUTCSR1 register.

22.10. Interrupt In Endpoint

An Interrupt IN endpoint is used to transfer periodic data from the function controller to the host.

An Interrupt IN endpoint uses the same protocol as a Bulk IN endpoint and can be used the same way.

Interrupt IN endpoints also support one feature that Bulk IN endpoints do not, in that they support continuous toggle of the data toggle bit. This feature is enabled by setting the FRCDATATOG bit in the INCSR2 register. When this bit is set to 1, the USB modules will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

22.11. Interrupt Out Endpoint

An Interrupt OUT endpoint is used to transfer periodic data from the host to a function controller.

An Interrupt OUT endpoint uses almost the same protocol as a Bulk OUT endpoint and can be used the same way.

22.12. Isochronous IN Endpoint

An Isochronous IN endpoint is used to transfer periodic data from the function controller to the host.

Three optional features are available for use with an Isochronous IN endpoint:

- Double packet buffering

Double packet buffering is automatically enabled when the value written to the INMAXP register is less than or equal to half the size of the FIFO allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Note: Double packet buffering is generally advisable for Isochronous IN endpoints in order to avoid data runderrun.

- AutoSet

When the AutoSet feature is enabled, the INPKTRDY bit will be automatically set when a packet of INMAXP bytes has been loaded into the FIFO. However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size and the INCSR1 register needs to be accessed following every packet to check for runderrun errors.

22.12.1. Setup Endpoint

Before using an Isochronous IN endpoint, the INMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRINE register should be set to 1 (if an interrupt is required for this endpoint) and the INCSR2 register should be set as shown below：

7	AUTOSET	0/1	Set to 1 if the AutoSet feature is required.
6	ISO	1	Set to 1 to enable Isochronous protocol.
3	FRCDATATOG	0	Ignored in Isochronous mode.

22.12.2. Enpoint Operation

An Isochronous endpoint does not support data retries, so if data runderrun is to be avoided, the data to be sent to the host must be loaded into the FIFO before the IN token is received. The host will send one IN token per frame, however the time within the frame can vary. If an IN token is received near the end of one frame and then at the start of the next frame, there will be little time to reload the FIFO. For this reason, double buffering is usually required for an Isochronous IN endpoint.

The AutoSet feature can be used with an Isochronous IN endpoint, in the same way as for a Bulk IN endpoint. However,unless the data arrives from the source at an absolutely consistent rate, synchronized to the host’s frame clock, the size of the packets sent to the host will have to increase or decrease from frame to frame to match the source data rate. This means that the actual packet sizes will not always be INMAXP in size, rendering the AutoSet feature useless.

An interrupt is generated whenever a packet is sent to the host and the software may use this interrupt to load the next packet into the FIFO and set the InPktRdy bit in the INCSR1 register in the same way as for a Bulk IN endpoint. As the interrupt could occur almost any time within a frame, depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using either the SOF interrupt (INTUSB.D3) or the external SOF_PULSE signal from the USB modules to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame when a SOF packet is received. (The USB modules also maintains an external frame counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to set the INPKTRDY bit in INCSR1 and to check for data overruns/underruns.

Starting up a double-buffered Isochronous IN pipe can be a source of problems. Double buffering requires that a data packet is not transmitted until the frame after it is loaded. There is no problem if the function loads the first data packet at least a frame before the host sets up the pipe (and therefore starts sending IN tokens). But if the host has already started sending IN tokens by the time the first packet is loaded, the packet may be transmitted in the same frame as it is loaded, depending on whether it is loaded before, or after, the IN token is received. This potential problem can be avoided by setting the ISOUD bit in the POWER register. When this bit is set to 1, any data packet loaded into an Isochronous IN endpoint FIFO will not be transmitted until after the next SOF packet has been received, thereby ensuring that the data packet is not sent too early.

22.12.3. Error Handling

If the endpoint has no data in its FIFO when an IN token is received, it will send a null data packet to the host and set the UNDERRUN bit in the INCSR1 register. This is an indication that the software is not supplying data fast enough for the host.It is up to the application to determine how this error condition is handled.

If the software is loading one packet per frame and it finds that the INPKTRDY bit in the INCSR1 register (D0) is set when it wants to load the next packet, this indicates that a data packet has not been sent (perhaps because an IN token from the host was corrupted). It is up to the application how it handles this condition: it may choose to flush the unsent packet by setting the FLUSHFIFO bit in the INCSR1 register, or it may choose to skip the current packet.

22.13. Isochronous OUT Endpoint

An Isochronous OUT endpoint is used to transfer periodic data from the host to the function controller.

Three optional features are available for use with an Isochronous OUT endpoint:

- Double packet buffering

Double packet buffering is automatically enabled when the value written to the OUTMAXPregister is less than or equal to half the size of the FIFO allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Note: Double packet buffering is generally advisable for Isochronous OUT endpoints in order to avoid data overrun.

- AutoClear

When the AutoClear feature is enabled, the OUTPKTRDYbit will be automatically cleared when a packet of OUTMAXP bytes has been unloaded from the FIFO. However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size and the OUTCSR1 register needs to be accessed following every packet to check for Overrun or CRC errors.

22.13.1. Setup Enpoint

Before using an Isochronous OUT endpoint, the OUTMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTROUTE register should be set to 1 (if an interrupt is required for this endpoint) and the OUTCSR2 register should be set as shown below：

7	AUTOCLR	0/1	Set to 1 if the AutoClear feature is required.
6	ISO	1	Set to 1 to enable Isochronous protocol.

22.13.2. Enpoint Operation

An Isochronous endpoint does not support data retries so, if a data overrun is to be avoided, there must be space in the FIFO to accept a packet when it is received. The host will send one packet per frame, however the time within the frame can vary. If a packet is received near the end of one frame and another arrives at the start of the next frame, there will be little time to unload the FIFO. For this reason, double buffering is usually required for an Isochronous OUT endpoint.

The AutoClear feature can be used with an Isochronous OUT endpoint, in the same way as for a Bulk OUT endpoint.However, unless the data sink receives data at an absolutely consistent rate and is synchronized to the host’s frame clock, the size of the packets sent from the host will have to increase or decrease from frame to frame to match the required data rate.This means that the actual packet sizes will not always be OUTMAXP in size, rendering the AutoClear feature useless.An interrupt is generated whenever a packet is received from the host and the software may use this interrupt to unload the packet from the FIFO and clear the OUTPKTRDY bit in the OUTCSR1 register in the same way as for a Bulk OUT endpoint.

As the interrupt could occur almost any time within a frame, depending on when the host has scheduled the transaction, the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware,

it may be better to minimize the requirement for additional buffering by waiting until the end of each frame before unloading the FIFO. This can be done by using either the SOF interrupt (INTRUSB.D3) or the external SOF_PULSE signal from the USB modules to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame when a SOF packet is received. (The USB modules also maintains an external frame counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to clear the OUTPKTRDY bit in OUTCSR1 and to check for data overruns/underruns

22.13.3. Error Handling

If there is no space in the FIFO to store a packet when it is received from the host, the OverRun bit in the OutCSR1 register will be set. This is an indication that the software is not unloading data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the USB modules finds that a received packet has a CRC error, it will still store the packet in the FIFO and set the OUTPKTRDY bit (OUTCSR1.D0) and the DATAERROR bit (OUTCSR1.D3). It is left up to the application how this error condition is handled.

22.14. USB Registers

Table 22-2. USB Registers

Name	Address offset	Description	Reset value
USB interrupt Registers			
INTRIN	0x02	USB interrupt register for IN endpoint	0x00

Name	Address offset	Description	Reset value
INTROUT	0x04	USB interrupt register for OUT endpoint	0x00
INTRUSB	0x06	USB interrupt flag register	0x00
INTRINE	0x07	USB interrupt enable register for INTRIN	0x00
INTROUTE	0x09	USB OUT interrupt enable register for INTROUT	0x00
INTRUSBE	0x0B	USB Interrupt enable registers	0x00
USB common register			
FADDR	0x00	USB function address register	0x00
POWER	0x01	Power management register	0x00
FRAMEL	0x0C	Frame number low register (bits 0 to 7)	0x00
FRAMEH	0x0D	Frame number high register (bits 8 to 15)	0x00
INDEX	0x0E	Index register for selecting the endpoint status and control registers.	0x00
FIFOx	0x20 + 0x04 × x	FIFOs for Endpoints 0 to 3 (x from 0 to 3)	NA
USB frame number registers			
INMAXP	0x10	Setup maximum packet size for IN endpoint	0x00
CSR0	0x11	Control Status register for Endpoint 0.	0x00
INCSR1	0x11	Control Status register 1 for IN endpoint	0x00
INCSR2	0x12	Control Status register 2 for IN endpoint	0x20
OUTMAXP	0x13	Setup maximum packet size for OUT endpoint	0x00
OUTCSR1	0x14	Control Status register 1 for OUT endpoint.	0x00
OUTCSR2	0x15	Control Status register 2 for OUT endpoint.	0x00
COUNT0	0x16	Number of received bytes in Endpoint 0 FIFO	0x00
OUTCOUNTL	0x16	Number of bytes in OUT endpoint FIFO (lower byte).	0x00
OUTCOUNTH	0x17	Number of bytes in OUT endpoint FIFO (upper byte).	0x00

22.14.1. USB Interrupt Registers

The USB interrupt Registers are used for interrupt handling of USB module.

USB interrupt register for IN endpoint (USB_INTRIN)

Address offset: 0x02

Reset value: 0x00

USB_INTRIN is an 8-bit read-only register that indicates which of the interrupts for IN endpoints 1 – 3 are currently active, it also indicates whether the endpoint 0 interrupt is currently active(Bits 4-7 are Reserved) . The bits corresponding to endpoint 0 and the IN endpoints included in the design are set to 1 when their IN interrupt event occurs.

Note: All active interrupts will be cleared when this register is read.

Bits	Fields	R/W	Description
7:4	-	R	Reserved
3	IN3	R	Interrupt flag of IN endpoint 3.
2	IN2	R	Interrupt flag of IN endpoint 2.
1	IN1	R	Interrupt flag of IN endpoint 1.
0	EP0	R	Interrupt flag of endpoint 0.

USB interrupt register for OUT endpoint (USB_INTROUT)

Address offset: 0x04

Reset value: 0x00

USB_INTRIN is an 8-bit read-only register that indicates which of the interrupts for OUT endpoints 1– 3 are currently active (Bits0, 4-7 are Reserved). The bits corresponding to the OUT endpoints are set to 1 when their OUT interrupt event occurs.

Note: All active interrupts will be cleared when this register is read.

Bits	Fields	R/W	Description
7:4	-	R	Reserved
3	OUT3	R	Interrupt flag of OUT endpoint 3.
2	OUT2	R	Interrupt flag of OUT endpoint 2.
1	OUT1	R	Interrupt flag of OUT endpoint 1.
0	-	R	Reserved, always return zero.

USB interrupt flag register (USB_INTRUSB)

Address offset: 0x06

Reset value: 0x00

USB_INTRIN is an 8-bit read-only register that indicates which USB interrupts are currently active. The bits corresponding to the USB interrupts are set to 1 when their interrupt event occurs.

Note: All active interrupts will be cleared when this register is read

Bits	Fields	R/W	Description
7:4	-	R	Reserved.
3	SOFIS	R	Set 1 at the start of each frame.
2	RSTIS	R	Set 1 when Reset signaling is detected on the bus.
1	RSUIS	R	Set 1 when Resume signaling is detected on the bus while the USB modules is in Suspend mode.
0	SUSIS	R	Set when Suspend signaling is detected on the bus.

USB interrupt enable register for INTRIN (USB_INTRINE)

Address offset: 0x06

Reset value: 0x0F

Bits	Fields	R/W	Description
7:4	-	R	Reserved.
3	IN3E	RW	Set whether to enable the interrupt of the IN endpoint 3. 0: Disable. 1: Enable.
2	IN2E	RW	Set whether to enable the interrupt of the IN endpoint 2. 0: Disable. 1: Enable.
1	IN1E	RW	Set whether to enable the interrupt of the IN endpoint 1. 0: Disable. 1: Enable.
0	EP0E	RW	Set whether to enable the interrupt of the endpoint 0. 0: Disable. 1: Enable.

USB interrupt enable register for INTROUT (USB_INTROUTE)

Address offset: 0x09

Reset value: 0x0E

Bits	Fields	R/W	Description
7:4	-	R	Reserved.
3	OUT3E	RW	Set whether to enable the interrupt of the OUT endpoint 3. 0: Disable. 1: Enable.
2	OUT2E	RW	Set whether to enable the interrupt of the OUT endpoint 2. 0: Disable. 1: Enable.
1	OUT1E	RW	Set whether to enable the interrupt of the OUT endpoint 1. 0: Disable. 1: Enable.
0	-	R	Reserved, always return zero.

USB Interrupt enable registers (USB_INTRUSBE)

Address offset: 0x0B

Reset value: 0x06

Bits	Fields	R/W	Description
7:4	-	R	Reserved.
3	SOFIE	RW	Set whether to enable the interrupt of each frame start. 0: Disable. 1: Enable.
2	RSTIE	RW	Set whether to enable the interrupt of reset. 0: Disable. 1: Enable.
1	RSUIE	RW	Set whether to enable the interrupt of resume. 0: Disable.

			1: Enable.
0	SUSIE	RW	Set whether to enable the interrupt of suspend. 0: Disable. 1: Enable.

22.14.2. USB common register

USB Function address register (USB_FADDR)

Address offset: 0x00

Reset value: 0x00

Bits	Fields	R/W	Description
7	UPDATE	R	Determine whether the function address has been updated. The WIP is set to 1 When the new address value is written into the FADDR register and is reset to 0 when the new address takes effect (at the end of the current transfer). 0: The last address written to FADDR has taken effect. 1: The address at which FADDR was last written has not yet taken effect.
6:0	FADDR	RW	The function address. Save the 7-bit function address of USB. When endpoint 0 receives the standard request of SET_ADDRESS, the CPU should write that address into the FADDR register. The new address takes effect upon completion of the device request.

USB Power management register (USB_POWER)

Address offset: 0x01

Reset value: 0x00

Bits	Fields	R/W	Description
7	ISOUD	RW	ISO updata. This bit is only used by endpoints performing Isochronous transfers. 0: The USB module will wait for an IN token from the time INRDY is set 1 before sending the packet. 1: When set by the CPU the USB modules will wait for an SOF token from the time INRDY is set 1 before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent.
6:4	-	R	Reserved, always return zero.
3	USBRST	R	Detects the reset signal on the USB bus. This read only bit is set while Reset signaling is present on the bus. 0: No reset signal detected on the bus. 1: Reset signal detected on the bus.
2	RESUME	RW	Resume. Set by the CPU to generate Resume signaling when the function is in Suspend mode (USB wake up the host). The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling.
1	SUSMD	R	Suspend Mode. Set by the USB device when Suspend mode is entered. Cleared when the CPU reads the interrupt register, or sets the Resume bit of this register. 0: The USB modules is not in suspend mode. 1: The USB modules is in suspend mode.
0	SUSEN	RW	Enable Suspend.

			Set by the software to enable entry into Suspend mode when Suspend signaling is received on the bus. 0: Disable. 1: Enable.

Frame Number Low Register (USB_FRAMEL)

Address offset: 0x0C

Reset value: 0x00

Bits	Fields	R/W	Description
7:0	FRAMEL	R	Holds the lower 8 bits of the last received frame number.

Frame Number High Register (USB_FRAMEH)

Address offset: 0x0D

Reset value: 0x00

Bits	Fields	R/W	Description
7:3	-	R	Reserved
2:0	FRAMEH	R	Holds the upper 3 bits of the last received frame number.

Index register for selecting the endpoint status and control registers. (USB_INDEX)

Address offset: 0x0E

Reset value: 0x00

Each IN endpoint and each OUT endpoint have their own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map.

Bits	Fields	R/W	Description
7:4	-	R	Reserved
3:0	EPSEL	RW	Selected endpoint. Index is a 4-bit register that determines which endpoint control/status registers are accessed at addresses 10h to 17h. 0000：endpoint 0 0001：endpoint 1 0010：endpoint 2 0011：endpoint 3 0100-1111：Reserved.

FIFOs for Endpoints 0 to 3 (USB_FIFOx)

Address offset: 0x20 + 0x04 × x (x from 0 to 3)

Access to the FIFO for each endpoint is achieved through the corresponding FIFOX register. Writing to these addresses loads data into the IN FIFO for the corresponding endpoint. Reading from these addresses unloads data from the OUT FIFO for the corresponding endpoint.

Bits	Fields	R/W	Description

7:0	FIFODATA	RW	Access to the FIFO for endpoint.

22.14.3. USB Frame Number Registers

Setup maximum packet size for IN endpoint (USB_INMAXP)

Address offset: 0x10

Reset value: 0x00

INMAXP is an 8-bit register that holds the maximum packet size for transactions through the currently-selected IN endpoint – in units of 8 bytes, except that a value of 128 sets the maximum packet size to 1023 (the maximum size for an Isochronous packet transferred in a Full-speed transaction) rather than 1024. In setting this value, you should note the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transactions in Full-speed operations.

There is an INMAXP register for each IN endpoint (except Endpoint 0).

The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint (see Universal Serial Bus Specification Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

If a value greater than the configured IN FIFO size for the endpoint is written to this register, the value will be automatically changed to the IN FIFO size. If the value written to this register is less than, or equal to, half the IN FIFO size, two IN packets can be buffered.

The register is reset to 0. If this register is changed after packets have been sent from the endpoint, the endpoint IN FIFO should be completely flushed (using the FLUSHFIFO bit (D3) in INCSR1) after writing the new value to this register.

Bits	Fields	R/W	Description
7:0	INMAXP	RW	Maximum Packet Size/transaction for IN endpoint.

Control Status register for Endpoint 0 (USB_CSR0)

Address offset: 0x11

Reset value: 0x00

CSR0 is an 8-bit register that provides control and status bits for Endpoint 0.

Bits	Fields	R/W	Description
7	SVDSETUPEND	S	The software writes a 1 to this bit to clear the SETUPEND bit. It is cleared automatically.
6	SVDOUTPKTRDY	S	The software writes a 1 to this bit to clear the OUTPKTRDY bit. It is cleared automatically.
5	SENDSTALL	S	The software writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically
4	SETUPEND	R	This bit will be set when a control transaction ends before the DATAEND bit has been set. The bit is cleared by the software writing a 1 to the SVDSETUPEND bit.
3	DATAEND	S	The software sets this bit: 1. When setting INPKTRDY for send the last data packet. 2. When the software sets 1 to SVDOUTPKTRDY after receive the last

Bits	Fields	R/W	Description
			data packet. 3. When setting INPKTRDY for a zero length data packet. It is cleared automatically.
2	SENTSTALL	R_W0	This bit is set by hardware when a STALL handshake is transmitted. The software should clear this bit.
1	INPKTRDY	RS	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated when the bit is cleared.
0	OUTPKTRDY	R	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The software clears this bit by setting the SVDOUTPKTRDY bit.

Control Status register 1 for IN endpoint (USB_INCSR1)

Address offset: 0x11

Reset value: 0x00

Bits	Fields	R/W	Description
7	-	R	Reserved, always return zero.
6	CLRDATATOG	S	The software writes a 1 to this bit to reset the endpoint IN data toggle to 0.
5	SENTSTALL	R_W0	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the INPKTRDY bit is cleared. The software should clear this bit.
4	SENDSTALL	RW	The software writes a 1 to this bit to issue a STALL handshake to an IN token. The software clears this bit to terminate the STALL condition. This bit has no effect if the IN endpoint is in ISO mode.
3	FLUSHFIFO	S	The software writes a 1 to this bit to flush the next packet to be transmitted from the endpoint IN FIFO. The FIFO pointer is reset and the InPktRdy bit (below) is cleared. Note: If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO.
2	UNDERRUN	R_W0	In ISO mode, this bit is set when a zero length data packet is sent after receiving an IN token with the InPktRdy bit not set. In Bulk/Interrupt mode, this bit is set when a NAK is returned in response to an IN token. The software should clear this bit.
1	FIFONE	R_W0	This bit is set when there is at least 1 packet in the IN FIFO. 0：IN FIFO be empty. 1：The IN FIFO contains one or more packets.
0	INPKTRDY	RS	The software sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

Control Status register 2 for IN endpoint (USB_INCSR2)

Address offset: 0x12

Reset value: 0x20

Bits	Fields	R/W	Description
7	AUTOSET	RW	If the CPU sets this bit, INPKTRDY will be automatically set when data of the maximum packet size (value in INMAXP) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, then InPktRdy will have to be set manually. When 2 packets are in the IN FIFO then

Bits	Fields	R/W	Description
			INPKTRDYwill also be automatically set when the first packet has been sent, if the second packet is the maximum packet size.
6	ISO	RW	The software sets this bit to enable the IN endpoint for isochronous transfers (ISO mode), and clears it to enable the IN endpoint for bulk/interrupt transfers. 0: Set the IN endpoint for bulk/interrupt transfers. 1: Set the IN endpoint for isochronous transfers.
5	-	R	Reserved.
4	-	R	Reserved, must be zero.
3	FRCDATATOG	RW	The software sets this bit to force the endpoints IN data toggle to switch after each data packet is sent regardless of whether an ACK was received. This can be used by interrupt IN endpoints which are used to communicate rate feedback for Isochronous endpoints 0： The endpoints IN data toggle to switch after an ACK was received. 1：The endpoints IN data toggle to switch after each data packet is sent regardless of whether an ACK was received.
2:0	-	R	Reserved, always return zero.

Setup maximum packet size for OUT endpoint (USB_OUTMAXP)

Address offset: 0x13

Reset value: 0x00

OutMaxP is an 8-bit register that holds the maximum packet size for transactions through the currently-selected OUT endpoint–in units of 8 bytes, except that a value of 128 sets the maximum packet size to 1023 (the maximum size for an isochronous packet) rather than 1024. In setting this value, you should note the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transactions in Full-speed operations.There is an OutMaxP register for each OUT endpoint (except Endpoint 0).

The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint (see Universal Serial Bus Specification Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the OUT endpoint, and should not exceed half the FIFO size if double-buffering is required. If a value greater than the configured OUT FIFO size for the endpoint is written to this register, the value will be automatically changed to the OUT FIFO size. If the value written to this register is less than, or equal to, half the OUT FIFO size, two OUT packets can be buffered.

Bits	Fields	R/W	Description
7:0	OUTMAXP	RW	Maximum Packet Size/transaction for OUT endpoint.

Control Status register 1 for OUT endpoint. (USB_OUTCSR1)

Address offset: 0x14

Reset value: 0x00

Bits	Fields	R/W	Description
7	CLRDATATOG	S	The software writes a 1 to this bit to reset the endpoint data toggle to 0.
6	SENTSTALL	R_W0	This bit is set when a STALL handshake is transmitted. The software should clear this bit.
5	SENDSTALL	RW	The software writes a 1 to this bit to issue a STALL handshake. The software clears this bit to terminate the stall condition. This bit has no effect if the OUT endpoint is in ISO mode.
4	FLUSHFIFO	S	The software writes a 1 to this bit to flush the next packet to be read from the endpoint OUT FIFO. Note: If the FIFO contains two packets, FLUSHFIFO will need to be set twice to completely clear the FIFO.
3	DATAERROR	R	This bit is set when OUTPKTRDY is set if the data packet has a CRC or bit-stuff error. it is cleared when OUTPKTRDY is cleared. The bit is only valid in ISO mode.
2	OVERRUN	R_W0	This bit is set if an OUT packet cannot be loaded into the OUT FIFO. The CPU should clear this bit. The bit is only valid in ISO mode.
1	FIFOFULL	R	This bit is set when no more packets can be loaded into the OUT FIFO.
0	OUTPKTRDY	R_W0	This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated when the bit is set.

Control Status register 2 for OUT endpoint (USB_OUTCSR2)

Address offset: 0x15

Reset value: 0x00

Bits	Fields	R/W	Description
7	AUTOCLR	RW	If the CPU sets this bit then the OutPktRdy bit will be automatically cleared when a packet of OutMaxP bytes has been unloaded from the OUT FIFO. When packets of less than the maximum packet size are unloaded, OutPktRdy will have to be cleared manually.
6	ISO	RW	The CPU sets this bit to enable the OUT endpoint for Isochronous transfers, and clears it to enable the OUT endpoint for Bulk/Interrupt transfers. 0: Set the OUT endpoint for Bulk/Interrupt transfers. 1: Set the OUT endpoint for Isochronous transfers.
5:4	-	R	Reserved, must be zero.
3:0	-	R	Reserved, always return zero.

Number of received bytes in Endpoint 0 FIFO (USB_COUNT0)

Address offset: 0x16

Reset value: 0x00

Bits	Fields	R/W	Description
7	-	R	Reserved.
6:0	COUNT0	R	The value returned is valid while OUTPKTRDY (CSR0.D0) is set.

Number of bytes in OUT endpoint FIFO (lower byte) (USB_OUTCOUNTL)

Address offset: 0x16

Reset value: 0x00

Bits	Fields	R/W	Description
7:0	OUTCOUNTL	R	The value returned is valid while OUTPKTRDY is set

Number of bytes in OUT endpoint FIFO (upper byte) (USB_OUTCOUNTH)

Address offset: 0x17

Reset value: 0x00

Bits	Fields	R/W	Description
7:3	-	R	Reserved.
2:0	OUTCOUNTH	R	The value returned is valid while OUTPKTRDY is set.

23. Serial peripheral interface (SPI)

23.1. Introduction

Serial peripheral interface (SPI) allows the chip to communicate with external devices in half /full duplex, synchronous and serial mode. This interface can be configured as a master mode or a slave mode to communicate with external devices.

There are two independent slave mode SPI modules and one master mode SPI module in the chip.

