GigaDevice Semiconductor Inc.

GD32F10x

ARM® Cortex™-M3 32-bit MCU

User Manual

Revision 2.3

(Mar. 2020)

Table of Contents

List of Figures

List of Tables

1. System and memory architecture

The devices of GD32F10x 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, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

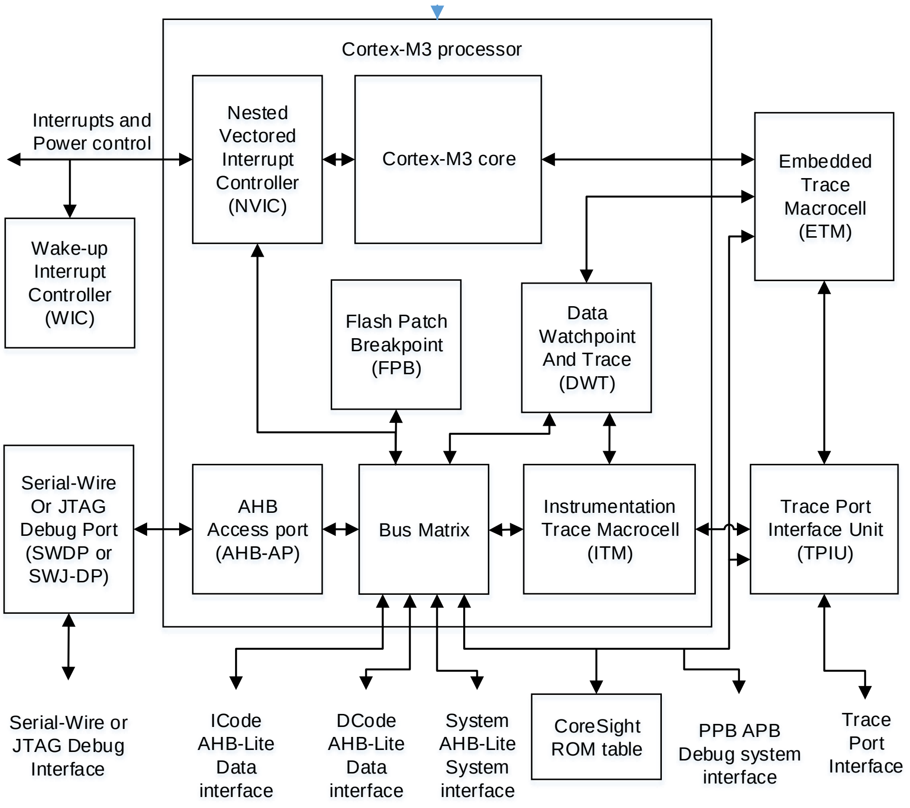
1.1. ARM Cortex-M3 processor

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 microcontrollers 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 accesses.
- Nested Vectored Interrupt Controller (NVIC).
- Flash Patch and Breakpoint (FPB).
- Data Watchpoint and Trace (DWT).
- Instrumentation Trace Macrocell (ITM).
- Embedded Trace Macrocell (ETM).
- Serial Wire JTAG Debug Port (SWJ-DP).
- Trace Port Interface Unit (TPIU).
- Memory Protection Unit (MPU).

Figure 1-1. The structure of the Cortex™-M3 processor shows the Cortex™-M3 processor block diagram. For more information, refer to the ARM® Cortex™-M3 Technical Reference Manual.

Figure 1-1. 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

Note: The functions of ETM and MPU do not exist in the GD32F101xx and GD32F103xx microcontrollers with the flash memory density less than 256 Kbytes.

1.2. System architecture

A 32-bit multilayer bus is implemented in the GD32F10x devices, which enables parallel access paths between multiple masters and slaves in the system. The multilayer 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 1-1. The interconnection relationship of the AHB interconnect matrix

	IBUS	DBUS	SBUS	DMA0	DMA1	ENET
FMC-I	1					
FMC-D		1		1	1	
SRAM	1	1	1	1	1	1
EXMC	1	1	1	1	1	1

	IBUS	DBUS	SBUS	DMA0	DMA1	ENET
AHB			1	1	1	
APB1			1	1	1	
APB2			1	1	1	

As is shown above, there are several masters connected with the AHB interconnect matrix, including IBUS, DBUS, SBUS, DMA0, DMA1 and ENET. 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. DMA0 and DMA1 are the buses of DMA0 and DMA1 respectively. ENET is the Ethernet.

There are also several slaves connected with the AHB interconnect matrix, including FMC-I, FMC-D, SRAM, EXMC, 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. EXMC is the external memory controller. 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 is limited to 54 MHz, APB2 operates at full speed (up to 108MHz depending on the device).

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

Figure 1-2. GD32F10x Medium-density series system architecture

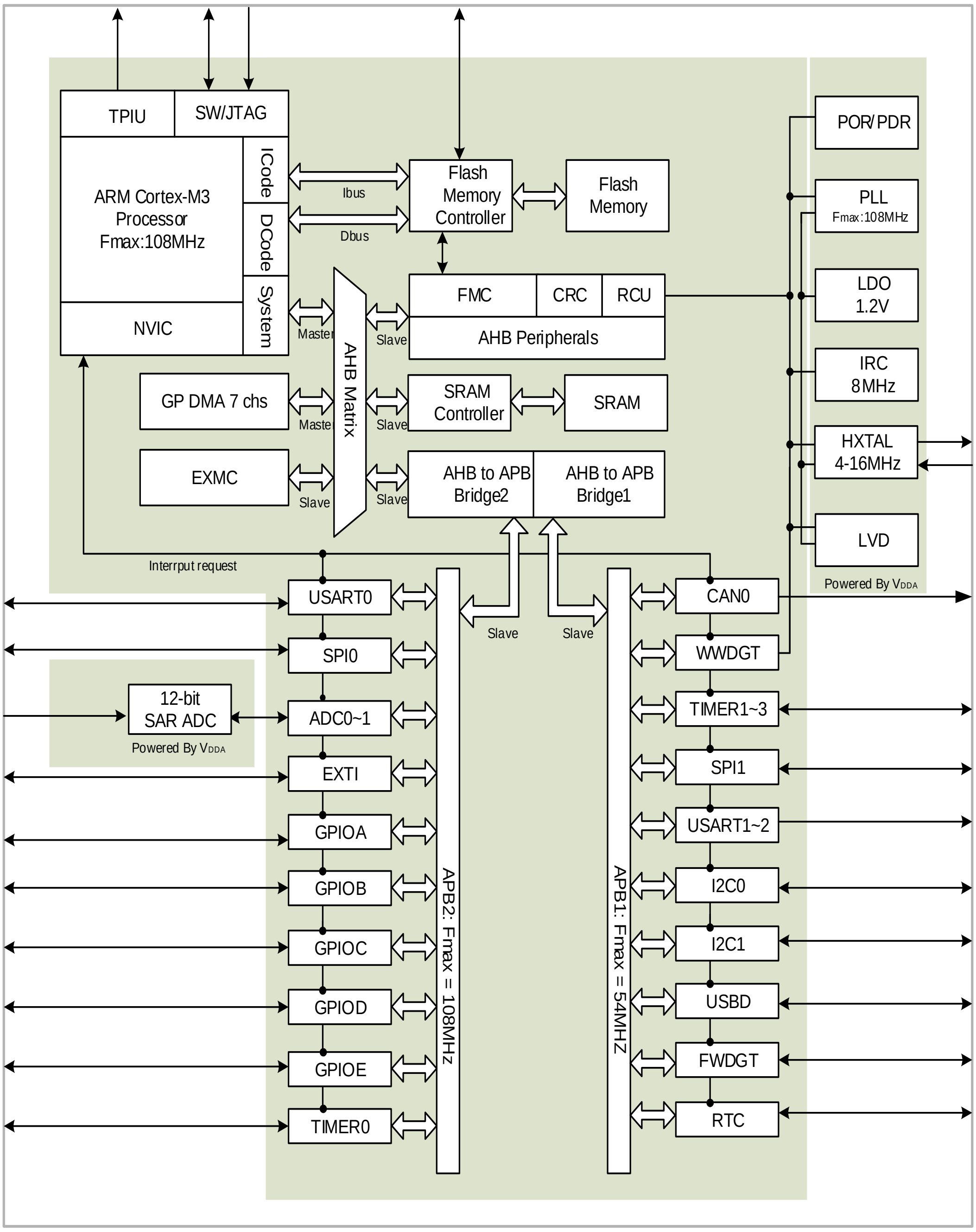


Figure 1-3. GD32F10x High-density series system architecture

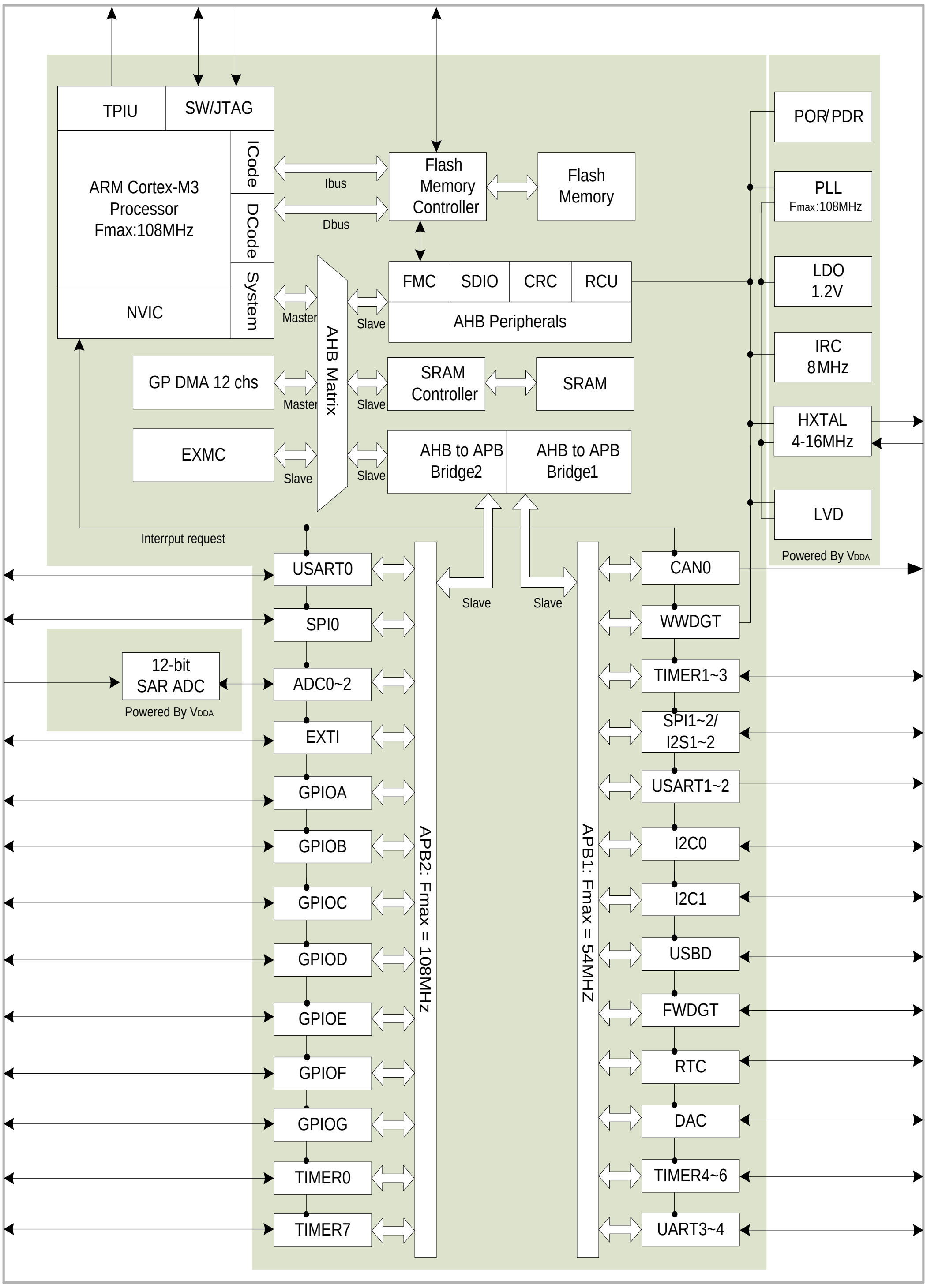


Figure 1-4. GD32F10x Extra-density series system architecture

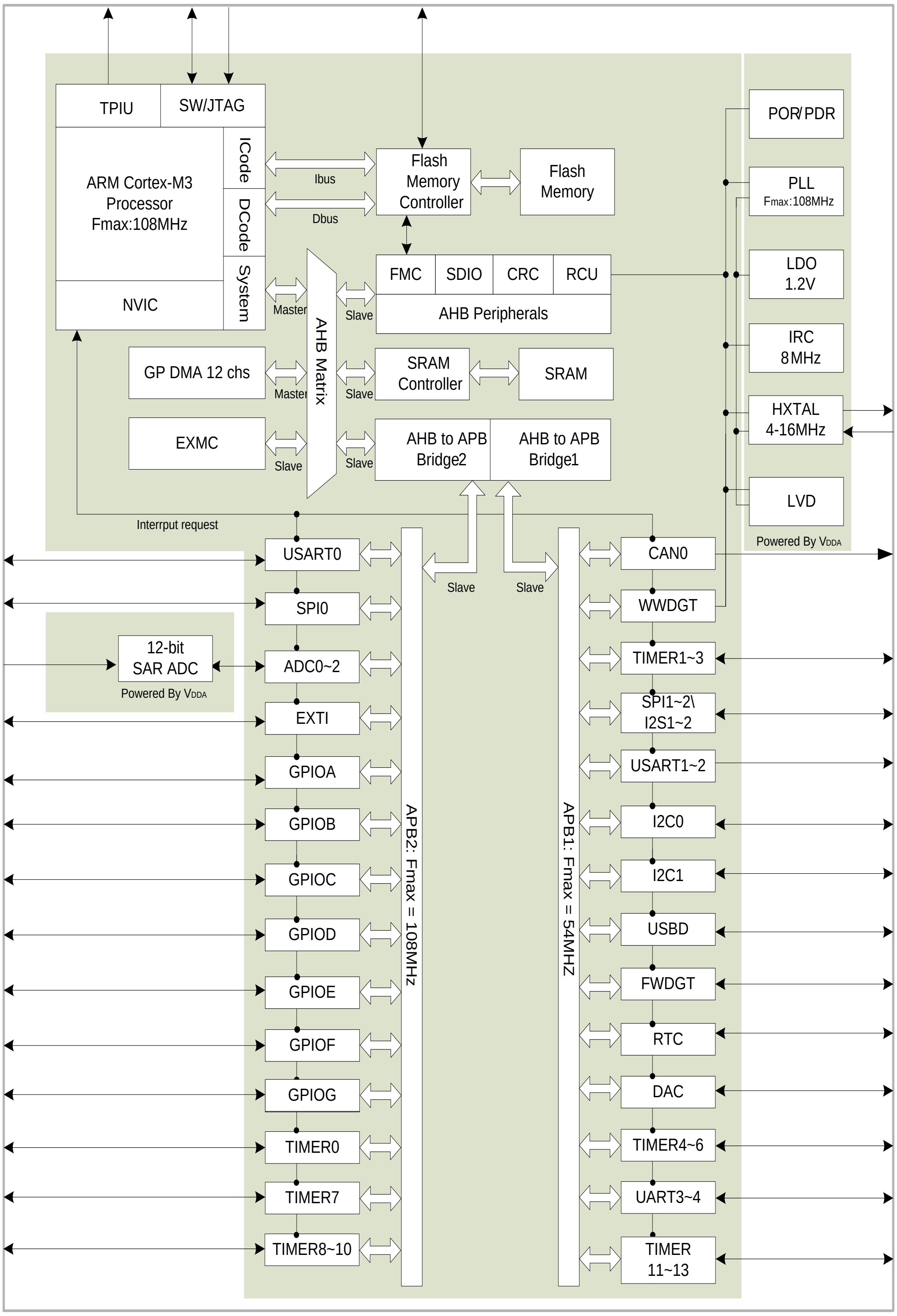
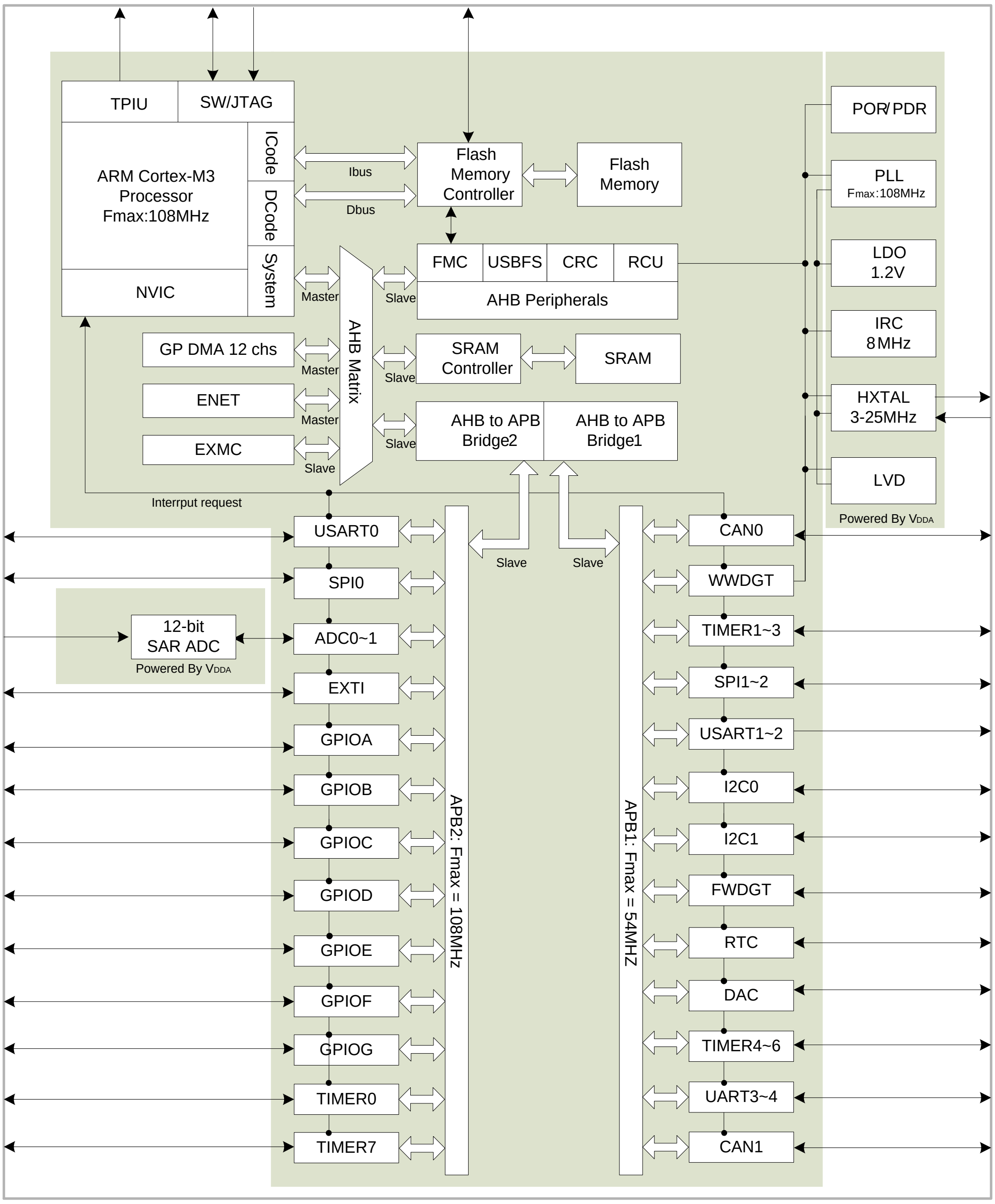


Figure 1-5. GD32F10x Connectivity line series system architecture



1.3. 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 1-2. Memory map of GD32F10x devices*** shows the memory map of the GD32F10x 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 1-2. Memory map of GD32F10x devices

Pre-defined Regions	Bus	Address	Peripherals
External device	AHB	0xA000 0000 - 0xA000 0FFF	EXMC - SWREG
External RAM		0x9000 0000 - 0x9FFF FFFF	EXMC - PC CARD
		0x7000 0000 - 0x8FFF FFFF	EXMC - NAND
		0x6000 0000 - 0x6FFF FFFF	EXMC - NOR/PSRAM/SRAM
Peripheral	AHB	0x5000 0000 - 0x5003 FFFF	USBFS
		0x4008 0000 - 0x4FFF FFFF	Reserved
		0x4004 0000 - 0x4007 FFFF	Reserved
		0x4002 BC00 - 0x4003 FFFF	Reserved
		0x4002 B000 - 0x4002 BBFF	Reserved
		0x4002 A000 - 0x4002 AFFF	Reserved
		0x4002 8000 - 0x4002 9FFF	ENET
		0x4002 6800 - 0x4002 7FFF	Reserved
		0x4002 6400 - 0x4002 67FF	Reserved
		0x4002 6000 - 0x4002 63FF	Reserved
		0x4002 5000 - 0x4002 5FFF	Reserved
		0x4002 4000 - 0x4002 4FFF	Reserved
		0x4002 3C00 - 0x4002 3FFF	Reserved
		0x4002 3800 - 0x4002 3BFF	Reserved
		0x4002 3400 - 0x4002 37FF	Reserved
		0x4002 3000 - 0x4002 33FF	CRC
		0x4002 2C00 - 0x4002 2FFF	Reserved
		0x4002 2800 - 0x4002 2BFF	Reserved
		0x4002 2400 - 0x4002 27FF	Reserved
		0x4002 2000 - 0x4002 23FF	FMC
		0x4002 1C00 - 0x4002 1FFF	Reserved
		0x4002 1800 - 0x4002 1BFF	Reserved
		0x4002 1400 - 0x4002 17FF	Reserved
		0x4002 1000 - 0x4002 13FF	RCU
		0x4002 0C00 - 0x4002 0FFF	Reserved

Pre-defined Regions	Bus	Address	Peripherals
		0x4002 0800 - 0x4002 0BFF	Reserved
		0x4002 0400 - 0x4002 07FF	DMA1
		0x4002 0000 - 0x4002 03FF	DMA0
		0x4001 8400 - 0x4001 FFFF	Reserved
		0x4001 8000 - 0x4001 83FF	SDIO
	APB2	0x4001 7C00 - 0x4001 7FFF	Reserved
		0x4001 7800 - 0x4001 7BFF	Reserved
		0x4001 7400 - 0x4001 77FF	Reserved
		0x4001 7000 - 0x4001 73FF	Reserved
		0x4001 6C00 - 0x4001 6FFF	Reserved
		0x4001 6800 - 0x4001 6BFF	Reserved
		0x4001 5C00 - 0x4001 67FF	Reserved
		0x4001 5800 - 0x4001 5BFF	Reserved
		0x4001 5400 - 0x4001 57FF	TIMER10
		0x4001 5000 - 0x4001 53FF	TIMER9
		0x4001 4C00 - 0x4001 4FFF	TIMER8
		0x4001 4800 - 0x4001 4BFF	Reserved
		0x4001 4400 - 0x4001 47FF	Reserved
		0x4001 4000 - 0x4001 43FF	Reserved
		0x4001 3C00 - 0x4001 3FFF	ADC2
		0x4001 3800 - 0x4001 3BFF	USART0
		0x4001 3400 - 0x4001 37FF	TIMER7
		0x4001 3000 - 0x4001 33FF	SPI0
		0x4001 2C00 - 0x4001 2FFF	TIMER0
		0x4001 2800 - 0x4001 2BFF	ADC1
		0x4001 2400 - 0x4001 27FF	ADC0
		0x4001 2000 - 0x4001 23FF	GPIOG
		0x4001 1C00 - 0x4001 1FFF	GPIOF
		0x4001 1800 - 0x4001 1BFF	GPIOE
		0x4001 1400 - 0x4001 17FF	GPIOD
		0x4001 1000 - 0x4001 13FF	GPIOC
		0x4001 0C00 - 0x4001 0FFF	GPIOB
		0x4001 0800 - 0x4001 0BFF	GPIOA
		0x4001 0400 - 0x4001 07FF	EXTI
		0x4001 0000 - 0x4001 03FF	AFIO
	APB1	0x4000 CC00 - 0x4000 FFFF	Reserved
		0x4000 C800 - 0x4000 CBFF	Reserved
		0x4000 C400 - 0x4000 C7FF	Reserved
		0x4000 C000 - 0x4000 C3FF	Reserved
		0x4000 8000 - 0x4000 BFFF	Reserved

Pre-defined Regions	Bus	Address	Peripherals
		0x4000 7C00 - 0x4000 7FFF	Reserved
		0x4000 7800 - 0x4000 7BFF	Reserved
		0x4000 7400 - 0x4000 77FF	DAC
		0x4000 7000 - 0x4000 73FF	PMU
		0x4000 6C00 - 0x4000 6FFF	BKP
		0x4000 6800 - 0x4000 6BFF	CAN1
		0x4000 6400 - 0x4000 67FF	CAN0
		0x4000 6000 - 0x4000 63FF	Shared USBD/CAN SRAM 512 bytes
		0x4000 5C00 - 0x4000 5FFF	USBD
		0x4000 5800 - 0x4000 5BFF	I2C1
		0x4000 5400 - 0x4000 57FF	I2C0
		0x4000 5000 - 0x4000 53FF	UART4
		0x4000 4C00 - 0x4000 4FFF	UART3
		0x4000 4800 - 0x4000 4BFF	USART2
		0x4000 4400 - 0x4000 47FF	USART1
		0x4000 4000 - 0x4000 43FF	Reserved
		0x4000 3C00 - 0x4000 3FFF	SPI2/I2S2
		0x4000 3800 - 0x4000 3BFF	SPI1/I2S1
		0x4000 3400 - 0x4000 37FF	Reserved
		0x4000 3000 - 0x4000 33FF	FWDGT
		0x4000 2C00 - 0x4000 2FFF	WWDGT
		0x4000 2800 - 0x4000 2BFF	RTC
		0x4000 2400 - 0x4000 27FF	Reserved
		0x4000 2000 - 0x4000 23FF	TIMER13
		0x4000 1C00 - 0x4000 1FFF	TIMER12
		0x4000 1800 - 0x4000 1BFF	TIMER11
		0x4000 1400 - 0x4000 17FF	TIMER6
		0x4000 1000 - 0x4000 13FF	TIMER5
		0x4000 0C00 - 0x4000 0FFF	TIMER4
		0x4000 0800 - 0x4000 0BFF	TIMER3
		0x4000 0400 - 0x4000 07FF	TIMER2
		0x4000 0000 - 0x4000 03FF	TIMER1
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 8000 - 0x2001 BFFF	Reserved
		0x2000 5000 - 0x2001 7FFF	SRAM

Pre-defined Regions	Bus	Address	Peripherals
		0x2000 0000 - 0x2000 4FFF	
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option Bytes
		0x1FFF F000 - 0x1FFF F7FF	Boot loader
		0x1FFF E000 - 0x1FFF EFFF	
		0x1FFF B000 - 0x1FFF DFFF	
		0x1FFF 7A10 - 0x1FFF AFFF	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 - 0x082F FFFF	Main Flash
		0x0030 0000 - 0x07FF FFFF	Reserved
		0x0010 0000 - 0x002F FFFF	Aliased to Main Flash or Boot loader
		0x0002 0000 - 0x000F FFFF	
		0x0000 0000 - 0x0001 FFFF	

1.3.1. 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:

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4) \qquad (1\text{-}1)$$

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:

bit_word_addr = 0x2200 0000 + (0x200 * 32)+ (7 * 4)= 0x2200 401C (1-2)

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.