23.2. SPI features

Each SPI module has the following characteristics:

- APB interface
- Support DMA transmission
- Two data FIFO s with depth of 4 and width of 16 bits, corresponding to Transfer and receiving respectively
- Four configurable clock modes
- The data format is right aligned
- Support Motorola SPI protocol
- Support Texas Instruments SSP protocol
- Support National Semiconductor Microwire protocol

Figure 23-1. SPI structure

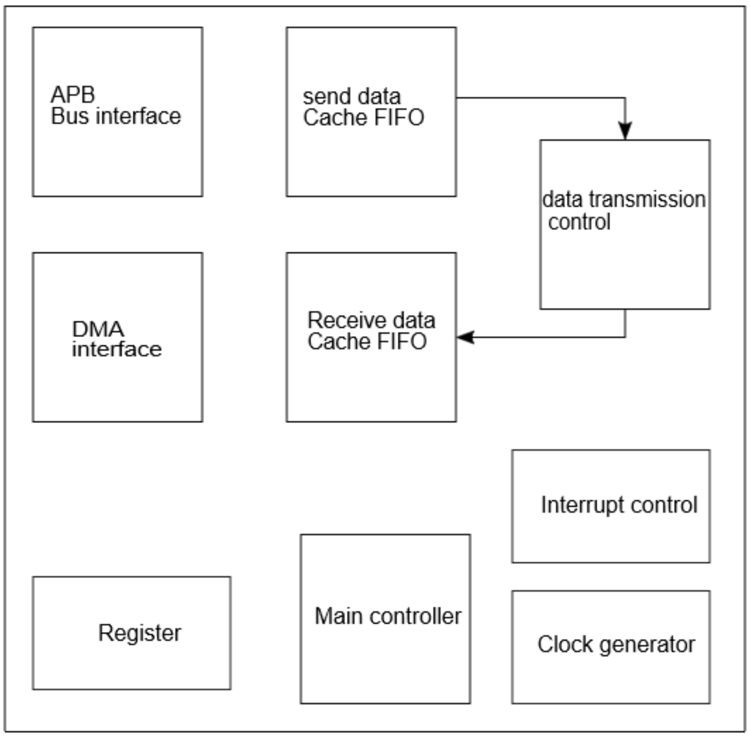
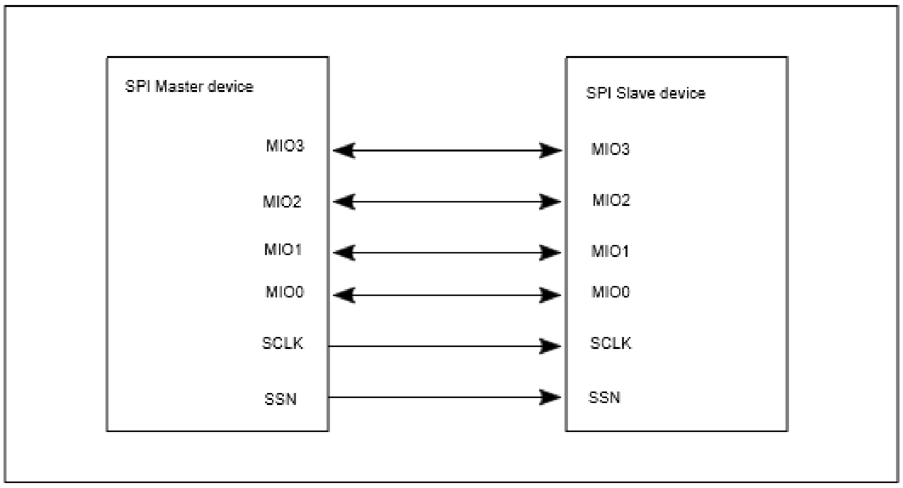


Figure 23-2. SPI connection diagram



23.3. Clock mode

Four clock modes are supported:

- CPOL,CPHA = 00
- CPOL,CPHA = 01
- CPOL,CPHA = 10
- CPOL,CPHA = 11

The CPOL and CPHA bits of SPI_CR0 register can be combined into four possible timing relationships.

The CPOL bit controls the level of the clock in the idle state. This bit is valid for both master mode and slave mode devices; If CPOL = 0, SCLK is idle If CPOL = 1, SCLK remains high in idle state.

The CPHA bit controls the phase of the clock. If CPHA = 0, the clock edge appears in the middle of the data bit, and when the data begins to transmit, the second edge of SCLK is used for data sampling; if CPHA = 1, the clock edge appears at the beginning of the data bit, and when the data begins to transmit, the second edge of SCLK appears is used for data sampling.

Figure 23-3. SPI clock mode: CPHA = 0

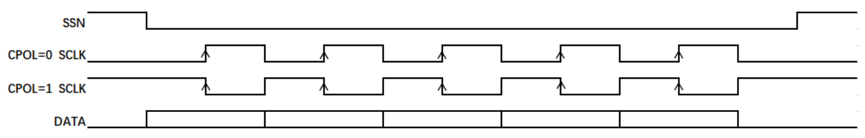
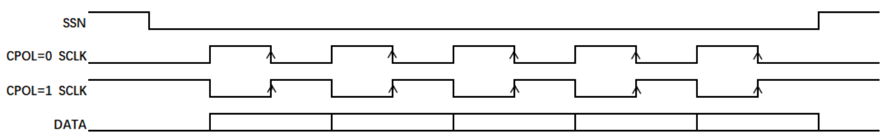


Figure 23-4. SPI clock mode: CPHA = 1



23.4. Transfer Modes

When transmitting data on the serial bus, three transfer modes are supported, which is controlled by TMOD bit of register SPI_CR0.

Transmit & Receive

When TMOD = 2‘b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

Transmit Only

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

Receive Only

When TMOD = 2‘b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The TXD output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

23.5. SPI slave mode

There are two independent slave mode SPI modules in the chip: SPIS1and SPIS2.

SPIS1 is on the APB1 bus and SPIS2 is on the APB2 bus. The two slave mode SPI modules can communicate with the master mode SPI device outside the chip. All serial transfers are initiated and controlled by the serial bus master. The SCLK clock is controlled by the master mode device, and the clock mode (CPOL, CPHA) of the slave mode device must be the same as that of the master mode device.

The serial slave enables its TXD data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line, driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

The serial slave remains in an idle state until selected by the bus master. When not actively transmitting data, the slave must hold its TXD line in a high impedance state to avoid interference with serial transfers to other slave devices. The slave continues to transfer data to and from the master device as long as it is selected.

Note: Before the master mode device sends the clock, the slave mode module must be configured and the data FIFO of them must be ready, otherwise unexpected data transmission error may occur.

GPIO configuration

- SPIS1 pin configuration. If SPIS1 is used, the following GPIO should be configured first:
    - SSN corresponds to GPIOA PORT4 or GPIOA PORT15, and the alternate mode should be configured to 6
    - SCLK corresponds to GPIOA PORT5 or GPIOB PORT3, and the alternate mode should be configured to 6
    - MISO corresponds to GPIOA PORT6 or GPIOB PORT4, and the alternate mode should be configured to 6
    - MOSI corresponds to GPIOA PORT7 or GPIOB PORT5, and the alternate mode should be configured to 6
- SPIS2 pin configuration. If SPIS2 is used, the following GPIO should be configured first:
    - SSN corresponds to GPIOB port 12 or GPIOC port 0, and the alternate mode of the port is configured to 6
    - SCLK corresponds to GPIOB port 13 or GPIOC port 1, and the alternate mode should be configured to 6
    - MISO corresponds to GPIOB port 14 or GPIOC port 2, and the alternate mode should be configures to 6
    - MOSI corresponds to GPIOB PORT 15 or GPIOC PORT 3, and the alternate mode

should be configured to 6

Slave SPI and SSP Serial Transfers

If the serial slave is receive only (TMOD=10), the transmit FIFO need not contain valid data because the data currently in the transmit shift register is resent each time the slave device is selected. The TXE error flag in the status register (SR) is not set when TMOD=01. You should mask the transmit FIFO empty interrupt when this mode is used.

If the serial slave transmits data to the master, you must ensure that data exists in the transmit FIFO before a transfer is initiated by the serial-master device. If the master initiates a transfer to the slave when no data exists in the transmit FIFO, an error flag (TXE) is set in the status register, and the previously transmitted data frame is resent on TXD. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to generate early interrupt to the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA Controller is used for APB accesses, the DMA transmit data level register (DMATDLR) can be used to early request the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer.

The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA Controller is used for APB accesses, the DMA receive data level register (DMARDLR) can be used to early request the DMA controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing a continuous serial transfer from a serial master to the slave is described as follows:

- If the slave is enabled, disable it by writing 0 to SSIENR.
- Set up the slave control registers for the transfer. These registers can be set in any order.
  - Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).
  - Write TXFTLR and RXFTLR to set FIFO threshold levels.
  - Write the IMR register to set up interrupt masks.
- Enable the serial slave by writing 1 to the SSIENR register.
- If the transfer mode is transmit and receive (TMOD=2'b00) or transmit only (TMOD=2'b01), write data for transmission to the master into the transmit FIFO (Write DR).
- If the transfer mode is receive only (TMOD=2'b10), there is no need to write data into the transmit FIFO; the current value in the transmit shift register is retransmitted.
- The slave is now ready for the serial transfer. The transfer begins when the slave is selected by a serial-master device.
- When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).

- The transfer ends when the serial master removes the select input to the slave. When the transfer is completed, the BUSY status is reset to 0.
- If the transfer mode is not transmit only (TMOD!=2’b01), read the receive FIFO until empty.
- Disable the slave by writing 0 to SSIENR.

Slave Microwire Serial Transfers

As a slave device, the Microwire protocol operates in much the same way as the SPI protocol. There is no decode of the control frame by the slave device.

23.6. SPI master mode

There is a master mode SPI module in the chip, which is located on the APB2 bus.

The master mode SPI control the SCLK clock, and use SPI_ BAUDR register to configure the baud rate. The master initiates and controls all serial transfers on the bus. When the master is disabled, no serial transfers can occur and the SCLK clock is held in inactive state.

Data transfers are started by the serial-master device. When the master is enabled (SSI_EN=1), at least one valid data entry is present in the transmit FIFO and a serial-slave device is selected. When actively transferring data, the busy flag (BUSY) in the status register (SR) is set. You must wait until the busy flag is cleared before attempting a new serial transfer.

The master mode module can drive three SSN output signals, corresponding to three external slave mode devices.

GPIO configuration

For the master mode pin configuration, the following GPIO should be configured: (please refer to the GPIO register section)

- SSN0 corresponds to GPIOB port 12 or GPIOC port 0. Configure the port as output mode and the port's alternate mode as 5
- SSN1 corresponds to GPIOB port 7 or GPIOB port 11. Configure the port as output mode and the port's alternate mode as 5
- SSN2 corresponds to GPIOB port 8 or GPIOC port 5. Configure the port as output mode and the port's alternate mode as 5
- SCLK corresponds to GPIOB port 13 or GPIOC port 1. Configure the port as output mode and the port's alternate mode as 5
- MISO corresponds to GPIOB port 14 or GPIOC port 2, configures the port as input mode, and configures the port's alternate mode as 5
- MOSI corresponds to GPIOB port 15 or GPIOC port 3, which is configured as output mode, and the port's alternate mode is configured as 5

Master SPI and SSP Serial Transfers

When the transfer mode is “transmit and receive” or “transmit only” (TMOD = 2'b00 or TMOD = 2'b01, respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty.

For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to generate early interrupt to the processor indicating that the transmit FIFO buffer is nearly empty. When a DMA is used for APB accesses, the transmit data level (DMATDLR) can be used to early request to the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. The user may also write a block of data (at least two FIFO entries) into the transmit FIFO before enabling a serial slave. This ensures that serial transmission does not begin until the number of data-frames that make up the continuous transfer are present in the transmit FIFO.

When the transfer mode is “receive only” (TMOD = 2'b10), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The TXD output is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “number of data frames” (NDF) field in control register 1 (CTRLR1).

If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value + 1. This transfer mode increases the bandwidth of the APB bus as the transmit FIFO never needs to be serviced during the transfer. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow.

The receive FIFO threshold level (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA is used for APB accesses, the receive data level (DMARDLR) can be used to early request to the DMA Controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing an SPI or SSP serial transfer from the serial master is outlined as follows:

- If the master is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
- Set up the master control registers for the transfer; these registers can be set in any order.
  - Write Control Register 0 (CTRLR0). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
  - If the transfer mode is receive only, write CTRLR1 (Control Register 1) with the number of frames in the transfer minus 1; for example, if you want to receive four data frames, write this register with 3.
  - Write the Baud Rate Select Register (BAUDR) to set the baud rate for the transfer.
  - Write the Transmit and Receive FIFO Threshold Level registers (TXFTLR and RXFTLR, respectively) to set FIFO threshold levels.
  - Write the IMR register to set up interrupt masks.
  - The Slave Enable Register (SER) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to

writing to the Data Register (DR), the transfer does not begin until a slave is enabled.

- Enable the master by writing 1 to the SSIENR register.
- Write data for transmission to the target slave into the transmit FIFO (write DR).If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
- Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately; for more information, see the note on page 52.If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
- The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is receive only (TMOD = 2'b10), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
- If the transfer mode is not transmit only (TMOD != 2’b01), read the receive FIFO until it is empty.
- Disable the master by writing 0 to SSIENR.

Master Microwire Serial Transfers

Microwire serial transfers from the serial master are controlled by the Microwire Control Register (MWCR). The MWHS bit enables and disables the Microwire handshaking interface. The MDD bit controls the direction of the data frame (the control frame is always transmitted by the master and received by the slave). The MWMOD bit defines whether the transfer is sequential or non-sequential.

All Microwire transfers are started by the serial master when there is at least one control word in the transmit FIFO and a slave is enabled. When the master transmits the data frame (MDD = 1), the transfer is terminated by the shift logic when the transmit FIFO is empty. When the master receives the data frame (MDD = 1), the termination of the transfer depends on the setting of the MWMOD bit. If the transfer is non-sequential (MWMOD = 0), it is terminated when the transmit FIFO is empty after shifting in the data frame from the slave. When the transfer is sequential (MWMOD = 1), it is terminated by the shift logic when the number of data frames received is equal to the value in the CTRLR1 register + 1.

When the handshaking interface on the master is enabled (MWHS =1), the status of the target slave is polled after transmission. Only when the slave reports a ready status does the master complete the transfer and clear its BUSY status. If the transfer is continuous, the next control/data frame is not sent until the slave device returns a ready status.

A typical software flow for completing a Microwire serial transfer from the serial master is outlined as follows:

- If the master is enabled, disable it by writing 0 to SSIENR.
- Set up the master control registers for the transfer. These registers can be set in any order. Write CTRLR0 to set transfer parameters.
  - If the transfer is sequential and the master receives data, write CTRLR1 with the

number of frames in the transfer minus 1; for instance, if you want to receive four data frames, write this register with 3.

- Write BAUDR to set the baud rate for the transfer.
- Write TXFTLR and RXFTLR to set FIFO threshold levels.
- Write the IMR register to set up interrupt masks.

  You can write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.

- Enable the master by writing 1 to the SSIENR register.
- If the master transmits data, write the control and data words into the transmit FIFO (write DR). If the master receives data, write the control word(s) into the transmit FIFO.
- If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
- Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately.
- If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
- The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is sequential and the master receives data, the transfer is stopped by the shift control logic when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0.
- If the master receives data, read the receive FIFO until it is empty.
- Disable the master by writing 0 to SSIENR.

23.7. DMA transmission

Transfer and receiving both support DMA transmission, and users can set DMA threshold separately for transfer and receive to trigger DMA transmission.

DMA transfer threshold: set by the register SPI_DMATDLR. When the amount of data in the transmission FIFO is equal to or less than the DMA transfer threshold, a DMA request will be generated to the DMA controller on the bus.

DMA receiving threshold: set by register SPI_DMARDLR. When the amount of data in the receiving FIFO is equal to or greater than the DMA receiving threshold, a DMA request will be generated to the DMA controller on the bus.

23.8. Interrupts

Each SPI module has five interrupt sources, but only one interrupt signal is output. When the CPU receives the interrupt, it needs to read the SPI interrupt register to determine the interrupt source.

- Interrupt source:
  - Transmit FIFO empty: generated when the data to be sent in the Transmit FIFO is equal to or less than the set threshold
  - Transmit FIFO overflow: occurs when the transmit FIFO is full and new data is attempted to be written from the APB bus.
  - Receive FIFO full: when the data stored in the receive FIFO is equal to or greater than the set threshold.
  - Receive FIFO overflow: when the receive FIFO is full and new data is received.
  - Receive FIFO underflow: occurs when the receive FIFO is empty and an attempt is made to read data from the APB bus.
- FIFO threshold:
  - Set thresholds for Transmit FIFO and receive FIFO to better control the amount of data. If you do not set thresholds, the default threshold is the maximum capacity of send FIFO and receive FIFO.

23.9. SPI Slave mode register

23.9.1. SPI Control register 0(SPI_CR0)

Address offset: 0x000

Reset value: 0x0100_0007

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:25	-	Res	Reserved
24	SSTE	RW	When CPHA = 0, SSTE sets the state of SSN signal between data frames. SSTE = 1: SSN remains 1 between two consecutive data frames. SSTE = 0: SSN remains 0 between two consecutive data frames
23:16	-	Res	Reserved
15:12	CFS	RW	Control Frame Size. Selects the length of the control word for the Microwire frame format. 0000: the control word is 1 bit 0001: the control word is 2 bits 0010: the control word is 3 bits .... 1111: the control word is 16 bits
11	-	Res	Reserved
10	SLVOE	RW	Slave output enable 0: disable output from salve mode device 1: Allow output from slave mode device
9:8	TMOD	RW	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit

Bits	Fields	R/W	Description
			data are valid. 00: both Transfer and receive are valid 01: send only valid 10: Receive only valid 11: Invalid setting, setting to 11 is not allowed
7	CPOL	RW	Clock polarity 0: SCLK remains low in idle state 1: In idle state, SCLK remains high
6	CPHA	RW	Serial Clock phase, 0: data sampling starts from the first clock edge 1: Data sampling starts at the edge of the second clock
5:4	FRF	RW	Protocol selection 00: Motorola SPI protocol 01: Texas Instruments SSP protocol 10: National Semiconductor Microwire protocol 11: Invalid setting, setting to 11 is not allowed
3:0	DFS	RW	Data frame length 0000: .Invalid setting, setting to 0000 is not allowed 0001: .Invalid setting, setting to 0001 is not allowed 0010: .Invalid setting, setting to 0010 is not allowed 0011: The data frame length is 4bit 0100: The data frame length is 5bit 0101: The data frame length is 6bit .... 1111: The data frame length is 16bit

23.9.2. SPI Enable register (SPI_SPIENR)

Address offset: 0x008

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	SPI_EN	RW	SPI enable

23.9.3. MICROWIRE Control register (SPI_MWCR)

Address offset: 0x00C

Reset value: 0x0000_0000

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1	MDD	RW	data transmission 0: receive data 1: send data

0	MWMOD	RW	Continuous transmission control 0: Single transmission 1: Continuous transmission

23.9.4. TX FIFO threshold register (SPI_TXFTLR)

Address offset: 0x018

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	TFT	RW	TX FIFO threshold Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

23.9.5. Receive FIFO threshold register (SPI_RXFTLR)

Address offset: 0x01C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	RFT	RW	Receive FIFO threshold Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

23.9.6. TX FIFO status register (SPI_TXFLR)

Address offset: 0x020

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	TXTFL	R	Contains the number of valid data entries in the transmit FIFO.

23.9.7. Receive FIFO status register (SPI_RXFLR)

Address offset: 0x024

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	RXTFL	R	Contains the number of valid data entries in the receive FIFO.

23.9.8. SPI Status register (SPI_SR)

Address offset: 0x028

Reset value: 0x0000_0006

Bits	Fields	R/W	Description
31:6	-	Res	Reserved
5	TXERR	RC_R	When the transmit FIFO is empty and transmit is started, it will be set to 1. Reading this register will clear this bit
4	RFF	R	Set 1 when the receive FIFO is full and 0 when the receive FIFO is not full
3	RFNE	R	Set 1 when the receive FIFO is not empty and 0 when the receive FIFO is empty
2	TFE	R	Set 1 when the transmit FIFO is empty and 0 when the transmit FIFO is not empty
1	TFNF	R	Set 1 when the transmit FIFO is not full and 0 when the transmit FIFO is full
0	BUSY	R	Set 1 when transmission is in progress and 0 when all transmission ends

23.9.9. Interrupt enable register (SPI_IER)

Address offset: 0x02C

Reset value: 0x0000_001F

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIE	RW	Write 0 to mask receive FIFO full interrupt
3	RXOIE	RW	Write 0 to mask receive FIFO overflow interrupt
2	RXUIE	RW	Write 0 to mask receive FIFO underflow interrupt
1	TXOIE	RW	Writing 0 masks the overflow interrupt on the send FIFO
0	TXEIE	RW	Write 0 to mask send FIFO null interrupt

23.9.10. Interrupt status register (SPI_ISR)

Address offset: 0x030

Reset value: 0x0000_0000

Description: This status register is affected by interrupt enable register

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIS	R	Receive FIFO full interrupt status
3	RXOIS	R	Receive FIFO overflow interrupt status
2	RXUIS	R	Receive FIFO overflow interrupt status

1	TXOIS	R	Send FIFO overflow interrupt status
0	TXEIS	R	Send FIFO null interrupt status

23.9.11. Raw interrupt status register (SPI_RISR)

Address offset: 0x034

Reset value: 0x0000_0000

Description: This status register ignores the interrupt enable register

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIR	R	Receive FIFO full raw interrupt status
3	RXOIR	R	Receive FIFO overflow raw interrupt status
2	RXUIR	R	Receive FIFO overflow raw interrupt status
1	TXOIR	R	Send FIFO overflow raw interrupt status
0	TXEIR	R	Send FIFO null raw interrupt status

23.9.12. Transmit Overflow interrupt clear register (SPI_TXOICR)

Address offset: 0x038

Reset value: 0x0000_0000

Description: Reading this register will clear the overflow interrupt on the send FIFO

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	TXOICR	R	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.9.13. Receive Overflow interrupt clear register (SPI_RXOICR)

Address offset: 0x03C

Reset value: 0x0000_0000

Description: Reading this register will clear the overflow interrupt on the receive FIFO

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXOICR	R	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.9.14. Receive underflow interrupt clear register (SPI_RXUICR)

Address offset: 0x040

Reset value: 0x0000_0000

Description:Reading this register will clear the receive FIFO underflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXUICR	R	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.9.15. Interrupt clear register (SPI_ICR)

Address offset: 0x048

Reset value: 0x0000_0000

Description: Reading this register will clear the overflow interrupt on the transmit FIFO, the overflow interrupt on the receive FIFO and the overflow interrupt under the receive FIFO at the same time Underflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	ICR	R	Clear interrupt

23.9.16. DMA ENABLE register (SPI_DMACR)

Address offset: 0x04C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1	TDMAE	RW	DMA Send enable This bit enables/disables the transmit FIFO DMA channel. 0: DISABEL DMA transmission 1: ENABLE DMA transmission
0	RDMAE	RW	DMA Receive enable This bit enables/disables the receive FIFO DMA channel. 0: DISABEL DMA receive 1: ENABLE DMA receive

23.9.17. DMA TX threshold register (SPI_DMATDLR)

Address offset: 0x050

Reset value: 0x0000_0000

Bits	Fields	R/W	Description

31:2	-	Res	Reserved
1:0	DMATDLR	RW	DMA TX threshold

23.9.18. DMA RX threshold register (SPI_DMARDLR)

Address offset: 0x054

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	DMARDL	RW	DMA RX threshold

23.9.19. SPI Data register (SPI_DR)

Address offset: 0x060 to 0x0EC

Reset value: 0x0000_0000

Description: The DR register is a continuous address space. Writing to this space will send the data into the transmit FIFO; reading from this space will read out the data in the receive FIFO in order.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	DR	RW	SPI data Writing this register will store the data in the send FIFO Reading this register will read data from the receive FIFO

23.10. SPI master mode register

23.10.1. SPI Control register0 (SPI_CR0)

Address offset: 0x000

Reset value: 0x0100_0007

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:25	-	Res	Reserved
24	SSTE	RW	When CPHA = 0, SSTE sets the state of SSN signal between data frames. SSTE = 1: SSN remains 1 between two consecutive data frames. SSTE = 0: SSN remains 0 between two consecutive data frames
23:16	-	Res	Reserved
15:12	CFS	RW	Control Frame Size. Selects the length of the control word for the Microwire frame format. 0000: The control word is 1 bit 0001: The control word is 2 bit 0010: The control word is 3bit

Bits	Fields	R/W	Description
			.... 1111: The control word is 16 bit
11:10	-	Res	Reserved
9:8	TMOD	RW	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. 00: Both Transfer and receiving are valid 01: Send only valid 10: Receive only valid 11: Invalid setting, setting to 11 is not allowed
7	CPOL	RW	Clock polarity 0: In idle state, SCLK remains low 1: In idle state, SCLK remains high
6	CPHA	RW	Serial Clock phase 0: Data sampling starts at the edge of the first clock 1: Data sampling starts at the edge of the second clock
5:4	FRF	RW	Protocol selection 00: Motorola SPI Protocol 01: Texas Instruments SSP Protocol 10: National Semiconductor MICROWIRE Protocol 11: Invalid setting, setting to 11 is not allowed
3:0	DFS	RW	Data frame length 0000: Invalid setting, setting to 0000 is not allowed 0001: Invalid setting, setting to 0001 is not allowed 0010: Invalid setting, setting to 0010 is not allowed 0011: The data frame length is 4bit 0100: The data frame length is 5bit 0101: The data frame length is 6bit .... 1111: The data frame length is 16bit

23.10.2. SPI Control Register 1(SPI_CR1)

Address offset: 0x004

Reset value: 0x0000_0000

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	NDF	RW	Maximum data Received continuously when TMOD=10.

23.10.3. SPI Enable Register (SPI_SPIENR)

Address offset: 0x008

Reset value: 0x0000_0000

Bits	Fields	R/W	Description

31:1	-	Res	Reserved
0	SPI_EN	RW	SPI enable

23.10.4. MICROWIRE Control Register (SPI_MWCR)

Address offset: 0x00C

Reset value: 0x0000_0000

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2	MHS	RW	Handshake control 0: Handshake prohibited 1: handshake Allow
1	MDD	RW	Data transmission 0: receive data 1: transmission data
0	MWMOD	RW	Continuous transmission control 0: Single transmission 1: Continuous transmission

23.10.5. Slave Select Register (SPI_SER)

Address offset: 0x010

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2	SE2	RW	Device 2 selection 0: Unchecked 1: Select the slave device
1	SE1	RW	Device 1 selection 0: Unchecked 1: Select the slave device
0	SE0	RW	Device 0 selection 0: Unchecked 1: Select the slave device

23.10.6. Baud Rate Register (SPI_BAUDR)

Address offset: 0x014

Reset value: 0x0000_0000

Description: The output SCLK clock of master device is obtained by SPI master device input clock (SPICLK). When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description

31:16	-	Res	Reserved
15:0	SCKDV	RW	Frequency division ratio: calculate the frequency of output SCLK clock according to the following formula FSCLK= FSPI_CLK/ SCKDV

23.10.7. TX FIFO Threshold Register (SPI_TXFTLR)

Address offset: 0x018

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	TFT	RW	TX FIFO threshold Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

23.10.8. RX FIFO Threshold Register (SPI_RXFTLR)

Address offset: 0x01C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	RFT	RW	Receive FIFO threshold Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

23.10.9. TX FIFO Status Register (SPI_TXFLR)

Address offset: 0x020

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	TXTFL	R	Contains the number of valid data entries in the transmit FIFO.

23.10.10. RX FIFO Status Register (SPI_RXFLR)

Address offset: 0x024

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	RXTFL	R	Contains the number of valid data entries in the receive FIFO.

23.10.11. SPI Status Register (SPI_SR)

Address offset: 0x028

Reset value: 0x0000_0006

Bits	Fields	R/W	Description
31:6	-	Res	Reserved
5	TXERR	RC_R	When the transmit FIFO is empty and transmit is started, it will be set to 1. Reading this register will clear this bit
4	RFF	R	Set 1 when the receive FIFO is full and 0 when the receive FIFO is not full
3	RFNE	R	Set 1 when the receive FIFO is not empty and 0 when the receive FIFO is empty
2	TFE	R	Set 1 when the transmit FIFO is empty and 0 when the transmit FIFO is not empty
1	TFNF	R	Set 1 when the transmit FIFO is not full and 0 when the transmit FIFO is full
0	BUSY	R	Set 1 when transmission is in progress and 0 when all transmission ends

23.10.12. Interrupt enable register (SPI_IER)

Address offset: 0x02C

Reset value: 0x0000_001F

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIE	RW	Write 0 to mask receive FIFO full interrupt
3	RXOIE	RW	Write 0 to mask receive FIFO overflow interrupt
2	RXUIE	RW	Write 0 to mask receive FIFO underflow interrupt
1	TXOIE	RW	Writing 0 masks the overflow interrupt on the send FIFO
0	TXEIE	RW	Write 0 to mask send FIFO null interrupt

23.10.13. Interrupt status register (SPI_ISR)

Address offset: 0x030

Reset value: 0x0000_0000

Description: This status register is affected by the interrupt enable register (SPI_IER)

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIS	R	Receive FIFO full interrupt status
3	RXOIS	R	Overflow interrupt on receive FIFO status
2	RXUIS	R	Receive FIFO underflow interrupt status
1	TXOIS	R	Send FIFO overflow interrupt status

0	TXEIS	R	Send FIFO null interrupt status

23.10.14. Raw interrupt status register (SPI_RISR)

Address offset: 0x034

Reset value: 0x0000_0000

Description: This status register ignores the interrupt enable register (SPI_ IER)

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIR	R	Receive FIFO full raw interrupt status
3	RXOIR	R	Receive FIFO overflow raw interrupt status
2	RXUIR	R	Receive FIFO underflow raw interrupt status
1	TXOIR	R	Send FIFO overflow raw interrupt status
0	TXEIR	R	Send FIFO null raw interrupt status

23.10.15. Overflow interrupt clear register (SPI_TXOICR)

Address offset: 0x038

Reset value: 0x0000_0000

Description: Reading this register will clear send FIFO overflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	TXOICR	R	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.10.16. RX Overflow interrupt clear register (SPI_RXOICR)

Address offset: 0x03C

Reset value: 0x0000_0000

Description: Overflow interrupt clear register on receive FIFO

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXOICR	R	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.10.17. RX underflow interrupt clear register (SPI_RXUICR)

Address offset: 0x040

Reset value: 0x0000_0000

Description: Reading this register will clear receive FIFO underflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXUICR	R	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

23.10.18. Interrupt clear register (SPI_ICR)

Address offset: 0x048

Reset value: 0x0000_0000

Description: Reading this register will clear the transmit FIFO overflow interrupt, receive FIFO overflow interrupt and receive FIFO underflow interrupt at the same time

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	ICR	R	Clear interrupt

23.10.19. DMA Enable register (SPI_DMACR)

Address offset: 0x04C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1	TDMAE	RW	DMA transmission enable 0: disable DMA transmission 1: enable DMA transmission
0	RDMAE	RW	DMA receive enable 0: disable DMA receive 1: enable DMA receive

23.10.20. DMA TX threshold register (SPI_DMATDLR)

Address offset: 0x050

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	DMATDL	RW	DMA TX threshold setting

23.10.21. DMA RX threshold register (SPI_DMARDLR)

Address offset: 0x054

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	DMARDL	RW	DMA RX threshold setting

23.10.22. SPI Data register (SPI_DR)

Address offset: 0x060 to 0x0EC

Reset value: 0x0000_0000

Description: DR register is a continuous address space. Writing to this space will store data into the transmit FIFO in order, Read from this space reads the data in the receive FIFO sequentially.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	DR	RW	SPI Data Writing this register will store the data in the send FIFO Reading this register will read data from the receive FIFO

24. 4-wire serial peripheral interface (QSPI)

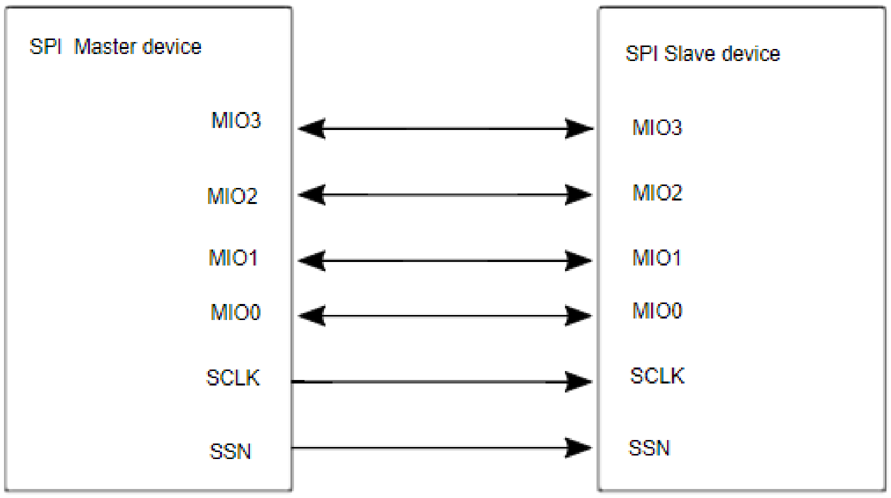
24.1. Introduction

QSPI is a SPI master mode device with four data lines, which is a special SPI protocol and also supports standard SPI protocol.

24.2. QSPI features

- APB interface
- Support DMA transmission
- Two data FIFO s with depth of 10 and width of 16 bits, corresponding to Transfer and receiving respectively
- Four configurable clock modes
- The data format is right aligned
- Support up to three 4-wire SPI slave mode devices
- Support standard SPI protocol (please refer to the chapter "serial peripheral interface SPI")

Figure 24-1. QSPI connection diagram



24.3. Clock phase and clock polarity

Four possible timing relationships can be chosen by software，Please refer to SPI section:

- CPOL,CPHA = 00
- CPOL,CPHA = 01
- CPOL,CPHA = 10
- CPOL,CPHA = 11

24.4. Transfer Modes

When transmitting data on the serial bus, three transfer modes are supported, which is controlled by TMOD bit of register SPI_CR0.

Transmit & Receive

When TMOD = 2‘b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

Transmit Only

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TXD line to the target device, which replies with data on the RXD line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

Receive Only

When TMOD = 2‘b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The TXD output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

24.5. QSPI Master Mode

QSPI needs to control the SCLK clock, and it must pass the register QSPI first_ BAUDR configures the baud rate of the master mode.

QSPI can output three SSN signals, corresponding to up to three external slave mode devices.

GPIO configuration

To use QSPI, the following GPIO should be configured first (please refer to the GPIO register section)

- SSN0 corresponds to GPIOA port 4 or GPIOA port 15, and the alternate mode of the port is configured to 5
- SSN1 corresponds to GPIOA port 13 or GPIOB port 6, and the alternate mode of this port is configured to 5
- SSN2 corresponds to GPIOA port 14 or GPIOB port 10, and the alternate mode of this port is configured as 5
- SCLK corresponds to GPIOA port 5 or GPIOB port 3, and the alternate mode of the

port is configured to 5

- MO_IO0 corresponds to GPIOA port 7 or GPIOB port 5, and the alternate mode of this port is configured to 5
- MI_IO1 corresponds to GPIOA port 6 or GPIOB port 4, and the alternate mode of this port is configured to 5
- IO2 corresponds to GPIOB port 0, and the alternate mode of this port is configured to 5
- IO3 corresponds to GPIOB port 1, and the alternate mode of this port is configured to 5

The QSPI master device will send QSPI instructions to the slave device first, and then send the address to be read and written (the instructions and addresses are collectively called control frames), and then send the QSPI instructions to the slave device

For the master SPI/SSP/Microwire serial transfers, please refer to section 23.6 for detail info. They are similar.

Data is sent to or received by the slave device.

24.6. DMA transmission

Transfer and receiving both support DMA transmission, and users can set DMA threshold separately for Transfer and receiving to trigger DMA transmission.

DMA TX threshold: via register QSPI_ DMATDLR, when the amount of data in the transmit FIFO is equal to or less than the DMA TX FIFO Threshold, a DMA request will be generated to the DMA controller on the bus.

DMA RX threshold: through register QSPI_ DMATDLR, when the amount of data in the receive FIFO is equal to or greater than the DMA RX FIFO Threshold, a DMA request is generated to the DMA controller on the bus

24.7. Interrupt

QSPI has five interrupt sources, but only one interrupt output. When the CPU receives the interrupt, it needs to read the QSPI interrupt register to determine the interrupt source. Interrupt source:

- Send FIFO empty: occurs when the data to be sent in the send FIFO is equal to or less than the set threshold
- Overflow: occurs when the transmit FIFO is full and new data is attempted to be written from the APB bus
- Full receive FIFO: when the data stored in the receive FIFO is equal to or greater than the set threshold
- Overflow of receive FIFO: occurs when the receive FIFO is full and new data is Received
- Underflow: occurs when the receive FIFO is empty and an attempt is made to read data from the APB bus

FIFO Threshold:

You can set thresholds for send FIFO and receive FIFO to better control the amount of data. If you do not set thresholds, the default threshold is the maximum capacity of send FIFO and receive FIFO.

24.8. QSPI Master Mode Register

24.8.1. QSPI Control Register 0(QSPI_CR0)

Address offset: 0x000

Reset value: 0x0100_0007

Description: When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:25	-	Res	Reserved
24	SSTE	RW	When CPHA = 0, SSTE sets the state of SSN signal between data frames. SSTE = 1: SSN remains 1 between two consecutive data frames. SSTE = 0: SSN remains 0 between two consecutive data frames
23	-	Res	Reserved
22:21	SPI_MODE	RW	SPI mode select 00: Standard SPI mode 01: Dual line SPI mode (Dual) 10: Four wire SPI mode (Quad) 11: Reserved
20:10	-	Res	Reserved
9:8	TMOD	RW	Control Frame Size. Selects the length of the control word for the Microwire frame format. 00: both Transfer and receiving are valid 01: send only valid 10: Receive only valid 11: Invalid setting, setting to 11 is not allowed
7	CPOL	RW	Clock polarity 0: SCLK remains low in idle state 1: In idle state, SCLK remains high
6	CPHA	RW	Clock phase 0: data sampling starts from the first clock edge 1: Data sampling starts at the edge of the second clock
5:4	FRF	RW	Protocol selection 00: Motorola SPI protocol 01: Texas Instruments SSP protocol 10: Invalid setting, setting to 10 is not allowed 11: Invalid setting, setting to 11 is not allowed
3:0	DFS	RW	The data frame length must be a multiple of 4 0000: invalid setting, setting to 0000 is not allowed 0001: invalid setting, setting to 0001 is not allowed 0010: invalid setting, setting to 0010 is not allowed 0011: the data frame length is 4 bits 0100: the data frame length is 5bit 0101: the data frame length is 6bit ....

			1111: the data frame length is 16bit

24.8.2. QSPI Control register 1(QSPI_CR1)

Address offset: 0x004

Reset value: 0x0000_0000

Description: This register only works when TMOD = 10 and is used to set the maximum amount of data continuously Received. When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	NDF	RW	Maximum amount of data continuously Received

24.8.3. QSPI Enable register (QSPI_SPIENR)

Address offset: 0x008

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	SPI_EN	RW	QSPI ENABLE

24.8.4. Slave select register (QSPI_SER)

Address offset: 0x010

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2	SE2	RW	device 2 selection 0: not selected 1: Select the slave device
1	SE1	RW	device 1 selection 0: not selected 1: Select the slave device
0	SE0	RW	device 0 selection 0: not selected 1: Select the slave device

24.8.5. Baud rate register (QSPI_BAUDR)

Address offset: 0x014

Reset value: 0x0000_0000

Description: The SCLK clock of QSPI output is controlled by the input clock (QSPI_CLK) frequency division. When SPI is enabled in SPI_SPIENR, this register cannot be written.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	SCKDV	RW	Frequency division ratio: calculate the frequency of output SCLK clock according to the following formula $f_{SCLK} = f_{QSPI\_CLK} / SCKDV$

24.8.6. TX FIFO threshold register (QSPI_TXFTLR)

Address offset: 0x018

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	TFT	RW	TX FIFO threshold Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

24.8.7. RX FIFO threshold register (QSPI_RXFTLR)

Address offset: 0x01C

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	RFT	RW	Receive FIFO threshold Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

24.8.8. TX FIFO status register (QSPI_TXFLR)

Address offset: 0x020

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	TXTFL	R	Contains the number of valid data entries in the transmit FIFO.

24.8.9. RX FIFO status register (QSPI_RXFLR)

Address offset: 0x024

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:3	-	Res	Reserved
2:0	RXTFL	R	Contains the number of valid data entries in the receive FIFO.

24.8.10. QSPI Status register (QSPI_SR)

Address offset: 0x028

Reset value: 0x0000_0006

Bits	Fields	R/W	Description
31:6	-	Res	Reserved
5	TXERR	RC_R	When the transmit FIFO is empty but transmit is started, it will be set to 1. Reading this register will clear this bit
4	RFF	R	Set 1 when the receive FIFO is full and 0 when the receive FIFO is not full
3	RFNE	R	Set 1 when the receive FIFO is not empty and 0 when the receive FIFO is empty
2	TFE	R	Set 1 when the transmit FIFO is empty and 0 when the transmit FIFO is not empty
1	TFNF	R	Set 1 when the transmit FIFO is not full and 0 when the transmit FIFO is full
0	BUSY	R	Set 1 when transmission is in progress and 0 when all transmission ends

24.8.11. Interrupt enable register (QSPI_IER)

Address offset: 0x02C

Reset value: 0x0000_001F

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIE	RW	Write 0 to mask receive FIFO full interrupt
3	RXOIE	RW	Write 0 to mask receive FIFO overflow interrupt
2	RXUIE	RW	Write 0 to mask receive FIFO underflow interrupt
1	TXOIE	RW	.Writing 0 masks the overflow interrupt on the send FIFO
0	TXEIE	RW	Write 0 to mask send FIFO null interrupt

24.8.12. Interrupt status register (QSPI_ISR)

Address offset: 0x030

Reset value: 0x0000_0000

Description: This status register is affected by the interrupt enable register (QSPIIER)

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIS	R	Receive FIFO full interrupt status
3	RXOIS	R	Receive FIFO overflow interrupt status
2	RXUIS	R	Receive FIFO underflow interrupt status

Bits	Fields	R/W	Description
1	TXOIS	R	Send FIFO overflow interrupt status
0	TXEIS	R	Send FIFO null interrupt status

24.8.13. Raw interrupt status register (QSPI_RISR)

Address offset: 0x034

Reset value: 0x0000_0000

Description: This status register ignores the interrupt enable register (QSPIIER)

Bits	Fields	R/W	Description
31:5	-	Res	Reserved
4	RXFIR	R	Receive FIFO full full raw interrupt status
3	RXOIR	R	Receive FIFO overflow full raw interrupt status
2	RXUIR	R	Receive FIFO underflow full raw interrupt status
1	TXOIR	R	Send FIFO overflow full raw interrupt status
0	TXEIR	R	Send FIFO null full raw interrupt status

24.8.14. TX overflow interrupt clear register (QSPI_TXOICR)

Address offset: 0x038

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	TXOICR	R	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

Description: Reading this register will clear the send FIFO overflow interrupt

24.8.15. RX overflow interrupt clear register (QSPI_RXOICR)

Address offset: 0x03C

Reset value: 0x0000_0000

Description: Reading this register will clear the receive FIFO overflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXOICR	R	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

24.8.16. Receive underflow interrupt clear register (QSPI_RXUICR)

Address offset: 0x040

Reset value: 0x0000_0000

Description: Reading this register will clear the receive FIFO underflow interrupt

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	RXUICR	R	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the interrupt; writing has no effect.

24.8.17. Interrupt clear register (QSPI_ICR)

Address offset: 0x048

Reset value: 0x0000_0000

Description: Reading this register will clear the transmit FIFO overflow interrupt, receive FIFO overflow interrupt and receive FIFO underflow interrupt at the same time

Bits	Fields	R/W	Description
31:1	-	Res	Reserved
0	ICR	R	Clear interrupt

24.8.18. DMA Enable register (QSPI_DMACR)

Address offset: 0x04C

Reset value:0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1	TDMAE	RW	DMA Send enable 0: disable DMA transmission 1: allow DMA transmission
0	RDMAE	RW	DMA Receive enable 0: disable DMA receive 1: enable DMA receive

24.8.19. DMA TX threshold register (QSPI_DMATDLR)

Address offset: 0x050

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	DMATDL	RW	DMA TX threshold setting

24.8.20. DMA RX threshold register (QSPI_DMARDLR)

Address offset: 0x054

Reset value: 0x0000_0000

Bits	Fields	R/W	Description
31:2	-	Res	Reserved
1:0	DMARDL	RW	DMA RX threshold setting

24.8.21. QSPI Data register (QSPI_DR)

Address offset: 0x060 to 0x0EC

Reset value: 0x0000_0000

Description: The DR register is a continuous address space. Writing to this address will store data into the transmit FIFO; reading from this space will read out the data in the receive FIFO.

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:0	DR	RW	QSPI Data Writing this register will store the data in the send FIFO Reading this register will read data from the receive FIFO

24.8.22. QSPI Enhanced mode configuration register (QSPI_ESPICR)

Address offset: 0x0F4

Reset value: 0x0000_0200

Bits	Fields	R/W	Description
31:16	-	Res	Reserved
15:11	WCYC	RW	QSPI waiting time When receiving, the QSPI master device needs a waiting time after Transfer the control frame (instruction and address). The slave device can receive the data only after the slave device is ready. The waiting time takes the SCLK cycle as a single bit
10	-	Res	Reserved
9:8	INSTL	RW	Instruction length 00: 0 bit,It means that there is no need to send instructions 01: 4 Bit instruction 10: 8 Bit instruction 11: 16 Bit instruction
7:6	-	Res	Reserved
5:2	ADDRL	RW	Address length 0000: 0 bit,This means that the Transfer address is not required 0001: 4 Bit address 0010: 8 Bit address 0011: 12 Bit address 0100: 16 Bit address

			0101: 20 Bit address 0110: 24 Bit address 0111: 28 Bit address 1000: 32 Bit address 1001: 36 Bit address 1010: 40 Bit address 1011: 44 Bit address 1100: 48 Bit address 1101: 52 Bit address 1110: 56 Bit address 1111: 60 Bit address
1:0	TRANST	RW	Control frame transmission format 00: Instructions and addresses are delivered in standard SPI format (single line SPI) 01: the instruction is sent in standard SPI format (one line SPI) and the address is sent in QSPI (four line SPI) 10: Instructions and addresses are sent in QSPI (4-wire SPI) 11: Setting to 11 is not allowed

25. Inter-IC Sound Bus (I2S)

25.1. I2S Introduction

MG32F10xx has an I2S bus interface which works in the Master mode, support a variety of audio transmission protocols and dual channel input/output.It provides four serial buses: master clock (MCLK), serial clock (SCLK), frame clock (WS) and serial data (SD0/SD1).

25.2. I2S Main Features

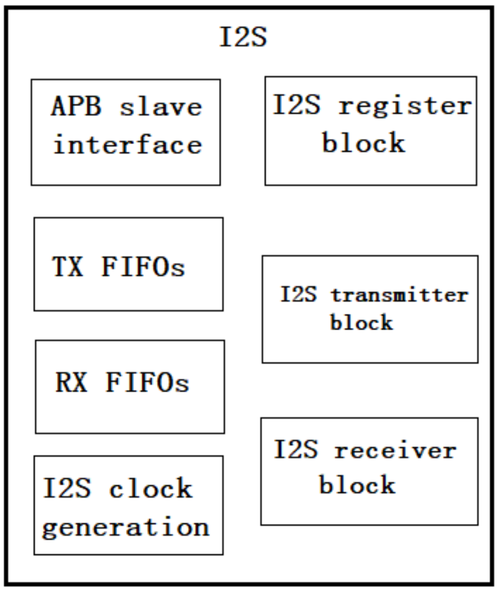
- Conforms to the Philips I2S serial protocol
- In addition to the 3-line serial bus SDI, SCLK and WS, we also provide the master clock MCLK, which enables better synchronization between systems.
- Dual channel input/output
- Full duplex communication due to the independence of transmitter and receiver
- Master mode of operation, provide sclk gating and enable signals
- Audio data resolutions of 12, 16, 20, 24, and 32 bits
- The Rx FIFO and Tx FIFO depth is 4 words, and the word size is 32 bit
- Configurable support for programmable DMA registers
- Programmable FIFO thresholds

25.3. I2S Functional Description

25.3.1. Block Diagram

Figure 25-1 shows I2S block diagram.

Figure 25-1. I2S Block Diagram



25.3.2. I2S transport protocols

The I2S consists of a serial data line (SDI), a word select line (WS), and a serial clock (SCLK) and we provide the master clock MCLK, which enables better synchronization between systems.

The serial data is transmitted in two’s complement format with the most significant bit (MSB) first.

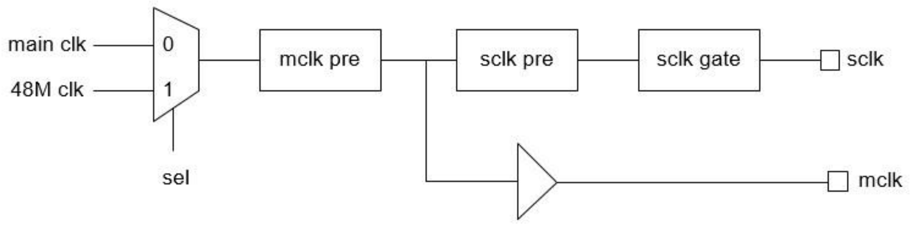
The I2S always transmits left stereo data first, then right channel data. When WS is low, the word being transferred is left stereo data; when WS is high, the word being transferred is right stereo data. The length of the WS can be configured with I2S_CCR.WSS, which supports three configurations: 16, 24 and 32 SCLK cycles.

When the data resolution is smaller than the WS length, I2S_CCR.SCLKG can be configured to turn off the SCLK corresponding to the unused bits.

25.3.3. I2S clock configuration

Figure 25-2 shows I2S clock configuration

Figure 25-2. I2S Time Clock configuration Description



The MCLK clock source can choose from Main CLK or 48MHz clock, configured through the MCLKSRC of the RCC modules. The MCLK prescaler can be configured via the MCLKPRE of the RCC module. The prescaler of SCLK can be configured via the I2SPRE of the RCC module. Please refer to the relevant section of the RCC module for details.

When the clock generator is turned off (I2S_CER[0]=0), SCLK and WS signals are stopped. You can configure SCLK and WS signals through register I2S_CCR when the clock generator is off.

25.3.4. I2S Enable

You need to set the I2S Enable (IEN) bit of the I2S Enable Register (I2S_IER) before using the I2S. To disable I2S, set I2S_IER[0] to 0.

After disable the I2S, the following events occur:

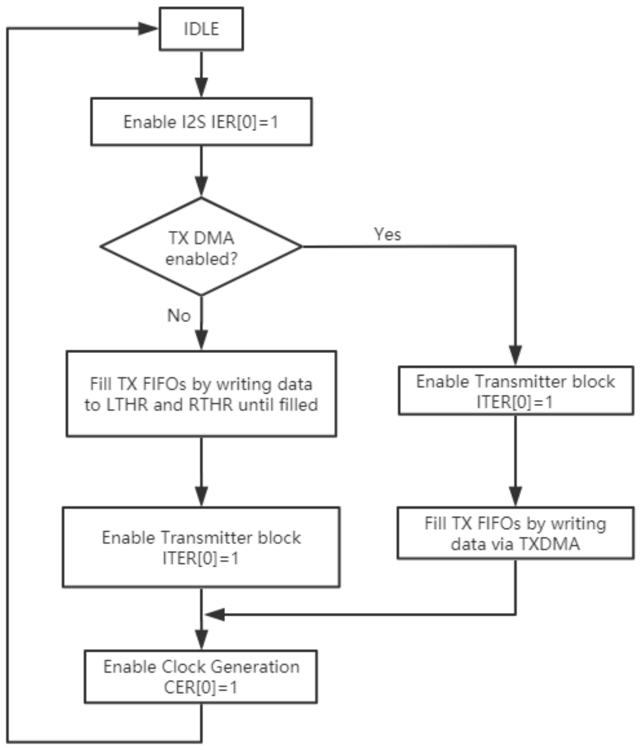
- TX and RX FIFOs are cleared;
- Any data in the process of being transmitted or received is lost;
- The enable bit in I2S_IRER/I2S_ITER/I2S_RERx/I2S_TERx is invalid;
- The SCLK output is turned off.

25.3.5. Transmit Mode

The I2S interface has two transmission channels which work independently.

Figure 25-3 shows the basic usage flow of the transmit mode.

Figure 25-3. The basic usage flow of the transmit mode


IDLE
Enable I2S IER[0]=1
TX DMA enabled?
Yes
No
Fill TX FIFOs by writing data to LTHR and RTHR until filled
Enable Transmitter block ITER[0]=1
Enable Transmitter block ITER[0]=1
Fill TX FIFOs by writing data via TXDMA
Enable Clock Generation CER[0]=1

The reason why the clock generator is finally turned on is that the transmission of the series starts from the WS change, and the I2S needs to be configured before this.

Transmit data

Each transmit channel has two FIFOs, storing left stereo data and right stereo data respectively. In non-DMA mode, You need to write left stereo data corresponding to stereo information to I2S_LTHRx (x=0,1) and right stereo data to I2S_RTHRx (x=0,1). In DMA mode, The I2S_TXDMA register allows access to all enabled transmit channels via a single point rather than through the LTHRx and RTHRx registers. The transmit channels are targeted in a cyclical fashion (starting at the lowest numbered enabled channel) and takes two writes (left and right stereo data) before the component points to the next channel.

For example, channel 0 and channel 1 are enabled, and the order of writing the transmit channels data is:

- Ch0 - left stereo data
- Ch0 - right stereo data
- Ch1 - left stereo data
- Ch1 - right stereo data
- Ch0 - left stereo data
- Ch0 - right stereo data

......

If you want to change the order during writing I2S_TXDMA and start again from the lowest channel, you can write the I2S_RTXDMA register. Note: This register has no effect in the middle of a stereo pair write (such as, when left stereo data has been written but not right stereo data). For example, channel 0 and channel 1 are enabled, and the order of writing the transmit data is:

- Ch0 - left stereo data
- Write 1 to I2S_RTXDMA (this write operation is invalid)
- Ch0 - right stereo data
- Ch1 - left stereo data
- Ch1 - right stereo data
- Ch0 - left stereo data
- Ch0 - right stereo data
- Write 1 to I2S_RTXDMA (this write operation is valid)
- Ch0 - left stereo data
- Ch0 - right stereo data

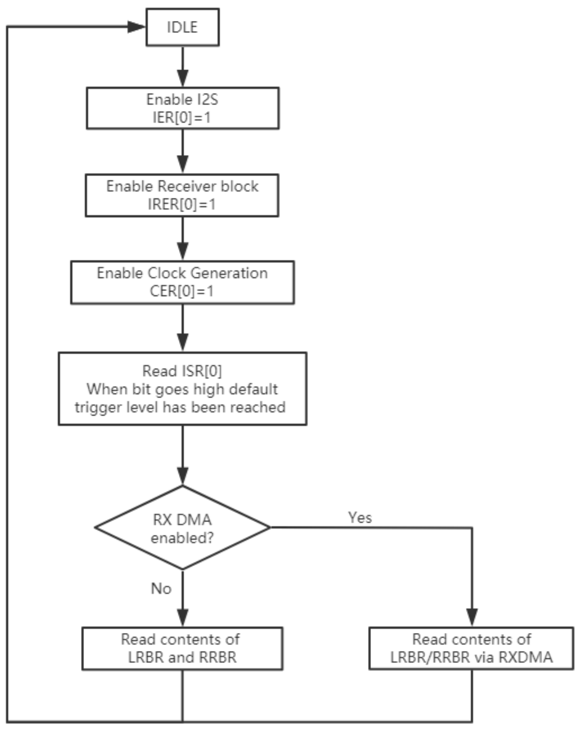
......

25.3.6. Receiver mode

The I2S supports two receive channels and can work independently.

Figure 25-4 shows how the I2S would be used as a receiver:

Figure 25-4. Basic use flow of receiver mode


IDLE
Enable I2S
IER[0]=1
Enable Receiver block
IRER[0]=1
Enable Clock Generation
CER[0]=1
Read ISR[0]
When bit goes high default trigger level has been reached
RX DMA enabled?
Yes
No
Read contents of LRBR and RRBR
Read contents of LRBR/RRBR via RXDMA

The reason why the clock generator is finally turned on is that the transmission of the series starts from the WS change, and the I2S needs to be configured before this.