1.3.2. On-chip SRAM memory

The GD32F10x series of devices contain up to 96 KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses.

1.3.3. On-chip flash memory overview

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

- Up to 3072KB of main flash memory.
- Up to 18KB of information blocks for the boot loader.
- Option bytes to configure the device.

Refer to ***Flash memory controller (FMC)*** Chapter for more details.

1.4. Boot configuration

The GD32F10x 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 CK_SYS 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 1-3. 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 F000) 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 GD32F10x devices, the boot loader can be activated through the USART0 interface.

1.5. Device electronic signature

Medium-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density ranges from 16 to 128 Kbytes.
High-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density ranges from 256 to 512 Kbytes.
Extra-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density larger than 512 Kbytes.
Connectivity line devices are GD32F105xx and GD32F107xx microcontrollers.

The device electronic signature contains memory size information and the 96-bit unique device ID. It is stored in the information block of the Flash memory. The 96-bit unique device ID is unique for any device. It can be used as serial numbers, or part of security keys, etc.

1.5.1. Memory density information

Base address: 0x1FFF F7E0

The value is factory programmed and can never be altered by user.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM_DENSITY[15:0]															
r															

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

Bits	Fields	Descriptions
31:16	SRAM_DENSITY [15:0]	SRAM density The value indicates the on-chip SRAM density of the device in Kbytes. Example: 0x0008 indicates 8 Kbytes.
15:0	FLASH_DENSITY [15:0]	Flash memory density The value indicates the Flash memory density of the device in Kbytes. Example: 0x0020 indicates 32 Kbytes.

1.5.2. Unique device ID (96 bits)

Base address: 0x1FFF F7E8

The value is factory programmed and can never be altered by user.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNIQUE_ID[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	UNIQUE_ID[31:0]	Unique device ID

Base address: 0x1FFF F7EC

The value is factory programmed and can never be altered by user.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNIQUE_ID[63:48]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNIQUE_ID[47:32]															
r															

Bits	Fields	Descriptions
31:0	UNIQUE_ID[63:32]	Unique device ID

Base address: 0x1FFF F7F0

The value is factory programmed and can never be altered by user.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNIQUE_ID[95:80]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNIQUE_ID[79:64]															
r															

Bits	Fields	Descriptions
31:0	UNIQUE_ID[95:64]	Unique device ID

1.6. System configuration registers

Base address: 0x4002 103C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEE	Reserved						
								rw							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	CEE	Code execution efficiency 0：Default code execution efficiency 1：Code execution efficiency enhancement
6:0	Reserved	Must be kept at reset value

Note:

1. Only bit[7] can be read-modify-write, other bits are not permitted.
2. Only GD32F10xC/D/E/F/G/I/K can be configured as code execution efficiency enhancement mode.

2. Flash memory controller (FMC)

2.1. Overview

The flash memory controller, FMC, provides all the necessary functions for the on-chip flash memory. There is no waiting time while CPU executes instructions stored in the first 256K bytes of the flash. It also provides page erase, mass erase, and word/half-word program operations for flash memory.

2.2. Characteristics

- Up to 3072KB of on-chip flash memory for instruction and data.
- No waiting time within first 256K bytes when CPU executes instructions. A long delay when CPU fetches the instructions out of the range.
- 2 banks adopted for GD32F10x_CL with flash more than 512KB and GD32F10x_XD. Bank0 is used for the first 512KB and bank1 is for the rest capacity.
- Only bank0 is adopted for GD32F10x_CL with flash no more than 512KB and GD32F10x_HD.
- The flash page size is 1KB for GD32F10x_MD, for other series, the page size is 2KB for bank0, 4KB for bank1.
- Word/half-word programming, page erase and mass erase operation.
- 16B option bytes block for user application requirements.
- Option bytes are uploaded to the option byte control registers on every system reset.
- Flash security protection to prevent illegal code/data access.
- Page erase/program protection to prevent unexpected operation.

2.3. Function overview

2.3.1. Flash memory architecture

For GD32F10x_MD, the page size is 1KB. For GD32F10x_CL with flash no more than 512KB and GD32F10x_HD, the page size is 2KB. For GD32F10x_CL and GD32F10x_XD, bank0 is used for the first 512KB where the page size is 2KB. Bank1 is used for the rest capacity where the page size is 4KB. Each page can be erased individually.

The following table shows the details of flash organization.

Table 2-1. GD32F10x_MD

Block	Name	Address Range	size (bytes)
Main Flash Block	Page 0	0x0800 0000 - 0x0800 03FF	1KB

Block	Name	Address Range	size (bytes)
	Page 1	0x0800 0400 - 0x0800 07FF	1KB
	Page 2	0x0800 0800 - 0x0800 0BFF	1KB
	…	…	…
	Page 127	0x0801 FC00 - 0x0801 FFFF	1KB
Information Block	Boot loader area	0x1FFF F000- 0x1FFF F7FF	2KB
Option bytes Block	Option bytes	0x1FFF F800 - 0x1FFF F80F	16B

Table 2-2. GD32F10x_CL and GD32F10x_HD, GD32F10x_XD

Block		Name	Address Range	size (bytes)
Main Flash Block		Page 0	0x0800 0000 - 0x0800 07FF	2KB
		Page 1	0x0800 0800 - 0x0800 0FFF	2KB
		Page 2	0x0800 1000 - 0x0800 17FF	2KB
		…	…	…
		Page 255	0x0807 F800 - 0x0807 FFFF	2KB
		Page 256	0x0808 0000 - 0x0808 0FFF	4KB
		Page 257	0x0808 1000 - 0x0808 1FFF	4KB
		…	…	…
		Page 895	0x082F F000 - 0x082F FFFF	4KB
Information Block	GD32F10x_HD	Boot loader area	0x1FFF F000- 0x1FFF F7FF	2KB
	GD32F10x_XD		0x1FFF E000- 0x1FFF F7FF	6KB
	GD32F10x_CL		0x1FFF B000- 0x1FFF F7FF	18KB
Option bytes Block		Option bytes	0x1FFF F800 - 0x1FFF F80F	16B

Note: The Information Block stores the boot loader. This block cannot be programmed or erased by user.

2.3.2. Read operations

The flash can be addressed directly as a common memory space. Any instruction fetch and the data access from the flash are through the IBUS or DBUS from the CPU.

2.3.3. Unlock the FMC_CTLx registers

After reset, the FMC_CTLx registers are not accessible in write mode, and the LK bit in FMC_CTLx register is 1. An unlocking sequence consists of two write operations to the FMC_KEY0 register to open the access to the FMC_CTL0 register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC_KEY0 register. After the two write operations, the LK bit in FMC_CTL0 register is reset to 0 by hardware. The software can lock the FMC_CTL0 again by setting the LK bit in FMC_CTL0 register to 1. Any wrong operations to the FMC_KEY0 will set the LK bit to 1, and lock FMC_CTL0 register, and lead to a bus error.

The OBPG bit and OBER bit in FMC_CTL0 are still protected even the FMC_CTL0 is unlocked. The unlocking sequence is two write operations, which are writing 0x45670123 and 0xCDEF89AB to FMC_OBKEY register. And then the hardware sets the OBWEN bit in FMC_CTL0 register to 1. The software can reset OBWEN bit to 0 to protect the OBPG bit and OBER bit in FMC_CTL0 register again.

For the GD32F10x_CL and GD32F10x_XD, the FMC_CTL0 register is used to configure the operations to bank0 and the option bytes block, while FMC_CTL1 register is used to configure the program and erase operations to bank1. The lock/unlock mechanism of FMC_CTL1 register is similar to FMC_CTL0 register. The unlock sequence should be written to FMC_KEY1 when unlocking FMC_CTL1.

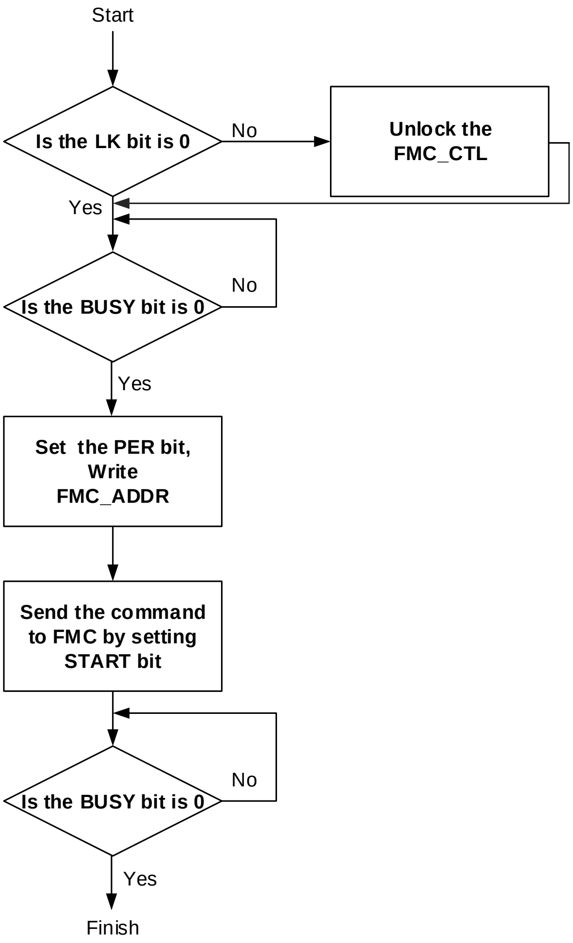
2.3.4. Page erase

The FMC provides a page erase function which is used to initialize the contents of a main flash memory page to a high state. Each page can be erased independently without affecting the contents of other pages. The following steps show the access sequence of the registers for a page erase operation.

- Unlock the FMC_CTLx registers if necessary.
- Check the BUSY bit in FMC_STATx registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set the PER bit in FMC_CTLx registers.
- Write the page absolute address (0x08XX XXXX) into the FMC_ADDRx registers.
- Send the page erase command to the FMC by setting the START bit in FMC_CTLx registers.
- Wait until all the operations have finished by checking the value of the BUSY bit in FMC_STATx registers.
- Read and verify the page if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTLx registers is set. Note that a correct target page address must be confirmed. Or the software may run out of control if the target erase page is being used to fetch codes or to access data. The FMC will not provide any notification when this occurs. Additionally, the page erase operation will be ignored on erase/program protected pages. In this condition, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTLx registers is set. The software can check the WPERR bit in the FMC_STATx registers to detect this condition in the interrupt handler. The following figure shows the page erase operation flow.

Figure 2-1. Process of page erase operation



For the GD32F10x_CL and GD32F10x_XD, FMC_STAT0 reflects the operation status of bank0, and FMC_ STAT1 reflects the operation status of bank1. The page erase procedure applied to bank1 is similar to the procedure applied to bank0. Especially, when erasing page in bank1 under security protection, the address should not only be written to FMC_ADRR1 but also to FMC_ADDR0.

2.3.5. Mass erase

The FMC provides a complete erase function which is used to initialize the main flash block contents. This erase can affect only on bank0 by setting MER bit to 1 in the FMC_CTL0 register, or only on bank1 by setting MER bit to 1 in the FMC_CTL1 register, or on entire flash by setting MER bits to 1 in FMC_CTL0 register and FMC_CTL1 register. The following steps show the mass erase register access sequence.

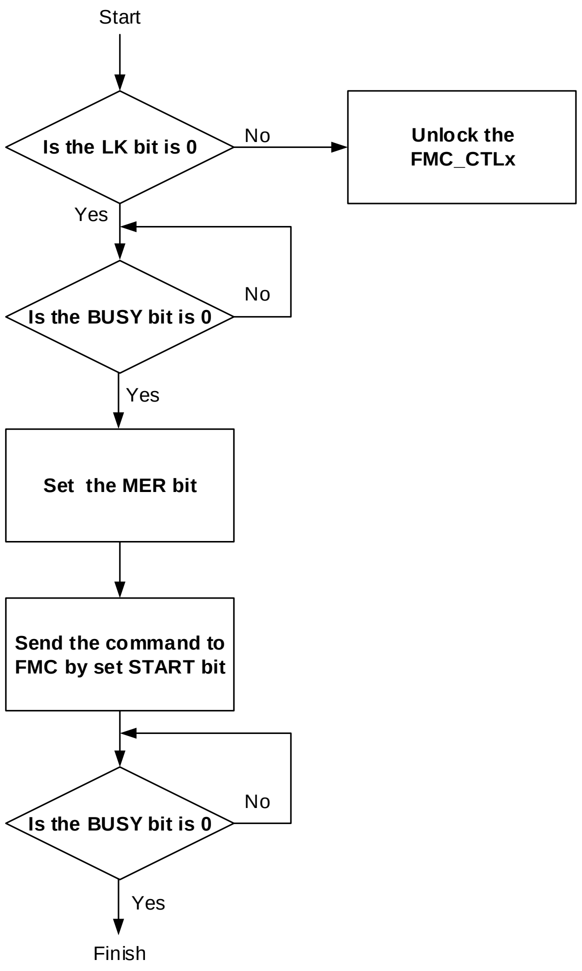
- Unlock the FMC_CTLx registers if necessary.
- Check the BUSY bit in FMC_STATx registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set MER bit in FMC_CTL0 register if erase bank0 only. Set MER bit in FMC_CTL1 register if erase bank1 only. Set MER bits in FMC_CTL0 register and FMC_CTL1 register if erase entire flash.
- Send the mass erase command to the FMC by setting the START bit in FMC_CTLx registers.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STATx registers.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTLx registers is set. Since all flash data will be modified to a value of 0xFFFF_FFFF, the mass erase operation can be implemented using a program that runs in SRAM or by using the debugging tool that accesses the FMC registers directly.

For the GD32F10x_CL and GD32F10x_XD, the mass erase procedure applied to bank1 is similar to the procedure applied to bank0.

The following figure indicates the mass erase operation flow.

Figure 2-2. Process of mass erase operation



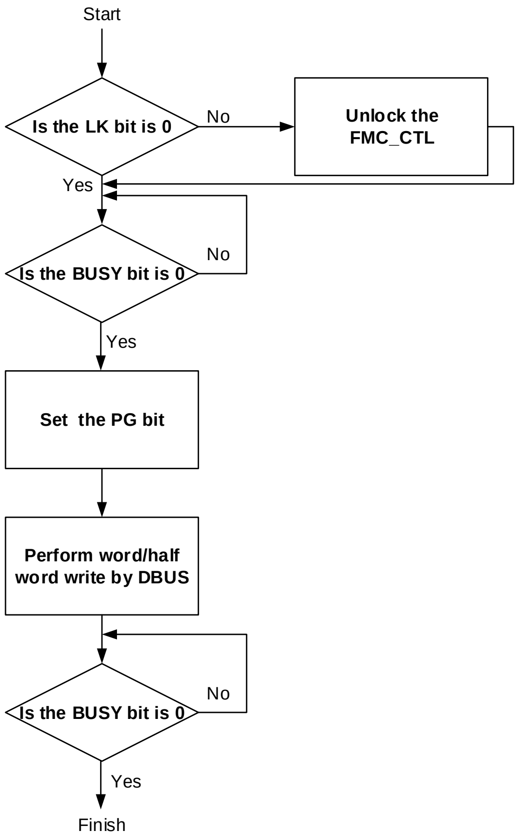
2.3.6. Main flash programming

The FMC provides a 32-bit word/16-bit half word programming function which is used to modify the main flash memory contents. The following steps show the register access sequence of the word programming operation.

- Unlock the FMC_CTLx registers if necessary.
- Check the BUSY bit in FMC_STATx registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set the PG bit in FMC_CTLx registers.
- Write a 32-bit word/16-bit half word to desired absolute address (0x08XX XXXX) by DBUS.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STATx registers.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTLx registers is set. Note that the word/half word programming operation checks the address if it has been erased. If the address has not been erased, PGERR bit in the FMC_STATx registers will be set when programming the address except 0x0. Note that the PG bit must be set before the word/half word programming operation. Additionally, the program operation will be ignored on erase/program protected pages and WPERR bit in FMC_STATx is set. In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTLx registers is set. The software can check the PGERR bit or WPERR bit in the FMC_STATx registers to detect which condition occurred in the interrupt handler. The following figure displays the word programming operation flow.

Figure 2-3. Process of word program operation



For the GD32F10x_CL and GD32F10x_XD, the program procedure applied to bank1 is similar to the procedure applied to bank0.

Note: Reading the flash should be avoided when a program/erase operation is ongoing in the same bank. And flash memory accesses failed if the CPU enters the power saving modes.

2.3.7. Option bytes Erase

The FMC provides an erase function which is used to initialize the option bytes block in flash. The following steps show the erase sequence.

- Unlock the FMC_CTL0 register if necessary.
- Check the BUSY bit in FMC_STAT0 register to confirm that no Flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has finished.
- Unlock the option bytes operation bits in FMC_CTL0 register if necessary.
- Wait until OBWEN bit is set in FMC_CTL0 register.
- Set OBER bit in FMC_CTL0 register.
- Send the option bytes erase command to the FMC by setting the START bit in FMC_CTL0 register.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT0 register.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successful, the ENDF in FMC_STAT0 register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL0 register is set.

2.3.8. Option bytes modify

The FMC provides an erase and then program function which is used to modify the option bytes block in flash. There are 8 pair option bytes. The MSB is the complement of the LSB in each pair. And when the option bytes are modified, the MSB is generated by FMC automatically, not the value of input data. The following steps show the erase sequence.

- Unlock the FMC_CTL0 register if necessary.
- Check the BUSY bit in FMC_STAT0 register to confirm that no Flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Unlock the option bytes operation bits in FMC_CTL0 register if necessary.
- Wait until OBWEN bit is set in FMC_CTL0 register.
- Set the OBPG bit in FMC_CTL0 register.
- A 32-bit word/16-bit half word write at desired address by DBUS.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT0 register.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC_STAT0 register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL0 register is set. Note that the word/half word programming operation checks the address if it has been erased. If the address has not been erased, PGERR bit in the FMC_STAT0 register will set when program the address except programming 0x0.

The modified option bytes only take effect after a system reset is generated.

2.3.9. Option bytes description

The option bytes block is reloaded to FMC_OBSTAT and FMC_WP registers after each system reset, and the option bytes take effect. The complement option bytes are the opposite of option bytes. When option bytes reload, if the complement option byte and option byte do not match, the OBERR bit in FMC_OBSTAT register is set, and the option byte is set to 0xFF. The OBERR bit is not set if both the option byte and its complement byte are 0xFF.The following table is the detail of option bytes.

Table 2-3. Option byte

Address	Name	Description
0x1fff f800	SPC	option byte Security Protection value 0xA5 : no security protection any value except 0xA5 : under security protection
0x1fff f801	SPC_N	SPC complement value
0x1fff f802	USER	[7:4]: reserved [3]: BB 0: boot from bank1 or bank0 if bank1 is void, when configured boot from main memory 1: boot from bank0, when configured boot from main memory [2]: nRST_STDBY 0: generate a reset instead of entering standby mode 1: no reset when entering standby mode [1]: nRST_DPSLP 0: generate a reset instead of entering Deep-sleep mode 1: no reset when entering Deep-sleep mode [0]: nWDG_HW 0: hardware free watchdog 1: software free watchdog
0x1fff f803	USER_N	USER complement value
0x1fff f804	DATA[7:0]	user defined data bit 7 to 0
0x1fff f805	DATA_N[7:0]	DATA complement value bit 7 to 0
0x1fff f806	DATA[15:8]	user defined data bit 15 to 8
0x1fff f807	DATA_N[15:8]	DATA complement value bit 15 to 8
0x1fff f808	WP[7:0]	Page Erase/Program Protection bit 7 to 0 0: protection active 1: unprotected
0x1fff f809	WP_N[7:0]	WP complement value bit 7 to 0
0x1fff f80a	WP[15:8]	Page Erase/Program Protection bit 15 to 8
0x1fff f80b	WP_N[15:8]	WP complement value bit 15 to 8
0x1fff f80c	WP[23:16]	Page Erase/Program Protection bit 23 to 16
0x1fff f80d	WP_N[23:16]	WP complement value bit 23 to 16
0x1fff f80e	WP[31:24]	Page Erase/Program Protection bit 31 to 24 WP[30:24]: Each bit is related to 4KB flash protection, that means 4 pages for GD32F10x_MD and 2 pages for GD32F10x_HD, GD32F10x_XD and GD32F10x_CL. Bit 0

Address	Name	Description
		configures the first 4KB flash protection, and so on. These bits totally controls the first 124KB flash protection. WP[31]: Bit 31 controls the protection of the rest flash memory.
0x1fff f80f	WP_N[31:24]	WP complement value bit 31 to 24

2.3.10. Page erase/program protection

The FMC provides page erase/program protection functions to prevent inadvertent operations on the Flash memory. The page erase or program will not be accepted by the FMC on protected pages. If the page erase or program command is sent to the FMC on a protected page, the WPERR bit in the FMC_STATx registers will then be set by the FMC. If the WPERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the Flash operation error interrupt will be triggered by the FMC to draw the attention of the CPU. The page protection function can be individually enabled by configuring the WP [31:0] bit field to 0 in the option bytes. If a page erase operation is executed on the option bytes block, all the Flash Memory page protection functions will be disabled. When WP in the option bytes is modified, a system reset followed is necessary.

2.3.11. Security protection

The FMC provides a security protection function to prevent illegal code/data access on the Flash memory. This function is useful for protecting the software/firmware from illegal users.

No protection: when setting SPC byte and its complement value to 0x5AA5, no protection performed. The main flash and option bytes block are accessible by all operations.

Under protection: when setting SPC byte and its complement value to any value except 0x5AA5, the security protection is performed. Note that a power reset should be followed instead of a system reset if the SPC modification is performed while the debug module is still connected to JTAG/SWD device. Under the security protection, the main flash can only be accessed by user code and the first 4KB flash is under erase/program protection. In debug mode, boot from SRAM or boot from boot loader mode, all operations to main flash is forbidden. If a read operation to main flash in debug, boot from SRAM or boot from boot loader mode, a bus error will be generated. If a program/erase operation to main flash in debug mode, boot from SRAM or boot from boot loader mode, the WPERR bit in FMC_STATx registers will be set. Option bytes block are accessible by all operations, which can be used to disable the security protection. If program back to no protection level by setting SPC byte and its complement value to 0x5AA5, a mass erase for main flash will be performed.

2.4. Register definition

2.4.1. Wait state register (FMC_WS)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	WSCNT[2:0]	Wait state counter These bits is set and reset by software. The WSCNT valid when WSEN bit in FMC_WSEN is set. 000: 0 wait state added 001: 1 wait state added 010: 2 wait state added 011~111:reserved

2.4.2. Unlock key register 0(FMC_KEY0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w															

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

Bits	Fields	Descriptions
31:0	KEY[31:0]	FMC_CTL0 unlock key These bits are only be written by software. Write KEY[31:0] with keys to unlock

FMC_CTL0 register

2.4.3. Option byte unlock key register (FMC_OBKEY)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OBKEY[31:16]															
w															

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

Bits	Fields	Descriptions
31:0	OBKEY[31:0]	FMC_ CTL0 option bytes operation unlock key These bits are only be written by software. Write OBKEY[31:0] with keys to unlock option bytes command in FMC_CTL0 register.

2.4.4. Status register 0 (FMC_STAT0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ENDF	WPERR	Reserved	PGERR	Reserved	BUSY
										rc_w1	rc_w1		rc_w1		r

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	ENDF	End of operation flag bit When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1.
4	WPERR	Erase/Program protection error flag bit When erase/program on protected pages, this bit is set by hardware. The software

		can clear it by writing 1.
3	Reserved	Must be kept at reset value
2	PGERR	Program error flag bit When program to the flash while it is not 0xFFFF, this bit is set by hardware. The software can clear it by writing 1.
1	Reserved	Must be kept at reset value
0	BUSY	The flash busy bit When the operation is in progress, this bit is set to 1. When the operation is end or an error is generated, this bit is cleared.

2.4.5. Control register 0 (FMC_CTL0)

Address offset: 0x10

Reset value: 0x0000 0080

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			ENDIE	Reserved	ERRIE	OBWEN	Reserved	LK	START	OBER	OBPG	Reserved	MER	PER	PG
			rw		rw	rw		rs	rs	rw	rw		rw	rw	rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	ENDIE	End of operation interrupt enable bit This bit is set or cleared by software 0: no interrupt generated by hardware. 1: end of operation interrupt enable
11	Reserved	Must be kept at reset value
10	ERRIE	Error interrupt enable bit This bit is set or cleared by software 0: no interrupt generated by hardware. 1: error interrupt enable
9	OBWEN	Option byte erase/program enable bit This bit is set by hardware when right sequence written to FMC_OBKEY register. This bit can be cleared by software.
8	Reserved	Must be kept at reset value

7	LK	FMC_CTL0 lock bit This bit is cleared by hardware when right sequence written to FMC_KEY0 register. This bit can be set by software.
6	START	Send erase command to FMC bit This bit is set by software to send erase command to FMC. This bit is cleared by hardware when the BUSY bit is cleared.
5	OBER	Option bytes erase command bit This bit is set or clear by software 0: no effect 1: option byte erase command
4	OBPG	Option bytes program command bit This bit is set or clear by software 0: no effect 1: option bytes program command
3	Reserved	Must be kept at reset value
2	MER	Main flash mass erase for bank0 command bit This bit is set or cleared by software 0: no effect 1: main flash mass erase command for bank0
1	PER	Main flash page erase for bank0 command bit This bit is set or clear by software 0: no effect 1: main flash page erase command for bank0
0	PG	Main flash program for bank0 command bit This bit is set or clear by software 0: no effect 1: main flash program command for bank0

Note: This register should be reset after the corresponding flash operation completed.

2.4.6. Address register 0 (FMC_ADDR0)

Address offset: 0x14

Reset value: 0x0000 0000.

This register has to be accessed by word (32-bit)

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

ADDR[15:0]
w

Bits	Fields	Descriptions
31:0	ADDR[31:0]	Flash erase/program command address bits These bits are configured by software. ADDR bits are the address of flash erase/program command

2.4.7. Option byte status register (FMC_OBSTAT)

Address offset: 0x1C

Reset value: 0x0XXX XXXX.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						DATA[15:6]									
						r									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[5:0]						USER[7:0]								SPC	OBERR
r						r								r	r

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25:10	DATA[15:0]	Store DATA of option bytes block after system reset.
9:2	USER[7:0]	Store USER of option bytes block after system reset.
1	SPC	Option bytes security protection code 0: no protection 1: protection
0	OBERR	Option bytes read error bit. This bit is set by hardware when the option bytes and its complement byte do not match, then the option bytes is set to 0xFF.

2.4.8. Erase/Program Protection register (FMC_WP)

Address offset: 0x20

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WP[31:16]															

r

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

Bits	Fields	Descriptions
31:0	WP[31:0]	Store WP of option bytes block after system reset

2.4.9. Unlock key register 1(FMC_KEY1)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w															

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

Bits	Fields	Descriptions
31:0	KEY[31:0]	FMC_CTL1 unlock register These bits are only be written by software Write KEY[31:0] with keys to unlock FMC_CTL1 register

2.4.10. Status register 1 (FMC_STAT1)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ENDF	WPERR	Reserved	PGERR	Reserved	BUSY
										rc_w1	rc_w1		rc_w1		r

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	ENDF	End of operation flag bit

		When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1.
4	WPERR	Erase/Program protection error flag bit When erase/program on protected pages, this bit is set by hardware. The software can clear it by writing 1.
3	Reserved	Must be kept at reset value
2	PGERR	Program error flag bit When program to the flash while it is not 0xFFFF, this bit is set by hardware. The software can clear it by writing 1.
1	Reserved	Must be kept at reset value
0	BUSY	The flash is busy bit. When the operation is in progress, this bit is set to 1. When the operation is end or an error is generated, this bit is cleared to 0.

2.4.11. Control register 1(FMC_CTL1)

Address offset: 0x50

Reset value: 0x0000 0080

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			ENDIE	Reserved	ERRIE	Reserved		LK	START	Reserved			MER	PER	PG
			rw		rw			rs	rs				rw	rw	rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	ENDIE	End of operation interrupt enable bit This bit is set or cleared by software 0: no interrupt generated by hardware. 1: end of operation interrupt enable
11	Reserved	Must be kept at reset value
10	ERRIE	Error interrupt enable bit This bit is set or cleared by software 0: no interrupt generated by hardware. 1: error interrupt enable

9:8	Reserved	Must be kept at reset value
7	LK	FMC_CTL1 lock bit This bit is cleared by hardware when right sequence written to FMC_KEY1 register. This bit can be set by software.
6	START	Send erase command to FMC bit This bit is set by software to send erase command to FMC. This bit is cleared by hardware when the BUSY bit is cleared.
5:3	Reserved	Must be kept at reset value
2	MER	Main flash mass erase for bank1 command bit This bit is set or cleared by software 0: no effect 1: main flash mass erase command for bank1
1	PER	Main flash page erase for bank1 command bit This bit is set or clear by software 0: no effect 1: main flash page erase command for bank1
0	PG	Main flash program for bank1 command bit This bit is set or clear by software 0: no effect 1: main flash program command for bank1

Note: This register should be reset after the corresponding flash operation completed.

2.4.12. Address register 1 (FMC_ADDR1)

Address offset: 0x54

Reset value: 0x0000 0000.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR[31:16]															
w															

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

Bits	Fields	Descriptions
31:0	ADDR[31:0]	Flash erase/program command address bits These bits are configured by software. ADDR bits are the address of flash erase/program command.

2.4.13. Wait state enable register (FMC_WSEN)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Resrved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															WSEN
															rw

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value
0	WSEN	FMC wait state enable register This bit is set and reset by software. This bit also protected by the FMC_KEYx register. It is necessary to writing 0x45670123 and 0xCDEF89AB to the FMC_KEYx register. 0: no wait state added when fetch flash 1: wait state added when fetch flash

2.4.14. Product ID register (FMC_PID)

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PID[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	PID[31:0]	Product reserved ID code register 0 These bits are read only by software. These bits are unchanged constant after power on. These bits are one time program when the chip produced.

3. Power management unit (PMU)

3.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32F10x series. According to the Power management unit (PMU), provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. For GD32F10x devices, there are three power domains, including V_{DD}/V_{DDA} domain, 1.2V domain, and Backup domain, as is shown in the following figure. The power of the V_{DD} domain is supplied directly by V_{DD}. An embedded LDO in the V_{DD}/V_{DDA} domain is used to supply the 1.2V domain power. A power switch is implemented for the Backup domain. It can be powered from the V_{BAT} voltage when the main V_{DD} supply is shut down.

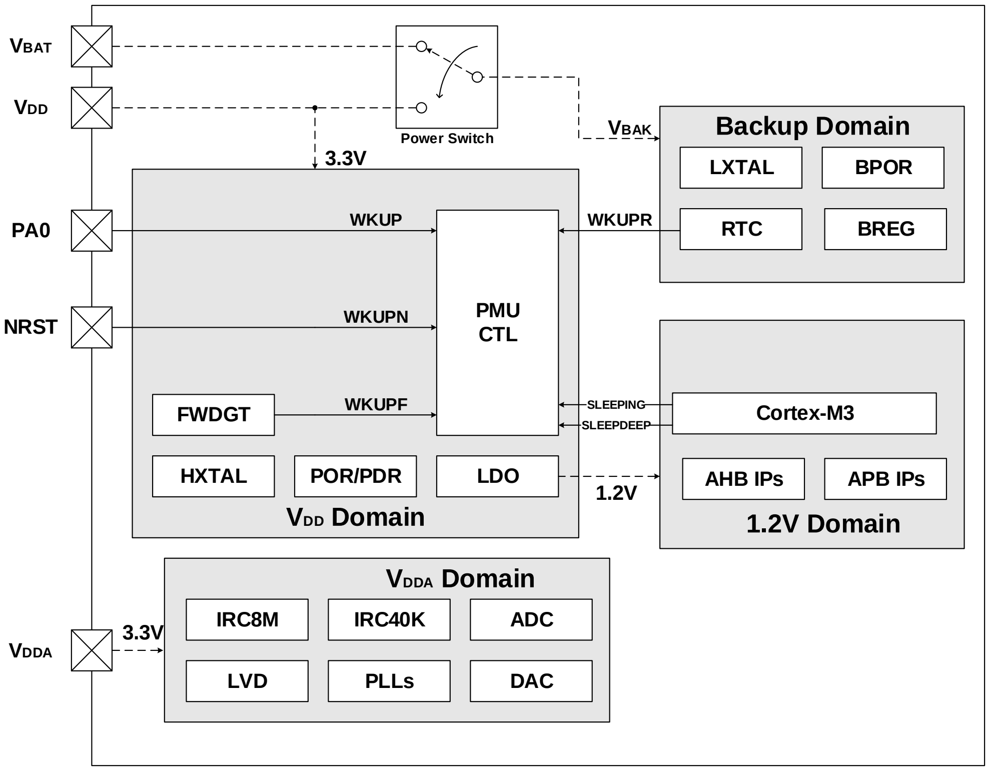
3.2. Characteristics

- Three power domains: V_{BAK}, V_{DD}/V_{DDA} and 1.2V power domains.
- Three power saving modes: Sleep, Deep-sleep and Standby modes.
- Internal Voltage regulator (LDO) supplies around 1.2V voltage source for 1.2V domain.
- Low Voltage Detector can issue an interrupt or event when the power is lower than a programmed threshold.
- Battery power (V_{BAT}) for Backup domain when V_{DD} is shut down.
- LDO output voltage select for power saving.

3.3. Function overview

Figure 3-1. Power supply overview provides details on the internal configuration of the PMU and the relevant power domains.

Figure 3-1. Power supply overview



LVD: Low Voltage Detector
POR: Power On Reset
LDO: Voltage Regulator
PDR: Power Down Reset
BPOR: V_{BAK} Power On Reset
BREG: Backup registers

3.3.1. Battery backup domain

The Backup domain is powered by the V_{DD} or the battery power source (V_{BAT}) selected by the internal power switch, and the V_{BAK} pin which drives Backup Domain, power supply for RTC unit, LXTAL oscillator, BPOR and BREG， and three pads, including PC13 to PC15. In order to ensure the content of the Backup domain registers and the RTC supply, when V_{DD} supply is shut down, V_{BAT} pin can be connected to an optional standby voltage supplied by a battery or by another source. The power switch is controlled by the Power Down Reset circuit in the V_{DD}/V_{DDA} domain. If no external battery is used in the application, it is recommended to connect V_{BAT} pin externally to V_{DD} pin with a 100nF external ceramic decoupling capacitor.

The Backup domain reset sources includes the Backup domain power-on-reset (BPOR) and the Backup Domain software reset. The BPOR signal forces the device to stay in the reset mode until V_{BAK} is completely powered up. Also the application software can trigger the Backup domain software reset by setting the BKPRST bit in the RCU_BDCTL register to reset the Backup domain.

The clock source of the Real Time Clock (RTC) circuit can be derived from the Internal 40KHz RC oscillator (IRC40K) or the Low Speed Crystal oscillator (LXTAL), or HXTAL clock divided by 128. When V_{DD} is shut down, only LXTAL is valid for RTC. Before entering the power saving mode by executing the WFI/WFE instruction, the Cortex™-M3 can setup the RTC register with an expected wakeup time and enable the wakeup function to achieve the RTC timer wakeup event. After entering the power saving mode for a certain amount of time, the RTC will wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the ***Real-time Clock (RTC***).

When the Backup domain is supplied by V_{DD} (V_{BAK} pin is connected to V_{DD}), the following functions are available:

- PC13 can be used as GPIO or RTC function pin described in the RTC chapter.
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

When the Backup domain is supplied by V_{BAT} (V_{BAK} pin is connected to V_{BAT}), the following functions are available:

- PC13 can be used as RTC function pin described in the RTC chapter.
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

Note: Since PC13, PC14, PC15 are supplied through the Power Switch, which can only be obtained by a small current, the speed of GPIOs PC13 to PC15 should not exceed 2MHz when they are in output mode(maximum load: 30pF)

3.3.2. VDD/VDDA power domain

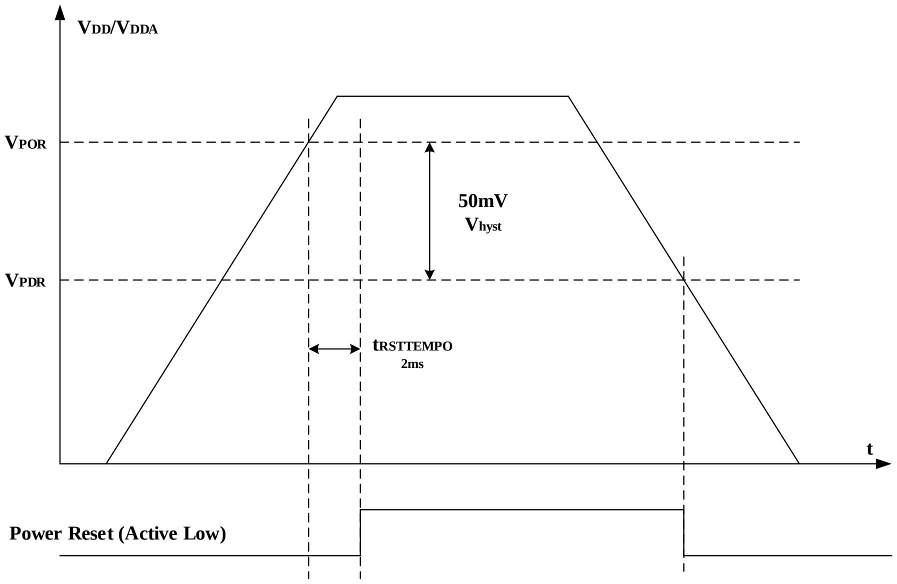
V_{DD}/V_{DDA} domain includes two parts: V_{DD} domain and V_{DDA} domain. V_{DD} domain includes HXTAL (High Speed Crystal oscillator), LDO (Voltage Regulator), POR/PDR (Power On/Down Reset), FWDGT (Free Watchdog Timer), all pads except PC13/PC14/PC15, etc. V_{DDA} domain includes ADC/DAC (AD/DA Converter), IRC8M (Internal 8MHz RC oscillator), IRC40K (Internal 40KHz RC oscillator), PLLs (Phase Locking Loop), LVD (Low Voltage Detector), etc.

VDD domain

The LDO, which is implemented to supply power for the 1.2V domain, is always enabled after reset. It can be configured to operate in three different status, including in the Sleep mode (full power on), in the Deep-sleep mode (on or low power), and in the Standby mode (power off).

The POR/PDR circuit is implemented to detect V_{DD}/V_{DDA} and generate the power reset signal which resets the whole chip except the Backup domain when the supply voltage is lower than the specified threshold. ***Figure 3-2. Waveform of the POR/PDR*** shows the relationship between the supply voltage and the power reset signal. V_{POR}, which typical value is 2.40V, indicates the threshold of power on reset, while V_{PDR}, which typical value is 2.35V, means the threshold of power down reset. The hysteresis voltage (V_{hyst}) is around 50mV.

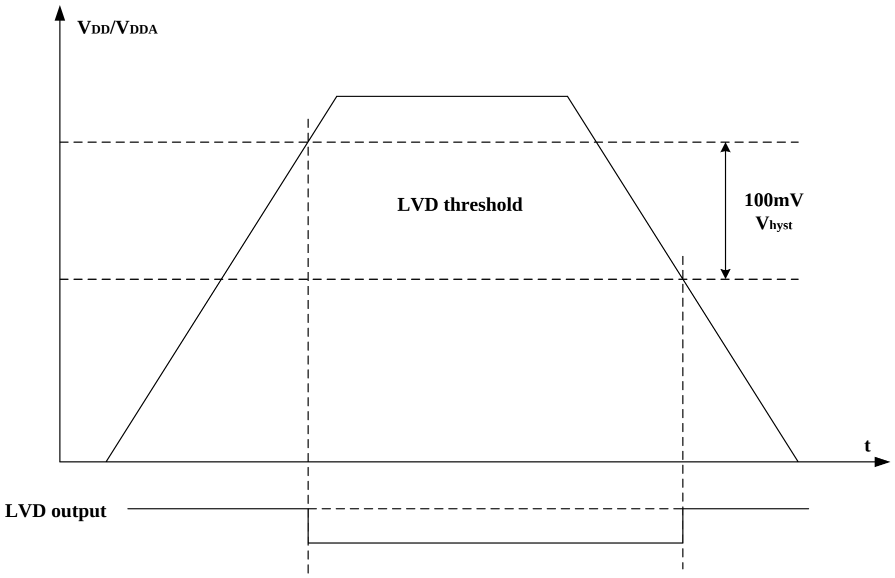
Figure 3-2. Waveform of the POR/PDR



VDDA domain

The LVD is used to detect whether the V_{DD}/V_{DDA} supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU_CTL). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register (PMU_CS), indicates if V_{DD}/V_{DDA} is higher or lower than the LVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. ***Figure 3-3. Waveform of the LVD threshold*** shows the relationship between the LVD threshold and the LVD output (LVD interrupt signal depends on EXTI line 16 rising or falling edge configuration). The following figure shows the relationship between the supply voltage and the LVD signal. The hysteresis voltage (V_{hyst}) is 100mV.

Figure 3-3. Waveform of the LVD threshold



Generally, digital circuits are powered by V_{DD}, while most of analog circuits are powered by V_{DDA}. To improve the ADC and DAC conversion accuracy, the independent power supply V_{DDA} is implemented to achieve better performance of analog circuits. V_{DDA} can be externally connected to V_{DD} through the external filtering circuit that avoids noise on V_{DDA}, and V_{SSA} should be connected to V_{SS} through the specific circuit independently. Otherwise, if V_{DDA} is different from V_{DD}, V_{DDA} must always be higher, but the voltage difference should not exceed 0.2V.

To ensure a high accuracy on ADC and DAC, the ADC/DAC independent external reference voltage should be connected to VREF+/VREF- pins. According to the different packages, VREF+ pin can be connected to VDDA pin, or external reference voltage which refers to ***Table 11-2. ADC pins definition*** and ***Table 12-1. DAC pins***, VREF- pin must be connected to VSSA pin. The VREF+ pin is only available on no less than 100-pin packages. On less than 100-pin packages, the VREF+ pin is not available and it is internally connected to VDDA. The VREF- pin is internally connected to VSSA.

3.3.3. 1.2V power domain

The main functions that include Cortex™-M3 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the V_{DD}/V_{DDA} domain, etc, are located in this power domain. Once the 1.2V is powered up, the POR will generate a reset sequence on the 1.2V power domain. If need to enter the expected power saving mode, the associated control bits must be configured. Then, once a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed, the device will enter an expected power saving mode which will be discussed in the following section.

3.3.4. Power saving modes

After a system reset or a power reset, the GD32F10x MCU operates at full function and all power domains are active. Users can achieve lower power consumption through slowing down the system clocks (HCLK, PCLK1, PCLK2) or gating the clocks of the unused peripherals. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, and Standby mode.

Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex™-M3. In Sleep mode, only clock of Cortex™-M3 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and execute a WFI or WFE instruction. If the Sleep mode is entered by executing a WFI instruction, any interrupt can wake up the system. If it is entered by executing a WFE instruction, any wakeup event can wake up the system (If SEVONPEND is 1, any interrupt can wake up the system, refer to Cortex-M3 Technical Reference Manual). The mode offers the lowest wakeup time as no time is wasted in interrupt entry or exit.

According to the SLEEPONEXIT bit in the Cortex™-M3 System Control Register, there are two options to select the Sleep mode entry mechanism.

- 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 from the lowest priority ISR.

Deep-sleep mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the Cortex™-M3. In Deep-sleep mode, all clocks in the 1.2V domain are off, and all of IRC8M, HXTAL and PLLs are disabled. The contents of SRAM and registers are preserved. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU_CTL register. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and clear the STBMOD bit in the PMU_CTL register. Then, the device enters the Deep-sleep mode after a WFI or WFE instruction is executed. If the Deep-sleep mode is entered by executing a WFI instruction, any interrupt from EXTI lines can wake up the system. If it is entered by executing a WFE instruction, any wakeup event from EXTI lines can wake up the system (If SEVONPEND is 1, any interrupt from EXTI lines can wake up the system, refer to Cortex™-M3 Technical Reference Manual). When exiting the Deep-sleep mode, the IRC8M is selected as the system clock. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

Note: In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI_PD register) and RTC Alarm must be reset. If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex™-M3, too. In Standby mode, the whole 1.2V domain is power off, the LDO is shut down, and all of IRC8M, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and set the STBMOD bit in the PMU_CTL register, and clear WUF bit in the PMU_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed, and the STBF status flag in the PMU_CS register indicates that the MCU has been in Standby mode. There are four wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC alarm, the FWDGT reset, and the rising edge on WKUP pin. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers in 1.2V power domain are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex™-M3 will execute instruction code from the 0x00000000 address.

Table 3-1. Power saving mode summary

Mode	Sleep	Deep-sleep	Standby
Description	Only CPU clock is off	1. All clocks in the 1.2V domain are off 2. Disable IRC8M, HXTAL and PLL	1. The 1.2V domain is power off 2. Disable IRC8M, HXTAL and PLL
LDO Status	On	On or in low power mode	Off

Configuration	SLEEPDEEP = 0	SLEEPDEEP = 1
		STBMOD = 0
			SLEEPDEEP = 1
			STBMOD = 1, WURST=1

Entry	WFI or WFE	WFI or WFE	WFI or WFE
Wakeup	Any interrupt for WFI Any event (or interrupt when SEVONPEND is 1) for WFE	Any interrupt from EXTI lines for WFI Any event(or interrupt when SEVONPEND is 1) from EXTI for WFE	1. NRST pin 2. WKUP pin 3. FWDGT reset 4. RTC
Wakeup Latency	None	IRC8M wakeup time, LDO wakeup time added if LDO is in low power mode	Power on sequence

3.4. Register definition

3.4.1. Control register (PMU_CTL)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

This register can be accessed by half-word(16-bit) or word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BKPWEN	LVDT[2:0]			LVDEN	STBRST	WURST	STBMOD	LDOLP
							rw	rw			rw	rc_w1	rc_w1	rw	rw

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	BKPWEN	Backup Domain Write Enable 0: Disable write access to the registers in Backup domain 1: Enable write access to the registers in Backup domain After reset, any write access to the registers in Backup domain is ignored. This bit has to be set to enable write access to these registers.
7:5	LVDT[2:0]	Low Voltage Detector Threshold 000: 2.2V 001: 2.3V 010: 2.4V 011: 2.5V 100: 2.6V 101: 2.7V 110: 2.8V 111: 2.9V
4	LVDEN	Low Voltage Detector Enable 0: Disable Low Voltage Detector 1: Enable Low Voltage Detector
3	STBRST	Standby Flag Reset 0: No effect 1: Reset the standby flag This bit is always read as 0.
2	WURST	Wakeup Flag Reset 0: No effect

		1: Reset the wakeup flag This bit is always read as 0.
1	STBMOD	Standby Mode 0: Enter the Deep-sleep mode when the Cortex™-M3 enters SLEEPDEEP mode 1: Enter the Standby mode when the Cortex™-M3 enters SLEEPDEEP mode
0	LDOLP	LDO Low Power Mode 0: The LDO operates normally during the Deep-sleep mode 1: The LDO is in low power mode during the Deep-sleep mode

3.4.2. Control and status register (PMU_CS)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

This register can be accessed by half-word(16-bit) or word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							WUPEN	Reserved					LVDF	STBF	WUF
							rw						r	r	r

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8	WUPEN	WKUP Pin Enable 0: Disable WKUP pin function 1: Enable WKUP pin function If WUPEN is set before entering the power saving mode, a rising edge on the WKUP pin wakes up the system from the power saving mode. As the WKUP pin is active high, the WKUP pin is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
7:3	Reserved	Must be kept at reset value
2	LVDF	Low Voltage Detector Status Flag 0: Low Voltage event has not occurred (V_{DD} is higher than the specified LVD threshold) 1: Low Voltage event occurred (V_{DD} is equal to or lower than the specified LVD threshold) **Note:** The LVD function is stopped in Standby mode.
1	STBF	Standby Flag

		0: The device has not entered the Standby mode 1: The device has been in the Standby mode This bit is cleared only by a POR/PDR or by setting the STBRST bit in the PMU_CTL register.
0	WUF	Wakeup Flag 0: No wakeup event has been received 1: Wakeup event occurred from the WKUP pin or the RTC wakeup event including RTC Tamper event, RTC alarm event，RTC Time Stamp event or RTC Wakeup This bit is cleared only by a POR/PDR or by setting the WURST bit in the PMU_CTL register.