Each receiving channel has two FIFOs that store left stereo data and right stereo data, respectively. In non-DMA mode, the user reads left stereo data from I2S_LRRBRX (x=0,1) stereo information pair and right stereo data from I2S_RRBRX (x=0,1). In DMA mode, The RXDMA register allows access to all enabled Receive channels via a single point rather than through the LRBRx and RRBRx registers. The Receive channels are targeted in a cyclical fashion (starting at the lowest numbered enabled channel) and takes two reads (left and right stereo data) before the component points to the next channel.

For example, channel 0 and channel 1 are enabled at the same time, and the order in which the data is read is:

- Ch0 - left stereo data
- Ch0 - right stereo data
- Ch1 - left stereo data
- Ch1 - right stereo data
- Ch0 - left stereo data
- Ch0 - right stereo data

......

The I2S_RRXDMA register can be written if you want to change the order during an RXDMA read and start again from the lowest channel. Note: Writing to this register has no effect if the component is performing a stereo pair read (such as, when left stereo data has been read but not right stereo data).

For example, channel 0 and channel 1 are enabled at the same time, and the order in which the data is read is:

Ch0 - left stereo data

I2S_ RRXDMA Reset - No effect (read not complete)

- Ch0 - right stereo data
- Ch1 - left stereo data
- Ch1 -right stereo data
- Ch0 - left stereo data
- Ch0 - right stereo data
- I2S_RRXDMA reset (complete valid)
- Ch0 - left stereo data
- Ch0 - right stereo data

......

25.3.7. Interrupt

The receive channel has two interrupts: RX FIFO data valid interrupt and data overflow interrupt.

The transmit channel has two interrupts: TX FIFO air break and data overflow interrupt.Each interrupt vector occupies a signal bit, Generates a uniform interrupt request signal INTR to CPU.The following table details all interrupt vectors:

Table 25-1. IC interrupt vectors

Channel	interrupt vectors	interrupt mask	Interrupt status	Interrupt generation	interrupt clear
RX CH0	Rx FIFO Data Available interrupt	IMR0[0]	ISR0[0]	The data in the Rx FIFO is greater than or equal to Trigger Level (You can configure Trigger Level with RFCR0.).	Reads data out of the RX FIFO until the number of data in the RX FIFO is less than the Trigger Level.
	Rx FIFO Overrun interrupt	IMR0[1]	ISR0[1]	This interrupt is asserted when an attempt is made to write received data to full RX FIFO (any data being written is lost while data in the FIFO is preserved).	To read the ROR0[0] bit
RX CH1	Rx FIFO Data Available interrupt	IMR1[0]	ISR1[0]	The number of data in the Rx FIFO is greater than or equal to Trigger Level (You can configure Trigger Level with RFCR1.)	Reads data out of the RX FIFO until the number of data in the RX FIFO is less than the configuration thresholds in the Trigger Level.
	Rx FIFO Overrun interrupt	IMR1[1]	ISR1[1]	This interrupt is asserted when an attempt is made to write received data to full RX FIFO (any data being	To read the ROR1[0] bit

Channel	interrupt vectors	interrupt mask	Interrupt status	Interrupt generation	interrupt clear
				written is lost while data in the FIFO is preserved).	
TX CH0	Tx FIFO Empty interrupt	IMR0[4]	ISR0[4]	The number of data in the Tx FIFO is greater than or equal to Trigger Level (You can configure Trigger Level with TFCR0.)	To write data in the Tx FIFO, until all the number of the data in Tx FIFO is greater than the Trigger Level.
	Tx FIFO Overrun interrupt	IMR0[5]	ISR0[5]	This interrupt is asserted when an attempt is made to write received data to full TX FIFO (any data being written is lost while data in the FIFO is preserved).	To read the TOR0[0] bit
TX CH1	Tx FIFO Empty interrupt	IMR1[4]	ISR1[4]	The number of data in the Tx FIFO is greater than or equal to (You can configure Trigger Level with TFCR0.).	To write data in the Tx FIFO, until all the number of the data in Tx FIFO is greater than the Trigger Level.
	Tx FIFO Overrun interrupt	IMR1[5]	ISR1[5]	This interrupt is asserted when an attempt is made to write to a full TX FIFO (any data being written is lost while data in the FIFO is preserved).	To read the TOR1[0]

25.4. Register Description

25.4.1. I2S Enable Register (I2S_IER)

Address offset: 0x00

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	IEN	RW	Set I2S enable or disable. 0:Disable 1:Enable

25.4.2. I2S Receiver Block Enable Register (I2S_IRER)

Address offset: 0x04

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RXEN	RW	This bit set the receiver enable or disable. 0:Disable 1:Enable

25.4.3. I2S Transmitter Block Enable Register (I2S_ITER)

Address offset: 0x08

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	TXEN	RW	This bit set the transmitter enable or disable. 0:Disable 1:Enable

25.4.4. Clock Enable Register (I2S_CER)

Address offset: 0x0C

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	CLKEN	RW	This bit set the clock enable or disable. 0:Disable 1:Enable

25.4.5. Clock Configuration Register (I2S_CCR)

Address offset: 0x10

Reset value: 0x02

The I2S Clock Generation block must be disabled (I2S_CER[0] = 0) before making any changes in this register.

Bits	Fields	R/W	Description
31:5	-	R	Reserved.
4:3	WSS	RW	These bits are used to program the number of sclk cycles for which the word select line (WS) stays in the left or right sample mode. 0x0: 16 sclk cycles 0x1: 24 sclk cycles 0x2: 32 sclk cycles
2:0	SCLKG	RW	These bits are used to program the gating of sclk. 0x0: Clock gating is disabled 0x1: Gating after 12 sclk cycles 0x2: Gating after 16 sclk cycles 0x3: Gating after 20 sclk cycles 0x4: Gating after 24 sclk cycles

25.4.6. Receiver Block FIFO Reset Register (I2S_RXFFR)

Address offset: 0x14

Reset value: 0x00

The I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RXFFR	W	Writing a 1 to this register flushes all the RX FIFOs (this is a self clearing bit).

25.4.7. Transmitter Block FIFO Reset Register (I2S_TXFFR)

Address offset: 0x18

Reset value: 0x00

The I2S_ITER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	TXFFR	W	Writing a 1 to this register flushes all the TX FIFOs (this is a self clearing bit).

25.4.8. Left Receive Buffer Register x (I2S_LRBRx x=0,1)

Address offset: 0x20, 0x60

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	LRBR	R	The left stereo data received serially from the receive channel input (sdix).

25.4.9. Left Transmit Holding Register x (I2S_LTHRx x=0,1)

Address offset: 0x20, 0x60

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	LTHR	W	The left stereo data to be transmitted serially through the transmit channel output (sdox) is written through this register.

25.4.10. Right Receive Buffer Register x (I2S_RRBRx x=0,1)

Address offset: 0x24, 0x64

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	RRBR	W	The right stereo data received serially from the receive channel input (sdix) is read through this register.

25.4.11. Right Transmit Holding Register x (I2S_RTHRx x=0,1)

Address offset: 0x24, 0x64

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	RTHR	W	The right stereo data to be transmitted serially through the transmit channel output (sdox) is written through this register.

25.4.12. Receive Enable Register x (I2S_RERx x=0,1)

Address offset: 0x28, 0x68

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RXCHEN	W	This bit set receive channel enable. This bit is valid when both I2S_IER[0] and I2S_IRER[0] are 1. 0: Disable 1: Enable

25.4.13. Transmit Enable Register x (I2S_TERx x=0,1)

Address offset: 0x2C,0x6C

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	TXCHEN	W	This bit set Transmit channel enable. This bit is valid when both I2S_IER[0] and I2S_ITER[0] are 1

25.4.14. Receive Configuration Register x (I2S_RCRx x=0,1)

Address offset: 0x30, 0x70

Reset value: 0x00

The I2S_RERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:3	-	R	Reserved.
2:0	WLEN	RW	These bits are used to program the desired data resolution of the receiver and enables the LSB of the incoming left (or right) word to be placed in the LSB of the I2S_LRBRx (or I2S_RRBRx) register： 0x0: Ignore the word length 0x1: 12-bit data resolution of the receiver. 0x2: 16-bit data resolution of the receiver. 0x3: 20-bit data resolution of the receiver. 0x4: 24-bit data resolution of the receiver. 0x5: 32-bit data resolution of the receiver.

25.4.15. Channel Configuration Register For TX (I2S_TCRx x=0,1)

Address offset: 0x34, 0x74

Reset value: 0x00

The I2S_TERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:3	-	R	Reserved.
2:0	WLEN	RW	These bits are used to program the data resolution of the transmitter and ensures the MSB of the data is transmitted first. 0x0: Ignore the word length 0x1: 12-bit data resolution of the receiver. 0x2: 16-bit data resolution of the receiver. 0x3: 20-bit data resolution of the receiver. 0x4: 24-bit data resolution of the receiver. 0x5: 32-bit data resolution of the receiver.

25.4.16. Channel Interrupt Status Register x (I2S_ISRx x=0,1)

Address offset: 0x34, 0x74

Reset value: 0x00

Bits	Fields	R/W	Description
31:6	-	R	Reserved.
5	TXFO	R	Status of Data Overrun interrupt for the TX channel. 0x0: TX FIFO write valid 0x1: TX FIFO write overflow
4	TXFE	R	Status of Transmit Empty Trigger interrupt.This bit specifies whether the TX FIFO trigger level has reached or not. TX FIFO is empty. 0x0: TX FIFO trigger level is reached 0x1: TX FIFO trigger level is not reached
3:2	-	R	Reserved.
1	RXFO	R	Status of Data Overrun interrupt for the RX channel. Incoming data lost due to a full RX FIFO. 0x0: RX FIFO write valid 0x1: RX FIFO write overrun
0	RXDA	R	Status of Receive Data Available interrupt. This bit denotes the status of the RX FIFO trigger level. 0x0: RX FIFO trigger level is reached. 0x1: RX FIFO trigger level is not reached.

25.4.17. Channel Interrupt Mask Register x (I2S_IMRx x=0,1)

Address offset: 0x3C, 0x7C

Reset value: 0x00

Bits	Fields	R/W	Description

Bits	Fields	R/W	Description
31:6	-	R	Reserved.
5	TXFOM	RW	Mask TX FIFO Overrun interrupt. This bit masks or unmasks a TX FIFO overrun interrupt. 1: Mask. 0: Unmask.
4	TXFEM	RW	Mask TX FIFO Empty interrupt. This bit masks or unmasks a TX FIFO Empty interrupt. 1: Mask 0: Unmask
3:2	-	R	Reserved.
1	RXFOM	RW	Mask RX FIFO Overrun interrupt. This bit masks or unmasks an RX FIFO Overrun interrupt 1: Mask. 0: Unmask.
0	RXDAM	RW	Mask RX FIFO Data Available interrupt. This bit masks or unmasks an RX FIFO Data Available interrupt. 1: Mask 0: Unmask

25.4.18. Receive Overrun Register x (I2S_RORx x=0,1)

Address offset: 0x40, 0x80

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RXCHO	R	Read this bit to clear the RX FIFO Data Overrun interrupt.

25.4.19. Transmit Overrun Register x (I2S_TORx x=0,1)

Address offset: 0x44, 0x84

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	TXCHO	R	Read this bit to clear the TX FIFO Data Overrun interrupt.

25.4.20. Receive FIFO Configuration Register x (I2S_RFCRx x=0,1)

Address offset: 0x48, 0x88

Reset value: 0x00

The I2S_RERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:4	-	R	Reserved.

Bits	Fields	R/W	Description
3:0	RXCHDT	RW	These bits program the trigger level in the RX FIFO at which the Received Data Available interrupt is generated. Trigger Level = Programmed Value + 1 Programmed Value: 0x00~0x03

25.4.21. Transmit FIFO Configuration Register x (I2S_TFCRx x=0,1)

Address offset: 0x4C,0x8C

Reset value: 0x00

The I2S_TERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:4	-	R	Reserved.
3:0	TXCHET	RW	These bits program the trigger level in the TX FIFO at which the Empty Threshold Reached Interrupt is generated. Trigger Level = TXCHET TXCHET Value:0x00~0x03

25.4.22. Receive FIFO Flush Register x (I2S_RFFx x=0,1)

Address offset: 0x50, 0x90

Reset value: 0x00

The I2S_RERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:1 0	-	R	Reserved.
	RXCHFR	W	Receive Channel FIFO Reset. Writing a 1 to this register flushes an individual RX FIFO (This is a self clearing bit.).

25.4.23. Transmit FIFO Flush Register x (I2S_TFFx x=0,1)

Address offset: 0x54, 0x94

Reset value: 0x00

The I2S_TERx[0] and the I2S_IRER[0] must be 0 before making any changes in this register.

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	TXCHFR	W	Transmit Channel FIFO Reset. Writing a 1 to this register flushes channel's TX FIFO (This is a self clearing bit.).

25.4.24. Receiver Block DMA Register (I2S_RXDMA)

Address offset: 0x1C0

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	RXDMA	R	Receiver Block DMA Register.These bits are used to cycle repeatedly through the enabled receive channels (from lowest numbered to highest), reading stereo data pairs.

25.4.25. Reset RX DMA Register (I2S_RRXDMA)

Address offset: 0x1C4

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RRXDMA	W	Reset Receiver Block DMA Register. Writing a 1 to this self-clearing register resets the RXDMA register mid-cycle to point to the lowest enabled Receive channel. Note: Writing to this register has no effect if the component is performing a stereo pair read (such as, when left stereo data has been read but not right stereo data).

25.4.26. Transmitter Block DMA Register (I2S_TXDMA)

Address offset: 0x1C8

Reset value: 0x00

Bits	Fields	R/W	Description
31:0	TXDMA	R	Transmitter Block DMA Register. The register bits can be used to cycle repeatedly through the enabled Transmit channels (from lowest numbered to highest) to allow writing of stereo data pairs.

25.4.27. Reset Transmitter Block DMA Register (I2S_RTXDMA)

Address offset: 0x1CC

Reset value: 0x00

Bits	Fields	R/W	Description
31:1	-	R	Reserved.
0	RRXDMA	W	Reset Transmitter Block DMA Register. Writing a 1 to this self-clearing register resets the TXDMA register mid-cycle to point to the lowest enabled Transmit channel. Note: This register has no effect in the middle of a stereo pair write (such as, when left stereo data has been written but not right stereo data).

26. Integrated Circuit Interface (I2C)

26.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard-mode, fast-mode and high-speed mode as well as SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

The device includes two I2C blocks: I2C1 and I2C2.

26.2. Characteristics

- Two-wire I2C serial interface – consists of a serial data line (SDA) and a serial clock line (SCL)
- Both master and slave functions with the same interface.
- Three speeds:
  - Standard mode (0 to 100 Kb/s)
  - Fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s)
  - High-speed mode (≤ 3.4 Mb/s)
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Handles Bit and Byte waiting at all bus speeds
- Interrupt or polled-mode operation
- Component parameters for configurable software driver support
- Programmable SDA hold time
- Support DMA mode
- Bus clear feature
- Device ID feature
- SMBus/PMBus Support
- SMBus Slave detects and responds to ARP commands.
- Support SMBUS Alarm and timeout detection

26.3. I2C implementation

MG32F103xx contains two I2C blocks, I2C1 and I2C2. The table below shows the supported features in detail.

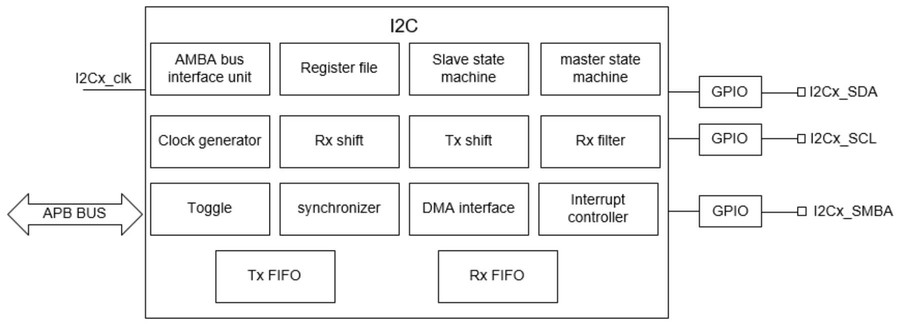
Table 26-1. I2C1 and I2C2 Features

I2C features	I2C1	I2C2
7 bit addressing	√	√
10 bit addressing	√	√
Standard mode	√	√
Fast mode/Fast Plus Mode	√	√
High speed mode	X	√
SMBUS	√	X

26.4. Block Diagram

The figure below shows the block diagram.

Figure 26-1. Block Diagram



26.5. Function Overview

26.5.1. Mode Selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, the device operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, it supports multi-master operation.

Start condition

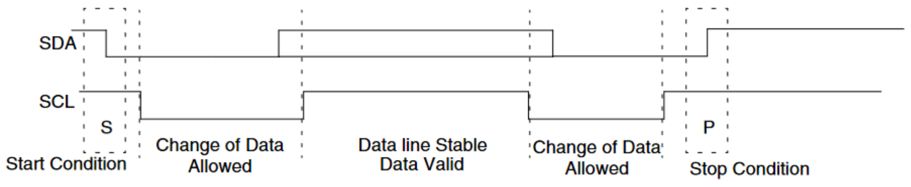
When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1.

Stop condition

When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1.

The figure below shows the start/stop condition in detail.

Figure 26-2. START and STOP Condition



Communication Flow

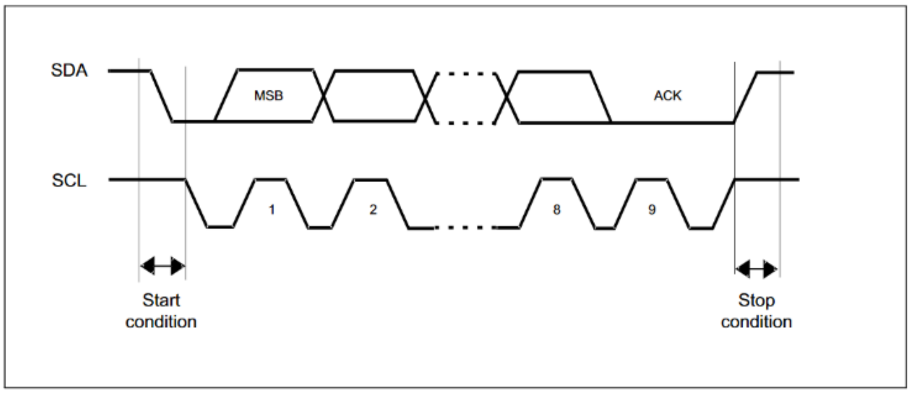
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The Reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure 26-3.

Figure 26-3. I2C Bus Protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

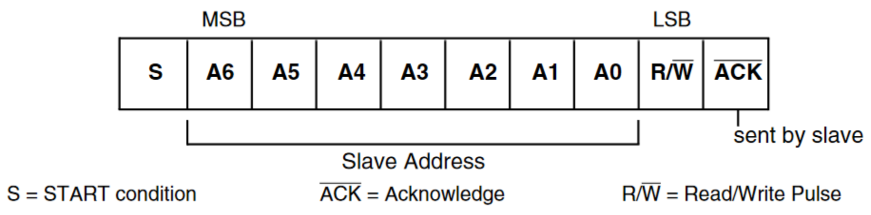
26.5.2. Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 26-4. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

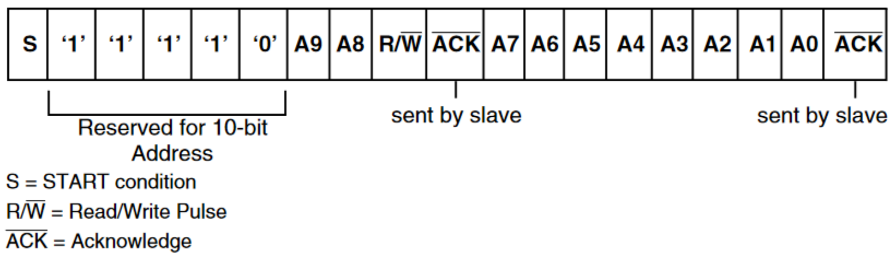
Figure 26-4. 7-bit Addressing Format



10-bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 26-5 shows the 10-bit address format.

Figure 26-5. 10-bit Address Format



The Table below defines the special purpose and Reserved first byte addresses.

Table 26-2. I2C/SMBus Definition of Bits in First Bye

Slave Address	R/W Bit	Description
0000 000	0	General Call Address. I2C places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte.
0000 001	X	CBUS address. I2C ignores these accesses.
0000 010	X	Reserved.
0000 011	X	Reserved
0000 1XX	X	High speed master mode.
1111 1XX	X	Reserved
1111 0XX	X	10-bit slave addressing

0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

I2C does not restrict you from using these Reserved addresses. However, if you use these Reserved addresses, you may run into incompatibilities with other I2C components.

26.5.3. Transmitting and Receiving Protocol

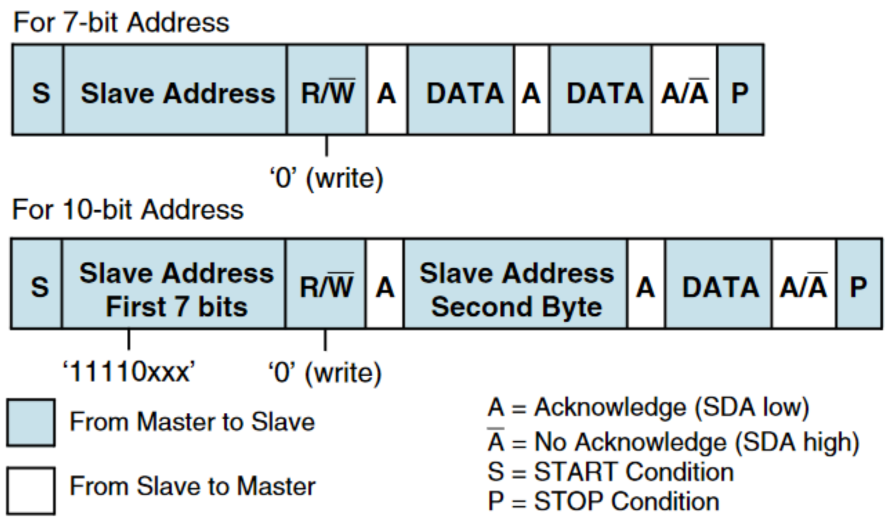
The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in Figure26-6, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

Figure 26-6. Master-Transmitter Protocol

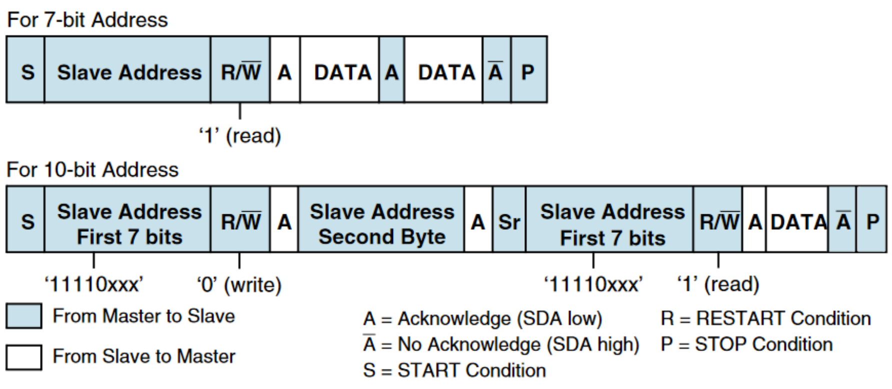


Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 26-7, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte.

The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition. When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse.

Figure 26-7. Master-Receiver Protocol

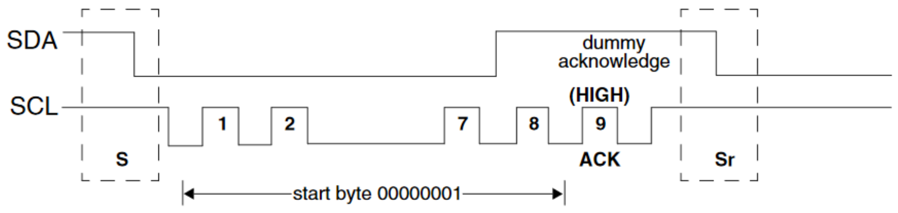


26.5.4. START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the I2C is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I2C is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it.

This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 26-8. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.

Figure 26-8. START BYTE Transfer



The START BYTE procedure is as follows:

- Master generates a START condition.
- Master transmits the START byte (0000 0001).
- Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
- No slave sets the ACK signal to 0.
- Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a Reserved address and reset after the RESTART condition is generated.

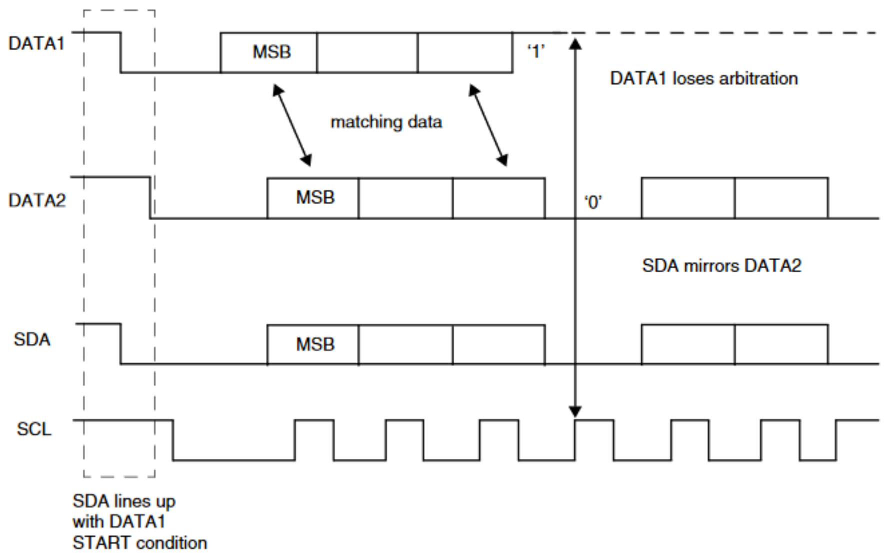
26.5.5. Multiple Master Arbitration

The I2C bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the I2C will stop generating SCL (ic_clk_oe). Figure 26-9 illustrates the timing of when two masters are arbitrating on the bus.

Figure 26-9. Multiple Master Arbitration



For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bitcode is defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, IC_HS_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition

Slaves are not involved in the arbitration process.

26.5.6. Slave Mode Operation

This section discusses slave mode procedures in detail.

Initial Configuration

To use the I2C as a slave mode, perform the following steps:

- Disable the I2C macro by writing a ‘0’ to bit 0 of the IC_ENABLE register.
- Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the I2C macro responds.
- Write to the IC_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I2C macro in slave-only mode by writing a ‘0’ into bit 6 (IC_SLAVE_DISABLE) and a ‘0’ to bit 0 (MASTER_MODE).
- Enable the I2C macro by writing a ‘1’ in bit 0 of the IC_ENABLE register.

Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses this I2C macro and requests data, the I2C macro acts as a slave-transmitter and the following steps occur:

- The other I2C master device initiates an I2C transfer with an address that matches the slave address in the IC_SAR register of this I2C macro.
- The I2C macro acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
- The I2C macro asserts the RD_REQ interrupt (bit 5 of the IC_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds.