4. Backup registers (BKP)

4.1. Overview

The Backup registers are located in the Backup domain that remains powered-on by V_{BAT} even if V_{DD} power is shut down, they are forty two 16-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 PMUEN and BKPIEN bits in the RCU_APB1EN register, and writing access to the registers in Backup domain should be enabled by setting the BKPWEN bit in the PMU_CTL register.

4.2. 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_TPCS) can control tamper detection with interrupt or event capability.

4.3. Function overview

4.3.1. RTC clock calibration

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

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

4.3.2. Tamper detection

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 TPEN bit in the BKP_TPCTL register. To prevent the tamper event from losing, the edge detection is logically ANDed with the TPEN 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_TPCS 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 TPEN bit), an extra tamper event is detected, while there was no rising/falling edge on the TAMPER pin after TPEN bit was set.

4.4. Register definition

4.4.1. Backup data register x (BKP_DATAx) (x= 0..41)

Address offset: 0x04 to 0x28, 0x40 to 0xBC
Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:0	DATA[15:0]	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.

4.4.2. RTC signal output control register (BKP_OCTL)

Address offset: 0x2C
Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ROSEL	ASOEN	COEN	RCCV[6:0]						
						rw	rw	rw	rw						

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9	ROSEL	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	ASOEN	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. This bit is reset only by a Backup domain reset.
7	COEN	RTC clock calibration output enable

		0: Disable RTC clock calibration output 1: Enable RTC clock Calibration output When enable, the TAMPER pin will output a clock with the frequency f_{RTCCLK}/64. ASOEN has the priority over COEN. When ASOEN is set, the TAMPER pin will output the RTC alarm or second signal whether COEN is set or not. This bit is reset only by a POR/PDR.
6:0	RCCV[6:0]	RTC clock calibration value The value indicates how many clock pulses are ignored or added every 2^20 RTC clock pulses.

4.4.3. Tamper pin control register (BKP_TPCTL)

Address offset: 0x30

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	TPAL	TAMPER pin active level 0: The TAMPER pin is active high 1: The TAMPER pin is active low
0	TPEN	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.

4.4.4. Tamper control and status register (BKP_TPCS)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved					TPIE	TIR	TER
						r	r						rw	w	w

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9	TIF	Tamper interrupt flag 0: No tamper interrupt occurred 1: A tamper interrupt occurred This bit is reset by writing 1 to the TIR bit or the TPIE bit being 0.
8	TEF	Tamper event flag 0: No tamper event occurred 1: A tamper event occurred This bit is reset by writing 1 to the TER bit.
7:3	Reserved	Must be kept at reset value
2	TPIE	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	TIR	Tamper interrupt reset 0: No effect 1: Reset the TIF bit This bit is always read as 0.
0	TER	Tamper event reset 0: No effect 1: Reset the TEF bit This bit is always read as 0.

5. Reset and clock unit (RCU)

Medium-, High- and Extra-density Reset and clock control unit (RCU)

GD32F101xx and GD32F103xx microcontrollers where the flash memory density ranges between 16 and 128 Kbytes are called Medium-density devices (GD32F10x_MD).

GD32F101xx and GD32F103xx microcontrollers where the flash memory density ranges between 256 and 512 Kbytes are called High-density devices (GD32F10x_HD).

GD32F101xx and GD32F103xx microcontrollers where the flash memory density is over 512 Kbytes are called Extra-density devices (GD32F10x_XD).

GD32F105xx and GD32F107xx microcontrollers are called connectivity line devices (GD32F10x_CL).

5.1. Reset control unit (RCTL)

5.1.1. Overview

GD32F10x Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power reset, known as a cold reset, resets the full system except the Backup domain. The system reset resets the processor core and peripheral IP components except for the SW-DP controller and the Backup domain. The backup domain reset resets the Backup domain. The resets can be triggered by an external signal, internal events and the reset generators. More information about these resets will be described in the following sections.

5.1.2. Function overview

Power reset

The Power reset is generated by either an external reset as Power On and Power Down reset (POR/PDR reset), or by the internal reset generator when exiting Standby mode. The power reset sets all registers to their reset values except the Backup domain. The Power reset whose active signal is low, it will be de-asserted when the internal LDO voltage regulator is ready to provide 1.2V power. The RESET service routine vector is fixed at address 0x0000 0004 in the memory map.

System reset

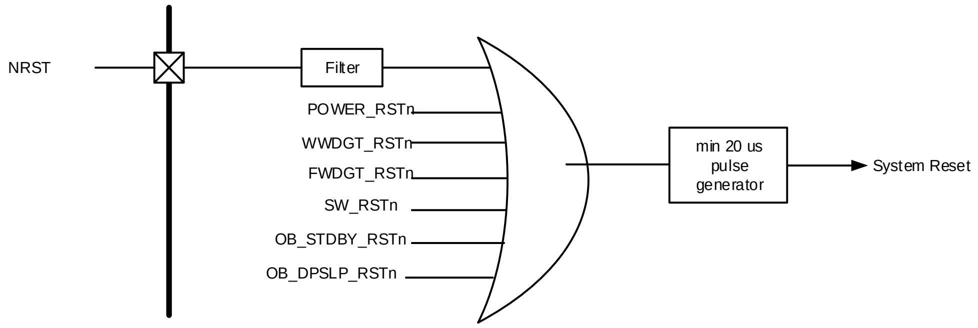
A system reset is generated by the following events:

- A power reset (POWER_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT_RSTn).
- A free watchdog timer reset (FWDGT_RSTn).
- The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register is set (SW_RSTn).
- Reset generated when entering Standby mode when resetting nRST_STDBY bit in User Option Bytes (OB_STDBY_RSTn).
- Reset generated when entering Deep-sleep mode when resetting nRST_DPSLP bit in User Option Bytes (OB_DPSLP_RSTn).

A system reset resets the processor core and peripheral IP components except for the SW-DP controller and the Backup domain.

A system reset pulse generator guarantees low level pulse duration of 20 μs for each reset source (external or internal reset).

Figure 5-1. The system reset circuit



Backup domain reset

A backup domain reset is generated by setting the BKPRST bit in the Backup domain control register or Backup domain power on reset (V_{DD} or V_{BAT} power on, if both supplies have previously been powered off).

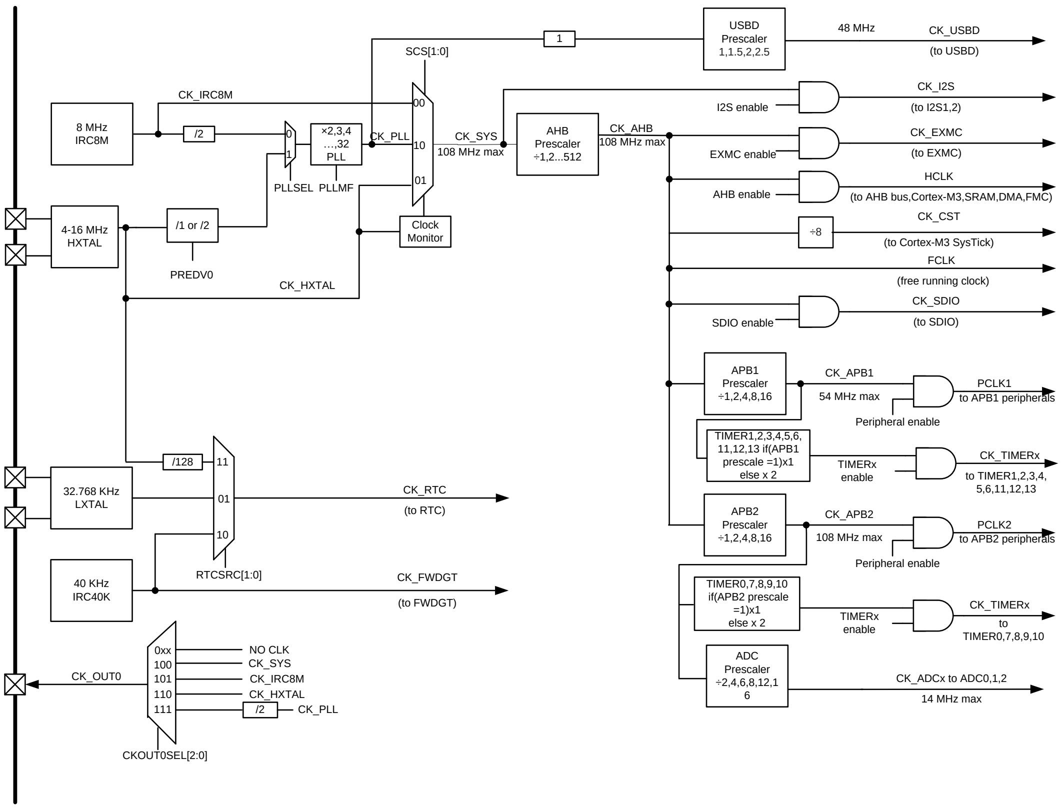
5.2. Clock control unit (CCTL)

5.2.1. Overview

The Clock Control unit provides a range of frequencies and clock functions. These include an Internal 8M RC oscillator (IRC8M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 40K RC oscillator (IRC40K), a Low Speed crystal oscillator (LXTAL), a Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex™-M3 are derived from the system clock (CK_SYS) which can source from the IRC8M, HXTAL or PLL. The maximum operating frequency of the system clock (CK_SYS) can be up to 108 MHz. The Free Watchdog Timer has independent clock source (IRC40K), and Real Time Clock (RTC) uses the IRC40K, LXTAL or HXTAL/128 as its clock source.

Figure 5-2. Clock tree



The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB, APB2 and APB1 domains is 108 MHz/108 MHz/54 MHz. The Cortex System Timer (SysTick) external clock is clocked with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the AHB clock (HCLK), configurable in the SysTick Control and Status Register.

The ADCs are clocked by the clock of APB2 divided by 2, 4, 6, 8, 12 or 16.

The SDIO, EXMC are clocked by the clock of CK_AHB.

The TIMERs are clocked by the clock divided from CK_APB2 and CK_APB1. The frequency of TIMERs clock is equal to CK_APBx(APB prescaler is 1), twice the CK_APBx(APB prescaler is not 1).

The USBD is clocked by the clock of CK_PLL as the clock source of 48MHz.

The I2S is clocked by the clock of CK_SYS.

The RTC is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 128 (defined which select by RTCSRC bit in Backup Domain Control Register (RCU_BDCTL). After the RTC select HXTAL clock divided by 128, the clock disappeared when the 1.2V core domain power off. After the RTC select IRC40K, the clock disappeared when V_{DD} power off. After the RTC select LXTAL, the clock disappeared when V_{DD} and V_{BAT} power off.

The FWDGT is clocked by IRC40K clock, which is forced on when FWDGT started.

5.2.2. Characteristics

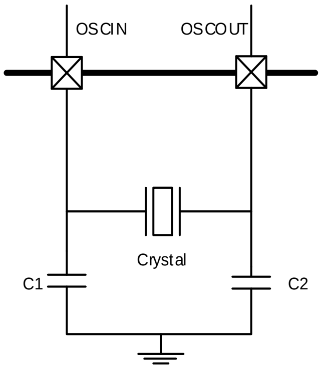
- 4 to 16 MHz High Speed crystal oscillator (HXTAL) .
- Internal 8 MHz RC oscillator (IRC8M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL or IRC8M.
- HXTAL clock monitor.

5.2.3. Function overview

High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), 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 HXTAL pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

Figure 5-3. HXTAL clock source



The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the Control Register RCU_CTL. The HXTALSTB flag in Control Register RCU_CTL indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator "Start-up time". As the HXTAL becomes stable, an interrupt will be generated if the related interrupt enable bit HXTALSTBIE in the Interrupt Register RCU_INT is set. At this point the HXTAL clock can be used directly as the system clock source or the PLL input clock.

Select external clock bypass mode by setting the HXTALBPS and HXTALEN bits in the Control Register RCU_CTL. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

Internal 8M RC oscillators (IRC8M)

The internal 8M RC oscillator, IRC8M, has a fixed frequency of 8 MHz and is the default clock source selection for the CPU when the device is powered up. The IRC8M oscillator provides a lower cost type clock source as no external components are required. The IRC8M RC oscillator can be switched on or off using the IRC8MEN bit in the Control Register RCU_CTL. The IRC8MSTB flag in the Control Register RCU_CTL is used to indicate if the internal 8M RC oscillator is stable. The start-up time of the IRC8M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC8MSTBIE, in the Clock Interrupt Register, RCU_INT, is set when the IRC8M becomes stable. The IRC8M clock can also be used as the system clock source or the PLL input clock.

The frequency accuracy of the IRC8M can be calibrated by the manufacturer, but its operating frequency is still less accurate than HXTAL. The application requirements, environment and cost will determine which oscillator type is selected.

If the HXTAL or PLL is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the IRC8M clock to be the system clock when the system initially wakes-up.

Phase locked loop (PLL)

There is one internal Phase Locked Loop, named PLL.

The PLL can be switched on or off by using the PLLEN bit in the RCU_CTL Register. The PLLSTB flag in the RCU_CTL Register will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the RCU_INT Register, is set as the PLL becomes stable.

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

Low speed crystal oscillator (LXTAL)

The low speed external crystal or ceramic resonator oscillator, which has a frequency of 32,768 Hz, produces a low power but highly accurate clock source for the Real Time Clock circuit. The LXTAL oscillator can be switched on or off using the LXTALEN bit in the Backup Domain Control Register (RCU_BDCTL). The LXTALSTB flag in the Backup Domain Control Register (RCU_BDCTL) will indicate if the LXTAL clock is stable. An interrupt can be generated if the related interrupt enable bit, LXTALSTBIE, in the Interrupt Register RCU_INT is set when the LXTAL becomes stable.

Select external clock bypass mode by setting the LXTALBPS and LXTALEN bits in the

Backup Domain Control Register (RCU_BDCTL). The CK_LXTAL is equal to the external clock which drives the OSC32IN pin.

Internal 40K RC oscillator (IRC40K)

The internal RC oscillator has a frequency of about 40 kHz and is a low power clock source for the Real Time Clock circuit or the Free Watchdog Timer. The IRC40K offers a low cost clock source as no external components are required. The IRC40K RC oscillator can be switched on or off by using the IRC40KEN bit in the Reset source/clock Register (RCU_RSTSCK). The IRC40KSTB flag in the Reset source/clock Register RCU_RSTSCK will indicate if the IRC40K clock is stable. An interrupt can be generated if the related interrupt enable bit IRC40KSTBIE in the Clock Interrupt Register (RCU_INT) is set when the IRC40K becomes stable.

The IRC40K can be trimmed by TIMER4_CH3, user can get the clocks frequency, and adjust the RTC and FWDGT counter. Please refer to TIMER4CH3_IREMAP in AFIO_PCF0 register.

System clock (CK_SYS) selection

After the system reset, the default CK_SYS source will be IRC8M and can be switched to HXTAL or CK_PLL by changing the System Clock Switch bits, SCS, in the Clock configuration register 0, RCU_CFG0. When the SCS value is changed, the CK_SYS will continue to operate using the original clock source until the target clock source is stable. When a clock source is directly or indirectly (by PLL) used as the CK_SYS, it is not possible to stop it.

HXTAL clock monitor (CKM)

The HXTAL clock monitor function is enabled by the HXTAL Clock Monitor Enable bit, CKMEN, in the Control Register (RCU_CTL). This function should be enabled after the HXTAL start-up delay and disabled when the HXTAL is stopped. Once the HXTAL failure is detected, the HXTAL will be automatically disabled. The HXTAL Clock Stuck interrupt Flag, CKMIF, in the Clock Interrupt Register, RCU_INT, will be set and the HXTAL failure event will be generated. This failure interrupt is connected to the Non-Maskable Interrupt, NMI, of the Cortex-M3. If the HXTAL is selected as the clock source of CK_SYS, PLL and CK_RTC, the HXTAL failure will force the CK_SYS source to IRC8M, the PLL will be disabled automatically. If the HXTAL is selected as the clock source of PLL, the HXTAL failure will force the PLL closed automatically. If the HXTAL is selected as the clock source of RTC, the HXTAL failure will reset the RTC clock selection.

Clock output capability

The clock output capability is ranging from 4 MHz to 108 MHz. There are several clock signals can be selected via the CK_OUT0 Clock Source Selection bits, CKOUT0SEL, in the Clock Configuration Register 0 (RCU_CFG0). The corresponding GPIO pin should be configured in the properly Alternate Function I/O (AFIO) mode to output the selected clock signal.

Table 5-1. Clock output 0 source select

Clock Source 0 Selection bits	Clock Source
0xx	NO CLK
100	CK_SYS
101	CK_IRC8M
110	CK_HXTAL
111	CK_PLL/2

Voltage control

The 1.2V domain voltage in Deep-sleep mode can be controlled by DSLPVS[1:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

Table 5-2. 1.2V domain voltage selected in deep-sleep mode

DSLPVS[1:0]	Deep-sleep mode voltage(V)
00	1.2
01	1.1
10	1.0
11	0.9

5.3. Register definition

5.3.1. Control register (RCU_CTL)

Address offset: 0x00

Reset value: 0x0000 xx83 where x is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLLSTB	PLLEN	Reserved				CKMEN	HXTALBPS	HXTALSTB	HXTALEN
						r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRC8MCALIB[7:0]								IRC8MADJ[4:0]					Reserved	IRC8MSTB	IRC8MEN
r								rw						r	rw

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value.
25	PLLSTB	PLL Clock Stabilization Flag Set by hardware to indicate if the PLL output clock is stable and ready for use. 0: PLL is not stable 1: PLL is stable
24	PLLEN	PLL enable Set and reset by software. This bit cannot be reset if the PLL clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL is switched off 1: PLL is switched on
23:20	Reserved	Must be kept at reset value.
19	CKMEN	HXTAL Clock Monitor Enable 0: Disable the High speed 4 ~ 16 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 4 ~ 16 MHz crystal oscillator (HXTAL) clock monitor When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC8M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software. **Note:** When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC8M internal RC oscillator regardless of the control bit, IRC8MEN, state.
18	HXTALBPS	High speed crystal oscillator (HXTAL) clock bypass mode enable The HXTALBPS bit can be written only if the HXTALEN is 0. 0: Disable the HXTAL Bypass mode

		1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock.
17	HXTALSTB	High speed crystal oscillator (HXTAL) clock stabilization flag Set by hardware to indicate if the HXTAL oscillator is stable and ready for use. 0: HXTAL oscillator is not stable 1: HXTAL oscillator is stable
16	HXTALEN	High Speed crystal oscillator (HXTAL) Enable Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL input clock when PLL clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: High speed 4 ~ 16 MHz crystal oscillator disabled 1: High speed 4 ~ 16 MHz crystal oscillator enabled
15:8	IRC8MCALIB[7:0]	Internal 8MHz RC Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC8MADJ[4:0]	Internal 8MHz RC Oscillator clock trim adjust value These bits are set by software. The trimming value is these bits (IRC8MADJ) added to the IRC8MCALIB[7:0] bits. The trimming value should trim the IRC8M to 8 MHz ± 1%.
2	Reserved	Must be kept at reset value.
1	IRC8MSTB	IRC8M Internal 8MHz RC Oscillator stabilization Flag Set by hardware to indicate if the IRC8M oscillator is stable and ready for use. 0: IRC8M oscillator is not stable 1: IRC8M oscillator is stable
0	IRC8MEN	Internal 8MHz RC oscillator Enable Set and reset by software. This bit cannot be reset if the IRC8M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when CKMEN is set. 0: Internal 8 MHz RC oscillator disabled 1: Internal 8 MHz RC oscillator enabled

5.3.2. Clock configuration register 0 (RCU_CFG0)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			ADCPSC[2]	PLLMF[4]	CKOUT0SEL[2:0]			USBDPSC[1:0]		PLLMF[3:0]				PREDV0	PLLSEL

			rw	rw		rw		rw			rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]			APB1PSC[2:0]			AHBPSC[3:0]				SCSS[1:0]		SCS[1:0]	
rw		rw			rw			rw				r		rw	

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28	ADCPSC[2]	Bit 2 of ADCPSC see bits 15:14 of RCU_CFG0
27	PLLMF[4]	Bit 4 of PLLMF see bits 21:18 of RCU_CFG0
26:24	CKOUT0SEL[2:0]	CKOUT0 Clock Source Selection Set and reset by software. 0xx: No clock selected 100: System clock selected 101: High Speed 8M Internal Oscillator clock selected 110: External High Speed oscillator clock selected 111: (CK_PLL / 2) clock selected
23:22	USBDPSC[1:0]	USBD clock prescaler selection Set and reset by software to control the USBD clock prescaler value. The USBD clock must be 48MHz. These bits can’t be reset if the USBD clock is enabled. 00: CK_USBD = CK_PLL / 1.5 01: CK_USBD = CK_PLL 10: CK_USBD = CK_PLL / 2.5 11: CK_USBD = CK_PLL / 2
21:18	PLLMF[3:0]	The PLL clock multiplication factor Bit 27 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor. **Caution**: The PLL output frequency must not exceed 108 MHz 00000: CK_SYS = CK_PLL x 2 00001: CK_SYS = CK_PLL x 3 00010: CK_SYS = CK_PLL x 4 00011: CK_SYS = CK_PLL x 5 00100: CK_SYS = CK_PLL x 6 00101: CK_SYS = CK_PLL x 7 00110: CK_SYS = CK_PLL x 8 00111: CK_SYS = CK_PLL x 9 01000: CK_SYS = CK_PLL x 10 01001: CK_SYS = CK_PLL x 11 01010: CK_SYS = CK_PLL x 12

		01011: CK_SYS = CK_PLL x 13 01100: CK_SYS = CK_PLL x 14 01101: CK_SYS = CK_PLL x 15 01110: CK_SYS = CK_PLL x 16 01111: CK_SYS = CK_PLL x 16 10000: CK_SYS = CK_PLL x 17 10001: CK_SYS = CK_PLL x 18 10010: CK_SYS = CK_PLL x 19 10011: CK_SYS = CK_PLL x 20 10100: CK_SYS = CK_PLL x 21 10101: CK_SYS = CK_PLL x 22 10110: CK_SYS = CK_PLL x 23 10111: CK_SYS = CK_PLL x 24 11000: CK_SYS = CK_PLL x 25 11001: CK_SYS = CK_PLL x 26 11010: CK_SYS = CK_PLL x 27 11011: CK_SYS = CK_PLL x 28 11100: CK_SYS = CK_PLL x 29 11101: CK_SYS = CK_PLL x 30 11110: CK_SYS = CK_PLL x 31 11111: CK_SYS = CK_PLL x 32
17	PREDV0	PREDV0 division factor This bit is set and reset by software. These bits can be written when PLL is disable. 0: PREDV0 input source clock not divided 1: PREDV0 input source clock divided by 2
16	PLLSEL	PLL Clock Source Selection Set and reset by software to control the PLL clock source. 0: (IRC8M / 2) clock selected as source clock of PLL 1: HXTAL selected as source clock of PLL
15:14	ADCPSC[1:0]	ADC clock prescaler selection These bits and bit 28 of RCU_CFG0 are written by software to define the ADC prescaler factor.Set and cleared by software. 000: (CK_APB2 / 2) selected 001: (CK_APB2 / 4) selected 010: (CK_APB2 / 6) selected 011: (CK_APB2 / 8) selected 100: (CK_APB2 / 2) selected 101: (CK_APB2 / 12) selected 110: (CK_APB2 / 8) selected 111: (CK_APB2 / 16) selected
13:11	APB2PSC[2:0]	APB2 prescaler selection

		Set and reset by software to control the APB2 clock division ratio. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
10:8	APB1PSC[2:0]	APB1 prescaler selection Set and reset by software to control the APB1 clock division ratio. Caution: The CK_APB1 output frequency must not exceed 60 MHz. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
7:4	AHBPSC[3:0]	AHB prescaler selection Set and reset by software to control the AHB clock division ratio 0xxx: CK_SYS selected 1000: (CK_SYS / 2) selected 1001: (CK_SYS / 4) selected 1010: (CK_SYS / 8) selected 1011: (CK_SYS / 16) selected 1100: (CK_SYS / 64) selected 1101: (CK_SYS / 128) selected 1110: (CK_SYS / 256) selected 1111: (CK_SYS / 512) selected
3:2	SCSS[1:0]	System clock switch status Set and reset by hardware to indicate the clock source of system clock. 00: select CK_IRC8M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLL as the CK_SYS source 11: reserved
1:0	SCS[1:0]	System clock switch Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC8M when leaving Deep-sleep and Standby mode or HXTAL failure is detected by HXTAL clock monitor when HXTAL is selected directly or indirectly as the clock source of CK_SYS 00: select CK_IRC8M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLL as the CK_SYS source 11: reserved

5.3.3. Clock interrupt register (RCU_INT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKMIC	Reserved		PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40KS TBIC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	Reserved		PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	CKMIC	HXTAL Clock Stuck Interrupt Clear Write 1 by software to reset the CKMIF flag. 0: Not reset CKMIF flag 1: Reset CKMIF flag
22:21	Reserved	Must be kept at reset value
20	PLLSTBIC	PLL stabilization Interrupt Clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL Stabilization Interrupt Clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M Stabilization Interrupt Clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL Stabilization Interrupt Clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag
16	IRC40KSTBIC	IRC40K Stabilization Interrupt Clear

		Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15:13	Reserved	Must be kept at reset value
12	PLLSTBIE	PLL Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL stabilization interrupt. 0: Disable the PLL stabilization interrupt 1: Enable the PLL stabilization interrupt
11	HXTALSTBIE	HXTAL Stabilization Interrupt Enable Set and reset by software to enable/disable the HXTAL stabilization interrupt 0: Disable the HXTAL stabilization interrupt 1: Enable the HXTAL stabilization interrupt
10	IRC8MSTBIE	IRC8M Stabilization Interrupt Enable Set and reset by software to enable/disable the IRC8M stabilization interrupt 0: Disable the IRC8M stabilization interrupt 1: Enable the IRC8M stabilization interrupt
9	LXTALSTBIE	LXTAL Stabilization Interrupt Enable LXTAL stabilization interrupt enable/disable control 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC40KSTBIE	IRC40K Stabilization interrupt enable IRC40K stabilization interrupt enable/disable control 0: Disable the IRC40K stabilization interrupt 1: Enable the IRC40K stabilization interrupt
7	CKMIF	HXTAL Clock Stuck Interrupt Flag Set by hardware when the HXTAL clock is stuck. Reset when setting the CKMIC bit by software. 0: Clock operating normally 1: HXTAL clock stuck
6:5	Reserved	Must be kept at reset value
4	PLLSTBIF	PLL stabilization interrupt flag Set by hardware when the PLL is stable and the PLLSTBIE bit is set. Reset when setting the PLLSTBIC bit by software. 0: No PLL stabilization interrupt generated 1: PLL stabilization interrupt generated
3	HXTALSTBIF	HXTAL stabilization interrupt flag Set by hardware when the High speed 4 ~ 16 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.