  If the RD_REQ interrupt has been masked, due to IC_INTR_MASK[5] register (M_RD_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the IC_RAW_INTR_STAT register.

  - Reads that indicate IC_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
  - Software must then act to satisfy the I2C transfer.
  - The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C macro can handle. For example, for 400 kb/s, the timing interval is 25us.
- If there is any data remaining in the Tx FIFO before receiving the read request, then the I2C macro asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO.

  If the TX_ABRT interrupt has been masked, due to of IC_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the IC_RAW_INTR_STAT register.

  - Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
  - There is no further action required from software.
  - The timing interval used should be similar to that described in the previous step for the IC_RAW_INTR_STAT[5] register.

- Software writes to the IC_DATA_CMD register with the data to be written (by writing a ‘0’ in bit 8).
- Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding.
- If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1.
- The I2C macro releases the SCL and transmits the byte.
- The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Note: Whenever a TX_ABRT event occurs, it is necessary for software to read the IC_CLR_TX_ABRT register before attempting to write into the Tx FIFO. See register IC_RAW_INTR_STAT for more details.

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses this I2C macro and is sending data, the I2C macro acts as a slave-receiver and the following steps occur:

- The other I2C master device initiates an I2C transfer with an address that matches its slave address in the IC_SAR register.
- The I2C macro acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C macro is acting as a slave-receiver.
- The I2C macro receives the transmitted byte and places it in the receive buffer.
- The I2C macro asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register).

  If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine (described in “Slave-Transmitter Operation for a Single Byte”) be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
- Software may read the byte from the IC_DATA_CMD register (bits 7:0).
- The other master device may hold the I2C bus by issuing a RESTART condition, or release the bus by issuing a STOP condition.

Slave-Transfer Operation for Bulk Transfers

In the standard I2C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave’s TXFIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter’s TXFIFO. This I2C macro is designed to handle more data in the TXFIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TXFIFO.

This mode only occurs when the I2C macro is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TXFIFO, the I2C macro holds the SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the TXFIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IC_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section. This timing routine is similar to that described in "Slave-Transmitter Operation for a Single Byte".

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TxFIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte. Then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses this I2C macro and requests data, the TxFIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the I2C macro slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the TxFIFO. There is no need to hold the SCL line low or to issue RD_REQ again. If the remote master is to receive n bytes from the I2C macro but the programmer wrote a number of bytes larger than n to the TxFIFO, then when the slave finishes sending the requested n bytes, it clears the TxFIFO and ignores any excess bytes.

The I2C macro generates a transmit abort (TX_ABRT) event to indicate the clearing of the Tx FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the Tx FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the TxFIFO is cleared at that time.

26.5.7. Master Mode Operation

This section discusses master mode procedures in detail.

Initial Configuration

To use the I2C macro as a master, perform the following steps:

- Disable the macro by writing 0 to bit 0 of the IC_ENABLE register.
- Write to the IC_CON register to set the maximum speed mode supported (bits 2:1) and the desired speed of the I2C macro master-initiated transfers, either 7-bit or 10-bit addressing (bit 4). Ensure that bit 6 (IC_SLAVE_DISABLE) is written with a '1' and bit 0 (MASTER_MODE) is written with a '1'.
- Write to the IC_TAR register the address of the I2C device to be addressed (bits 9:0). This register also indicates whether a General Call or a START BYTE command is going to be performed by I2C.

- Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired master code for the I2C macro. The master code is programmer-defined.
- Enable the DW_apb_i2c by writing a 1 to bit 0 of the IC_ENABLE register.
- Now write transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the DW_apb_i2c is enabled, the data and commands are lost as the buffers are kept cleared when DW_apb_i2c is disabled.

This step generates the START condition and the address byte on the I2C macro. Once the I2C macro is enabled and there is data in the TX FIFO, the I2C macro starts reading the data.

Master Transmit and Master Receive

The I2C macro supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The CMD bit [8] should be written to 0 for write operations. Subsequently, a read command may be issued by writing “don’t cares” to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The I2C macro (as a master) continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty—depending on the value of IC_EMPTYFIFO_HOLD_MASTER_EN, the master either inserts a STOP condition after completing the current transfers, or it checks to see if IC_DATA_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO.

26.5.8. Disable I2C

To disable I2C, user can clear bit 0 of IC_ENABLE. Then the user should poll the register IC_ENABLE_STATUS to unambiguously determine when the hardware has completely shut down.

26.5.9. Aborting I2C Transfers

The ABORT control bit of the IC_ENABLE register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation.

To abort the I2C transfers, follow the steps below:

- Stop filling the Tx FIFO (IC_DATA_CMD) with new commands.
- When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
- Set bit 1 of the IC_ENABLE register (ABORT) to 1.
- Wait for the M_TX_ABRT interrupt.
- Read the IC_TX_ABRT_SOURCE register to identify the source as ABRT_USER_ABRT.

26.5.10. Spike Suppression

The I2C macro contains programmable spike suppression logic, and the logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of ic_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic_clk cycles can be programmed by the user and should be calculated taking into account the frequency of ic_clk and the relevant spike length specification.

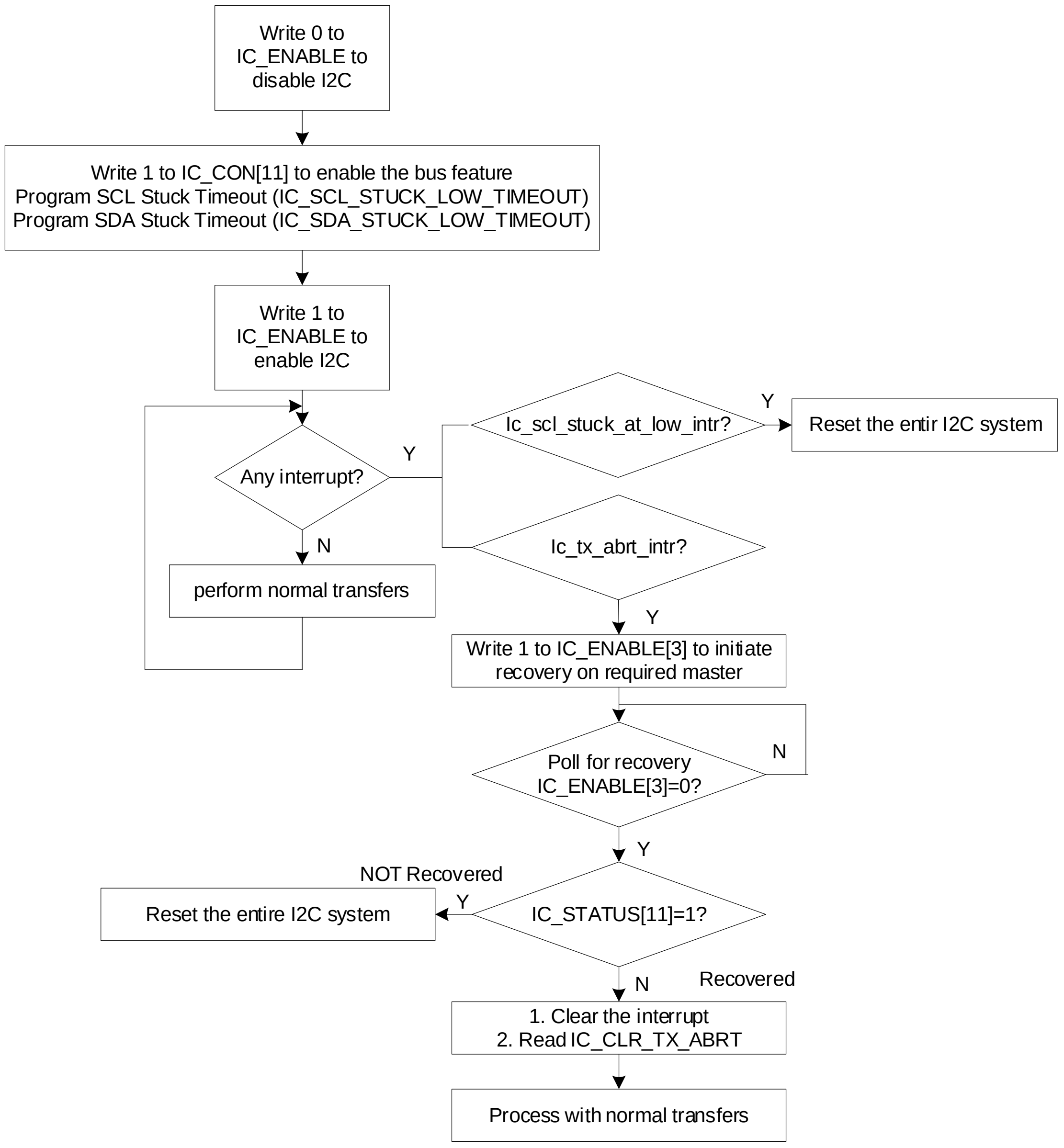
Based on I2C bus protocol, the maximum spike length for SS/FS mode is 50ns, and is 10ns for HS.

26.6. I2C Bus Clear

The I2C macro supports the bus clear feature that provides graceful recovery of data (SDA) and clock (SCL) lines during unlikely events in which either the clock or data line is stuck at LOW.

The figure 26-10 below shows the steps in detail:

Figure 26-10. Bus Clear Flow


Write 0 to IC_ENABLE to disable I2C
Write 1 to IC_CON[11] to enable the bus feature
Program SCL Stuck Timeout (IC_SCL_STUCK_LOW_TIMEOUT)
Program SDA Stuck Timeout (IC_SDA_STUCK_LOW_TIMEOUT)
Write 1 to IC_ENABLE to enable I2C
Any interrupt?
Y
N
perform normal transfers
Ic_scl_stuck_at_low_intr?
Y
Reset the entir I2C system
Ic_tx_abrt_intr?
Y
Write 1 to IC_ENABLE[3] to initiate recovery on required master
Poll for recovery IC_ENABLE[3]=0?
N
Y
NOT Recovered
Reset the entire I2C system
Y
IC_STATUS[11]=1?
N
Recovered
1. Clear the interrupt
2. Read IC_CLR_TX_ABRT
Process with normal transfers

26.7. Device ID

A Device ID field is an optional 3-byte read-only (24 bits) word, which provides the following information:

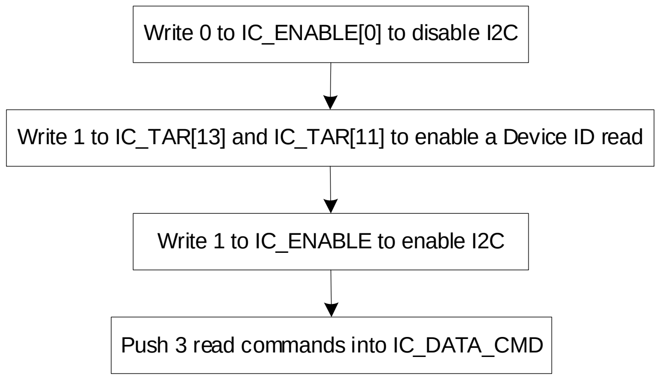
- Twelve bits with the manufacturer’s name, which is unique for every manufacturer.
- Nine bits with the part identification, which is assigned by the manufacturer.
- Three bits with the die revision, which is assigned by the manufacturer.

For reading the Device ID of a particular slave, the master can follow the procedure in figure 26-11. The Device ID that is read will be available in RX FIFO, which can be read using IC_DATA_CMD register.

In case of a slave, the user can read the Device ID of the slave using IC_DEVICE_ID register.

Device ID is not supported for 10-bit addressing and High Speed transfers (HS mode).

Figure 26-11. Reading Device ID


Write 0 to IC_ENABLE[0] to disable I2C
Write 1 to IC_TAR[13] and IC_TAR[11] to enable a Device ID read
Write 1 to IC_ENABLE to enable I2C
Push 3 read commands into IC_DATA_CMD

26.8. SMBUS Protocol

A typical SMBus device will have a set of commands by which data can be read and written. All commands are one byte long while their arguments and return values can vary in length. In accordance with the SMBus specification, the most significant bit (MSB) is transferred first. There are eleven possible command protocols for any given device. These commands are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call.

SMBus protocols for message transactions are generally different from I2C data transfer commands. It is still possible to program a SMBus master to deliver I2C data transfer commands. The following table describes the derivation of SMBus Bus Protocols through Tx-FIFO commands in this macro.

In the SMBus Master mode, all the receive data bytes will be available in Rx-FIFO. In the SMBus Slave mode, all the bus protocol command codes and data bytes will received in the Rx-FIF0 and read request data bytes must be sent using Tx-FIFO, similar to the I2C mode.

The macro slave can be enabled to receive only Quick command through enabling the SLAVE_QUICK_CMD_EN bit in the IC_CON Register. Whenever this bit is selected the slave only receives quick commands and will not accept other Bus Protocols. The macro slave issues the SMBUS_QUICK_DET interrupt upon receiving the QUICK command.

SMBus introduces a Packet Error checking Mechanism through appending PEC Byte at the end of the Bus Protocol. This can be achieved through adding an extra command (PEC byte) while transferring and decoding it while receiving by the software.

26.9. SMBUS Address Resolution Protocol

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device by the Host. This feature allows the devices to be 'hot-plugged' in to the system.

SMBus introduces a 128-bit Unique Device ID (UDID) for each device in the system to isolate each device for the purpose of address assignment. This macro uses the IC_SMBUS_UDID_MSB parameter for upper constant 96 bits and 'IC_SMBUS_ARP_UDID_LSB' register for lower variable 32 bits of the UDID.

This macro uses the PERSISTANT_SLV_ADDR_EN register bit in IC_CON register to indicate whether it supports persistent slave address.

This master can issue general and directed Address Resolution Protocol (ARP) commands to assign the dynamic address for the slaves in the SMBus system.

Note:

The macro slave hardware handles the generation, detection, and NACKing of the wrong PEC (CRC8 C(X)=X8+X2+X1+1) for the ARP Commands, but don't handle the PEC for Non-ARP commands.

The macro master hardware does not handle PEC for both APR and non-ARP commands

26.10. SMBus Alert

This macro includes the SMBus alter signal (SMBALERT#) specified by the SMBus standard. It can be used by simple devices to request the attention of the host.

In slave mode, user can set SMBUS_ALERT_CTRL bit (IC_ENABLE[18]) to enable it. Once asserted by user, the user waits for alert response address to be sent by master. Upon receiving it, contents of IC_SAR[7:0] register are sent to the master. When successful, this macro clears the SMBUS_ALERT_CTRL bit to stop sending the alert.

If working as host, user needs to service the alert detect interrupt by sending read byte command with Alert Response Address. Current alarm status can be read from SMBUS_ALERT_STATUS bit (IC_STATUS[20]).

26.11. SDA Hold Time

It is defined in I2C protocol that SDA needs to meet a certain holding time. The user can configure the IC_ SDA_HOLD register to dynamically adjust the hold time of SDA. When the transmission speed of I2C bus changes, it is necessary to adjust the SDA dynamic holding time at the same time.

IC_SDA_TX_HOLD uses IC_ SDA_ Hold [15:0] to adjust the holding time of SDA in transmission mode. However, in the transmission mode, there is a minimum SDA holding time. As the main transmitter, the minimum SDA holding time is 1 ic_clk. As a slave transmitter, the minimum SDA holding time is SPKLEN + 7 ic_clk period (SPKLEN refers to IC_FS_SPKLEN or IC_ HS_ SPKLEN)。

26.12. IC_CLK Frequency Configuration

When this macro is configured as a Standard (SS), Fast (FS)/Fast-Mode Plus (FM+), or High Speed (HS) master, the *CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The *CNT registers are:

- IC_SS_SCL_HCNT
- IC_SS_SCL_LCNT
- IC_FS_SCL_HCNT
- IC_FS_SCL_LCNT

- IC_HS_SCL_HCNT
- IC_HS_SCL_LCNT

When configure this registers, there are some requirements:

- IC_SS_SCL_LCNT and IC_FS_SCL_LCNT register values must be larger than IC_FS_SPKLEN + 7.
- IC_SS_SCL_HCNT and IC_FS_SCL_HCNT register values must be larger than IC_FS_SPKLEN + 5.
- If the component is programmed to support HS, IC_HS_SCL_LCNT register value must be larger than IC_HS_SPKLEN + 7.
- If the component is programmed to support HS, IC_HS_SCL_HCNT register value must be larger than IC_HS_SPKLEN + 5

The table below show the configurations in different working mode.

mode	ic_clk freq(MHz)	smallest value of IC_*SPK_LEN	SCL low time	SCL low configure	SCL high time	SCL high configure
SS	2.7	1	4.7 us	12	5.2 us	6
FS	12	1	1.33 us	15	1.16 us	6
HS(400pf)	51	1	333ns	16	274 ns	6
HS(100pf)	105.4	1	161ns	16	132 ns	6

26.13. DMA interfaces

This macro supports DMA transmission and can send DMA application to DMAC. The user can write the IC_ DMA_ CR to turn on the send DMA and receive DMA functions. When the data in the transmit buffer is less than or equal to the value defined in IC_ DMA_TDLR, a transmit DMA requests will be generated. When the number of data in RXFIFO is larger than or equal to the number specified in the IC_DMA_RDLR, a receive DMA request is generated.

Before DMA transmission, users need to configure DMAC. Please refer to DMAC related chapters for details.

26.14. Interrupts

For the interrupt types in I2C mode and SMBus mode, please refer to the corresponding register (I2C_RAW_INTR_STAT and SMBUS_RAW_INTR_STAT) for detail info.

26.15. I2C Registers

26.15.1. I2C_CON

Name: I2C Control Register.

Address Offset: 0x0

Default Value: 0x0000 0024(I2C1) 0x0000 003E(I2C2)

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:20	-	R	Reserved
19	SMBUS_PERSISTENT_SLV_ADDR_EN	RW	This bit is applicable only in Slave mode Values: 0x1 (ENABLED): SMBus Persistent Slave address control is enabled. 0x0 (DISABLED): SMBus Persistent Slave address control is disabled.
18	SMBUS_ARP_EN	RW	This bit is applicable only in Slave mode. Values: 0x1 (ENABLED): SMBus ARP control is enabled. 0x0 (DISABLED): SMBus ARP control is disabled.
17	SMBUS_SLAVE_QUICK_EN	RW	This bit is applicable only in slave mode. Values: 0x1 (ENABLED): SMBus Slave is enabled to receive Quick command. 0x0 (DISABLED): SMBus Slave is disabled to receive Quick command.
16	OPTIONAL_SAR_CTRL		Enables the usage of IC_OPTIONAL_SAR register. Values: 0x1 (ENABLED): Optional SAR Address Register is enabled. 0x0 (DISABLED): Optional SAR Address Register is disabled.
15:12	-	R	Reserved
11	BUS_CLEAR_FEATURE_CTRL	RW	In Master mode: 1'b1: Bus Clear Feature is enabled. 1'b0: Bus Clear Feature is Disabled. In Slave mode, this register bit is not applicable
10	STOP_DET_IF_MASTER_ACTIVE	RW	In Master mode: 1'b1: issues the STOP_DET interrupt only when master is active. 1'b0: issues the STOP_DET irrespective of whether master is active or not. In Slave mode, this register bit is not applicable
9	RX_FIFO_FULL_HLD_CTRL	RW	Values: 0x1 (ENABLED): Hold bus when RX_FIFO is full 0x0 (DISABLED): Overflow when RX_FIFO is full
8	TX_EMPTY_CTRL	RW	Values: 0x1 (ENABLED): Controlled generation of TX_EMPTY interrupt 0x0 (DISABLED): Default behavior of TX_EMPTY interrupt
7	STOP_DET_IFADDRESSED	RW	In slave mode: 1'b1: issues the STOP_DET interrupt only when it is addressed. 0'b0: issues the STOP_DET irrespective of whether it's addressed or not.
6	IC_SLAVE_DISABLE	RW	This bit controls whether I2C has its slave disabled. Values: 0x1 (SLAVE_DISABLED): Slave mode is disabled 0x0 (SLAVE_ENABLED): Slave mode is enabled Note: Software should ensure that if this bit is written with 0, then bit 0 should also be written with a 0.
5	IC_RESTART_EN	RW	Determines whether RESTART conditions may be sent when acting as a master. When RESTART is disabled, the master is

Bits	Fields	R/W	Description
			prohibited from performing the following functions: Sending a START BYTE Performing any high-speed mode operation High-speed mode operation Performing direction changes in combined format mode Performing a read operation with a 10-bit address Values: 0x1 (ENABLED): Master restart enabled 0x0 (DISABLED): Master restart disabled
4	IC_10BITADDR_MASTER	RW	This controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. Values: 0x1 (ADDR_10BITS): Master 10Bit addressing mode 0x0 (ADDR_7BITS): Master 7Bit addressing mode
3	IC_10BITADDR_SLAVE	RW	When acting as a slave, this bit controls whether the I2C macro responds to 7- or 10-bit addresses Values: 0x1 (ADDR_10BITS): Slave 10Bit addressing 0x0 (ADDR_7BITS): Slave 7Bit addressing
2:1	SPEED	RW	These bits control at which speed the I2C macro operates, and they are only be valid when operating in master mode. Values: 0x1 (STANDARD): Standard Speed mode of operation 0x2 (FAST): Fast or Fast Plus mode of operation 0x3 (HIGH): High Speed mode of operation Others: Reserved
0	MASTER_MODE	RW	This bit controls whether the I2C macro master is enabled. Values: 0x1 (ENABLED): Master mode is enabled 0x0 (DISABLED): Master mode is disabled

26.15.2. I2C_TAR

Name: I2C Target Address Register

Address Offset: 0x4

Default Value: 0x00000000

Description: It is only writable when IC_ENABLE[0]=0. It is not needed to configure this register if the macro works in slave mode.

Bits	Fields	R/W	Description
31:17	-	R	Reserved
16	SMBUS_QUICK_CMD	RW	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Quick command is to be performed. Values: 0x1 (ENABLED): Enable programming of QUICK-CMD transmission 0x0 (DISABLED): Disable programming of QUICK-CMD transmission

Bits	Fields	R/W	Description
15:14	-	R	Reserved
13	DEVICE_ID	RW	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Device-ID of a particular slave mentioned in IC_TAR[9:0] is to be performed. Values: 0x1 (ENABLED): Enables programming of DEVICE-ID transmission 0x0 (DISABLED): Disables programming of DEVICE-ID transmission
12	IC_10BITADDR_MASTER	RW	This bit controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. Values: 0x1 (ADDR_10BITS): Address 10Bit transmission format 0x0 (ADDR_7BITS): Address 7Bit transmission format
11	SPECIAL	RW	This bit indicates whether software performs a Device-ID or General Call or START BYTE command. Values: 0x1 (ENABLED): Enables programming of GENERAL_CALL or START_BYTE transmission 0x0 (DISABLED): Disables programming of GENERAL_CALL or START_BYTE transmission
10	GC_OR_START	RW	If bit 11 (SPECIAL) is set to 1 and bit 13(Device-ID) is set to 0, then this bit indicates whether a General Call or START byte command is to be performed. Values: 0x1 (START_BYTE): START byte transmission 0x0 (GENERAL_CALL): GENERAL_CALL byte
9:0	IC_TAR	RW	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits.

26.15.3. I2C_SAR

Name: Slave Address Register

Address Offset: 0x8

Default Value: 0x0000 03F0

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:10	-	R	Reserved
9:0	IC_SAR	RW	The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used.

26.15.4. I2C_HS_MADDR

Name: I2C High Speed Master Mode Code Address Register

Address Offset: 0xC

Default Value: 0x00000001

Description: It is only writable when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	IC_HS_MAR	RW	This bit field holds the value of the I2C HS mode master code. HS-mode master codes are Reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high-speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7.

26.15.5. I2C_DATA_CMD

Name: I2C Data and Command Register

Address Offset: 0x10

Default Value: 0x00000000

Description: It is only writable when IC_ENABLE[0]=0

Bits	Fields	R/W	Description
31:12	-	R	Reserved
11	FIRST_DATA_BYTE	R	Indicates the first data byte received after the address phase for receive transfer in Master receiver or Slave receiver mode. Values: 0x1 (ACTIVE): Non sequential data byte received 0x0 (INACTIVE): Sequential data byte received
10	RESTART	W	This bit controls whether a RESTART is issued before the byte is sent or received. Values: 0x1 (ENABLE): a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command 0x0 (DISABLE): a RESTART is issued only if the transfer direction is changing from the previous command
9	STOP	W	This bit controls whether a STOP is issued after the byte is sent or received. Values: 0x1 (ENABLE): STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0x0 (DISABLE): STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.

Bits	Fields	R/W	Description
8	CMD	RW	This bit controls whether a read or a write is performed. It controls only the direction when it acts as a master. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared. If a "1" is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs. Values: 0x1 (READ): Master Read Command 0x0 (WRITE): Master Write Command
7:0	DAT	RW	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the macro. However, when you read this register, these bits return the value of data received on the I2C interface.

26.15.6. I2C_SS_SCL_HCNT

Name: Standard Speed I2C Clock SCL High Count Register

Address Offset: 0x14

Default Value: 0x0000 0320

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_SS_SCL_HCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed, based on ic_clk period. The valid value is 6 ~ 65525.

26.15.7. I2C_SS_SCL_LCNT

Name: Standard Speed I2C Clock SCL Low Count Register

Address Offset: 0x18

Default Value: 0x0000 03AC

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_SS_SCL_LCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for standard speed, based on ic_clk period. The minimum value is 8.

26.15.8. I2C_FS_SCL_HCNT

Name: Full Speed I2C Clock SCL High Count Register.

Address Offset: 0x1C

Default Value: 0x0000 0078

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_FS_SCL_HCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed, based on ic_clk period. The minimum value is 6.

26.15.9. I2C_FS_SCL_LCNT

Name: Full Speed I2C Clock SCL Low Count Register

Address Offset: 0x20

Default Value: 0x0000 0104

Description: It is only writable when IC_ENABLE[0]=0.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_FS_SCL_LCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed, based on ic_clk period. The minimum value is 8.

26.15.10. I2C_HS_SCL_HCNT

Name: High Speed I2C Clock SCL High Count Register

Address Offset: 0x24

Default Value: 0x0000 000C

Description: It is only writable when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_HS_SCL_HCNT	R	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for high speed, based on ic_clk period. The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns. The minimum value is 6.

26.15.11. I2C_HS_SCL_LCNT

Name: High Speed I2C Clock SCL Low Count Register

Address Offset: 0x28

Default Value: 0x00000020

Description: It is only writable when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	IC_HS_SCL_LCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for high speed, based on ic_clk period. The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 160ns; for 400pF loading, the SCL High time is 320ns. The minimum value is 6.

26.15.12. I2C_INTR_STAT

Name: I2C Interrupt Status Register

Address Offset: 0x2C

Default Value: 0x00000000

Description: Each bit in this register has a corresponding mask bit in the IC_INTR_MASK register. These bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the IC_RAW_INTR_STAT register. Please refer to SFR IC_RAW_INTR_STAT for a detailed description of each interrupt bit.