		Reset when setting the HXTALSTBIC bit by software. 0: No HXTAL stabilization interrupt generated 1: HXTAL stabilization interrupt generated
2	IRC8MSTBIF	IRC8M stabilization interrupt flag Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set. Reset when setting the IRC8MSTBIC bit by software. 0: No IRC8M stabilization interrupt generated 1: IRC8M stabilization interrupt generated
1	LXTALSTBIF	LXTAL stabilization interrupt flag Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set. Reset when setting the LXTALSTBIC bit by software. 0: No LXTAL stabilization interrupt generated 1: LXTAL stabilization interrupt generated
0	IRC40KSTBIF	IRC40K stabilization interrupt flag Set by hardware when the Internal 40kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set. Reset when setting the IRC40KSTBIC bit by software. 0: No IRC40K stabilization clock ready interrupt generated 1: IRC40K stabilization interrupt generated

5.3.4. APB2 reset register (RCU_APB2RST)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIMER10RST	TIMER9RST	TIMER8RST	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2RST	USART0RST	TIMER7RST	SPI0RST	TIMER0RST	ADC1RST	ADC0RST	PGRST	PFRST	PERST	PDRST	PCRST	PBRST	PARST	Reserved	AFRST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TIMER10RST	Timer 10 reset

		This bit is set and reset by software. 0: No reset 1: Reset the TIMER10
20	TIMER9RST	Timer 9 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER9
19	TIMER8RST	Timer 8 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER8
18:16	Reserved	Must be kept at reset value
15	ADC2RST	ADC2 reset This bit is set and reset by software. 0: No reset 1: Reset the ADC2
14	USART0RST	USART0 Reset This bit is set and reset by software. 0: No reset 1: Reset the USART0
13	TIMER7RST	Timer 7 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER7
12	SPI0RST	SPI0 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI0
11	TIMER0RST	Timer 0 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER0
10	ADC1RST	ADC1 reset This bit is set and reset by software. 0: No reset 1: Reset the ADC1
9	ADC0RST	ADC0 reset

		This bit is set and reset by software. 0: No reset 1: Reset the ADC0
8	PGRST	GPIO port G reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port G
7	PFRST	GPIO portF reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port F
6	PERST	GPIO port E reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port E
5	PDRST	GPIO port D reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port D
4	PCRST	GPIO port C reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port C
3	PBRST	GPIO port B reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port B
2	PARST	GPIO port A reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port A
1	Reserved	Must be kept at reset value
0	AFRST	Alternate function I/O reset This bit is set and reset by software. 0: No reset 1: Reset Alternate Function I/O

5.3.5. APB1 reset register (RCU_APB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PMURST	BKPIRST	Reserved	CAN0RST	Reserved	USBDRST	I2C1RST	I2C0RST	UART4RST	UART3RST	USART2RST	USART1RST	Reserved
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2RST	SPI1RST	Reserved		WWDGTRST	Reserved		TIMER13RST	TIMER12RST	TIMER11RST	TIMER6RST	TIMER5RST	TIMER4RST	TIMER3RST	TIMER2RST	TIMER1RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DACRST	DAC reset This bit is set and reset by software. 0: No reset 1: Reset DAC unit
28	PMURST	Power control reset This bit is set and reset by software. 0: No reset 1: Reset power control unit
27	BKPIRST	Backup interface reset This bit is set and reset by software. 0: No reset 1: Reset backup interface
26	Reserved	Must be kept at reset value
25	CAN0RST	CAN0 reset This bit is set and reset by software. 0: No reset 1: Reset the CAN0
24	Reserved	Must be kept at reset value
23	USBDRST	USBD reset This bit is set and reset by software. 0: No reset

		1: Reset the USBD
22	I2C1RST	I2C1 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C1
21	I2C0RST	I2C0 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C0
20	UART4RST	UART4 reset This bit is set and reset by software. 0: No reset 1: Reset the UART4
19	UART3RST	UART3 reset This bit is set and reset by software. 0: No reset 1: Reset the UART3
18	USART2RST	USART2 reset This bit is set and reset by software. 0: No reset 1: Reset the USART2
17	USART1RST	USART1 reset This bit is set and reset by software. 0: No reset 1: Reset the USART1
16	Reserved	Must be kept at reset value
15	SPI2RST	SPI2 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI2
14	SPI1RST	SPI1 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI1
13:12	Reserved	Must be kept at reset value
11	WWDGTRST	WWDGT reset This bit is set and reset by software.

		0: No reset 1: Reset the WWDGT
10:9	Reserved	Must be kept at reset value
8	TIMER13RST	TIMER13 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER13
7	TIMER12RST	TIMER12 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER12
6	TIMER11RST	TIMER11 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER11
5	TIMER6RST	TIMER6 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER6
4	TIMER5RST	TIMER5 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER5
3	TIMER4RST	TIMER4 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER4
2	TIMER3RST	TIMER3 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER3
1	TIMER2RST	TIMER2 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER2
0	TIMER1RST	TIMER1 reset This bit is set and reset by software. 0: No reset

1: Reset the TIMER1

5.3.6. AHB enable register (RCU_AHBEN)

Address offset: 0x14

Reset value: 0x0000 0014

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SDIOEN	Reserved	EXMCEN	Reserved	CRCEN	Reserved	FMCSPEN	Reserved	SRAMSPEN	DMA1EN	DMA0EN
					rw		rw		rw		rw		rw	rw	rw

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	SDIOEN	SDIO clock enable This bit is set and reset by software. 0: Disabled SDIO clock 1: Enabled SDIO clock
9	Reserved	Must be kept at reset value
8	EXMCEN	EXMC clock enable This bit is set and reset by software. 0: Disabled EXMC clock 1: Enabled EXMC clock
7	Reserved	Must be kept at reset value
6	CRCEN	CRC clock enable This bit is set and reset by software. 0: Disabled CRC clock 1: Enabled CRC clock
5	Reserved	Must be kept at reset value
4	FMCSPEN	FMC clock enable when sleep mode This bit is set and reset by software to enable/disable FMC clock during Sleep mode. 0: Disabled FMC clock during Sleep mode 1: Enabled FMC clock during Sleep mode

3	Reserved	Must be kept at reset value
2	SRAMSPEN	SRAM interface clock enable when sleep mode This bit is set and reset by software to enable/disable SRAM interface clock during Sleep mode. 0: Disabled SRAM interface clock during Sleep mode. 1: Enabled SRAM interface clock during Sleep mode
1	DMA1EN	DMA1 clock enable This bit is set and reset by software. 0: Disabled DMA1 clock 1: Enabled DMA1 clock
0	DMA0EN	DMA0 clock enable This bit is set and reset by software. 0: Disabled DMA0 clock 1: Enabled DMA0 clock

5.3.7. APB2 enable register (RCU_APB2EN)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIMER10EN	TIMER9EN	TIMER8EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2EN	USART0EN	TIMER7EN	SPI0EN	TIMER0EN	ADC1EN	ADC0EN	PGEN	PFEN	PEEN	PDEN	PCEN	PBEN	PAEN	Reserved	AFEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TIMER10EN	TIMER10 clock enable This bit is set and reset by software. 0: Disabled TIMER10 clock 1: Enabled TIMER10 clock
20	TIMER9EN	TIMER9 clock enable This bit is set and reset by software. 0: Disabled TIMER9 clock

		1: Enabled TIMER9 clock
19	TIMER8EN	TIMER8 clock enable This bit is set and reset by software. 0: Disabled TIMER8 clock 1: Enabled TIMER8 clock
18:16	Reserved	Must be kept at reset value
15	ADC2EN	ADC2 clock enable This bit is set and reset by software. 0: Disabled ADC2 clock 1: Enabled ADC2 clock
14	USART0EN	USART0 clock enable This bit is set and reset by software. 0: Disabled USART0 clock 1: Enabled USART0 clock
13	TIMER7EN	TIMER7 clock enable This bit is set and reset by software. 0: Disabled TIMER7 clock 1: Enabled TIMER7 clock
12	SPI0EN	SPI0 clock enable This bit is set and reset by software. 0: Disabled SPI0 clock 1: Enabled SPI0 clock
11	TIMER0EN	TIMER0 clock enable This bit is set and reset by software. 0: Disabled TIMER0 clock 1: Enabled TIMER0 clock
10	ADC1EN	ADC1 clock enable This bit is set and reset by software. 0: Disabled ADC1 clock 1: Enabled ADC1 clock
9	ADC0EN	ADC0 clock enable This bit is set and reset by software. 0: Disabled ADC0 clock 1: Enabled ADC0 clock
8	PGEN	GPIO port G clock enable This bit is set and reset by software. 0: Disabled GPIO port G clock

		1: Enabled GPIO port G clock
7	PFEN	GPIO port F clock enable This bit is set and reset by software. 0: Disabled GPIO port F clock 1: Enabled GPIO port F clock
6	PEEN	GPIO port E clock enable This bit is set and reset by software. 0: Disabled GPIO port E clock 1: Enabled GPIO port E clock
5	PDEN	GPIO port D clock enable This bit is set and reset by software. 0: Disabled GPIO port D clock 1: Enabled GPIO port D clock
4	PCEN	GPIO port C clock enable This bit is set and reset by software. 0: Disabled GPIO port C clock 1: Enabled GPIO port C clock
3	PBEN	GPIO port B clock enable This bit is set and reset by software. 0: Disabled GPIO port B clock 1: Enabled GPIO port B clock
2	PAEN	GPIO port A clock enable This bit is set and reset by software. 0: Disabled GPIO port A clock 1: Enabled GPIO port A clock
1	Reserved	Must be kept at reset value
0	AFEN	Alternate function IO clock enable This bit is set and reset by software. 0: Disabled Alternate Function IO clock 1: Enabled Alternate Function IO clock

5.3.8. APB1 enable register (RCU_APB1EN)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved		DACEN	PMUEN	BKPIEN	Reserved	CAN0EN	Reserved	USBDEN	I2C1EN	I2C0EN	UART4EN	UART3EN	USART2EN	USART1EN	Reserved
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2EN	SPI1EN	Reserved		WWDGTEN	Reserved		TIMER13EN	TIMER12EN	TIMER11EN	TIMER6EN	TIMER5EN	TIMER4EN	TIMER3EN	TIMER2EN	TIMER1EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DACEN	DAC clock enable This bit is set and reset by software. 0: Disabled DAC clock 1: Enabled DAC clock
28	PMUEN	PMU clock enable This bit is set and reset by software. 0: Disabled PMU clock 1: Enabled PMU clock
27	BKPIEN	Backup interface clock enable This bit is set and reset by software. 0: Disabled Backup interface clock 1: Enabled Backup interface clock
26	Reserved	Must be kept at reset value
25	CAN0EN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock 1: Enabled CAN0 clock
24	Reserved	Must be kept at reset value
23	USBDEN	USBD clock enable This bit is set and reset by software. 0: Disabled USBD clock 1: Enabled USBD clock
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable This bit is set and reset by software.

		0: Disabled I2C0 clock 1: Enabled I2C0 clock
20	UART4EN	UART4 clock enable This bit is set and reset by software. 0: Disabled UART4 clock 1: Enabled UART4 clock
19	UART3EN	UART3 clock enable This bit is set and reset by software. 0: Disabled UART3 clock 1: Enabled UART3 clock
18	USART2EN	USART2 clock enable This bit is set and reset by software. 0: Disabled USART2 clock 1: Enabled USART2 clock
17	USART1EN	USART1 clock enable This bit is set and reset by software. 0: Disabled USART1 clock 1: Enabled USART1 clock
16	Reserved	Must be kept at reset value
15	SPI2EN	SPI2 clock enable This bit is set and reset by software. 0: Disabled SPI2 clock 1: Enabled SPI2 clock
14	SPI1EN	SPI1 clock enable This bit is set and reset by software. 0: Disabled SPI1 clock 1: Enabled SPI1 clock
13:12	Reserved	Must be kept at reset value
11	WWDGTEN	WWDGT clock enable This bit is set and reset by software. 0: Disabled WWDGT clock 1: Enabled WWDGT clock
10:9	Reserved	Must be kept at reset value
8	TIMER13EN	TIMER13 clock enable This bit is set and reset by software. 0: Disabled TIMER13 clock 1: Enabled TIMER13 clock

7	TIMER12EN	TIMER12 clock enable This bit is set and reset by software. 0: Disabled TIMER12 clock 1: Enabled TIMER12 clock
6	TIMER11EN	TIMER11 clock enable This bit is set and reset by software. 0: Disabled TIMER11 clock 1: Enabled TIMER11 clock
5	TIMER6EN	TIMER6 clock enable This bit is set and reset by software. 0: Disabled TIMER6 clock 1: Enabled TIMER6 clock
4	TIMER5EN	TIMER5 clock enable This bit is set and reset by software. 0: Disabled TIMER5 clock 1: Enabled TIMER5 clock
3	TIMER4EN	TIMER4 clock enable This bit is set and reset by software. 0: Disabled TIMER4 clock 1: Enabled TIMER4 clock
2	TIMER3EN	TIMER3 clock enable This bit is set and reset by software. 0: Disabled TIMER3 clock 1: Enabled TIMER3 clock
1	TIMER2EN	TIMER2 clock enable This bit is set and reset by software. 0: Disabled TIMER2 clock 1: Enabled TIMER2 clock
0	TIMER1EN	TIMER1 clock enable This bit is set and reset by software. 0: Disabled TIMER1 clock 1: Enabled TIMER1 clock

5.3.9. Backup domain control register (RCU_BDCTL)

Address offset: 0x20
Reset value: 0x0000 0000, reset by Backup domain Reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

Note: The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the Backup domain control register (RCU_BDCTL) are only reset after a Backup domain Reset. These bits can be modified only when the BKPWEN bit in the Power control register (PMU_CTL) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BKPRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSRC[1:0]		Reserved					LXTALBPS	LXTALSTB	LXTALEN
rw						rw							rw	r	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets Backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value
9:8	RTCSRC[1:0]	RTC clock entry selection Set and reset by software to control the RTC clock source. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. 00: No clock selected 01: CK_LXTAL selected as RTC source clock 10: CK_IRC40K selected as RTC source clock 11: (CK_HXTAL / 128) selected as RTC source clock
7:3	Reserved	Must be kept at reset value
2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL Bypass mode 1: Enable the LXTAL Bypass mode
1	LXTALSTB	Low speed crystal oscillator stabilization flag Set by hardware to indicate if the LXTAL output clock is stable and ready for use. 0: LXTAL is not stable

Bits	Fields	Descriptions
		1: LXTAL is stable
0	LXTALEN	LXTAL enable Set and reset by software. 0: Disable LXTAL 1: Enable LXTAL

5.3.10. Reset source/clock register (RCU_RSTSCK)

Address offset: 0x24

Reset value: 0x0C00 0000, ALL reset flags reset by power Reset only, RSTFC/IRC40KEN reset by system reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP RSTF	WWDGT RSTF	FWDGT RSTF	SW RSTF	POR RSTF	EP RSTF	Reserved	RSTFC	Reserved							
r	r	r	r	r	r		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														IRC40K STB	IRC40KE N
														r	rw

Bits	Fields	Descriptions
31	LPRSTF	Low-power reset flag Set by hardware when Deep-sleep /standby reset generated. Reset by writing 1 to the RSTFC bit. 0: No Low-power management reset generated 1: Low-power management reset generated
30	WWDGTRSTF	Window watchdog timer reset flag Set by hardware when a window watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No window watchdog reset generated 1: Window watchdog reset generated
29	FWDGTRSTF	Free watchdog timer reset flag Set by hardware when a free watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No free watchdog timer reset generated 1: free Watchdog timer reset generated
28	SWRSTF	Software reset flag Set by hardware when a software reset generated.

		Reset by writing 1 to the RSTFC bit. 0: No software reset generated 1: Software reset generated
27	PORRSTF	Power reset flag Set by hardware when a Power reset generated. Reset by writing 1 to the RSTFC bit. 0: No Power reset generated 1: Power reset generated
26	EPRSTF	External PIN reset flag Set by hardware when an External PIN reset generated. Reset by writing 1 to the RSTFC bit. 0: No External PIN reset generated 1: External PIN reset generated
25	Reserved	Must be kept at reset value
24	RSTFC	Reset flag clear This bit is set by software to clear all reset flags. 0: Not clear reset flags 1: Clear reset flags
23:2	Reserved	Must be kept at reset value
1	IRC40KSTB	IRC40K stabilization flag Set by hardware to indicate if the IRC40K output clock is stable and ready for use. 0: IRC40K is not stable 1: IRC40K is stable
0	IRC40KEN	IRC40K enable Set and reset by software. 0: Disable IRC40K 1: Enable IRC40K

5.3.11. Deep-sleep mode voltage register (RCU_DSV)

Address offset: 0x34

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													DSLPVS[2:0]		

rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	DSLPVS[2:0]	Deep-sleep mode voltage select These bits are set and reset by software 000 : The core voltage is 1.2V in Deep-sleep mode 001 : The core voltage is 1.1V in Deep-sleep mode 010 : The core voltage is 1.0V in Deep-sleep mode 011 : The core voltage is 0.9V in Deep-sleep mode 1xx : Reserved

Connectivity line devices: Reset and clock control unit (RCU)

GD32F101xx and GD32F103xx microcontrollers where the flash memory density ranges between 16 and 128 Kbytes are called Medium-density devices (GD32F10x_MD).

GD32F101xx and GD32F103xx microcontrollers where the flash memory density ranges between 256 and 512 Kbytes are called High-density devices (GD32F10x_HD).

GD32F101xx and GD32F103xx microcontrollers where the flash memory density is over 512 Kbytes are called Extra-density devices (GD32F10x_XD).

GD32F105xx and GD32F107xx microcontrollers are called connectivity line devices (GD32F10x_CL).

5.4. Reset control unit (RCTL)

5.4.1. Overview

GD32F10x Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power reset, known as a cold reset, resets the full system except the Backup domain. The system reset resets the processor core and peripheral IP components except for the SW-DP controller and the Backup domain. The backup domain reset resets the Backup domain. The resets can be triggered by an external signal, internal events and the reset generators. More information about these resets will be described in the following sections.

5.4.2. Function overview

Power reset

The Power reset is generated by either an external reset as Power On and Power Down reset (POR/PDR reset) or by the internal reset generator when exiting Standby mode. The power reset sets all registers to their reset values except the Backup domain. The Power reset whose active signal is low, it will be de-asserted when the internal LDO voltage regulator is ready to provide 1.2V power. The RESET service routine vector is fixed at address 0x0000 0004 in the memory map.

System reset

A system reset is generated by the following events:

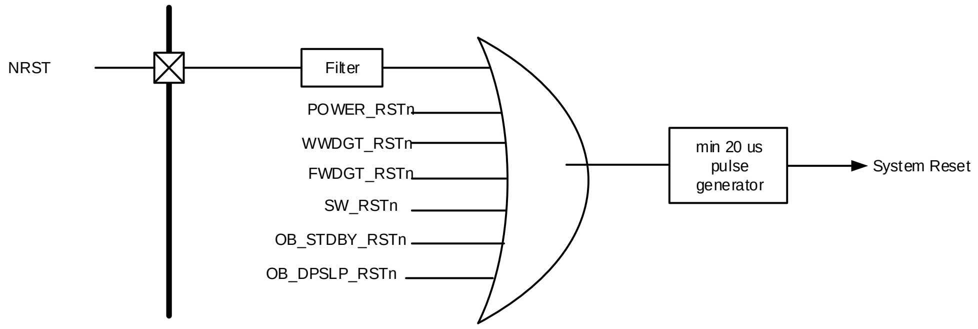
- A power reset (POWER_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT_RSTn).

- A free watchdog timer reset (FWDGT_RSTn).
- The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register is set (SW_RSTn).
- Reset generated when entering Standby mode when resetting nRST_STDBY bit in User Option Bytes (OB_STDBY_RSTn).
- Reset generated when entering Deep-sleep mode when resetting nRST_DPSLP bit in User Option Bytes (OB_DPSLP_RSTn).

A system reset resets the processor core and peripheral IP components except for the SW-DP controller and the Backup domain.

A system reset pulse generator guarantees low level pulse duration of 20 μs for each reset source (external or internal reset).

Figure 5-4. The system reset circuit



Backup domain reset

A backup domain reset is generated by setting the BKPRST bit in the Backup domain control register or Backup domain power on reset (V_{DD} or V_{BAT} power on, if both supplies have previously been powered off).

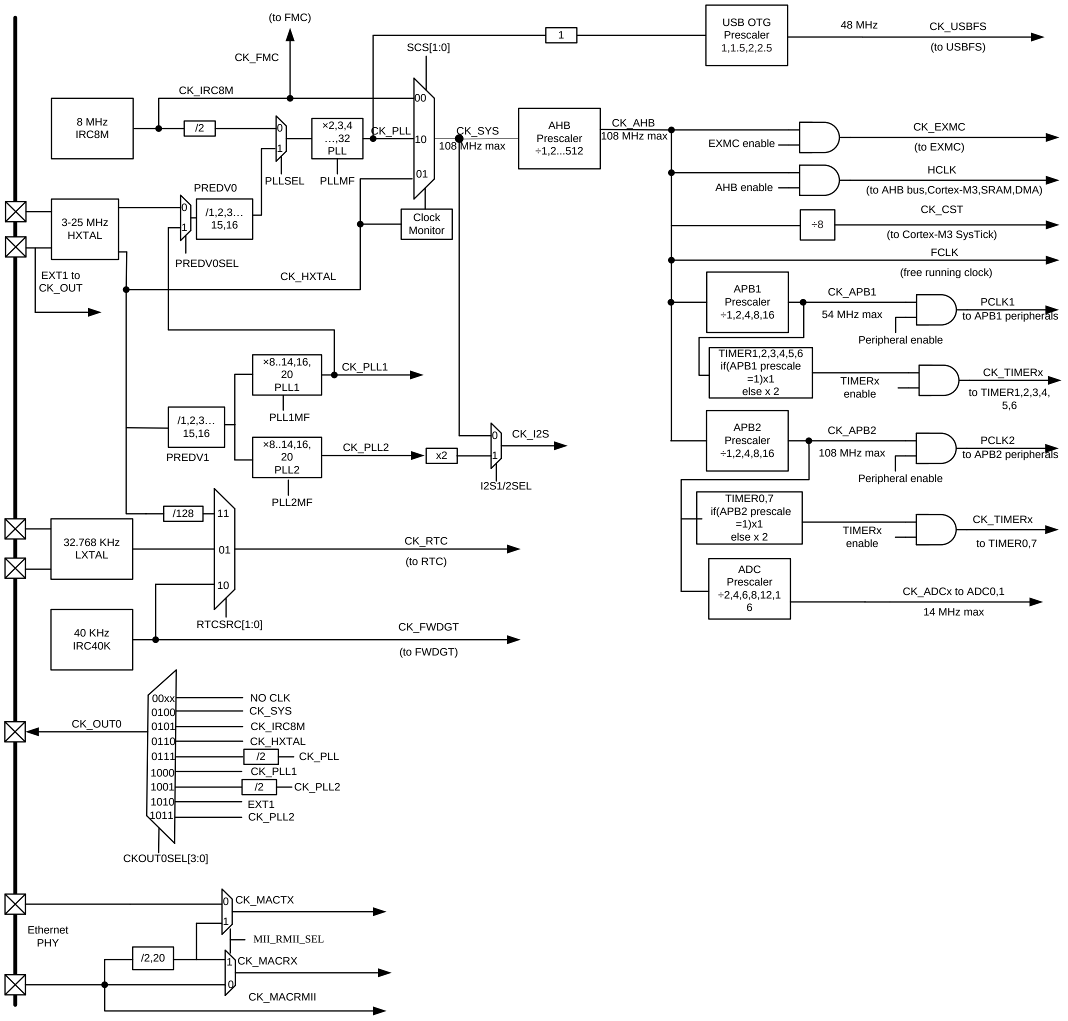
5.5. Clock control unit (CCTL)

5.5.1. Overview

The Clock Control unit provides a range of frequencies and clock functions. These include a Internal 8M RC oscillator (IRC8M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 40K RC oscillator (IRC40K), a Low Speed crystal oscillator (LXTAL), three Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex™-M3 are derived from the system clock (CK_SYS) which can source from the IRC8M, HXTAL or PLL. The maximum operating frequency of the system clock (CK_SYS) can be up to 108 MHz. The Free Watchdog Timer has independent clock source (IRC40K), and Real Time Clock (RTC) uses the IRC40K, LXTAL or HXTAL/128 as its clock source.

Figure 5-5. Clock tree



The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB, APB2 and APB1 domains is 108 MHz/108 MHz/54 MHz. The Cortex System Timer (SysTick) external clock is clocked with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the AHB clock (HCLK), configurable in the SysTick Control and Status Register.

The ADCs are clocked by the clock of APB2 divided by 2, 4, 6, 8, 12, 16.

The TIMERs are clocked by the clock divided from CK_APB2 and CK_APB1. The frequency of TIMERs clock is equal to CK_APBx(APB prescaler is 1), twice the CK_APBx(APB prescaler is not 1).

The USBD is clocked by the clock of CK_PLL as the clock source of 48MHz.

The I2S is clocked by the clock of CK_SYS or PLL2*2 which defined by I2SxSEL bit in RCU_CFG1 register.

The ENET TX/RX are clocked by External PIN (ENET_TX_CLK / ENET_RX_CLK), which select by ENET_PHY_SEL bit in AFIO_PCF0 register.

The RTC is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 128 (defined which select by RTCSRC bit in Backup Domain Control Register (RCU_BDCTL). After the RTC select HXTAL clock divided by 128, the clock disappeared when the 1.2V core domain power off. After the RTC select IRC40K, the clock disappeared when V_{DD} power off. After the RTC select LXTAL, the clock disappeared when V_{DD} and V_{BAT} power off.

The FWDGT is clocked by IRC40K clock, which is forced on when FWDGT started.

The FMC is clocked by IRC8M clock, which is forced on when IRC8M started.

5.5.2. Characteristics

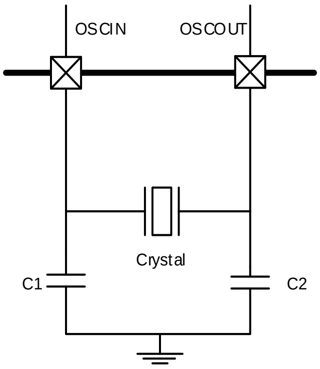
- 3 to 25 MHz High Speed crystal oscillator (HXTAL) .
- Internal 8 MHz RC oscillator (IRC8M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL or IRC8M.
- HXTAL clock monitor.

5.5.3. Function overview

High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 3 to 25 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 HXTAL pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

Figure 5-6. HXTAL clock source



The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the Control Register RCU_CTL. The HXTALSTB flag in Control Register RCU_CTL indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator “Start-up time”. As the HXTAL becomes stable, an interrupt will be generated if the related interrupt enable bit HXTALSTBIE in the Interrupt Register RCU_INT is set. At this point the HXTAL clock can be used directly as the system clock source or the PLL input clock.

Select external clock bypass mode by setting the HXTALBPS and HXTALEN bits in the Control Register RCU_CTL. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

Internal 8M RC oscillators (IRC8M)

The internal 8M RC oscillator, IRC8M, has a fixed frequency of 8 MHz and is the default clock source selection for the CPU when the device is powered up. The IRC8M oscillator provides a lower cost type clock source as no external components are required. The IRC8M RC oscillator can be switched on or off using the IRC8MEN bit in the Control Register RCU_CTL. The IRC8MSTB flag in the Control Register RCU_CTL is used to indicate if the internal 8M RC oscillator is stable. The start-up time of the IRC8M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC8MSTBIE, in the Clock Interrupt Register, RCU_INT, is set when the IRC8M becomes stable. The IRC8M clock can also be used as the system clock source or the PLL input clock.

The frequency accuracy of the IRC8M can be calibrated by the manufacturer, but its operating frequency is still less accurate than HXTAL. The application requirements, environment and cost will determine which oscillator type is selected.

If the HXTAL or PLL is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the IRC8M clock to be the system clock when the system initially wakes-up.

Phase locked loop (PLL)

There are three internal Phase Locked Loop, including PLL, PLL1 and PLL2.

The PLL can be switched on or off by using the PLLEN bit in the RCU_CTL Register. The PLLSTB flag in the RCU_CTL Register will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the RCU_INT Register, is set as the PLL becomes stable.

The PLL1 can be switched on or off by using the PLL1EN bit in the RCU_CTL Register. The PLL1STB flag in the RCU_CTL Register will indicate if the PLL1 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL1STBIE, in the RCU_INT Register, is set as the PLL1 becomes stable.

The PLL2 can be switched on or off by using the PLL2EN bit in the RCU_CTL Register. The PLL2STB flag in the RCU_CTL Register will indicate if the PLL2 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL2STBIE, in the RCU_INT Register, is set as the PLL2 becomes stable.

The three PLLs are closed by hardware when entering the Deepsleep/Standby mode or

HXTAL monitor fail when HXTAL used as the source clock of the PLLs.

Low speed crystal oscillator (LXTAL)

The low speed external crystal or ceramic resonator oscillator, which has a frequency of 32,768 Hz, produces a low power but highly accurate clock source for the Real Time Clock circuit. The LXTAL oscillator can be switched on or off using the LXTALEN bit in the Backup Domain Control Register (RCU_BDCTL). The LXTALSTB flag in the Backup Domain Control Register (RCU_BDCTL) will indicate if the LXTAL clock is stable. An interrupt can be generated if the related interrupt enable bit, LXTALSTBIE, in the Interrupt Register RCU_INT is set when the LXTAL becomes stable.

Select external clock bypass mode by setting the LXTALBPS and LXTALEN bits in the Backup Domain Control Register (RCU_BDCTL). The CK_LXTAL is equal to the external clock which drives the OSC32IN pin.

Internal 40K RC oscillator (IRC40K)

The internal RC oscillator has a frequency of about 40 kHz and is a low power clock source for the Real Time Clock circuit or the Free Watchdog Timer. The IRC40K offers a low cost clock source as no external components are required. The IRC40K RC oscillator can be switched on or off by using the IRC40KEN bit in the Reset source/clock Register (RCU_RSTSCK). The IRC40KSTB flag in the Reset source/clock Register RCU_RSTSCK will indicate if the IRC40K clock is stable. An interrupt can be generated if the related interrupt enable bit IRC40KSTBIE in the Clock Interrupt Register (RCU_INT) is set when the IRC40K becomes stable.

The IRC40K can be trimmed by TIMER4_CH3, user can get the clocks frequency, and adjust the RTC and FWDGT counter. Please refer to TIMER4CH3_IREMAP in AFIO_PCF0 register.

System clock (CK_SYS) selection

After the system reset, the default CK_SYS source will be IRC8M and can be switched to HXTAL or CK_PLL by changing the System Clock Switch bits, SCS, in the Clock configuration register 0, RCU_CFG0. When the SCS value is changed, the CK_SYS will continue to operate using the original clock source until the target clock source is stable. When a clock source is directly or indirectly (by PLL) used as the CK_SYS, it is not possible to stop it.

HXTAL clock monitor (CKM)

The HXTAL clock monitor function is enabled by the HXTAL Clock Monitor Enable bit, CKMEN, in the Control Register (RCU_CTL). This function should be enabled after the HXTAL start-up delay and disabled when the HXTAL is stopped. Once the HXTAL failure is detected, the HXTAL will be automatically disabled. The HXTAL Clock Stuck interrupt Flag, CKMIF, in the Clock Interrupt Register, RCU_INT, will be set and the HXTAL failure event will be generated. This failure interrupt is connected to the Non-Maskable Interrupt, NMI, of the

Cortex-M3. If the HXTAL is selected as the clock source of CK_SYS, PLL and CK_RTC, the HXTAL failure will force the CK_SYS source to IRC8M, the PLL will be disabled automatically. If the HXTAL is selected as the clock source of PLL, the HXTAL failure will force the PLL closed automatically. If the HXTAL is selected as the clock source of RTC, the HXTAL failure will reset the RTC clock selection.

Clock output capability

The clock output capability is ranging from 0.09375 MHz to 108 MHz. There are several clock signals can be selected via the CK_OUT0 Clock Source Selection bits, CKOUT0SEL, in the Clock Configuration Register 0 (RCU_CFG0). The corresponding GPIO pin should be configured in the properly Alternate Function I/O (AFIO) mode to output the selected clock signal..

Table 5-3. Clock output 0 source select

Clock Source 0 Selection bits	Clock Source
00xx	NO CLK
0100	CK_SYS
0101	CK_IRC8M
0110	CK_HXTAL
0111	CK_PLL/2
1000	CK_PLL1
1001	CK_PLL2/2
1010	EXT1
1011	CK_PLL2

Voltage control

The 1.2V domain voltage in Deep-sleep mode can be controlled by DSLPVS[1:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

Table 5-4. 1.2V domain voltage selected in deep-sleep mode

DSLPVS[1:0]	Deep-sleep mode voltage(V)
00	1.2
01	1.1
10	1.0
11	0.9

5.6. Register definition

5.6.1. Control register (RCU_CTL)

Address offset: 0x00

Reset value: 0x0000 xx83 where x is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLL2STB	PLL2EN	PLL1STB	PLL1EN	PLLSTB	PLLEN	Reserved				CKMEN	HXTALBPS	HXTALSTB	HXTALEN
		r	rw	r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRC8MCALIB[7:0]								IRC8MADJ[4:0]					Reserved	IRC8MSTB	IRC8MEN
r								rw						r	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLL2STB	PLL2 Clock Stabilization Flag Set by hardware to indicate if the PLL2 output clock is stable and ready for use. 0: PLL2 is not stable 1: PLL2 is stable
28	PLL2EN	PLL2 enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL2 is switched off 1: PLL2 is switched on
27	PLL1STB	PLL1 Clock Stabilization Flag Set by hardware to indicate if the PLL1 output clock is stable and ready for use. 0: PLL1 is not stable 1: PLL1 is stable
26	PLL1EN	PLL1 enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL1 is switched off 1: PLL1 is switched on
25	PLLSTB	PLL Clock Stabilization Flag Set by hardware to indicate if the PLL output clock is stable and ready for use. 0: PLL is not stable

		1: PLL is stable
24	PLLEN	PLL enable Set and reset by software. This bit cannot be reset if the PLL clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL is switched off 1: PLL is switched on
23:20	Reserved	Must be kept at reset value.
19	CKMEN	HXTAL Clock Monitor Enable 0: Disable the High speed 3 ~ 25 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 3 ~ 25 MHz crystal oscillator (HXTAL) clock monitor When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC8M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software. **Note:** When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC8M internal RC oscillator regardless of the control bit, IRC8MEN, state.
18	HXTALBPS	High speed crystal oscillator (HXTAL) clock bypass mode enable The HXTALBPS bit can be written only if the HXTALEN is 0. 0: Disable the HXTAL Bypass mode 1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock.
17	HXTALSTB	High speed crystal oscillator (HXTAL) clock stabilization flag Set by hardware to indicate if the HXTAL oscillator is stable and ready for use. 0: HXTAL oscillator is not stable 1: HXTAL oscillator is stable
16	HXTALEN	High Speed crystal oscillator (HXTAL) Enable Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL input clock when PLL clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: High speed 3 ~ 25 MHz crystal oscillator disabled 1: High speed 3 ~ 25 MHz crystal oscillator enabled
15:8	IRC8MCALIB[7:0]	Internal 8MHz RC Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC8MADJ[4:0]	Internal 8MHz RC Oscillator clock trim adjust value These bits are set by software. The trimming value is these bits (IRC8MADJ) added to the IRC8MCALIB[7:0] bits. The trimming value should trim the IRC8M to 8 MHz ± 1%.

2	Reserved	Must be kept at reset value.
1	IRC8MSTB	IRC8M Internal 8MHz RC Oscillator stabilization Flag Set by hardware to indicate if the IRC8M oscillator is stable and ready for use. 0: IRC8M oscillator is not stable 1: IRC8M oscillator is stable
0	IRC8MEN	Internal 8MHz RC oscillator Enable Set and reset by software. This bit cannot be reset if the IRC8M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when CKMEN is set. 0: Internal 8 MHz RC oscillator disabled 1: Internal 8 MHz RC oscillator enabled

5.6.2. Clock configuration register 0 (RCU_CFG0)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLLMF[4]	ADCPSC[2]	CKOUT0SEL[3:0]				USBFSPSC[1:0]		PLLMF[3:0]				PREDV0_LSB	PLLSEL
		rw	rw	rw				rw		rw				rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]			APB1PSC[2:0]			AHBPSC[3:0]				SCSS[1:0]		SCS[1:0]	
rw		rw			rw			rw				r		rw	

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLLMF[4]	Bit 4 of PLLMF see bits 21:18 of RCU_CFG0
28	ADCPSC[2]	Bit 2 of ADCPSC see bits 15:14 of RCU_CFG0
27:24	CKOUT0SEL[3:0]	CKOUT0 Clock Source Selection Set and reset by software. 00xx: No clock selected 0100: System clock selected 0101: High Speed 8M Internal Oscillator clock selected 0110: External High Speed oscillator clock selected 0111: (CK_PLL / 2) clock selected 1000: CK_PLL1 clock selected

		1001: CK_PLL2 clock divided by 2 selected 1010: EXT1 selected, to provide the external clock for ENET 1011: CK_PLL2 clock selected
23:22	USBFSPSC[1:0]	USBFS clock prescaler selection Set and reset by software to control the USBFS clock prescaler value. The USBFS clock must be 48MHz. These bits can’t be reset if the USBFS clock is enabled. 00: CK_USBFS = CK_PLL / 1.5 01: CK_USBFS = CK_PLL 10: CK_USBFS = CK_PLL / 2.5 11: CK_USBFS = CK_PLL / 2
21:18	PLLMF[3:0]	The PLL clock multiplication factor Bit 29 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor **Caution**: The PLL output frequency must not exceed 108 MHz 00000: (PLL source clock x 2) 00001: (PLL source clock x 3) 00010: (PLL source clock x 4) 00011: (PLL source clock x 5) 00100: (PLL source clock x 6) 00101: (PLL source clock x 7) 00110: (PLL source clock x 8) 00111: (PLL source clock x 9) 01000: (PLL source clock x 10) 01001: (PLL source clock x 11) 01010: (PLL source clock x 12) 01011: (PLL source clock x 13) 01100: (PLL source clock x 14) 01101: (PLL source clock x 6.5) 01110: (PLL source clock x 16) 01111: (PLL source clock x 16) 10000: (PLL source clock x 17) 10001: (PLL source clock x 18) 10010: (PLL source clock x 19) 10011: (PLL source clock x 20) 10100: (PLL source clock x 21) 10101: (PLL source clock x 22) 10110: (PLL source clock x 23) 10111: (PLL source clock x 24) 11000: (PLL source clock x 25) 11001: (PLL source clock x 26) 11010: (PLL source clock x 27) 11011: (PLL source clock x 28)

		11100: (PLL source clock x 29) 11101: (PLL source clock x 30) 11110: (PLL source clock x 31) 11111: (PLL source clock x 32)
17	PREDV0_LSB	The LSB of PREDV0 division factor This bit is the same bit as PREDV0 division factor bit [0] from RCU_CFG1. Changing the PREDV0 division factor bit [0] from RCU_CFG1, this bit is also changed. When the PREDV0 division factor bits [3:1] are not set, this bit controls PREDV0 input clock divided by 2 or not.
16	PLLSEL	PLL Clock Source Selection Set and reset by software to control the PLL clock source. 0: (IRC8M / 2) clock selected as source clock of PLL 1: HXTAL selected as source clock of PLL
15:14	ADCPSC[1:0]	ADC clock prescaler selection These bits and bit 28 of RCU_CFG0 are written by software to define the ADC prescaler factor.Set and cleared by software. 000: (CK_APB2 / 2) selected 001: (CK_APB2 / 4) selected 010: (CK_APB2 / 6) selected 011: (CK_APB2 / 8) selected 100: (CK_APB2 / 2) selected 101: (CK_APB2 / 12) selected 110: (CK_APB2 / 8) selected 111: (CK_APB2 / 16) selected
13:11	APB2PSC[2:0]	APB2 prescaler selection Set and reset by software to control the APB2 clock division ratio. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
10:8	APB1PSC[2:0]	APB1 prescaler selection Set and reset by software to control the APB1 clock division ratio. Caution: The CK_APB1 output frequency must not exceed 60 MHz. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
7:4	AHBPSC[3:0]	AHB prescaler selection

		Set and reset by software to control the AHB clock division ratio 0xxx: CK_SYS selected 1000: (CK_SYS / 2) selected 1001: (CK_SYS / 4) selected 1010: (CK_SYS / 8) selected 1011: (CK_SYS / 16) selected 1100: (CK_SYS / 64) selected 1101: (CK_SYS / 128) selected 1110: (CK_SYS / 256) selected 1111: (CK_SYS / 512) selected
3:2	SCSS[1:0]	System clock switch status Set and reset by hardware to indicate the clock source of system clock. 00: select CK_IRC8M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLL as the CK_SYS source 11: reserved
1:0	SCS[1:0]	System clock switch Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC8M when leaving Deep-sleep and Standby mode or HXTAL failure is detected by HXTAL clock monitor when HXTAL is selected directly or indirectly as the clock source of CK_SYS 00: select CK_IRC8M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLL as the CK_SYS source 11: reserved

5.6.3. Clock interrupt register (RCU_INT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKMIC	PLL2 STBIC	PLL1 STBIC	PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40K STBIC
								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLL2 STBIE	PLL1 STBIE	PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	PLL2 STBIF	PLL1 STBIF	PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	CKMIC	HXTAL Clock Stuck Interrupt Clear Write 1 by software to reset the CKMIF flag. 0: Not reset CKMIF flag 1: Reset CKMIF flag
22	PLL2STBIC	PLL2 stabilization Interrupt Clear Write 1 by software to reset the PLL2STBIF flag. 0: Not reset PLL2STBIF flag 1: Reset PLL2STBIF flag
21	PLL1STBIC	PLL1 stabilization Interrupt Clear Write 1 by software to reset the PLL1STBIF flag. 0: Not reset PLL1STBIF flag 1: Reset PLL1STBIF flag
20	PLLSTBIC	PLL stabilization Interrupt Clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL Stabilization Interrupt Clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M Stabilization Interrupt Clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL Stabilization Interrupt Clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag
16	IRC40KSTBIC	IRC40K Stabilization Interrupt Clear Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15	Reserved	Must be kept at reset value
14	PLL2STBIE	PLL2 Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL2 stabilization interrupt.

		0: Disable the PLL2 stabilization interrupt 1: Enable the PLL2 stabilization interrupt
13	PLL1STBIE	PLL1 Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL1 stabilization interrupt. 0: Disable the PLL1 stabilization interrupt 1: Enable the PLL1 stabilization interrupt
12	PLLSTBIE	PLL Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL stabilization interrupt. 0: Disable the PLL stabilization interrupt 1: Enable the PLL stabilization interrupt
11	HXTALSTBIE	HXTAL Stabilization Interrupt Enable Set and reset by software to enable/disable the HXTAL stabilization interrupt 0: Disable the HXTAL stabilization interrupt 1: Enable the HXTAL stabilization interrupt
10	IRC8MSTBIE	IRC8M Stabilization Interrupt Enable Set and reset by software to enable/disable the IRC8M stabilization interrupt 0: Disable the IRC8M stabilization interrupt 1: Enable the IRC8M stabilization interrupt
9	LXTALSTBIE	LXTAL Stabilization Interrupt Enable LXTAL stabilization interrupt enable/disable control 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC40KSTBIE	IRC40K Stabilization interrupt enable IRC40K stabilization interrupt enable/disable control 0: Disable the IRC40K stabilization interrupt 1: Enable the IRC40K stabilization interrupt
7	CKMIF	HXTAL Clock Stuck Interrupt Flag Set by hardware when the HXTAL clock is stuck. Reset when setting the CKMIC bit by software. 0: Clock operating normally 1: HXTAL clock stuck
6	PLL2STBIF	PLL2 stabilization interrupt flag Set by hardware when the PLL2 is stable and the PLL2STBIE bit is set. Reset when setting the PLL2STBIC bit by software. 0: No PLL2 stabilization interrupt generated 1: PLL2 stabilization interrupt generated
5	PLL1STBIF	PLL1 stabilization interrupt flag Set by hardware when the PLL1 is stable and the PLL1STBIE bit is set.

		Reset when setting the PLL1STBIC bit by software. 0: No PLL1 stabilization interrupt generated 1: PLL1 stabilization interrupt generated
4	PLLSTBIF	PLL stabilization interrupt flag Set by hardware when the PLL is stable and the PLLSTBIE bit is set. Reset when setting the PLLSTBIC bit by software. 0: No PLL stabilization interrupt generated 1: PLL stabilization interrupt generated
3	HXTALSTBIF	HXTAL stabilization interrupt flag Set by hardware when the High speed 3 ~ 25 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set. Reset when setting the HXTALSTBIC bit by software. 0: No HXTAL stabilization interrupt generated 1: HXTAL stabilization interrupt generated
2	IRC8MSTBIF	IRC8M stabilization interrupt flag Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set. Reset when setting the IRC8MSTBIC bit by software. 0: No IRC8M stabilization interrupt generated 1: IRC8M stabilization interrupt generated
1	LXTALSTBIF	LXTAL stabilization interrupt flag Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set. Reset when setting the LXTALSTBIC bit by software. 0: No LXTAL stabilization interrupt generated 1: LXTAL stabilization interrupt generated
0	IRC40KSTBIF	IRC40K stabilization interrupt flag Set by hardware when the Internal 40kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set. Reset when setting the IRC40KSTBIC bit by software. 0: No IRC40K stabilization clock ready interrupt generated 1: IRC40K stabilization interrupt generated

5.6.4. APB2 reset register (RCU_APB2RST)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART0RST	TIMER7RST	SPI0RST	TIMER0RST	ADC1RST	ADC0RST	PGRST	PFRST	PERST	PDRST	PCRST	PBRST	PARST	Reserved	AFRST
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	USART0RST	USART0 Reset This bit is set and reset by software. 0: No reset 1: Reset the USART0
13	TIMER7RST	Timer 7 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER7
12	SPI0RST	SPI0 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI0
11	TIMER0RST	Timer 0 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER0
10	ADC1RST	ADC1 reset This bit is set and reset by software. 0: No reset 1: Reset the ADC1
9	ADC0RST	ADC0 reset This bit is set and reset by software. 0: No reset 1: Reset the ADC0
8	PGRST	GPIO port G reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port G
7	PFRST	GPIO port F reset

		This bit is set and reset by software. 0: No reset 1: Reset the GPIO port F
6	PERST	GPIO port E reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port E
5	PDRST	GPIO port D reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port D
4	PCRST	GPIO port C reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port C
3	PBRST	GPIO port B reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port B
2	PARST	GPIO port A reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port A
1	Reserved	Must be kept at reset value
0	AFRST	Alternate function I/O reset This bit is set and reset by software. 0: No reset 1: Reset Alternate Function I/O

5.6.5. APB1 reset register (RCU_APB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PMURST	BKPIRST	CAN1RST	CAN0RST	Reserved		I2C1RST	I2C0RST	UART4RST	UART3RST	USART2RST	USART1RST	Reserved
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2RST	SPI1RST	Reserved		WWDGT RST	Reserved					TIMER6R ST	TIMER5R ST	TIMER4R ST	TIMER3R ST	TIMER2R ST	TIMER1R ST
rw	rw			rw						rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DACRST	DAC reset This bit is set and reset by software. 0: No reset 1: Reset DAC unit
28	PMURST	Power control reset This bit is set and reset by software. 0: No reset 1: Reset power control unit
27	BKPIRST	Backup interface reset This bit is set and reset by software. 0: No reset 1: Reset backup interface
26	CAN1RST	CAN1 reset This bit is set and reset by software. 0: No reset 1: Reset the CAN1
25	CAN0RST	CAN0 reset This bit is set and reset by software. 0: No reset 1: Reset the CAN0
24:23	Reserved	Must be kept at reset value
22	I2C1RST	I2C1 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C1
21	I2C0RST	I2C0 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C0
20	UART4RST	UART4 reset This bit is set and reset by software.

		0: No reset 1: Reset the UART4
19	UART3RST	UART3 reset This bit is set and reset by software. 0: No reset 1: Reset the UART3
18	USART2RST	USART2 reset This bit is set and reset by software. 0: No reset 1: Reset the USART2
17	USART1RST	USART1 reset This bit is set and reset by software. 0: No reset 1: Reset the USART1
16	Reserved	Must be kept at reset value
15	SPI2RST	SPI2 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI2
14	SPI1RST	SPI1 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI1
13:12	Reserved	Must be kept at reset value
11	WWDGTRST	WWDGT reset This bit is set and reset by software. 0: No reset 1: Reset the WWDGT
10:6	Reserved	Must be kept at reset value
5	TIMER6RST	TIMER6 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER6
4	TIMER5RST	TIMER5 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER5

3	TIMER4RST	TIMER4 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER4
2	TIMER3RST	TIMER3 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER3
1	TIMER2RST	TIMER2 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER2
0	TIMER1RST	TIMER1 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER1

5.6.6. AHB enable register (RCU_AHBEN)

Address offset: 0x14

Reset value: 0x0000 0014

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															ENETRXEN
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENETTXEN	ENETEN	Reserved	USBFSEN	Reserved			EXMCEN	Reserved	CRCEN	Reserved	FMCSPEN	Reserved	SRAMSPEN	DMA1EN	DMA0EN
rw	rw		rw				rw		rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	ENETRXEN	Ethernet RX clock enable This bit is set and reset by software. 0: Disabled Ethernet RX clock 1: Enabled Ethernet RX clock
15	ENETTXEN	Ethernet TX clock enable This bit is set and reset by software.