Bits	Fields	R/W	Description
31:15	-	R	Reserved
14	R_SCL_STUCK_AT_LOW	R	Values: 0x1 (ACTIVE): R_SCL_STUCK_AT_LOW interrupt is active 0x0 (INACTIVE): R_SCL_STUCK_AT_LOW interrupt is inactive
13	-	R	Reserved
12	R_RESTART_DET	R	Values: 0x1 (ACTIVE): R_MASTER_ON_HOLD interrupt is active 0x0 (INACTIVE): R_MASTER_ON_HOLD interrupt is inactive
11	R_GEN_CALL	R	Values: 0x1 (ACTIVE): R_GEN_CALL interrupt is active 0x0 (INACTIVE): R_GEN_CALL interrupt is inactive
10	R_START_DET	R	Values: 0x1 (ACTIVE): R_START_DET interrupt is active 0x0 (INACTIVE): R_START_DET interrupt is inactive
9	R_STOP_DET	R	Values: 0x1 (ACTIVE): R_STOP_DET interrupt is active 0x0 (INACTIVE): R_STOP_DET interrupt is inactive

Bits	Fields	R/W	Description
8	R_ACTIVITY	R	Values: 0x1 (ACTIVE): R_ACTIVITY interrupt is active 0x0 (INACTIVE): R_ACTIVITY interrupt is inactive
7	R_RX_DONE	R	Values: 0x1 (ACTIVE): R_RX_DONE interrupt is active 0x0 (INACTIVE): R_RX_DONE interrupt is inactive
6	R_TX_ABRT	R	Values: 0x1 (ACTIVE): R_TX_ABRT interrupt is active 0x0 (INACTIVE): R_TX_ABRT interrupt is inactive
5	R_RD_REQ	R	Values: 0x1 (ACTIVE): R_RD_REQ interrupt is active 0x0 (INACTIVE): R_RD_REQ interrupt is inactive
4	R_TX_EMPTY	R	Values: 0x1 (ACTIVE): R_TX_EMPTY interrupt is active 0x0 (INACTIVE): R_TX_EMPTY interrupt is inactive
3	R_TX_OVER	R	Values: 0x1 (ACTIVE): R_TX_OVER interrupt is active 0x0 (INACTIVE): R_TX_OVER interrupt is inactive
2	R_RX_FULL	R	Values: 0x1 (ACTIVE): R_RX_FULL interrupt is active 0x0 (INACTIVE): R_RX_FULL interrupt is inactive
1	R_RX_OVER	R	Values: 0x1 (ACTIVE): R_RX_OVER interrupt is active 0x0 (INACTIVE): R_RX_OVER interrupt is inactive
0	R_RX_UNDER	R	Values: 0x1 (ACTIVE): RX_UNDER interrupt is active 0x0 (INACTIVE): RX_UNDER interrupt is inactive

26.15.13. I2C_INTR_MASK

Name: I2C Interrupt Mask Register

Address Offset: 0x30

Default Value: 0x000048FF

Description: These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmasks the interrupt.

Bits	Fields	R/W	Description
31:15	-	R	Reserved
14	M_SCL_STUCK_AT_L OW	RW	Mask bit for interrupt SCL_STUCK_AT_LOW
13	-	R	Reserved
12	M_RESTART_DET	RW	Mask bit for interrupt RS_DET
11	M_GEN_CALL	RW	Mask bit for interrupt GEN_CALL
10	M_START_DET	RW	Mask bit for interrupt START_DET
9	M_STOP_DET	RW	Mask bit for interrupt STOP_DET

Bits	Fields	R/W	Description
8	M_ACTIVITY	RW	Mask bit for interrupt ACTIVITY
7	M_RX_DONE	RW	Mask bit for interrupt RX_DONE
6	M_TX_ABRT	RW	Mask bit for interrupt TX_ABRT
5	M_RD_REQ	RW	Mask bit for interrupt RD_REQ
4	M_TX_EMPTY	RW	Mask bit for interrupt TX_EMPTY
3	M_TX_OVER	RW	Mask bit for interrupt TX_OVER
2	M_RX_FULL	RW	Mask bit for interrupt RX_FULL
1	M_RX_OVER	RW	Mask bit for interrupt RX_OVER
0	M_RX_UNDER	RW	Mask bit for interrupt RX_UNDER

26.15.14. I2C_RAW_INTR_STAT

Name: I2C RAW Interrupt Status Register

Address Offset: 0x34

Default Value: 0x0000 0000

Description: Unlike the IC_INTR_STAT register, these bits are not masked so they always show the true status. The interrupt bits are high active.

Bits	Fields	R/W	Description
31:15	-	R	Reserved
14	SCL_STUCK_AT_LOW	R	Indicates whether the SCL Line is stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT number of ic_clk periods.
13	-	R	Reserved
12	RESTART_DET	R	Indicates whether a RESTART condition has occurred on the I2C interface when the macro is operating in Slave mode and the slave is being addressed. Note: However, in high-speed mode or during a START BYTE transfer, the RESTART comes before the address field as per the I2C protocol. In this case, the slave is not the addressed slave when the RESTART is issued, therefore the macro does not generate the RESTART_DET
11	GEN_CALL	R	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling the macro or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. The macro stores the received data in the Rx buffer.
10	START_DET	R	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether the macro is operating in slave or master mode
9	STOP_DET	R	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether the macro is operating in slave or master mode. In Slave Mode: If IC_CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET

Bits	Fields	R/W	Description
			interrupt will be issued only if slave is addressed If IC_CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether it is being addressed. In Master Mode: If IC_CON[10]=1'b1 (STOP_DET_IF_MASTER_ACTIVE), the STOP_DET interrupt will be issued only if Master is active. If IC_CON[10]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt will be issued irrespective of whether master is active or not.
8	ACTIVITY	R	This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it: Disabling the macro Reading the IC_CLR_ACTIVITY register Reading the IC_CLR_INTR register System reset
7	RX_DONE	R	When the macro is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.
6	TX_ABRT	R	This bit indicates if the macro, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. Note: The macro flushes/resets/empties only the TX_FIFO whenever there is a transmit abort caused by any of the events tracked by the IC_TX_ABRT_SOURCE register. The Tx FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface.
5	RD_REQ	R	This bit is set to 1 when the macro is acting as a slave and another I2C master is attempting to read data from it. The macro holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register.
4	TX_EMPTY	R	The behavior of the TX_EMPTY interrupt status differs based on the TX_EMPTY_CTRL selection in the IC_CON register. When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. When TX_EMPTY_CTRL = 1:This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register and the transmission of the address/data from the internal shift register for the most recently popped command is completed. It is automatically cleared by hardware when the buffer level goes above the threshold. When IC_ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no

Bits	Fields	R/W	Description
			data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with ic_en=0, this bit is set to 0.
3	TX_OVER	R	Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.
2	RX_FULL	R	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues.
1	RX_OVER	R	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Note: If bit 9 of the IC_CON register (RX_FIFO_FULL_HLD_CTRL) is programmed to HIGH, then the RX_OVER interrupt never occurs, because the Rx FIFO never overflows.
0	RX_UNDER	R	Set if the processor attempts to read the receiver buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.

26.15.15. I2C_RX_TL

Name: I2C Receive FIFO Threshold Register

Address Offset: 0x38

Default Value: 0x0000 0005

Description:

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	RX_TL	RW	Receive FIFO Threshold Level. Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets

Bits	Fields	R/W	Description
			the threshold for 256 entries

26.15.16. I2C_TX_TL

Name: I2C Transmit FIFO Threshold Register

Address Offset: 0x3C

Default Value: 0x0000 0005

Description:

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	TX_TL	RW	Transmit FIFO Threshold Level. Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries

26.15.17. I2C_CLR_INTR

Name: I2C Clear Combined and Individual Interrupt Register

Address Offset: 0x40

Default Value: 0x0000 0000

Description: Clear Combined and Individual Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_INTR	R	Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE.

26.15.18. I2C_CLR_RX_UNDER

Name: Clear RX_UNDER Interrupt Register

Address Offset: 0x44

Default Value: 0x0000 0000

Description: Clear RX_UNDER Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved

Bits	Fields	R/W	Description
0	CLR_RX_UNDER	R	Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register.

26.15.19. I2C_CLR_RX_OVER

Name: Clear RX_OVER Interrupt Register

Address Offset: 0x48

Default Value: 0x0000 0000

Description: Clear RX_OVER Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_RX_OVER	R	Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register.

26.15.20. I2C_CLR_TX_OVER

Name: Clear TX_OVER Interrupt Register

Address Offset: 0x4C

Default Value: 0x0000 0000

Description: Clear TX_OVER Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_TX_OVER	R	Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register.

26.15.21. I2C_CLR_RD_REQ

Name: Clear RD_REQ Interrupt Register

Address Offset: 0x50

Default Value: 0x0000 0000

Description: Clear RD_REQ Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_RD_REQ	R	Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register.

26.15.22. I2C_CLR_TX_ABRT

Name: Clear TX_ABRT Interrupt Register

Address Offset: 0x54

Default Value: 0x0000 0000

Description: Clear TX_ABRT Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_TX_ABRT	R	Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE.

26.15.23. I2C_CLR_RX_DONE

Name: Clear RX_DONE Interrupt Register

Address Offset: 0x58

Default Value: 0x0000 0000

Description: Clear RX_DONE Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_RX_DONE	R	Read this register to clear the RD_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register.

26.15.24. I2C_CLR_ACTIVITY

Name: Clear ACTIVITY Interrupt Register

Address Offset: 0x5C

Default Value: 0x0000 0000

Description: Clear ACTIVITY Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_ACTIVITY	R	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register.

26.15.25. I2C_CLR_STOP_DET

Name: Clear STOP_DET Interrupt Register

Address Offset: 0x60

Default Value: 0x0000 0000

Description: Clear STOP_DET Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_STOP_DET	R	Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register.

26.15.26. I2C_CLR_START_DET

Name: Clear START_DET Interrupt Register

Address Offset: 0x64

Default Value: 0x0000 0000

Description: Clear START_DET Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_START_DET	R	Read this register to clear the START_DET interrupt (bit 10 of the IC_RAW_INTR_STAT register.

26.15.27. I2C_CLR_GEN_CALL

Name: Clear GEN_CALL Interrupt Register

Address Offset: 0x68

Default Value: 0x0000 0000

Description: Clear GEN_CALL Interrupt Register

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_GEN_CALL	R	Read this register to clear the GEN_CALL interrupt (bit 11 of the IC_RAW_INTR_STAT register.

26.15.28. I2C_ENABLE

Name: I2C Enable Register

Address Offset: 0x6C

Default Value: 0x0000 0000

Description: I2C_ENABLE Register

Bits	Fields	R/W	Description
31:19	-	R	Reserved
18	SMBUS_ALTER_EN	RW	The SMBUS_ALERT_CTRL register bit is used to control assertion of SMBALERT signal. This register bit is auto-cleared after detection of Acknowledgement from master for Alert Response address. Values:

Bits	Fields	R/W	Description
			0x1 (ALERT_ENABLED): Slave initates the Alert signal to indicate SMBus Host 0x0 (SUSPEND_DISABLED): Slave will not initates the Alert signal to indicate SMBus Host.
17	SMBUS_SUSPEND_EN	RW	The SMBUS_SUSPEND_EN register bit is used to control assertion and de-assertion of SMBSUS signal. Values: 0x1 (ENABLED): Host/Master initates the SMBUS system to enter Suspend Mode. 0x0 (DISABLED): Host/Master will not initates the SMBUS system to enter Suspend Mode.
16	SMBUS_CLK_RESET	RW	This bit is used in SMBus Host mode to initiate the SMBus Master Clock Reset. This bit should be enabled only when Master is in idle. Whenever this bit is enabled, the SMBCLK is held low for the IC_SCL_STUCK_TIMEOUT ic_clk cycles to reset the SMBus slave devices.
15:4	-	R	
3	SDA_STUCK_RECOVERY_ENABLE	RW	If SDA is stuck at low indicated through the TX_ABORT interrupt (IC_TX_ABRT_SOURCE[17]), then this bit is used as a control knob to initiate the SDA Recovery Mechanism (that is, send at most 9 SCL clocks and STOP to release the SDA line) and then this bit gets auto clear. Values: 0x1 (SDA_STUCK_RECOVERY_ENABLED): Master initates the SDA stuck at low recovery mechanism. 0x0 (SDA_STUCK_RECOVERY_DISABLED): Master disabled the SDA stuck at low recovery mechanism.
2	TX_CMD_BLOCK	RW	In Master mode: 1'b1: Blocks the transmission of data on I2C bus even if Tx FIFO has data to transmit. 1'b0: The transmission of data starts on I2C bus automatically, as soon as the first data is available in the Tx FIFO. Note: To block the execution of Master commands, set the TX_CMD_BLOCK bit only when Tx FIFO is empty (IC_STATUS[2]==1) and Master is in Idle state (IC_STATUS[5] == 0). Any further commands put in the Tx FIFO are not executed until TX_CMD_BLOCK bit is unset
1	ABORT	RW	When set, the controller initiates the transfer abort. 0: ABORT not initiated or ABORT done 1: ABORT operation in progressThe software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation.
0	ENABLE	RW	Values: 0x1 (ENABLED): I2C is enabled

Bits	Fields	R/W	Description
			0x0 (DISABLED): I2C is disabled

26.15.29. I2C_STATUS

Name: I2C_STATUS Register

Address Offset: 0x70

Default Value: 0x0000 0006

Description: This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt.

When the I2C is disabled by writing 0 in bit 0 of the IC_ENABLE register:

- Bits 1 and 2 are set to 1
- Bits 3 and 10 are set to 0

Bits	Fields	R/W	Description
31:21	-	R	Reserved
20	SMBUS_ALTER_STATUS	R	This bit indicates the status of the SMBus Alert signal (ic_smbalert_in_n). Values: 0x1 (ACTIVE): SMBUS Alert is asserted. 0x0 (INACTIVE): SMBUS Alert is not asserted.
19	SMBUS_SUSPEND_STATUS	R	This bit indicates the status of the SMBus Suspend signal (ic_smbsus_in_n). Values: 0x1 (ACTIVE): SMBUS System is in Suspended mode. 0x0 (INACTIVE): SMBUS System is not in Suspended mode.
18	SMBUS_SLV_ADDR_RESOLVED	R	This bit indicates whether the slave address (ic_sar) is resolved by the ARP Master. Values: 0x1 (ACTIVE): SMBUS Slave Address is Resolved. 0x0 (INACTIVE): SMBUS Slave Address is not Resolved.
17	SMBUS_SLV_ADDR_VLD	R	This bit indicates whether the slave address (ic_sar) is valid or not. Values: 0x1 (ACTIVE): SMBUS Slave Address is Valid. 0x0 (INACTIVE): SMBUS Slave Address is not valid.
16	SMBUS_QUICK_CMD_BIT	R	This bit indicates the R/W bit of the Quick command received. This bit will be cleared after the user has read this bit. Values: 0x1 (ACTIVE): SMBUS QUICK CMD Read/write is set to 1. 0x0 (INACTIVE): SMBUS QUICK CMD Read/write is set to 0.
15:12	-	R	Reserved
11	SDA_STUCK_NOT_RECOVERED	R	This bit indicates that SDA stuck at low is not recovered after the recovery mechanism. In Slave mode, this register bit is not applicable. Values:

Bits	Fields	R/W	Description
			0x1 (ACTIVE): SDA Stuck at low is recovered after recovery mechanism. 0x0 (INACTIVE): SDA Stuck at low is not recovered after recovery mechanism.
10	SLV_HOLD_RX_FIFO_FULL	R	This bit indicates the BUS Hold in Slave mode due to Rx FIFO is Full and an additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1). Values: 0x1 (ACTIVE): Slave holds the bus due to Rx FIFO is full 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Rx FIFO is full.
9	SLV_HOLD_TX_FIFO_EMPTY	R	This bit indicates the BUS Hold in Slave mode for the Read request when the Tx FIFO is empty. The Bus is in hold until the Tx FIFO has data to Transmit for the read request. Values: 0x1 (ACTIVE): Slave holds the bus due to Tx FIFO is empty 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Tx FIFO is empty
8	MST_HOLD_RX_FIFO_FULL	R	This bit indicates the BUS Hold in Master mode due to Rx FIFO is Full and additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1). Values: 0x1 (ACTIVE): Master holds the bus due to Rx FIFO is full 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Rx FIFO is full
7	MST_HOLD_TX_FIFO_EMPTY	R	The I2C master stalls the write transfer when Tx FIFO is empty, and the last byte does not have the Stop bit set. This bit indicates the BUS hold when the master holds the bus because of the Tx FIFO being empty, and the the previous transferred command does not have the Stop bit set. Values: 0x1 (ACTIVE): Master holds the bus due to Tx FIFO is empty 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Tx FIFO is empty
6	SLV_ACTIVITY	R	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. Values: 0x1 (ACTIVE): Slave not idle 0x0 (IDLE): Slave is idle
5	MST_ACTIVITY	R	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. Values: 0x1 (ACTIVE): Master not idle 0x0 (IDLE): Master is idle Note: IC_STATUS[0]-that is, ACTIVITY bit-is the OR of SLV_ACTIVITY and MST_ACTIVITY bits
4	RFF	R	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. Values: 0x1 (FULL): Rx FIFO is full

Bits	Fields	R/W	Description
			0x0 (NOT_FULL): Rx FIFO not full
3	RFNE	R	Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. Values: 0x1 (NOT_EMPTY): Rx FIFO not empty 0x0 (EMPTY): Rx FIFO is empty
2	TFE	R	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. Values: 0x1 (EMPTY): Tx FIFO is empty 0x0 (NON_EMPTY): Tx FIFO not empty
1	TFNF	R	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. Values: 0x1 (NOT_FULL): Tx FIFO not full 0x0 (FULL): Tx FIFO is full
0	ACTIVITY	R	I2C Activity Status. Values: 0x1 (ACTIVE): I2C is active 0x0 (INACTIVE): I2C is idle

26.15.30. I2C_TXFLR

Name: I2C Transmit FIFO Level Register

Address Offset: 0x74

Default Value: 0x0000 0000

Description: This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

The I2C is disabled

There is a transmit abort - that is, TX_ABRT bit is set in the IC_RAW_INTR_STAT register

The slave bulk transmit mode is aborted

The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	TXFLR	R	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

26.15.31. I2C_RXFLR

Name: I2C Receive FIFO Level Register

Address Offset: 0x78

Default Value: 0x0000 0000

Description: This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever:

- The I2C is disabled
- Whenever there is a transmit abort caused by any of the events tracked in IC_TX_ABRT_SOURCE

The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	RXFLR	R	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

26.15.32. I2C_SDA_HOLD

Name: I2C SDA Hold Time Length Register

Address Offset: 0x7C

Default Value: 0x0000 0002

Description: It is only writable when IC_ENABLE[0]=0.

The values in this register are in units of ic_clk period. The value programmed in IC_SDA_TX_HOLD must be greater than the minimum hold time in each mode one cycle in master mode, seven cycles in slave mode for the value to be implemented.

The programmed SDA hold time during transmit (IC_SDA_TX_HOLD) cannot exceed at any time the duration of the low part of SCL. Therefore the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the SCL period measured in ic_clk cycles.

Bits	Fields	R/W	Description
31:24	-	R	Reserved
23:16	IC_SDA_RX_HOLD	RW	Sets the required SDA hold time in units of ic_clk period, when the macro acts as a receiver.

26.15.33. I2C_TX_ABRT_SOURCE

Name: I2C Transmit Abort Source Register

Address Offset: 0x80

Default Value: 0x0000 0000

Description: This register has 32 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; RESTART must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]).

Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

Bits	Fields	R/W	Description
31:23	TX_FLUSH_CNT	R	This field indicates the number of Tx FIFO Data Commands which are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled.
22:21	-	R	Reserved
20	ABRT_DEVICE_WRITE	R	This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the Tx-FIFO consists of write commands.
19	ABRT_DEVICE_SLVADDR_NOACK	R	This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the slave address sent was not acknowledged by any slave.
18	ABRT_DEVICE_NOACK	R	This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the device id sent was not acknowledged by any slave
17	ABRT_SDA_STUCK_AT_LOW	R	This is a master-mode-only bit. Master detects the SDA Stuck at low for the IC_SDA_STUCK_AT_LOW_TIMEOUT value of ic_clks.
16	ABRT_USER_ABRT	R	This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1])
15	ABRT_SLVRD_INTX	R	Values: 0x1 (ABRT_SLVRD_INTX_GENERATED): Slave trying to transmit to remote master in read mode 0x0 (ABRT_SLVRD_INTX_VOID): Slave trying to transmit to remote master in read mode- scenario not present
14	ABRT_SLV_ARBLOST	R	This field indicates that a Slave has lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. Note: Even though the slave never 'owns' the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_i2c no longer own the bus
13	ABRT_SLVFLUSH_TXFIFO	R	This field specifies that the Slave has received a read command and some data exists in the TX FIFO, so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.
12	ABRT_LOST	R	This field specifies that the Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration.
11	ABRT_MASTER_DIS	R	This field indicates that the User tries to initiate a Master operation with the Master mode disabled.
10	ABRT_10B_RD_NORSTRT	R	This field indicates that the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the master sends a read command in 10-bit addressing mode.
9	ABRT_SBYTE_NORSTRT	R	Values: 0x1 (ABRT_SBYTE_NORSTRT_GENERATED): User trying to send START byte when RESTART disabled 0x0 (ABRT_SBYTE_NORSTRT_VOID): User trying to send START byte when RESTART disabled- scenario not present
8	ABRT_HS_NORSTRT	R	This field indicates that the restart is disabled (IC_RESTART_EN bit

Bits	Fields	R/W	Description
			(IC_CON[5]) =0) and the user is trying to use the master to transfer data in High Speed mode.
7	ABRT_SBYTE_ACKDET	R	This field indicates that the Master has sent a START Byte and the START Byte was acknowledged (wrong behavior).
6	ABRT_HS_ACKDET	R	This field indicates that the Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior).
5	ABRT_GCALL_READ	R	This field indicates that DW_apb_i2c in the master mode has sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1).
4	ABRT_GCALL_NOACK	R	This field indicates that DW_apb_i2c in master mode has sent a General Call and no slave on the bus acknowledged the General Call.
3	ABRT_TXDATA_NOACK	R	This field indicates the master-mode only bit. When the master receives an acknowledgement for the address, but when it sends data byte(s) following the address, it did not receive an acknowledge from the remote slave(s).
2	ABRT_10ADDR2_NOACK	R	This field indicates that the Master is in 10-bit address mode and that the second address byte of the 10-bit address was not acknowledged by any slave.
1	ABRT_10ADDR1_NOACK	R	This field indicates that the Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave.
0	ABRT_7B_ADDR_NOACK	R	This field indicates that the Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.

26.15.34. I2C_SLV_DATA_NACK_ONLY

Name: Generate Slave Data NACK Register

Address Offset: 0x84

Default Value: 0x0000 0000

Description: The register is used to generate a NACK for the data part of a transfer when this macro is acting as a slave-receiver.

A write can occur on this register if both of the following conditions are met:

DW_apb_i2c is disabled (IC_ENABLE[0] = 0)

Slave part is inactive (IC_STATUS[6] = 0)

Note: The IC_STATUS[6] is a register read-back location for the internal slv_activity signal; the user should poll this before writing the ic_slv_data_nack_only bit.

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	NACK	RW	Generate NACK. This NACK generation only occurs when the macro is a slave-receiver. Values: 0x1 (ENABLED): Slave receiver generates NACK upon data reception only

Bits	Fields	R/W	Description
			0x0 (DISABLED): Slave receiver generates NACK normally

26.15.35. I2C_DMA_CR

Name: DMA Control Register

Address Offset: 0x88

Default Value: 0x0000 0000

Description: This register is used to enable the DMA Controller interface operation.

Bits	Fields	R/W	Description
31:2	-	R	Reserved
1	TDMAE	RW	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel.
0	RDMAE	RW	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel.

26.15.36. I2C_DMA_TDLR

Name: DMA Transmit Data Level Register

Address Offset: 0x8C

Default Value: 0x0000 0000

Description:

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	DMATDL	RW	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

26.15.37. I2C_DMA_RDLR

Name: DMA Receive Data Level Register

Address Offset: 0x90

Default Value: 0x0000 0000

Description:

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2:0	DMARDL	RW	Receive Data Level. This bit field controls the level at which a DMA

Bits	Fields	R/W	Description
			request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE =1. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.

26.15.38. I2C_SDA_SETUP

Name: I2C SDA Setup Register

Address Offset: 0x94

Default Value: 0x0000 0000

Description: This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the rising edge of SCL - relative to SDA changing - when this macro services a read request in a slave-transmitter operation. This register must be programmed with a value equal to or greater than 2.

Writes to this register succeed only when IC_ENABLE[0] = 0.

Note: The length of setup time is calculated using [(IC_SDA_SETUP - 1) * (ic_clk_period)], so if the user requires 10 ic_clk periods of setup time, they should program a value of 11. The IC_SDA_SETUP register is only used when operating as a slave transmitter.

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	SDA_SETUP	RW	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2.

26.15.39. I2C_ACK_GENERAL_CALL

Name: I2C ACK General Call Register

Address Offset: 0x98

Default Value: 0x0000 0001

Description: The register controls whether this macro responds with an ACK or NACK when it receives an I2C General Call address.

This register is applicable only when the DW_apb_i2c is in slave mode.

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	ACK_GEN_CALL	RW	ACK General Call. When set to 1, the macro responds with an ACK when it receives a General Call. Otherwise, the macro responds with a NACK.

26.15.40. I2C_ENABLE_STATUS

Name: I2C Enable Status Register

Address Offset: 0x9C

Default Value: 0x0000 0000

Description: The register is used to report the macro hardware status when the IC_ENABLE[0] register is set from 1 to 0.

If IC_ENABLE[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.

If IC_ENABLE[0] has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

Note: When IC_ENABLE[0] has been set to 0, a delay occurs for bit 0 to be read as 0 because disabling the macro depends on I2C bus activities.

Bits	Fields	R/W	Description
31:3	-	R	Reserved
2	SLV_RX_DATA_LOST	R	Slave Received Data Lost. Values: 0x1 (ACTIVE): Slave RX Data is lost 0x0 (INACTIVE): Slave RX Data is not lost
1	SLV_DISABLED_WHILE_BUSY	R	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting bit 0 of the IC_ENABLE register from 1 to 0. Values: 0x1 (ACTIVE): Slave is disabled when it is active 0x0 (INACTIVE): Slave is disabled when it is idle
0	IC_EN	R	Values: 0x1 (ENABLED): I2C enabled 0x0 (DISABLED): I2C disabled

26.15.41. I2C_FS_SPKLEN

Name: I2C SS, FS spike suppression limit

Address Offset: 0xA0

Default Value: 0x0000 0001 (I2C1) / 0x0000 0005(I2C2)

Description: This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS, FS modes.

This register can be written only when IC_ENABLE[0]=0

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	IC_FS_SPKLEN	RW	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic.

26.15.42. I2C_HS_SPKLEN

Name: I2C HS spike suppression limit register

Address Offset: 0xA4

Default Value: 0x0000 0001

Description: This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in HS modes.

This register can be written only when IC_ENABLE[0]=0

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	IC_HS_SPKLEN	RW	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic

26.15.43. I2C_CLR_RESTART_DET

Name: Clear RESTART_DET Interrupt Register

Address Offset: 0xA8

Default Value: 0x0000 0000

Description:

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_RESTART_DET	R	Read this register to clear the RESTART_DET interrupt (bit 12) of IC_RAW_INTR_STAT register.

26.15.44. I2C_SCL_STUCK_AT_LOW_TIMEOUT

Name: I2C SCL Stuck at Low Timeout register

Address Offset: 0xAC

Default Value: 0xFFFFFFFF

Description: This register can be written only when IC_ENABLE[0]=0

Bits	Fields	R/W	Description
31:0	IC_SCL_STUCK_LOW_TIMEOUT	RW	This register is used to store the duration, measured in ic_clk cycles, used to Generate an Interrupt (SCL_STUCK_AT_LOW) if SCL is held low for the IC_SCL_STUCK_LOW_TIMEOUT duration.