		0: Disabled Ethernet TX clock 1: Enabled Ethernet TX clock
14	ENETEN	Ethernet clock enable This bit is set and reset by software. 0: Disabled Ethernet clock 1: Enabled Ethernet clock
13	Reserved	Must be kept at reset value
12	USBFSEN	USBFS clock enable This bit is set and reset by software. 0: Disabled USBFS clock 1: Enabled USBFS clock
11:9	Reserved	Must be kept at reset value
8	EXMCEN	EXMC clock enable This bit is set and reset by software. 0: Disabled EXMC clock 1: Enabled EXMC clock
7	Reserved	Must be kept at reset value
6	CRCEN	CRC clock enable This bit is set and reset by software. 0: Disabled CRC clock 1: Enabled CRC clock
5	Reserved	Must be kept at reset value
4	FMCSPEN	FMC clock enable when sleep mode This bit is set and reset by software to enable/disable FMC clock during Sleep mode. 0: Disabled FMC clock during Sleep mode 1: Enabled FMC clock during Sleep mode
3	Reserved	Must be kept at reset value
2	SRAMSPEN	SRAM interface clock enable when sleep mode This bit is set and reset by software to enable/disable SRAM interface clock during Sleep mode. 0: Disabled SRAM interface clock during Sleep mode. 1: Enabled SRAM interface clock during Sleep mode
1	DMA1EN	DMA1 clock enable This bit is set and reset by software. 0: Disabled DMA1 clock

		1: Enabled DMA1 clock
0	DMA0EN	DMA0 clock enable This bit is set and reset by software. 0: Disabled DMA0 clock 1: Enabled DMA0 clock

5.6.7. APB2 enable register (RCU_APB2EN)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART0EN	TIMER7EN	SPI0EN	TIMER0EN	ADC1EN	ADC0EN	PGEN	PFEN	PEEN	PDEN	PCEN	PBEN	PAEN	Reserved	AFEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	USART0EN	USART0 clock enable This bit is set and reset by software. 0: Disabled USART0 clock 1: Enabled USART0 clock
13	TIMER7EN	TIMER7 clock enable This bit is set and reset by software. 0: Disabled TIMER7 clock 1: Enabled TIMER7 clock
12	SPI0EN	SPI0 clock enable This bit is set and reset by software. 0: Disabled SPI0 clock 1: Enabled SPI0 clock
11	TIMER0EN	TIMER0 clock enable This bit is set and reset by software. 0: Disabled TIMER0 clock 1: Enabled TIMER0 clock
10	ADC1EN	ADC1 clock enable

		This bit is set and reset by software. 0: Disabled ADC1 clock 1: Enabled ADC1 clock
9	ADC0EN	ADC0 clock enable This bit is set and reset by software. 0: Disabled ADC0 clock 1: Enabled ADC0 clock
8	PGEN	GPIO port G clock enable This bit is set and reset by software. 0: Disabled GPIO port G clock 1: Enabled GPIO port G clock
7	PFEN	GPIO port F clock enable This bit is set and reset by software. 0: Disabled GPIO port F clock 1: Enabled GPIO port F clock
6	PEEN	GPIO port E clock enable This bit is set and reset by software. 0: Disabled GPIO port E clock 1: Enabled GPIO port E clock
5	PDEN	GPIO port D clock enable This bit is set and reset by software. 0: Disabled GPIO port D clock 1: Enabled GPIO port D clock
4	PCEN	GPIO port C clock enable This bit is set and reset by software. 0: Disabled GPIO port C clock 1: Enabled GPIO port C clock
3	PBEN	GPIO port B clock enable This bit is set and reset by software. 0: Disabled GPIO port B clock 1: Enabled GPIO port B clock
2	PAEN	GPIO port A clock enable This bit is set and reset by software. 0: Disabled GPIO port A clock 1: Enabled GPIO port A clock
1	Reserved	Must be kept at reset value
0	AFEN	Alternate function IO clock enable This bit is set and reset by software.

0: Disabled Alternate Function IO clock
1: Enabled Alternate Function IO clock

5.6.8. APB1 enable register (RCU_APB1EN)

Address offset: 0x1C
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACEN	PMUEN	BKPIEN	CAN1EN	CAN0EN	Reserved		I2C1EN	I2C0EN	UART4EN	UART3EN	USART2EN	USART1EN	Reserved
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2EN	SPI1EN	Reserved		WWDGTEN	Reserved					TIMER6EN	TIMER5EN	TIMER4EN	TIMER3EN	TIMER2EN	TIMER1EN
rw	rw			rw						rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DACEN	DAC clock enable This bit is set and reset by software. 0: Disabled DAC clock 1: Enabled DAC clock
28	PMUEN	PMU clock enable This bit is set and reset by software. 0: Disabled PMU clock 1: Enabled PMU clock
27	BKPIEN	Backup interface clock enable This bit is set and reset by software. 0: Disabled Backup interface clock 1: Enabled Backup interface clock
26	CAN1EN	CAN1 clock enable This bit is set and reset by software. 0: Disabled CAN1 clock 1: Enabled CAN1 clock
25	CAN0EN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock

		1: Enabled CAN0 clock
24:23	Reserved	Must be kept at reset value
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable This bit is set and reset by software. 0: Disabled I2C0 clock 1: Enabled I2C0 clock
20	UART4EN	UART4 clock enable This bit is set and reset by software. 0: Disabled UART4 clock 1: Enabled UART4 clock
19	UART3EN	UART3 clock enable This bit is set and reset by software. 0: Disabled UART3 clock 1: Enabled UART3 clock
18	USART2EN	USART2 clock enable This bit is set and reset by software. 0: Disabled USART2 clock 1: Enabled USART2 clock
17	USART1EN	USART1 clock enable This bit is set and reset by software. 0: Disabled USART1 clock 1: Enabled USART1 clock
16	Reserved	Must be kept at reset value
15	SPI2EN	SPI2 clock enable This bit is set and reset by software. 0: Disabled SPI2 clock 1: Enabled SPI2 clock
14	SPI1EN	SPI1 clock enable This bit is set and reset by software. 0: Disabled SPI1 clock 1: Enabled SPI1 clock
13:12	Reserved	Must be kept at reset value
11	WWDGTEN	WWDGT clock enable

		This bit is set and reset by software. 0: Disabled WWDGT clock 1: Enabled WWDGT clock
10:6	Reserved	Must be kept at reset value
5	TIMER6EN	TIMER6 clock enable This bit is set and reset by software. 0: Disabled TIMER6 clock 1: Enabled TIMER6 clock
4	TIMER5EN	TIMER5 clock enable This bit is set and reset by software. 0: Disabled TIMER5 clock 1: Enabled TIMER5 clock
3	TIMER4EN	TIMER4 clock enable This bit is set and reset by software. 0: Disabled TIMER4 clock 1: Enabled TIMER4 clock
2	TIMER3EN	TIMER3 clock enable This bit is set and reset by software. 0: Disabled TIMER3 clock 1: Enabled TIMER3 clock
1	TIMER2EN	TIMER2 clock enable This bit is set and reset by software. 0: Disabled TIMER2 clock 1: Enabled TIMER2 clock
0	TIMER1EN	TIMER1 clock enable This bit is set and reset by software. 0: Disabled TIMER1 clock 1: Enabled TIMER1 clock

5.6.9. Backup domain control register (RCU_BDCTL)

Address offset: 0x20

Reset value: 0x0000 0018, reset by Backup domain Reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

Note: The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the Backup domain control register (RCU_BDCTL) are only reset after a Backup domain Reset. These bits can be modified only when the BKPWEN bit in the Power control register (PMU_CTL) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved	BKPRST
	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSRC[1:0]		Reserved					LXTALBPS	LXTALSTB	LXTALEN
rw						rw							rw	r	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets Backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value
9:8	RTCSRC[1:0]	RTC clock entry selection Set and reset by software to control the RTC clock source. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. 00: No clock selected 01: CK_LXTAL selected as RTC source clock 10: CK_IRC40K selected as RTC source clock 11: (CK_HXTAL / 128) selected as RTC source clock
7:3	Reserved	Must be kept at reset value
2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL Bypass mode 1: Enable the LXTAL Bypass mode
1	LXTALSTB	Low speed crystal oscillator stabilization flag Set by hardware to indicate if the LXTAL output clock is stable and ready for use. 0: LXTAL is not stable 1: LXTAL is stable
0	LXTALEN	LXTAL enable Set and reset by software. 0: Disable LXTAL

1: Enable LXTAL

5.6.10. Reset source/clock register (RCU_RSTSCK)

Address offset: 0x24

Reset value: 0x0C00 0000, ALL reset flags reset by power Reset only, RSTFC/IRC40KEN reset by system reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP RSTF	WWDGT RSTF	FWDGT RSTF	SW RSTF	POR RSTF	EP RSTF	Reserved	RSTFC	Reserved							
r	r	r	r	r	r		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														IRC40K STB	IRC40KE N
														r	rw

Bits	Fields	Descriptions
31	LPRSTF	Low-power reset flag Set by hardware when Deep-sleep /standby reset generated. Reset by writing 1 to the RSTFC bit. 0: No Low-power management reset generated 1: Low-power management reset generated
30	WWDGTRSTF	Window watchdog timer reset flag Set by hardware when a window watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No window watchdog reset generated 1: Window watchdog reset generated
29	FWDGTRSTF	Free watchdog timer reset flag Set by hardware when a free watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No free watchdog timer reset generated 1: free Watchdog timer reset generated
28	SWRSTF	Software reset flag Set by hardware when a software reset generated. Reset by writing 1 to the RSTFC bit. 0: No software reset generated 1: Software reset generated
27	PORRSTF	Power reset flag

		Set by hardware when a Power reset generated. Reset by writing 1 to the RSTFC bit. 0: No Power reset generated 1: Power reset generated
26	EPRSTF	External PIN reset flag Set by hardware when an External PIN reset generated. Reset by writing 1 to the RSTFC bit. 0: No External PIN reset generated 1: External PIN reset generated
25	Reserved	Must be kept at reset value
24	RSTFC	Reset flag clear This bit is set by software to clear all reset flags. 0: Not clear reset flags 1: Clear reset flags
23:2	Reserved	Must be kept at reset value
1	IRC40KSTB	IRC40K stabilization flag Set by hardware to indicate if the IRC40K output clock is stable and ready for use. 0: IRC40K is not stable 1: IRC40K is stable
0	IRC40KEN	IRC40K enable Set and reset by software. 0: Disable IRC40K 1: Enable IRC40K

5.6.11. AHB reset register (RCU_AHBRST)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ENETRST	Reserved	USBFSRST	Reserved											
	rw		rw												

Bits	Fields	Descriptions

31:15	Reserved	Must be kept at reset value
14	ENETRST	ENET reset This bit is set and reset by software. 0: No reset 1: Reset the ENET
13	Reserved	Must be kept at reset value
12	USBFSRST	USBFS reset This bit is set and reset by software. 0: No reset 1: Reset the USBFS
11:0	Reserved	Must be kept at reset value

5.6.12. Clock configuration register 1 (RCU_CFG1)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													I2S2SEL	I2S1SEL	PREDV0 SEL
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2MF[3:0]				PLL1MF[3:0]				PREDV1[3:0]				PREDV0[3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18	I2S2SEL	I2S2 Clock Source Selection Set and reset by software to control the I2S2 clock source. 0: System clock selected as I2S2 source clock 1: (CK_PLL2 x 2) selected as I2S2 source clock
17	I2S1SEL	I2S1 Clock Source Selection Set and reset by software to control the I2S1 clock source. 0: System clock selected as I2S1 source clock 1: (CK_PLL2 x 2) selected as I2S1 source clock
16	PREDV0SEL	PREDV0 input Clock Source Selection Set and reset by software. 0: HXTAL selected as PREDV0 input source clock

		1: CK_PLL1 selected as PREDV0 input source clock
15:12	PLL2MF[3:0]	The PLL2 clock multiplication factor Set and reset by software. 00xx: reserve 010x: reserve 0110: (PLL2 source clock x 8) 0111: (PLL2 source clock x 9) 1000 :(PLL2 source clock x 10) 1001: (PLL2 source clock x 11) 1010: (PLL2 source clock x 12) 1011: (PLL2 source clock x 13) 1100: (PLL2 source clock x 14) 1101: (PLL2 source clock x 15) 1110: (PLL2 source clock x 16) 1111: (PLL2 source clock x 20)
11:8	PLL1MF[3:0]	The PLL1 clock multiplication factor Set and reset by software. 00xx: reserve 010x: reserve 0110: (PLL1 source clock x 8) 0111: (PLL1 source clock x 9) 1000 :(PLL1 source clock x 10) 1001: (PLL1 source clock x 11) 1010: (PLL1 source clock x 12) 1011: (PLL1 source clock x 13) 1100: (PLL1 source clock x 14) 1101: (PLL1 source clock x 15) 1110 :(PLL1 source clock x 16) 1111: (PLL1 source clock x 20)
7:4	PREDV1[3:0]	PREDV1 division factor This bit is set and reset by software. These bits can be written when PLL1 and PLL2 are disable 0000: PREDV1 input source clock not divided 0001: PREDV1 input source clock divided by 2 0010: PREDV1 input source clock divided by 3 0011: PREDV1 input source clock divided by 4 0100: PREDV1 input source clock divided by 5 0101: PREDV1 input source clock divided by 6 0110: PREDV1 input source clock divided by 7 0111: PREDV1 input source clock divided by 8 1000: PREDV1 input source clock divided by 9

		1001: PREDV1 input source clock divided by 10 1010: PREDV1 input source clock divided by 11 1011: PREDV1 input source clock divided by 12 1100: PREDV1 input source clock divided by 13 1101: PREDV2 input source clock divided by 14 1110: PREDV2 input source clock divided by 15 1111: PREDV2 input source clock divided by 16
3:0	PREDV0[3:0]	PREDV0 division factor This bit is set and reset by software. These bits can be written when PLL is disable. **Note:** The bit 0 of PREDV0 is same as bit 17 of RCU_CFG0, so modifying Bit 17 of RCU_CFG0 also modifies bit 0 of RCU_CFG1. 0000: PREDV0 input source clock not divided 0001: PREDV0 input source clock divided by 2 0010: PREDV0 input source clock divided by 3 0011: PREDV0 input source clock divided by 4 0100: PREDV0 input source clock divided by 5 0101: PREDV0 input source clock divided by 6 0110: PREDV0 input source clock divided by 7 0111: PREDV0 input source clock divided by 8 1000: PREDV0 input source clock divided by 9 1001: PREDV0 input source clock divided by 10 1010: PREDV0 input source clock divided by 11 1011: PREDV0 input source clock divided by 12 1100: PREDV0 input source clock divided by 13 1101: PREDV0 input source clock divided by 14 1110: PREDV0 input source clock divided by 15 1111: PREDV0 input source clock divided by 16

5.6.13. Deep-sleep mode voltage register (RCU_DSV)

Address offset: 0x34

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions

31:3	Reserved	Must be kept at reset value
2:0	DSLPVS[2:0]	Deep-sleep mode voltage select These bits are set and reset by software 000 : The core voltage is 1.2V in Deep-sleep mode 001 : The core voltage is 1.1V in Deep-sleep mode 010 : The core voltage is 1.0V in Deep-sleep mode 011 : The core voltage is 0.9V in Deep-sleep mode 1xx : Reserved

6. Interrupt/event controller (EXTI)

6.1. Overview

Cortex-M3 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and controls power management. It’s tightly coupled to the processer core. You can read the Technical Reference Manual of Cortex-M3 for more details about NVIC.

EXTI (interrupt/event controller) contains up to 20 independent edge detectors and generates interrupt requests or events to the processer. 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.

6.2. Characteristics

- Cortex-M3 system exception.
- Up to 68 maskable peripheral interrupts.
- 4 bits interrupt priority configuration—16 priority levels.
- Efficient interrupt processing.
- Support exception pre-emption and tail-chaining.
- Wake up system from power saving mode.
- Up to 20 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

6.3. Function overview

The ARM Cortex-M3 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).

The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. The following tables list all exception types.

Table 6-1. NVIC exception types in Cortex-M3

Exception Type	Vector Number	Priority (a)	Vector Address	Description
-	0	-	0x0000_0000	Reserved
Reset	1	-3	0x0000_0004	Reset
NMI	2	-2	0x0000_0008	Non maskable interrupt.
HardFault	3	-1	0x0000_000C	All class of fault
MemManage	4	Programmable	0x0000_0010	Memory management
BusFault	5	Programmable	0x0000_0014	Prefetch fault, memory access fault
UsageFault	6	Programmable	0x0000_0018	Undefined instruction or illegal state
-	7-10	-	0x0000_001C - 0x0000_002B	Reserved
SVCall	11	Programmable	0x0000_002C	System service call via SWI instruction
Debug Monitor	12	Programmable	0x0000_0030	Debug Monitor
-	13	-	0x0000_0034	Reserved
PendSV	14	Programmable	0x0000_0038	Pendable request for system service
SysTick	15	Programmable	0x0000_003C	System tick timer

Table 6-2. Interrupt vector table

Interrupt Number	Vector Number	Non-connectivity devices Interrupt Description	Connectivity devices Interrupt Description	Vector Address
IRQ 0	16	WWDGT interrupt	WWDGT interrupt	0x0000_0040
IRQ 1	17	LVD from EXTI interrupt	LVD from EXTI interrupt	0x0000_0044
IRQ 2	18	Tamper interrupt	Tamper interrupt	0x0000_0048
IRQ 3	19	RTC global interrupt	RTC global interrupt	0x0000_004C
IRQ 4	20	FMC global interrupt	FMC global interrupt	0x0000_0050
IRQ 5	21	RCU global interrupt	RCU global interrupt	0x0000_0054
IRQ 6	22	EXTI Line0 interrupt	EXTI Line0 interrupt	0x0000_0058
IRQ 7	23	EXTI Line1 interrupt	EXTI Line1 interrupt	0x0000_005C
IRQ 8	24	EXTI Line2 interrupt	EXTI Line2 interrupt	0x0000_0060
IRQ 9	25	EXTI Line3 interrupt	EXTI Line3 interrupt	0x0000_0064
IRQ 10	26	EXTI Line4 interrupt	EXTI Line4 interrupt	0x0000_0068
IRQ 11	27	DMA0 channel0 global interrupt	DMA0 channel0 global interrupt	0x0000_006C
IRQ 12	28	DMA0 channel1 global interrupt	DMA0 channel1 global interrupt	0x0000_0070

Interrupt Number	Vector Number	Non-connectivity devices Interrupt Description	Connectivity devices Interrupt Description	Vector Address
IRQ 13	29	DMA0 channel2 global interrupt	DMA0 channel2 global interrupt	0x0000_0074
IRQ 14	30	DMA0 channel3 global interrupt	DMA0 channel3 global interrupt	0x0000_0078
IRQ 15	31	DMA0 channel4 global interrupt	DMA0 channel4 global interrupt	0x0000_007C
IRQ 16	32	DMA0 channel5 global interrupt	DMA0 channel5 global interrupt	0x0000_0080
IRQ 17	33	DMA0 channel6 global interrupt	DMA0 channel6 global interrupt	0x0000_0084
IRQ 18	34	ADC0 and ADC1 global interrupt	ADC0 and ADC1 global interrupt	0x0000_0088
IRQ 19	35	USBD High Priority or CAN0 TX interrupts	CAN0 TX interrupts	0x0000_008C
IRQ 20	36	USBD Low Priority or CAN0 RX0 interrupts	CAN0 RX0 interrupts	0x0000_0090
IRQ 21	37	CAN0 RX1 interrupts	CAN0 RX1 interrupts	0x0000_0094
IRQ 22	38	CAN0 EWMC interrupts	CAN0 EWMC interrupts	0x0000_0098
IRQ 23	39	EXTI line[9:5] interrupts	EXTI line[9:5] interrupts	0x0000_009C
IRQ 24	40	TIMER0 break interrupt and TIMER8 global interrupt	TIMER0 break interrupt	0x0000_00A0
IRQ 25	41	TIMER0 update interrupt and TIMER9 global interrupt	TIMER0 update interrupt	0x0000_00A4
IRQ 26	42	TIMER0 trigger and channel commutation interrupts and TIMER10 global interrupt	TIMER0 trigger and channel commutation interrupts	0x0000_00A8
IRQ 27	43	TIMER0 channel capture compare interrupt	TIMER0 channel capture compare interrupt	0x0000_00AC
IRQ 28	44	TIMER1 global interrupt	TIMER1 global interrupt	0x0000_00B0
IRQ 29	45	TIMER2 global interrupt	TIMER2 global interrupt	0x0000_00B4
IRQ 30	46	TIMER3 global interrupt	TIMER3 global interrupt	0x0000_00B8
IRQ 31	47	I2C0 event interrupt	I2C0 event interrupt	0x0000_00BC
IRQ 32	48	I2C0 error interrupt	I2C0 error interrupt	0x0000_00C0
IRQ 33	49	I2C1 event interrupt	I2C1 event interrupt	0x0000_00C4
IRQ 34	50	I2C1 error interrupt	I2C1 error interrupt	0x0000_00C8
IRQ 35	51	SPI0 global interrupt	SPI0 global interrupt	0x0000_00CC
IRQ 36	52	SPI1 global interrupt	SPI1 global interrupt	0x0000_00D0
IRQ 37	53	USART0 global interrupt	USART0 global interrupt	0x0000_00D4
IRQ 38	54	USART1 global interrupt	USART1 global interrupt	0x0000_00D8

Interrupt Number	Vector Number	Non-connectivity devices Interrupt Description	Connectivity devices Interrupt Description	Vector Address
IRQ 39	55	USART2 global interrupt	USART2 global interrupt	0x0000_00DC
IRQ 40	56	EXTI line[15:10] interrupts	EXTI line[15:10] interrupts	0x0000_00E0
IRQ 41	57	RTC alarm from EXTI interrupt	RTC alarm from EXTI interrupt	0x0000_00E4
IRQ 42	58	USBD wakeup from EXTI interrupt	USBFS wakeup from EXTI interrupt	0x0000_00E8
IRQ 43	59	TIMER7 break interrupt and TIMER11 global interrupt	TIMER7 break interrupt	0x0000_00EC
IRQ 44	60	TIMER7 update interrupt and TIMER12 global interrupt	TIMER7 update interrupt	0x0000_00F0
IRQ 45	61	TIMER7 trigger and channel commutation interrupts and TIMER13 global interrupt	TIMER7 trigger and channel commutation interrupts	0x0000_00F4
IRQ 46	62	TIMER7 channel capture compare interrupt	TIMER7 channel capture compare interrupt	0x0000_00F8
IRQ 47	63	ADC2 global interrupt	reserved	0x0000_00FC
IRQ 48	64	EXMC global interrupt	EXMC global interrupt	0x0000_0100
IRQ 49	65	SDIO global interrupt	reserved	0x0000_0104
IRQ50	66	TIMER4 global interrupt	TIMER4 global interrupt	0x0000_0108
IRQ51	67	SPI2 global interrupt	SPI2 global interrupt	0x0000_010C
IRQ52	68	UART3 global interrupt	UART3 global interrupt	0x0000_0110
IRQ53	69	UART4 global interrupt	UART4 global interrupt	0x0000_0114
IRQ54	70	TIMER5 global interrupt	TIMER5 global interrupt	0x0000_0118
IRQ55	71	TIMER6 global interrupt	TIMER6 global interrupt	0x0000_011C
IRQ56	72	DMA1 channel0 global interrupt	DMA1 channel0 global interrupt	0x0000_0120
IRQ57	73	DMA1 channel1 global interrupt	DMA1 channel1 global interrupt	0x0000_0124
IRQ58	74	DMA1 channel2 global interrupt	DMA1 channel2 global interrupt	0x0000_0128
IRQ59	75	DMA1 channel3 and DMA1 channel4 global interrupt	DMA1 channel3 global interrupt	0x0000_012C
IRQ60	76	reserved	DMA1 channel4 global interrupt	0x0000_0130
IRQ61	77	reserved	ENET global interrupt	0x0000_0134
IRQ62	78	reserved	ENET wakeup from EXTI interrupt	0x0000_0138
IRQ63	79	reserved	CAN1 TX interrupt	0x0000_013C
IRQ64	80	reserved	CAN1 RX0 interrupt	0x0000_0140

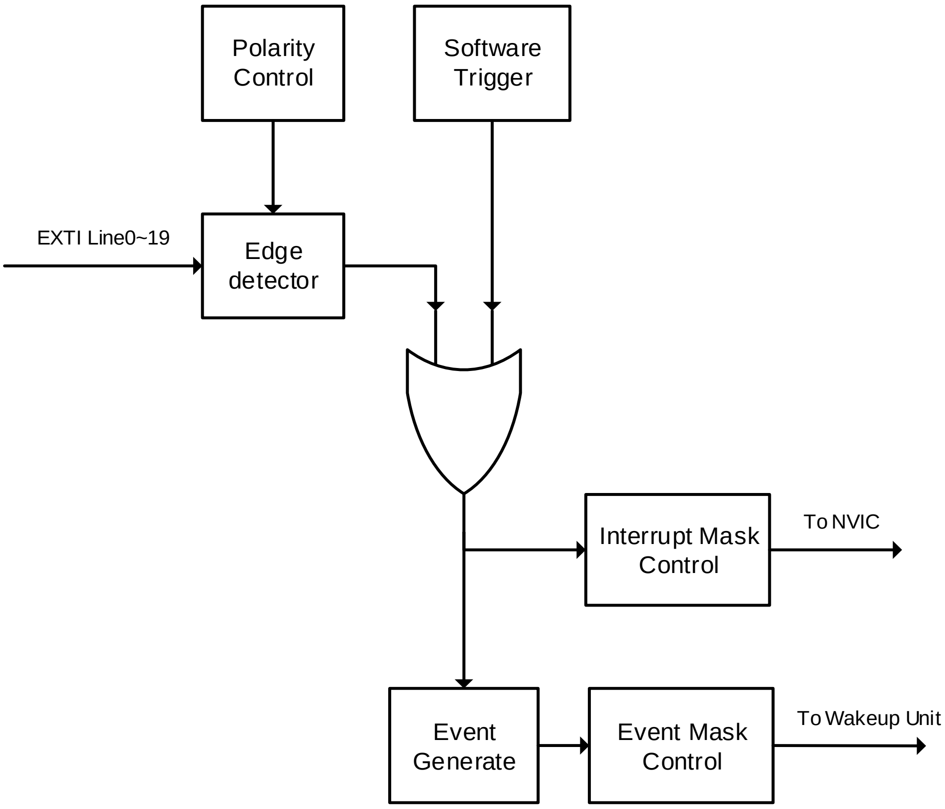
Interrupt Number	Vector Number	Non-connectivity devices Interrupt Description	Connectivity devices Interrupt Description	Vector Address
IRQ65	81	reserved	CAN1 RX1 interrupt	0x0000_0144
IRQ66	82	reserved	CAN1 EWMC interrupt	0x0000_0148
IRQ67	83	reserved	USBFS global interrupt	0x0000_014C

Note:

1. IRQ0 ~ 42 are available in MD devices, but when the flash memory is less than 64KB, IRQ30, IRQ33, IRQ34, IRQ36 and IRQ39 are not available.
2. IRQ0 ~ 59 are available in HD and XD devices, but the TIMER8 to TIMER13 global interrupts (IRQ24、IRQ25、IRQ26、IRQ43、IRQ44、IRQ45) are available only in the XD devices.
3. At non-connectivity devices, USB and CAN (IRQ19,IRQ 20) function cannot be used at the same time

6.4. External interrupt and event (EXTI) block diagram

Figure 6-1. Block diagram of EXTI



6.5. External Interrupt and Event function overview

The EXTI contains up to 20 independent edge detectors and generates interrupts request or event to the processer. 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, Ethernet Wakeup). All GPIO pins can be selected as an EXTI trigger source by configuring AFIO_EXTISSx registers in GPIO module (please refer to GPIO and AFIO section for detail).