26.15.45. I2C_SDA_STUCK_AT_LOW_TIMEOUT

Name: I2C SDA Stuck at Low Timeout register

Address Offset: 0xB0

Default Value: 0xFFFFFFFF

Description: This register can be written only when IC_ENABLE[0]=0

Bits	Fields	R/W	Description
31:0	IC_SDA_STUCK_LOW_TIMEOUT	RW	This register is used to store the duration, measured in ic_clk cycles, used to Recover the Data (SDA) line through sending SCL pulses if SDA is held low for the mentioned duration.

26.15.46. I2C_CLR_SCL_STUCK_DET

Name: Clear SCL Stuck at Low Detect interrupt Register

Address Offset: 0xB4

Default Value: 0x0000 0000

Description:

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	CLR_SCL_STUCK_DET	R	Read this register to clear the SCL_STUCT_AT_LOW interrupt (bit 15) of the IC_RAW_INTR_STAT register.

26.15.47. SMBUS_CLK_LOW_SEXT

Name: SMBus Slave Clock Extend Timeout register

Address Offset: 0xBC

Default Value: 0xFFFF FFFF

Description: This Register contains the Timeout value used to determine the Slave Clock Extend Timeout in one transfer (from START to STOP).

This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:0	SMBUS_CLK_LOW_SEXT_TIMEOUT	RW	This field is used to detect the Slave Clock Extend timeout (tLOW:SEXT) in master mode extended by the slave device in one message from the initial START to the STOP. The values in this register are in units of ic_clk period.

26.15.48. SMBUS_CLK_LOW_MEXT

Name: SMBus Master Clock Extend Timeout register

Address Offset: 0xC0

Default Value: 0xFFFF FFFF

Description: This Register contains the Timeout value used to determine the Master Clock Extend Timeout in one byte of transfer.

This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:0	SMBUS_CLK_LOW_MEXT_TIMEOUT	RW	This field is used to detect the Master extend SMBus clock (SCLK) timeout defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP in Master mode. The values in this register are in units of ic_clk period.

26.15.49. SMBUS_THIGH_MAX_BUS_IDLE_CNT

Name: SMBus Master THIGH MAX Bus-idle count Register

Address Offset: 0xC4

Default Value: 0x0000 FFFF

Description: This register programs the Bus-idle time period used when a master has been dynamically added to the bus or when a master has generated a clock reset on the bus. This register is used to store the duration, measured in ic_clk cycles, used to detect the Bus Idle condition if SCL and SDA are held high for the mentioned duration.

This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:0	SMBUS_THIGH_MAX_BUS_IDLE_CNT	RW	This field is used to set the required Bus-Idle time period used when a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress The values in this register are in units of ic_clk period.

26.15.50. SMBUS_INTR_STAT

Name: SMBus Interrupt Status Register

Address Offset: 0xC8

Default Value: 0x0000 0000

Description: Each bit in this register has a corresponding mask bit in the IC_SMBUS_INTR_MASK register. These bits are cleared by writing the matching SMBus interrupt clear register(IC_CLR_SMBUS_INTR) bits. The unmasked raw versions of these bits are available in the IC_SMBUS_RAW_INTR_STAT register. Refer to IC_SMBUS_INTR_RAW_STATUS for detail description of each bit. The interrupt status bits are high active.

This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:12	-	R	Reserved
10	R_SMBUS_ALERT_DET	R	Status for interrupt SMBUS_ALERT_DET
9	R_SMBUS_SUSPEND_DET	R	Status for SMBUS_SUSPEND_DET

Bits	Fields	R/W	Description
8	R_SLV_RX_PEC_NACK	R	Status for SLV_RX_PEC_NACK
7	R_ARP_ASSGN_ADDR_CMD_DET	R	Status for ARP_ASSGN_ADDR_CMD_DET
6	R_ARP_GET_UDID_CMD_DET	R	Status for ARP_GET_UDID_CMD_DET
5	R_ARP_RST_CMD_DET	R	Status for ARP_RST_CMD_DET
4	R_ARP_PREPARE_CMD_DET	R	Status for ARP_PREPARE_CMD_DET
3	R_HOST_NOTIFY_MST_DET	R	Status for HOST_NOTIFY_MST_DET
2	R_QUICK_CMD_DET	R	Status for QUICK_CMD_DET
1	R_MST_CLOCK_EXTND_TIMEOUT	R	Status for MST_CLOCK_EXTND_TIMEOUT
0	R_SLV_CLOCK_EXTND_TIMEOUT	R	Status for SLV_CLOCK_EXTND_TIMEOUT

26.15.51. SMBUS_INTR_MASK

Name: SMBus Interrupt Mask Register

Address Offset: 0xCC

Default Value: 0x0000 0000

Description: These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmasks the interrupt.

This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:12	-	R	Reserved
10	M_SMBUS_ALERT_DET	RW	Mask bit for SMBUS_ALERT_DET
9	M_SMBUS_SUSPEND_DET	RW	Mask bit for SMBUS_SUSPEND_DET
8	M_SLV_RX_PEC_NACK	RW	Mask bit for SLV_RX_PEC_NACK
7	M_ARP_ASSGN_ADDR_CMD_DET	RW	Mask bit for ARP_ASSGN_ADDR_CMD_DET
6	M_ARP_GET_UDID_CMD_DET	RW	Mask bit for ARP_GET_UDID_CMD_DET
5	M_ARP_RST_CMD_DET	RW	Mask bit for ARP_RST_CMD_DET
4	M_ARP_PREPARE_CMD_DET	RW	Mask bit for ARP_PREPARE_CMD_DET
3	M_HOST_NOTIFY_MST_DET	RW	Mask bit for HOST_NOTIFY_MST_DET
2	M_QUICK_CMD_DET	RW	Mask bit for QUICK_CMD_DET
1	M_MST_CLOCK_EXTND_TIMEOUT	RW	Mask bit for MST_CLOCK_EXTND_TIMEOUT
0	M_SLV_CLOCK_EXTND_TIMEOUT	RW	Mask bit for SLV_CLOCK_EXTND_TIMEOUT

26.15.52. SMBUS_RAW_INTR_STAT

Name: SMBus Raw Interrupt Status Register

Address Offset: 0xD0

Default Value: 0x0000 0000

Description: This register only exists in I2C1, and doesn't exist in I2C2.

Bits	Fields	R/W	Description
31:12	-	R	Reserved
10	SMBUS_ALERT_DET	R	Indicates whether a SMBALERT (ic_smbalert_in_n) signal is driven low by the slave.
9	SMBUS_SUSPEND_DET	R	Indicates whether a SMBSUS (ic_smbsus_in_n) signal is driven low by the Host
8	SLV_RX_PEC_NACK	R	Indicates whether a NACK has been sent due to PEC mismatch while working as ARP slave.
7	ARP_ASSGN_ADDR_CMD_DET	R	Indicates whether an Assign Address ARP command has been received.
6	ARP_GET_UDID_CMD_DET	R	Indicates whether a Get UDID ARP command has been received.
5	ARP_RST_CMD_DET	R	Indicates whether a General or Directed Reset ARP command has been received.
4	ARP_PREPARE_CMD_DET	R	Indicates whether a prepare to ARP command has been received.
3	HOST_NOTIFY_MST_DET	R	Indicates whether a Notify ARP Master ARP command has been received.
2	QUICK_CMD_DET	R	Indicates whether a Quick command has been received on the SMBus interface regardless of whether this macro is operating in slave or master mode. Enabled only when IC_SMBUS=1 is set to 1.

1	MST_CLOCK_EXTND_TIMEOUT	R	Indicates whether the Master device transaction (START-to-ACK, ACK-to-ACK, or ACK-to-STOP) from START to STOP exceeds IC_SMBUS_CLOCK_LOW_MEXT time with in each byte of message.
			This bit is enabled only when:
			IC_SMBUS=1
			IC_CON[0]=1
			IC_EMPTYFIFO_HOLD_MASTER_EN=1 or
			IC_RX_FULL_HLD_BUS_EN=1

0	SLV_CLOCK_EXTND_TIMEOUT	R	Indicates whether the transaction from Slave (i.e from START to STOP) exceeds IC_SMBUS_CLK_LOW_SEXT time.
			This bit is enabled only when:
			IC_SMBUS=1
			IC_CON[0]=1

26.15.53. CLR_SMBUS_INTR

Name: Clear SMBus Interrupt Register

Address Offset: 0xD4

Default Value: 0x0000 0000

Description: This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:12	-	R	Reserved
10	CLR_SMBUS_ALERT_DET	W	Clear bit for SMBUS_ALERT_DET
9	CLR_SMBUS_SUSPEND_DET	W	Clear bit for SMBUS_SUSPEND_DET
8	CLR_SLV_RX_PEC_NACK	W	Clear bit for SLV_RX_PEC_NACK
7	CLR_ARP_ASSGN_ADDR_CMD_DET	W	Clear bit for ARP_ASSGN_ADDR_CMD_DET
6	CLR_ARP_GET_UDID_CMD_DET	W	Clear bit for ARP_GET_UDID_CMD_DET
5	CLR_ARP_RST_CMD_DET	W	Clear bit for ARP_RST_CMD_DET
4	CLR_ARP_PREPARE_CMD_DET	W	Clear bit for ARP_PREPARE_CMD_DET
3	CLR_HOST_NOTIFY_MST_DET	W	Clear bit for HOST_NOTIFY_MST_DET
2	CLR_QUICK_CMD_DET	W	Clear bit for QUICK_CMD_DET
1	CLR_MST_CLOCK_EXTND_TIMEOUT	W	Clear bit for MST_CLOCK_EXTND_TIMEOUT
0	CLR_SLV_CLOCK_EXTND_TIMEOUT	W	Clear bit for SLV_CLOCK_EXTND_TIMEOUT

26.15.54. I2C_OPTIONAL_SAR

Name: I2C Optional Slave Address Register

Address Offset: 0xD8

Default Value: 0x0000 0000

Description: This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:7	-	R	Reserved
6:0	OPTIONAL_SAR	RW	Optional Slave address for DW_apb_i2c when operating as a slave in SMBus Mode.

26.15.55. SMBUS_UDID_LSB

Name: SMBUS ARP UDID LSB Register

Address Offset: 0xDC

Default Value: 0xFFFFFFFF

Description: This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn’t exist in I2C2.

Bits	Fields	R/W	Description
31:1	-	R	Reserved
0	SMBUS_UDID_LSB	R	This field is used to store the LSB 32 bit value of slave unique device identifier used in Address Resolution Protocol.

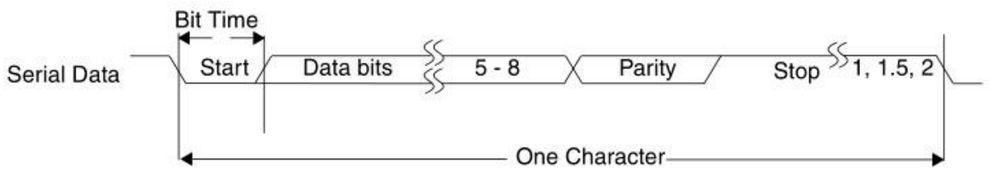
27. Universal asynchronous transceiver (UART)

27.1. Introduction

27.1.1. RS232 Serial communication protocol

UART port supports RS232 serial communication protocol. The format of the protocol is as follows, which can be divided into starting bit, data bit, parity bit and stop bit. Parity bit is an option, and stop bit can be 1, 1.5 or 2 bit width.

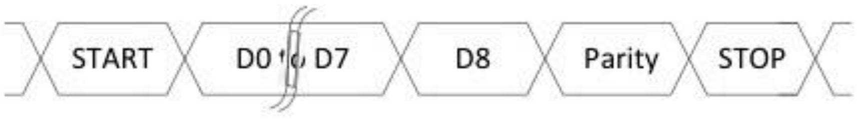
Figure 27-1. RS232 Serial data format



27.1.2. 9-bit data transmission

UART port supports 9-bit data transmission mode. The ninth bit is after the 8th bit of data and before the parity bit and stop bit. The following figure shows the format of 9-bit transmission.

Figure 27-2. 9-bit transmission data format



By enabling 9-bit transmission mode, the system can support the connection between a master controller and multiple slave devices. When the master controller needs to communicate with a slave device, it will first send an address data to select the slave device to communicate with. The difference between address and data lies in the 9th bit. When the 9th bit is set to "1", the data transmitted represents the address of the slave device; when the 9th bit is set to "0", the data is transmitted. When receiving the address, all slave devices will compare with their own address. If the address matches, the device will continue to receive the following data; if not, the slave device will ignore the following data until it receives the new address.

When receiving, the received address can be automatically matched by hardware, and the received address can also be matched by software.

27.1.3. Fractional baud rate

UART supports fractional baud rate configuration to reduce the bit error rate in high-speed transmission, which can ensure that the baud rate error of UART is less than 2%. The factors affecting baud rate are as follows ：

- The working frequency of UART module clock Fuart
- Expected baud rate
- The value of baud rate generation register DIVISOR.
- Acceptable baud rate error

The calculation formula of baud rate is as follows:

$$Baud\ Rate = \frac{F_{uart}}{16 \times DIVISOR} \tag{27.1}$$

Among them, DIVISOR is composed of DLH and DLL registers.

The formula for calculating DIVISOR as follows:

$$DIVISOR = \frac{F_{uart}}{16 \times Baud\ Rate} \tag{27.2}$$

The decimal part of the divisor can be calculated from the value in the DLF register:

Table 27-1. Decimal values for divisor latch

DLF value	fraction	Decimal value
0000	0/16	0.0000
0001	1/16	0.0625
0010	2/16	0.125
0011	3/16	0.1875
0100	4/16	0.25
0101	5/16	0.3125
0110	6/16	0.375
0111	7/16	0.4375
1000	8/16	0.5
1001	9/16	0.5625
1010	10/16	0.625
1011	11/16	0.6875
1100	12/16	0.75
1101	13/16	0.8125
1110	14/16	0.875
1111	15/16	0.9375

Compared with the traditional integer baud rate generator, the programmable fractional baud rate generator can produce higher precision baud rate. This allows programming to the integer and fractional parts of the baud rate. The formula is as follows:

$$Baud\ Rate\ Divisor = \frac{Fuart}{16\times Baud\ Rate} = BRD_1 + BRD_2 \tag{27.3}$$

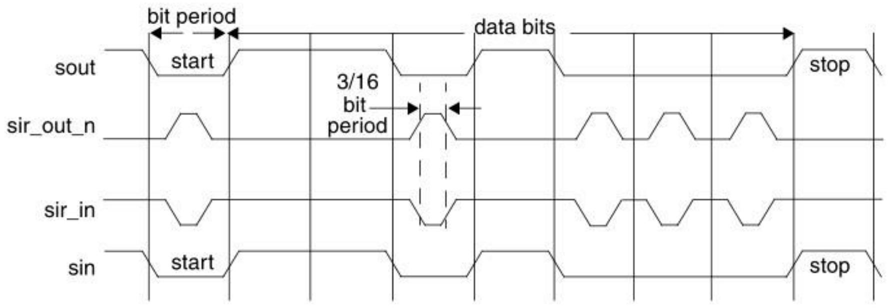
Among the formula，the integer part of the BRD1 divisor. The fractional part of the BRD2 divisor.

27.1.4. IrDA SIR Agreement

UART supports IrDA 1.0 SIR mode for two-way data transmission through infrared, and the maximum baud rate can reach 115.2kbps. The data format is the same as the standard serial communication format. Each data byte is sent in the following order:

- Start with a start bit
- Send 8-bit data
- End with an end bit
- In this way, the width of the data that can be sent is fixed, the transmission of parity bits is not supported, and there is only one end bit.
- The meaning of sending or not sending an infrared pulse is as follows:
- Send an infrared pulse to represent logic 0
- Do not send an infrared pulse to represent logic 1
- The width of each pulse is 3 / 16 of the normal bit width, so each new byte transmission starts with an infrared pulse. However, the polarity of the received data is opposite to that of the transmitted data, which is mainly determined by the characteristics of the infrared receiver. The following figure shows the format of the infrared data.

Figure 27-3. Infrared transmission data format



27.1.5. Automatic flow control

If the automatic RTS mode is enabled, the UART receiver FIFO hardware can control the RTS output of the UART. If the automatic CTS mode is enabled, the UART TSR hardware will start transmitting only when the CTS input signal is valid.

Automatic RTS

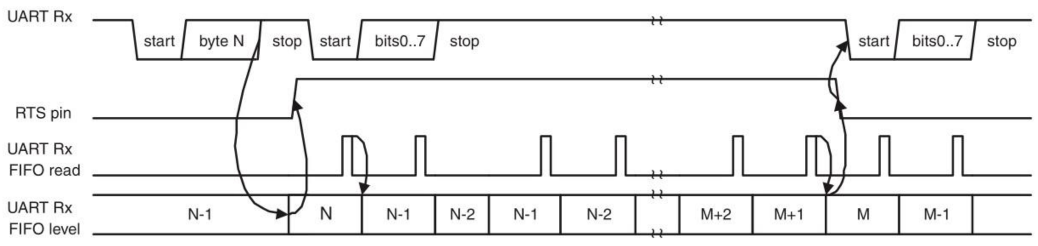
The automatic RTS function can be enabled by setting the AFCE bit. Automatic RTS data flow control is generated in the RBR module and linked to the programmed receiver FIFO trigger threshold. If automatic RTS is enabled, the data flow is controlled as follows:

When the FIFO value of the receiver reaches the programmed trigger threshold, RTS fails (becomes a high level value). When sending UART reaches the trigger threshold, it may send an extra byte (assuming that sending UART has extra bytes to send), because it may not know RTS failure until it starts sending extra bytes. As soon as the receiver FIFO reaches the previous trigger threshold, RTS will automatically take effective again (become a low level value).

After the RTS signal sending UART takes effective again, it can continue to send data.

If the automatic RTS mode is disabled, the RTS bit controls the RTS output of the UART. If the automatic RTS mode is enabled, the hardware can control the RTS output and copy the actual value of RTS to the RTS control bit of UART. If automatic RTS is enabled, only software can read only the value of RTS control bit.

Figure 27-4. Automatic RTS function sequence diagram

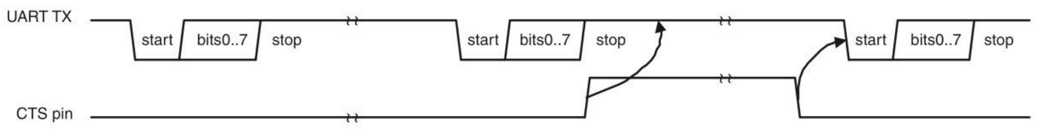


Automatic CTS

The automatic CTS function can be enabled by setting AFCE bit. If automatic CTS is enabled, the transmitter circuit will check the CTS input before sending the next data byte. When CTS is active (low level), the transmitter sends the next byte. In order for the sender to stop sending the following bytes, CTS must be released before the last stop bit currently being sent is sent half way. In automatic CTS mode, the change of CTS signal will not trigger modem status interrupt unless CTS interrupt enable bit is set, but Delta CTS bit in MSR will be set.

The automatic CTS function can reduce the interruption of the host system. When flow control is enabled, the change of CTS status will not trigger host interrupt, because the device will automatically control its transmitter. If the automatic CTS is not used, the transmitter will send out all the data in the transmit FIFO, resulting in the overflow error of the receiver. Figure 27.5 shows the timing of the automatic CTS function.

Figure 27-5. Automatic CTS function sequence diagram



27.1.6. DMA operation

By using micro DMA, the user can choose to operate the transmission and / or reception of UART. The DMA mode is determined by the DMA mode selection bit in the FCR register. This bit is valid only when FIFO is enabled by the FIFO enable bit in the FCR register

UART Receiver DMA

In DMA mode, when the FIFO level of the receiver is equal to or greater than the trigger level, or when a character timeout occurs, the DMA request of the receiver will take effect. Please refer to the description of Rx trigger level above. The receiver DMA request is cleared by the DMA controller.

UART Transmitter DMA

In DMA mode, when the transmitter FIFO becomes not full, the transmitter DMA request will take effect. The transmitter DMA request is cleared by the DMA controller.

27.2. UART Register

27.2.1. Receiver buffer register (UART_RBR) (When DLAB = 0)

Address offset: 0x000

Reset value: NA

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8:0	RBR	R	UART receiver buffer register contains the earliest received byte in UART RX FIFO.

RBR is the highest byte of UART RX FIFO. The highest byte of Rx FIFO contains the earliest received character and can be read through the bus interface. LSB (bit 0) indicates the received data bit. If less than 9 characters are received, the unused MSB is filled with 0.

To access RBR, the divisor latch access bit (DLAB) in LCR must be 0. RBR is always read-only.

Because the parity error (PE), frame error (FE) and interval interrupt (BI) bits correspond to the byte at the top of RBR FIFO (that is, the byte to be read next time when reading from RBR), to correctly extract the effective received byte pair and its status bit, the contents of LSR register should be read first, and then the bytes in RBR should be read.

27.2.2. Transmitter hold register (UART_THR) (when DLAB = 0)

Address offset: 0x000

Reset value: NA

Bits	Fields	R/W	Description
31:9	-	R	Reserved
8:0	THR	W	Writing to the UART transmit hold register will save the data to the UART transmit FIFO. When the byte reaches the bottom of FIFO and the transmitter is available, the byte is sent.

THR is the highest byte of UART TX FIFO. The highest byte is the latest character in TX FIFO, which can be written through the bus interface. LSB represents the first bit to be sent.

To access THR, the divisor latch access bit (DLAB) in LCR must be 0. THR is always write only.

27.2.3. Divisor latch LSB register (UART_DLL) (when DLAB = 1)

Address offset: 0x000

Reset value: NA

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	DLLSB	RW	UART divisor latch LSB register and DLH register determine the baud rate of UART.

27.2.4. Divider latch MSB register (UART_DLH) (when DLAB = 1)

Address offset: 0x004

Reset value: NA

Bits	Fields	R/W	Description
31:8	-	R	Reserved
7:0	DLHSB	RW	UART divisor latch LSB register and DLH register determine the baud rate of UART.

UART divisor latch is a part of UART baud rate generator, and stores the used value. It is used together with fractional frequency divider to implement UART division_ PCLK clock is divided to generate baud rate clock, which must be 16 times of the required baud rate. The DLL and DLH registers together form a 16 bit divisor, in which the DLL contains the lower 8 bits of the divisor and the DLH contains the upper 8 bits of the divisor. The value of 0x0000 is treated as the value of 0x0001 because the divisor cannot be 0. To access the UART divisor latch, the divisor latch access bit (DLAB) in the LCR must be 1.

27.2.5. Interrupt enable register (UART_IER) (when DLAB = 0)

Address offset: 0x004

Reset value: NA

Description: IER is used to enable four UART interrupt sources.

Bits	Fields	R/W	description
31:8	-	R	Reserved
7	PTIME	RW	Control programmable THRE interrupt. 0: Disable programmable THRE interrupt. 1: Enable programmable THRE interrupt.
6:5	-	R	Reserved
4	LSRCLRMD	RW	This bit is used to control the clearing mode of status bit in LSR register. Only for OE, PE, Fe and Bi status bits. 0: when reading RBR or LSR register, clear LSR status bit. 1: Clear the LSR status bit only when reading the LSR register.

Bits	Fields	R/W	description
3	MSIE	RW	Modem interrupt control. 0: Disable modem interrupt. 1: Enable modem interrupt.
2	RLSIE	RW	RX line interrupt enabled. Enable UART RX line status interrupt. The status of the interrupt can be read from LSR [4:1]. 0: disable RX line state interrupt. 1: Enable RX line status interrupt.
1	THREIE	RW	THRE interrupt enable. THRE interrupt to enable UART. The status of the interrupt can be read from LSR [5]. 0: disable THRE interrupt. 1: Enable THRE interrupt.
0	RDAIE	RW	RBR interrupt enable. UART enabled receive data available interrupt. It also controls the character receive timeout interrupt. 0: disable RDA interrupt. 1: Enable RDA interrupt.

27.2.6. Interrupt identification register (UART_IIR)

Address offset: 0x008

Reset value: 0x001

Bits	Fields	R/W	description
31:8	-	R	Reserved
7:6	FIFOSE	R	FIFO enable control. 00: FIFO is disabled. 11: FIFO enabled.
5:4	-	R	Reserved. User software should not write 1 to the Reserved bit.
3：0	INTID	R	Interrupt identification. LER[3:0] identifies the highest priority interrupt pending. Other combinations of IER[3:1] not listed below are Reserved values (100, 101, 111). 0xc: byte timeout interrupt. 0x7: busy state detection interrupt. 0x6: receive line status interrupt. 0x4: receive data available interrupt. 0x2: THR null interrupt. 0x1: no pending interrupt. 0x0: modem interrupt.

Table 27-2. UART interrupt handling

IIR[3:0]	priority	interrupt type	Interrupt source	Interrupt reset
0001	-	NULL	NULL	-
0110	highest	RX line status / error	OE、PE、FE or BI	LSR read operation
0100	second	RX data available	RX data available or reach FIFO trigger level (fcr0 = 1)	RBR read operation or UART FIFO below trigger level

IIR[3:0]	priority	interrupt type	Interrupt source	Interrupt reset
1100	second	Character timeout indication	There is at least one character in the RX FIFO, and there is no character input or output for a period of time. The length of the time depends on the number of characters in the FIFO and the trigger value in the period of 3.5 to 4.5 characters. The actual time is:[（ Word length ） × 7 - 2] × 8 + [（ Trigger level - Number of characters ） × 8 + 1] RCLK	RBR read operation
0010	third	THRE	THRE	IIR read operation or THR write operation

27.2.7. FIFO Control register (UART_FCR)

Address offset: 0x008

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7:6	RXTL	W	RX trigger level. These two bits determine the number of characters that the receiver UART FIFO must write before activating the interrupt. 00: trigger level 0 (1 character or 0x01). 01: trigger level 1 (4 characters or 0x04). 10: Trigger level 2 (8 characters or 0x08). 11: Trigger level 3 (14 characters or 0x0e).
5:4	TXTL	W	TX trigger level. These two bits determine the number of characters in the FIFO before the UART FIFO is activated. 00: trigger level 0 (0 characters or 0x00). 01: trigger level 1 (2 characters or 0x02). 10: Trigger level 2 (4 characters or 0x04). 11: Trigger level 3 (8 characters or 0x08).
3	-	R	Reserved
2	TXFIFORS	W	TX FIFO reset 0: It has no effect on both UART FIFOs. 1: Writing logic 1 to FCR [2] will clear all bytes in UART TX FIFO and reset pointer logic. This bit can be cleared automatically.
1	RXFIFORS	W	RX FIFO reset 0: no effect on both UART FIFOs. 1: Writing logic 1 to FCR [1] will clear all bytes in UART RX FIFO and reset pointer logic. This bit can be cleared automatically.
0	FIFOEN	W	FIFO ENABLE 0: UART FIFO is disabled. 1: The high level is valid, enabling access to UART Rx, TX FIFO and FCR [7:1]. This bit must be set for correct UART operation. Any change in this bit will automatically clear the UART FIFO

27.2.8. Line control register (UART_LCR)

Address offset: 0x00C

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7	DLAB	RW	Divisor latch access bit (DLAB). 0: disable access to divisor latch. 1: Enables access to the divisor latch.
6	BC	RW	Interval control. 0: disable interval transfer. 1: Enable interval transmission. When LCR [6] is high level active, output pin UART TXD is forced to logic 0.
5:4	PS	RW	Parity check selection. 0x0: odd test. The number of logical 1 in the transmitted characters and additional check bits is odd. 0x1: even test. The number of logical 1 in the transmitted characters and additional check bits is even. 0x2: parity bit is forced to 1. 0x3: the parity bit is forced to 0.
3	PE	RW	Parity check enable. 0: disable parity generation and checking. 1: Enables parity generation and checking.
2	SBS	RW	Stop bit selection. 0: 1 stop bits. 1: 2 stop bits (1.5 stop bits if LCR [1:0] = 00).
1:0	WLS	RW	Word length selection. 00: 5 Bit character length. 01: 6 Bit character length. 10: 7 Bit character length. 11: 8 Bit character length.

27.2.9. Modem control register (UART_MCR)

Address offset: 0x010

Reset value: 0x000

Bits	Fields	R/W	description
31:7	-	R	Reserved
6	SIRE	RW	IrDA SIR Mode enable 0: disable IrDA SIR mode. 1: Enable IrDA SIR mode.
5	AFCE	RW	Automatic process control enable. 0: Disable automatic process control. 1: Enable automatic process control.
4:2	-	R	Reserved
1	RTS	RW	Modem output pin RTS source. When modem loop-back mode is active, this bit is read as 0.

0	-	R	Reserved

27.2.10. 27.2.10 Line status register (UART_LSR)

Address offset: 0x014

Reset value: 0x060

Bits	Fields	R/W	description
31:9	-	R	Reserved
8	ADDR_RCVD	R	Address receive bit. When the 9bit data mode is valid, this bit refers to the 9th bit received. This bit can also represent the received address or data. 0: received address. 1: Data received.
7	RFE	R	RX FIFO error. When a character with RX error (such as frame error, parity error or interval interrupt) is loaded into RBR, LSR [7] will be set. When the LSR register is read and UART This bit is cleared when there are no subsequent errors in FIFO. 0: RBR does not contain UART RX error or FCR [0] = 0. 1: UART RBR contains at least one UART RX error.
6	TEMT	R	The transmitter is empty. When THR (or TX FIFO) and transmitter shift register are empty at the same time, TEMT will be set; when transmitter shift register or THR (or TX FIFO) contains valid data, TEMT will be cleared. 0: THR (or TX FIFO) or transmitter shift register contains valid data. 1: THR (or TX FIFO) and transmitter shift register are empty.
5	THRE	R	The transmitter keeps the register empty. When the programmable THRE interrupt mode is not enabled (IER[7] = 0), this bit indicates whether the THR or TX FIFO is empty, regardless of whether the FIFO is enabled or not. 0: THR or TX FIFO contains valid data. 1: THR or TX FIFO is empty. When both the programmable THRE mode and FIFO are enabled (IER[7] = 1 and FCR [0] = 1), this bit indicates whether the transmit FIFO is full. 0: TX FIFO is not full. 1: TX FIFO is full.
4	BI	R	The interval is interrupted. In the process of sending the whole character (start, data, parity, stop), If rxd1 remains in the interval state (all "0"), the interval interrupt will occur. Once the interval condition is detected, the receiver will immediately enter the idle state until rxd1 enters the marked state (all "1"). The LSR read operation will clear the status bit. The interval detection time depends on FCR [0]. Note: the interval interrupt is related to the character at the top of UART RBR FIFO. 0: the interval interrupt state is invalid. 1: The interval interrupt status is valid.
3	FE	R	Framing error. When the stop bit of the received character is logical 0, a framing error occurs. The LSR read operation will clear LSR [3]. The framing error detection time depends on fcr0. When a framing

Bits	Fields	R/W	description
			error is detected, RX will try to resynchronize with the data and assume that the stop bit of the error is actually a leading start bit. However, even if there is no framing error, it is impossible to assume that the next received byte is correct. Note: framing errors are related to the characters at the top of UART RBR FIFO. 0: invalid framing error state. 1: The framing error state is valid.
2	PE	R	Parity error. A parity error occurs when the parity bit of a received character is in an error state. The LSR read operation will clear LSR [2]. Parity error detection time depends on FCR [0] Note: parity error is related to the character at the top of UART RBR FIFO. 0: invalid parity error state. 1: Parity error status is valid.
1	OE	R	Overflow error. The overflow error condition is set immediately after the error occurs. The LSR read operation will clear the LSR [1]. When UART RSR has a new character combination and UART RBR FIFO is full, LSR [1] will be set. In this case, UART RBR FIFO will not be covered and the characters in UART RSR will be lost. 0: invalid overflow error state. 1: Overflow error status is valid.
0	DR	R	Receiver data ready: when RBR contains unread characters, LSR [0] will be set; when UART RBR When FIFO is empty, LSR [0] will be cleared. 0: RBR is empty. 1: RBR contains valid data.

27.2.11. Modem status register (UART_MSR)

Address offset: 0x018

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7	DCD	R	Data carrier detection status. Enter the complement of the DCD. In modem loopback mode, this bit is connected to MCR [3]
6	RI	R	Ringing indicator status. Enter the complement of RI. In modem loopback mode, this bit is connected to MCR [2].
5	DSR	R	Data set ready state. The complement of input signal DSR. In modem return mode, this bit is connected to MCR[0].
4	CTS	R	Clear send status. The complement of input signal CTS. In modem loopback mode, this bit is connected to MCR [1]
3	DDCD	R	Delta DCD。 This bit is set when the state of the input DCD changes. The MSR read operation clears the bit. 0: no state change detected on modem input DCD. 1: A state change on the modem input DCD has been detected.
2	TERI	R	Back edge RI. This bit is set when the input RI is converted from low level to high level. The MSR read operation clears the bit.

			0: no state change detected on modem input RI. 1: A low to high level transition on the RI is detected.
1	DDSR	R	Delta DSR。 This bit is set when the state of the input DSR changes. The MSR read operation clears the bit. 0: invalid overflow error state. 1: Overflow error status is valid.
0	DCTS	R	Delta CTS。 This bit is set when the state of the input CTS changes. Reading MSR will clear this bit. 0: no state change detected on modem input CTS. 1: A state change on the modem input CTS has been detected.

27.2.12. High speed temporary register (UART_SCR)

Address offset: 0x01C

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7:0	PAD	RW	A byte that can be read and written.

27.2.13. UART Status register (UART_USR)

Address offset: 0x07C

Reset value: 0x006

Bits	Fields	R/W	description
31:5	-	R	Reserved
4	RFF	R	Receive FIFO full. Used to indicate whether the receive FIFO is full.
3	RFNE	R	Receive FIFO is not empty. Used to indicate whether the receive FIFO is empty.
2	TFE	R	The transmit FIFO is empty. Used to indicate whether the transmit FIFO is empty.
1	TFNF	R	The transmit FIFO is full. Used to indicate whether the transmit FIFO is full.
0	BUSY	R	Busy state. Used to indicate that serial data transmission is in progress.

27.2.14. Transmit FIFO level register (UART_TFL)

Address offset: 0x080

Reset value: 0x000

Bits	Fields	R/W	description
31:4	-	R	Reserved

3：0	TFL	R	Used to indicate the number of data in transmit FIFO.

27.2.15. Receive FIFO level register (UART_RFL)

Address offset: 0x084

Reset value: 0x000

Bits	Fields	R/W	description
31:4	-	R	Reserved
3：0	RFL	R	Used to indicate the number of data in the receive FIFO.

27.2.16. Software restart register (UART_SRR)

Address offset: 0x088

Reset value: 0x000

Bits	Fields	R/W	description
31:3	-	R	Reserved
2	XFR	RW	This is the shadow bit that sends the FIFO restart control bit FCR [2].
1	RFR	RW	This is the shadow bit of the receive FIFO restart control bit FCR [1].
0	UR	RW	This is the UART module restart control bit.

27.2.17. Shadow TX request register (UART_SRTS)

Address offset: 0x08C

Reset value: 0x000

Bits	Fields	R/W	description
31:1	-	R	Reserved
0	SRTS	RW	This is the shadow bit of the transmit request control bit RTS (MCR [1]).

27.2.18. Shadow interrupt control register (UART_SBCR)

Address offset: 0x090

Reset value: 0x000

Bits	Fields	R/W	description
31:1	-	R	Reserved
0	SBCB	RW	This is the shadow bit of the interrupt control bit (LCR [6]).

27.2.19. Shadow FIFO enable register (UART_SFE)

Address offset: 0x098

Reset value: 0x000

Bits	Fields	R/W	description
31:1	-	R	Reserved
0	SFE	RW	This is the shadow bit of the FIFO enable control bit (FCR [0]).

27.2.20. Shadow receive trigger register (UART_SRT)

Address offset: 0x09C

Reset value: 0x000

Bits	Fields	R/W	description
31:0	-	R	Reserved
1:0	SRT	RW	This is the shadow bit of the receive trigger control bit (FCR [7:6]).

27.2.21. Shadow TX trigger register (UART_STET)

Address offset: 0x0A0

Reset value: 0x000

Bits	Fields	R/W	description
31:0	-	R	Reserved
1:0	STET	RW	This is the shadow bit of the transmit memory empty trigger control bit (FCR [7:4]).

27.2.22. DMA Software abort register (UART_DMASA)

Address offset: 0x0A8

Reset value: 0x000

Bits	Fields	R/W	description
31:1	-	R	Reserved
0	DMASA	RW	When DMA error occurs, set this bit to abort DMA transmission. This bit will be cleared automatically.

27.2.23. Fractional latch register (UART_DLF)

Address offset: 0x0C0

Reset value: 0x000

Bits	Fields	R/W	description

31:4	-	R	Reserved
3：0	DLF	RW	This register is used to store the fractional part of the baud rate generator.

27.2.24. Receive address register (UART_RAR)

Address offset: 0x0C4

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7：0	RAR	R	This register is used to store the address of the comparison in receive mode. When bit 9 is set to "1", the next 8 bits are compared with the address in the RAR register. When matching, the next data with the 9th bit of "0" will be received.

27.2.25. TX address register (UART_TAR)

Address offset: 0x0C8

Reset value: 0x000

Bits	Fields	R/W	description
31:8	-	R	Reserved
7：0	TAR	R	This register is used to store addresses in send mode. When WLS_ E position "1", UART will send 9-bit data, the 9th position "1", the next data will be taken out from the tar register.

27.2.26. Extended control register (UART_EXTLCR)

Address offset: 0x0CC

Reset value: 0x000

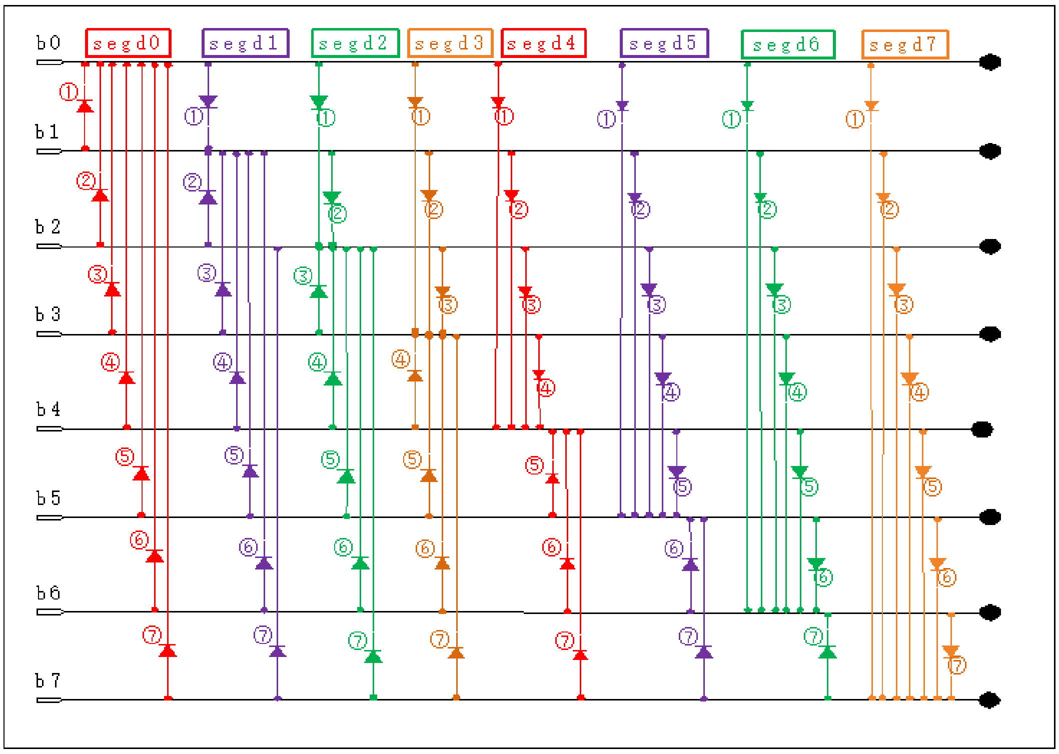
Bits	Fields	R/W	description
31:4	-	R	Reserved
3	TRANSMIT_MODE	RW	It is used to control whether the transmission mode is 9 bit or not. 1: The transmit register is 9 bits wide. 0: the transmit register is 8 bits wide.
2	SEND_ADDR	RW	Send address control bit. 1: Send 9-bit wide data, the 9th position is "1", and the address is taken from tar. 0: Send 9-bit wide data, the 9th position is "0", and the data is taken from TxFIFO.
1	ADDR_MATCH	RW	Address matching enable. 1: Allow address matching. 0: normal transmission mode.
0	WLS_E	RW	WLS extension. Used to enable 9-bit data transmission

28. LED Driver Controller (LED)

28.1. LED Introduction

LED driver controller contains 8-segment LED driver circuit, it uses 8 output port to drive up to 56 LED which saves lots of IO resource.

Figure 28-1. LED driver usage



28.2. LED Driver Controller Main Features

The main characteristics of the drive controller are as follows:

- Using Tri-state IO port, IO port can output low level, high level, high impedance;
- Module clock uses APB2_CLK;
- The LED driver scan period is configurable;
- The output can be switched dynamically;

28.3. Scan Width and Cycle

The scanning period of LED Driver is determined by LED_CYC after configuring the LED driver clock (APB2_CLK). Since the LED driver module is in serial output mode, only one LED can be lighted at the same time.

The LED scan cycle time is calculated as follows:

$$56 * t_{APB2\_CLK} * (LED\_CYC + 1)$$

28.4. LED Drive Register

28.4.1. LED Control Register (LED_CON)

Address offset: 0x00

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:6	-	R	Reserved
5:4	LED_CHNUM	RW	The LED Drive channel control. 00: switch on channel led0,led1,led2,led3,led4,led5,led6 and led7; 01: switch on channel led0,led1,led2,led3,led4 and led5; 10: switch on channel led0,led1,led2,led3,led4,led5 and led6 11: switch on channel led0,led1,led2,led3,led4,led5,led6 and led7. Note: If the LED driver channel is not enabled, it can be used for other functions.
3:1	-	R	Reserved
0	LED_RUN	RW	LED Run Enable Control 0: Disable. 1: Enable.

28.4.2. LED Cycle Register (LED_CYC)

Address offset: 0x04

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	LED_CYC	RW	The LED_CYC register determines the time interval to drive two LEDs, and the scan cycle time of the whole 56-segment LED is: T= 56 * ∆t.

28.4.3. LED Display Time control register (LED_ECO)

Address offset: 0x08

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:16	-	R	Reserved
15:0	LED_ECO	RW	The LED_ECO register determines the time (brightness) to drive the LEDs, and LED_ECO must be less than LED_CYC

28.4.4. LED segment High Register (LED_SEGH)

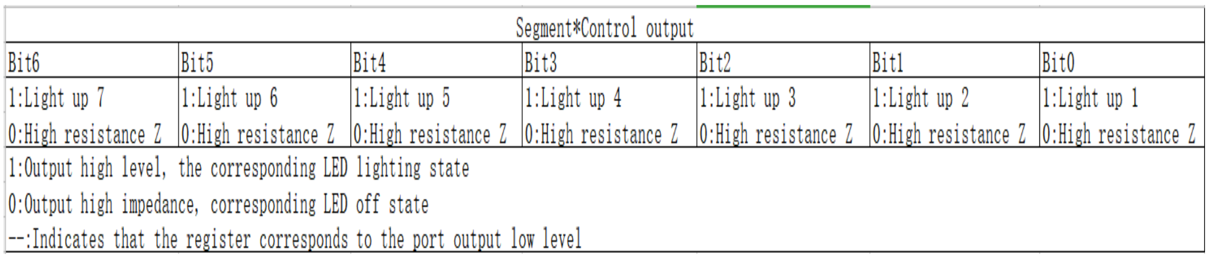
Address offset: 0x08

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description

Bits	Fields	R/W	Description
31:24	LED_SEGD7	RW	Segment 7 Control output 1: Drive high level 0: Drive high impedance note: bit 31 must be 0
23:16	LED_SEGD6	RW	Segment 6 Control output 1: Drive high level 0: Drive high impedance note: bit 22 must be 0
15:8	LED_SEGD5	RW	Segment 5 Control output 1: Drive high level 0: Drive high impedance note: bit 13 must be 0
7:0	LED_SEGD4	RW	Segment 4 Control output 1: Drive high level 0: Drive high impedance note: bit 4 must be 0

Figure 28-2. LED Segment drive control



Segment*Control output						
Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1:Light up 7 0:High resistance Z	1:Light up 6 0:High resistance Z	1:Light up 5 0:High resistance Z	1:Light up 4 0:High resistance Z	1:Light up 3 0:High resistance Z	1:Light up 2 0:High resistance Z	1:Light up 1 0:High resistance Z
1:Output high level, the corresponding LED lighting state 0:Output high impedance, corresponding LED off state --:Indicates that the register corresponds to the port output low level						

28.4.5. LED segment Low Register (LED_SEGL)

Address offset: 0x10

Reset Value: 0x0000 0000

Bits	Fields	R/W	Description
31:24	LED_SEGD3	RW	Segment 3 Control output 1: Drive high level 0: Drive high impedance note: bit 27 must be 0。
23:16	LED_SEGD2	RW	Segment 2 Control output 1: Drive high level 0: Drive high impedance note: bit 18 must be 0。
15:8	LED_SEGD1	RW	Segment 1 Control output 1: Drive high level 0: Drive high impedance note: bit 9 must be
7:0	LED_SEGD0	RW	Segment 0 Control output 1: Drive high level 0: Drive high impedance note: bit 0 must be 0

29. Device electronic signature

29.1. Overview

This Section applies to the whole MG32F10xx family, unless otherwise specified.

The electronic signature is stored in the System memory area in the Flash memory module, and can be read using the SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the MG32F10xx microcontroller.

29.2. Memory size registers

Flash size register

Base address: 0x4001 6404

Read only = 0xXXXX where X is factory-programmed

Please refer to section 3.3.2 in SYS for detail

29.3. Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Address: 0x1FFFF7E8

Read only = 0xXXXX where X is factory-programmed

Bits	Fields	Descriptions
95:0	U_ID[95:0]	Unique ID bits.

29.4. MCU device ID register (MCU_ID)

Address: 0x4001 6400

Read only = 0xXXXX where X is factory-programmed

Please refer to section 3.3.1 in SYS for detail

30. Debug (DBG)

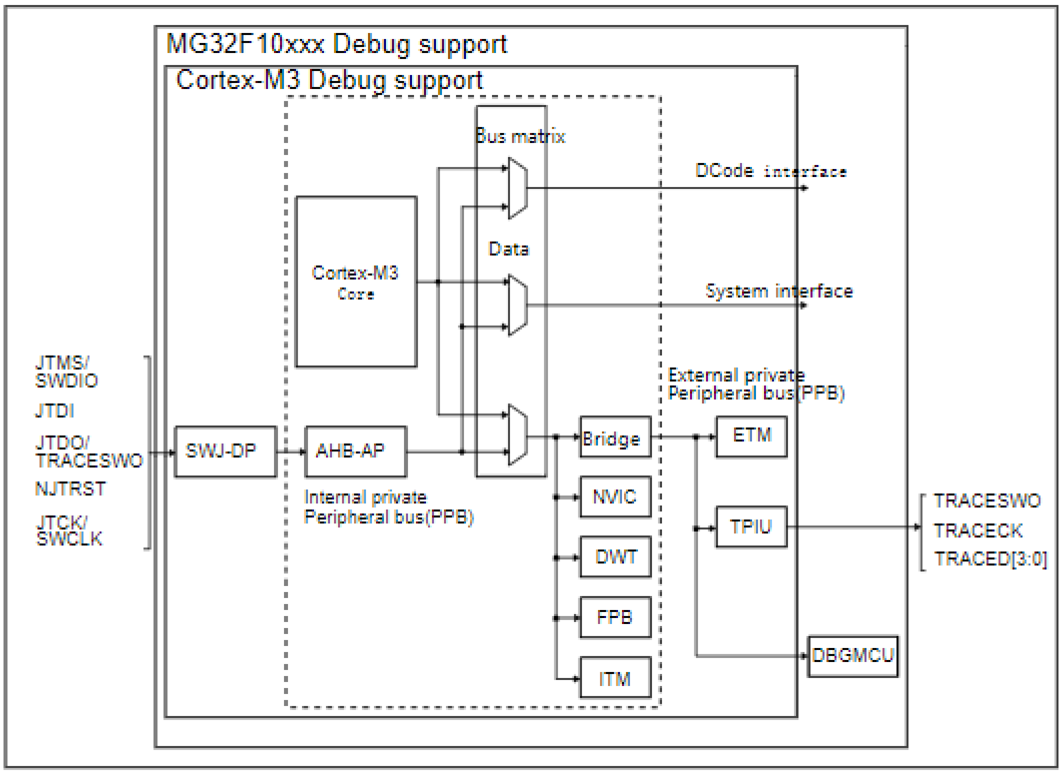
30.1. Overview

The MG32F10xx series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the ARM CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the ARM Cortex-M3. The debug system supports serial wire debug (SWD) and trace functions in addition to standard JTAG debug. The debug and trace functions refer to the following documents:

- Cortex-M3 Technical Reference Manual
- ARM Debug Interface v5 Architecture Specification

The DBG unit helps debugger to debug power saving mode, TIMER, WWDG, IWDG. When corresponding bit is set, provide clock when in power saving mode or hold the state for TIMER, WWDG, IWDG.

Figure 30-1. MG32F10xx debugging block diagram



30.2. JTAG/SW function overview

Debug capabilities can be accessed by a debug tool via Serial Wire (SW - Debug Port) or JTAG interface (JTAG - Debug Port).

30.2.1. Switch JTAG or SW interface

By default, the JTAG interface is active. The sequence for switching from JTAG to SWD is:

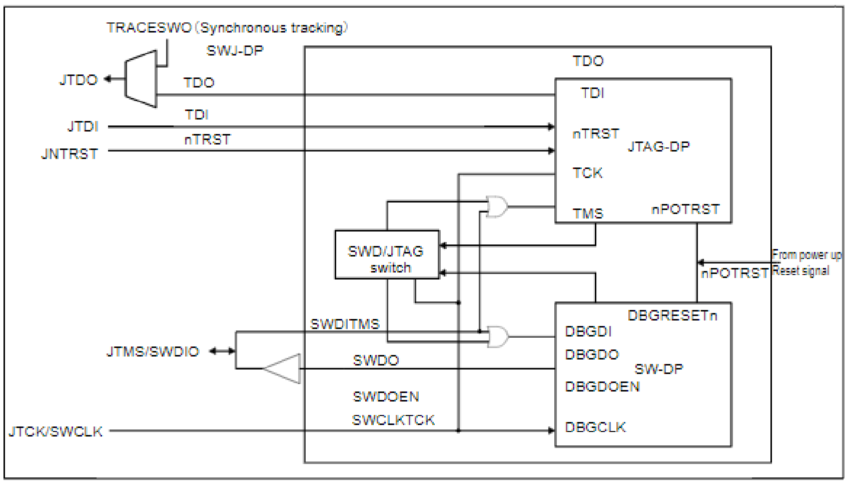
- Send 50 or more TCK cycles with TMS = 1.
- Send the 16-bit sequence on TMS = 1110011110011110 (0xE79E LSB first).
- Send 50 or more TCK cycles with TMS = 1.

The sequence for switching from SWD to JTAG is:

Send 50 or more TCK cycles with TMS = 1.

- Send the 16-bit sequence on TMS = 1110011100111100 (0xE73C LSB first).
- Send 50 or more TCK cycles with TMS = 1.

Figure 30-2. MG32F10xx only support debug with SWD interface.



30.2.2. Pin assignment

The serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK). The pin assignment are:

PA14: JTCK/SWCLK

PA13: JTMS/SWDIO

PB3: TRACESWO

By default, 2-pin SWD debug mode is chosen after reset. If switch to other function with the GPIOx_MODER, the PA14/PA13/PB3 are released to other GPIO functions.

30.2.3. JEDEC-106 ID code

The Cortex-M3 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00F_F000_0xE00F_FFFF.

30.3. Debug hold function overview

30.3.1. Debug support for power saving mode

When DBG_STANDBY bit in DBG control register (DBGMCU_CR) is set and entering the standby mode, the clock of AHB bus and system clock are provided by HSI, and the debugger can debug in standby mode. When exit the standby mode, a system reset generated.

When DBG_STOP bit in DBG control register (DBGMCU_CR) is set and entering the Deep-sleep mode, the clock of AHB bus and system clock are provided by HSI, and the debugger can debug in Deep-sleep mode.

When DBG_SLEEP bit in DBG control register (DBGMCU_CR) is set and entering the sleep mode, the clock of AHB bus for CPU is not closed, and the debugger can debug in sleep mode.

30.3.2. Debug support for TIMER, WWDG, IWDG

When the core halted and the corresponding bit in DBG control register (DBGMCU_CR) is set, the following behaved.

For TIMER, the timer counters stopped and hold for debug.

For WWDG or IWDG, the counter clock stopped for debug.

30.4. Register definition

30.4.1. ID code register (DBG_ID)

Address: 0xE004 2000

Read only

Bits	Fields	RW	Descriptions
31:0	ID_CODE[31:0]	R	DBG ID code register These bits read by software, These bits are unchanged constant

30.4.2. Control register (DBGMCU_CR)

Address offset: 0x04

Reset value: 0x0000 0000

Power On reset only

Bits	Fields	RW	Descriptions
31:14	Reserved	R	Must be kept at reset value
13	DBG_TIM4_STOP	RW	TIMER 4 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 4 counter for debug when core halted
12	DBG_TIM3_STOP	RW	TIMER 3 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 3 counter for debug when core halted
11	DBG_TIM2_STOP	RW	TIMER 2 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 2 counter for debug when core halted

Bits	Fields	RW	Descriptions
10	DBG_TIM1_STOP	RW	TIMER 1 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 1 counter for debug when core halted
9	DBG_WWDG_STOP	RW	WWDG hold bit This bit is set and reset by software 0: no effect 1: hold the WWDG counter clock for debug when core halted
8	DBG_IWDG_STOP	RW	IWDG hold bit This bit is set and reset by software 0: no effect 1: hold the IWDG counter clock for debug when core halted
7:6	Reserved	R	Must be kept at reset value
5	TRACE_IOEN	RW	Trace pin allocation enable This bit is set and reset by software 0: Trace pin allocation disable 1: Trace pin allocation enable
4:3	Reserved	R	Must be kept at reset value
2	DBG_STANDBY	RW	Standby mode hold register This bit is set and reset by software 0: no effect 1: At the standby mode, the clock of AHB bus and system clock are provided by HSI, a system reset generated when exit standby mode
1	DBG_STOP	RW	Deep-sleep mode hold register This bit is set and reset by software 0: no effect 1: At the Deep-sleep mode, the clock of AHB bus and system clock are provided by HSI
0	DBG_SLEEP	RW	Sleep mode hold register This bit is set and reset by software 0: no effect 1: At the sleep mode, the clock of AHB is on.