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 6-3. EXTI source

EXTI Line Number	Source
0	PA0/PB0/PC0/PD0/PE0/PF0/PG0
1	PA1/PB1/PC1/PD1/PE1/PF1/PG1
2	PA2/PB2/PC2/PD2/PE2/PF2/PG2
3	PA3/PB3/PC3/PD3/PE3/PF3/PG3
4	PA4/PB4/PC4/PD4/PE4/PF4/PG4
5	PA5/PB5/PC5/PD5/PE5/PF5/PG5
6	PA6/PB6/PC6/PD6/PE6/PF6/PG6
7	PA7/PB7/PC7/PD7/PE7/PF7/PG7
8	PA8/PB8/PC8/PD8/PE8/PF8/PG8
9	PA9/PB9/PC9/PD9/PE9/PF9/PG9
10	PA10/PB10/PC10/PD10/PE10/PF10/PG10
11	PA11/PB11/PC11/PD11/PE11/PF11/PG11
12	PA12/PB12/PC12/PD12/PE12/PF12/PG12
13	PA13/PB13/PC13/PD13/PE13/PF13/PG13
14	PA14/PB14/PC14/PD14/PE14/PF14/PG14
15	PA15/PB15/PC15/PD15/PE15/PF15/PG15
16	LVD
17	RTC Alarm
18	USB Wakeup
19	Ethernet Wakeup

Note: The EXTI line19 is available only in the GD32F107xx device.

6.6. Register definition

6.6.1. Interrupt enable register (EXTI_INTEN)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												INTEN19	INTEN18	INTEN17	INTEN16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN15	INTEN14	INTEN13	INTEN12	INTEN11	INTEN10	INTEN9	INTEN8	INTEN7	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19: 0	INTENx	Interrupt enable bit 0: Interrupt from Linex is disabled. 1: Interrupt from Linex is enabled.

6.6.2. Event enable register (EXTI_EVEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												EVEN19	EVEN18	EVEN17	EVEN16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVEN15	EVEN14	EVEN13	EVEN12	EVEN11	EVEN10	EVEN9	EVEN8	EVEN7	EVEN6	EVEN5	EVEN4	EVEN3	EVEN2	EVEN1	EVEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19: 0	EVENx	Event enable bit 0: Event from Linex is disabled. 1: Event from Linex is enabled.

6.6.3. Rising edge trigger enable register (EXTI_RTEN)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												RTEN19	RTEN18	RTEN17	RTEN16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTEN15	RTEN14	RTEN13	RTEN12	RTEN11	RTEN10	RTEN9	RTEN8	RTEN7	RTEN6	RTEN5	RTEN4	RTEN3	RTEN2	RTEN1	RTEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19:0	RTENx	Rising edge trigger enable 0: Rising edge of Linex is invalid 1: Rising edge of Linex is valid as an interrupt/event request

6.6.4. Falling edge trigger enable register (EXTI_FTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												FTEN19	FTEN18	FTEN17	FTEN16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTEN15	FTEN14	FTEN13	FTEN12	FTEN11	FTEN10	FTEN9	FTEN8	FTEN7	FTEN6	FTEN5	FTEN4	FTEN3	FTEN2	FTEN1	FTEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31: 20	Reserved	Must be kept at reset value
19: 0	FTENx	Falling edge trigger enable 0: Falling edge of Linex is invalid 1: Falling edge of Linex is valid as an interrupt/event request

6.6.5. Software interrupt event register (EXTI_SWIEV)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												SWIEV19	SWIEV18	SWIEV17	SWIEV16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIEV15	SWIEV14	SWIEV13	SWIEV12	SWIEV11	SWIEV10	SWIEV9	SWIEV8	SWIEV7	SWIEV6	SWIEV5	SWIEV4	SWIEV3	SWIEV2	SWIEV1	SWIEV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19: 0	SWIEVx	Interrupt/Event software trigger 0: Deactivate the EXTIx software interrupt/event request 1: Activate the EXTIx software interrupt/event request

6.6.6. Pending register (EXTI_PD)

Address offset: 0x14
Reset value: undefined

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												PD19	PD18	PD17	PD16
												rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31: 20	Reserved	Must be kept at reset value
19: 0	PDx	Interrupt pending status 0: EXTI Linex is not triggered 1: EXTI Linex is triggered. This bit is cleared to 0 by writing 1 to it.

7. General-purpose and alternate-function I/Os (GPIO and AFIO)

7.1. Overview

There are up to 112 general purpose I/O pins (GPIO), named PA0 ~ PA15, PB0 ~ PB15, PC0 ~ PC15, PD0 ~ PD15, PE0 ~ PE15, PF0 ~ PF15 and PG0 ~ PG15 for the device to implement logic input/output functions. Each GPIO port has related control and configuration registers to satisfy the requirements of specific applications. The external interrupt on the GPIO pins of the device have related control and configuration registers in the Interrupt/event Controller Unit (EXTI).

The GPIO ports are pin-shared with other alternative functions (AFs) to obtain maximum flexibility on the package pins. The GPIO pins can be used as alternative functional pins by configuring the corresponding registers regardless of the AF input or output pins.

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), input, peripheral alternate function or analog mode. Each GPIO pin can be configured as pull-up, pull-down or no pull-up/pull-down. All GPIOs are high-current capable except for analog mode.

7.2. Characteristics

- Input/output direction control.
- Schmitt trigger input function enable control.
- Each pin weak pull-up/pull-down function.
- Output push-pull/open drain enable control.
- Output set/reset control.
- External interrupt with programmable trigger edge – using EXTI configuration registers.
- Analog input/output configuration.
- Alternate function input/output configuration.
- Port configuration lock.

7.3. Function overview

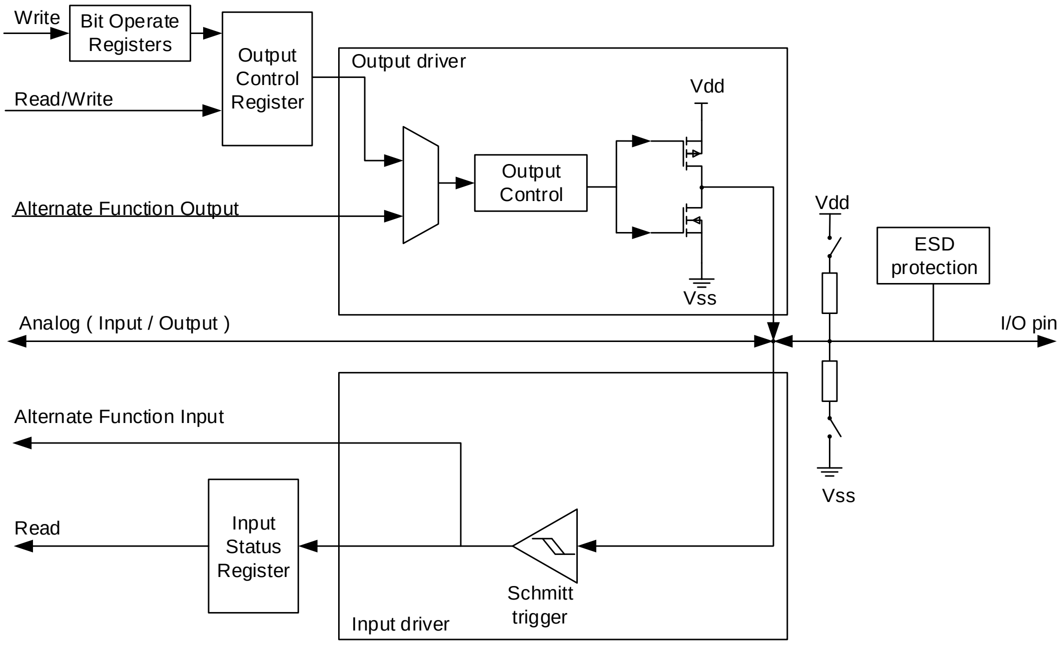
Each of the general-purpose I/O ports can be configured as 8 modes: analog inputs, input floating, input pull-down/pull-up, GPIO push-pull/open-drain or AFIO push-pull/open-drain mode by two GPIO configuration registers (GPIOx_CTL0/GPIOx_CTL1), and two 32-bits data registers (GPIOx_ISTAT and GPIOx_OCTL).***Table 7-1. GPIO configuration table*** shows the details.

Table 7-1. GPIO configuration table

Configuration mode		CTL[1:0]	MD[1:0]	OCTL
Input	Analog	00	0	don't care
	Input floating	01		don't care
	Input pull-down	10		0
	Input pull-up	10		1
General purpose Output (GPIO)	Push-pull	00	00: Reserved 01: Speed up to 10MHz 10: Speed up to 2MHz 11: Speed up to 50MHz	0 or 1
	Open-drain	01		0 or 1
Alternate Function Output (AFIO)	Push-pull	10		don't care
	Open-drain	11		don't care

Figure 7-1. Basic structure of a standard I/O port bit shows the basic structure of an I/O port bit.

Figure 7-1. Basic structure of a standard I/O port bit



7.3.1. GPIO pin configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured into the input floating mode that input disabled without Pull-Up (PU)/Pull-Down (PD) resistors. But the JTAG/Serial-Wired Debug pins are in input PU/PD mode after reset:

PA15: JTDI in PU mode.
PA14: JTCK / SWCLK in PD mode.
PA13: JTMS / SWDIO in PU mode.
PB4: NJTRST in PU mode.

PB3: JTDO in Floating mode.

The GPIO pins can be configured as inputs or outputs. When the GPIO pins are configured as input pins, all GPIO pins have an internal weak pull-up and weak pull-down which can be chosen. And the data on the external pins can be captured at every APB2 clock cycle to the port input status register (GPIOx_ISTAT).

When the GPIO pins are configured as output pins, user can configure the speed of the ports. And chooses the output driver mode: Push-Pull or Open-Drain mode. The value of the port output control register (GPIOx_OCTL) is output on the I/O pin.

There is no need to read-then-write when programming the GPIOx_OCTL at bit level, user can modify only one or several bits in a single atomic APB2 write access by programming ‘1’ to the bit operate register (GPIOx_BOP, or for clearing only GPIOx_BC). The other bits will not be affected.

7.3.2. External interrupt/event lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

7.3.3. Alternate functions (AF)

When the port is configured as AFIO (set CTLy bits to “0b10” or “0b11”, and set MDy bits to “0b01”, “0b10”, or “0b11”, which is in GPIOx_CTL0/GPIOx_CTL1 registers), the port is used as peripheral alternate functions. The detail alternate function assignments for each port are in the device datasheet.

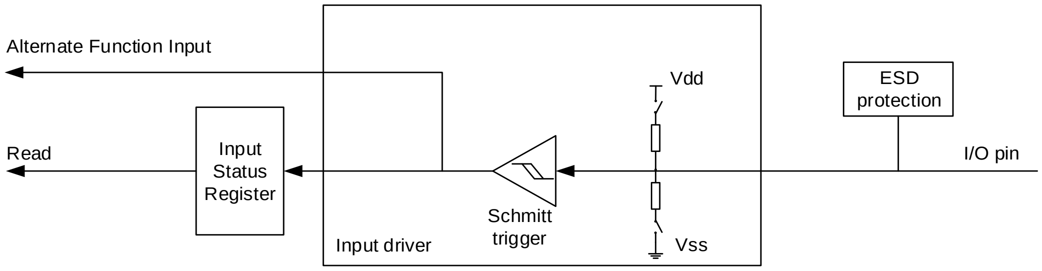
7.3.4. Input configuration

When GPIO pin is configured as Input:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen.
- Every APB2 clock cycle the data present on the I/O pin is got to the port input status Register.
- The output buffer is disabled.

Figure 7-2. Input configuration shows the input configuration.

Figure 7-2. Input configuration



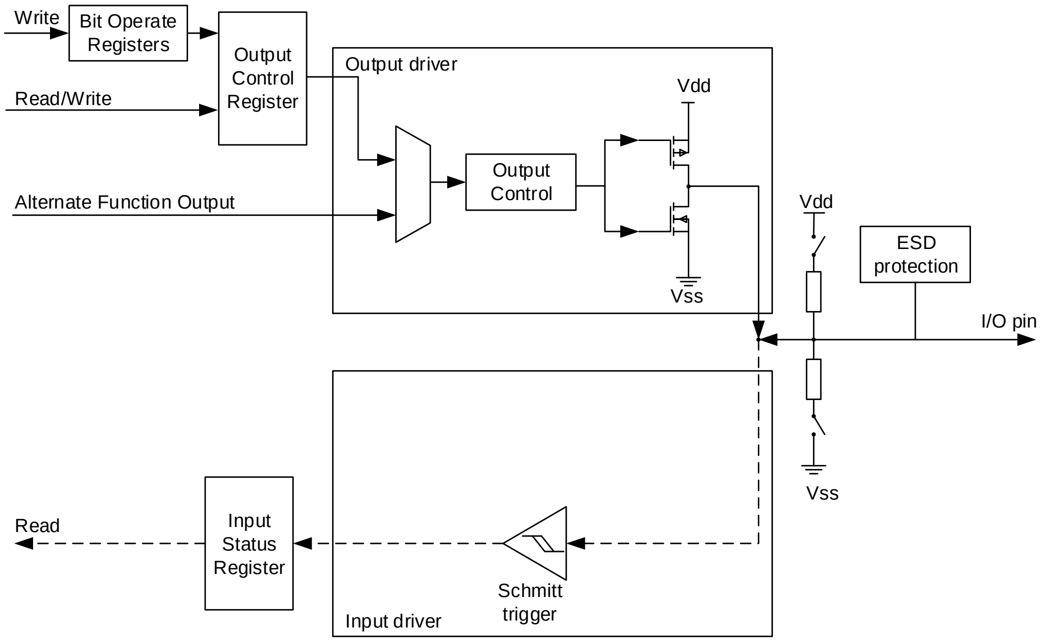
7.3.5. Output configuration

When GPIO pin is configured as output:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors are disabled.
- The output buffer is enabled.
- Open Drain Mode: The pad output low level when a “0” in the output control register. while the pad leaves Hi-Z when a “1” in the output control register.
- Push-Pull Mode: The pad output low level when a “0” in the output control register; while the pad output high level when a “1” in the output control register.
- A read access to the port output control register gets the last written value.
- A read access to the port input status register gets the I/O state.

Figure 7-3. Output configuration shows the output configuration.

Figure 7-3. Output configuration



7.3.6. Analog configuration

When GPIO pin is used as analog configuration:

- The weak pull-up and pull-down resistors are disabled.
- The output buffer is disabled.
- The schmitt trigger input is disabled.
- The port input status register of this I/O port bit is “0”.

Figure 7-4. Analog configuration shows the analog configuration.

Figure 7-4. Analog configuration

ESD protection
Analog (Input / Output)
I/O pin

7.3.7. Alternate function (AF) configuration

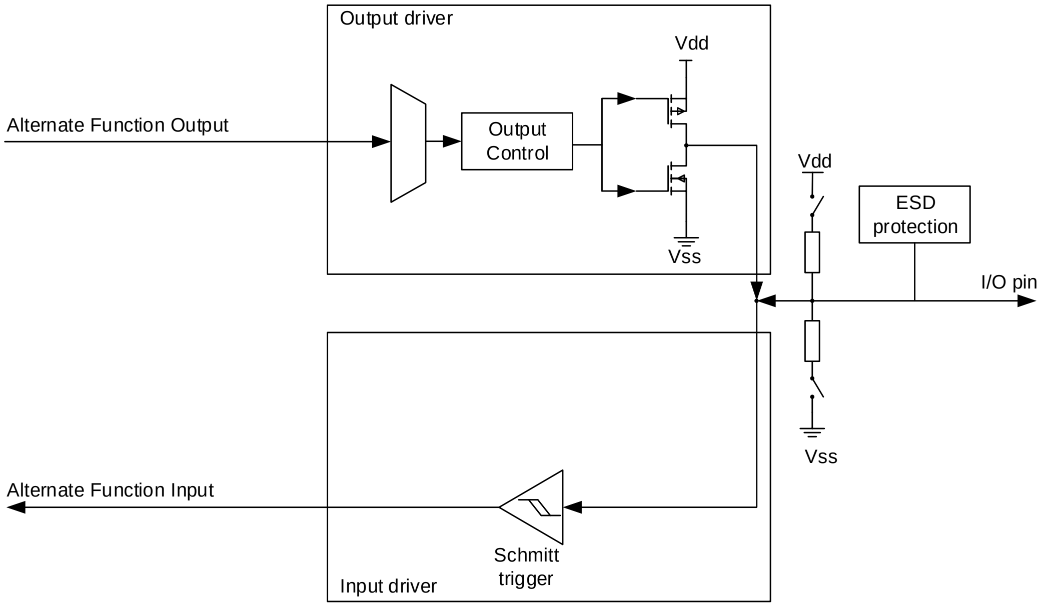
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

When be configured as alternate function:

- The output buffer is enabled in Open-Drain or Push-Pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen when input.
- The I/O pin data is stored into the port input status register every APB2 clock.
- A read access to the port input status register gets the I/O state.
- A read access to the port output control register gets the last written value.

Figure 7-5. Alternate function configuration shows the alternate function configuration.

Figure 7-5. Alternate function configuration



7.3.8. IO pin function selection

Each IO pin can implement many functions, each function selected by GPIO registers.

GPIO:

Each IO pin can be used for GPIO input function by configuring MDy bits to 0b00 in GPIOx_CTL0/GPIOx_CTL1 registers. And set output function by configuring MDy bits to 0b01, 0b10, or 0b11 and configuring CTLy bits of corresponding port in GPIOx_CTL0/GPIOx_CTL1 register to 0b00 (for GPIO push-pull output) or 0b01 (for GPIO open-drain output).

Alternate function:

Each IO pin can be used for AF input function by configuring MDy bits to 0b00 in GPIOx_CTL0/GPIOx_CTL1 registers. And set output function by configuring MDy bits to 0b01, 0b10, or 0b11 and configuring CTLy bits of corresponding port in GPIOx_CTL0/GPIOx_CTL1 register to 0b10 (for AF push-pull output) or 0b11 (for AF open-drain output).

7.3.9. GPIO locking function

The locking mechanism allows the IO configuration to be protected.

The protected registers are GPIOx_CTL0, GPIOx_CTL1. It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx_LOCK). When the special LOCK sequence has been occurred on LKK bit in GPIOx_LOCK register and the LKy bit is set in GPIOx_LOCK register, the corresponding port is locked and the corresponding port configuration cannot be modified until the next reset. It should be recommended to be used in the configuration of driving a power module.

7.4. Remapping function I/O and debug configuration

7.4.1. Introduction

In order to expand the flexibility of the GPIO or the usage of peripheral functions, each I/O pin can be configured to have up to four different functions by setting the AFIO Port Configuration Register (AFIO_PCF0/AFIO_PCF1). Suitable pinout locations can be selected using the peripheral IO remapping function. Additionally, various GPIO pins can be selected to be the EXTI interrupt line by setting the relevant EXTI Source Selection Register (AFIO_EXTISSx) to trigger an interrupt or event.

7.4.2. Main features

- APB slave interface for register access.
- EXTI source selection.
- Each pin has up to four alternative functions for configuration.

7.4.3. JTAG/SWD alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in table below.

Table 7-2. Debug interface signals

Alternate function	GPIO port
JTMS / SWDIO	PA13
JTCK / SWCLK	PA14
JTDI	PA15
JTDO / TRACESWO	PB3
NJTRST	PB4
TRACECK	PE2
TRACECK0	PE3
TRACECK1	PE4
TRACECK2	PE5
TRACECK3	PE6

To reduce the number of GPIOs used to debug, user can configure SWJ_CFG [2:0] bits in the AFIO_PCF0 to different value. Refer to table below.

Table 7-3. Debug port mapping

SWJ _CFG [2:0]	Available debug ports	SWJ I/O pin assigned				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/S WCLK	PA15/ JTDI	PB3/ JTDO/ TRACE SWO	PB4/ NJTRST
000	Full SWJ (JTAG-DP + SW-DP) (Reset state)	●	●	●	●	●
001	Full SWJ (JTAG-DP + SW-DP) but without NJTRST	●	●	●	●	X
010	JTAG-DP Disabled and SW-DP Enabled	●	●	X	X [1]	X
100	JTAG-DP Disabled and SW-DP Disabled	X	X	X	X	X
Other	Forbidden					

1. Released only if not using asynchronous trace.

7.4.4. ADC AF remapping

Refer to AFIO Port Configuration Register 0 (AFIO_ PCF0).

Table 7-4. ADC0 external trigger inserted conversion AF remapping [1]

Alternate function	ADC0_ETRGINS_REMAP = 0	ADC0_ETRGINS_REMAP = 1
ADC0 external trigger inserted conversion	ADC0 external trigger inserted conversion is connected to EXTI15	ADC0 external trigger inserted conversion is connected to TIMER7_CH3

1. Remap available only for High-density and Extra-density devices

Table 7-5. ADC0 external trigger regular conversion AF remapping [1]

Alternate function	ADC0_ETRGREG_REMAP = 0	ADC0_ETRGREG_REMAP = 1
ADC0 external trigger regular conversion	ADC0 external trigger regular conversion is connected to EXTI11	ADC0 external trigger regular conversion is connected to TIMER7_TRGO

1. Remap available only for High-density and Extra-density devices

Table 7-6. ADC1 external trigger inserted conversion AF remapping [1]

Alternate function	ADC1_ETRGINS_REMAP = 0	ADC1_ETRGINS_REMAP = 1
ADC1 external trigger inserted conversion	ADC1 external trigger inserted conversion is connected to EXTI15	ADC1 external trigger inserted conversion is connected to TIMER7_CH3

1. Remap available only for High-density and Extra-density devices

Table 7-7. ADC1 external trigger regular conversion AF remapping [1]

Alternate function	ADC1_ETRGREG_REMAP = 0	ADC1_ETRGREG_REMAP = 1
ADC1 external trigger regular conversion	ADC1 external trigger regular conversion is connected to EXTI11	ADC1 external trigger regular conversion is connected to TIMER7_TRGO

1. Remap available only for High-density and Extra-density devices

7.4.5. TIMER AF remapping

Table 7-8. TIMER0 alternate function remapping

Alternate function	TIMER0_REMAP [1:0] = "00" (no remap)	TIMER0_REMAP [1:0] = "01" (partial remap)	TIMER0_REMAP [1:0] = "11" (full remap) [1]
TIMER0_ETI	PA12		PE7
TIMER0_CH0	PA8		PE9
TIMER0_CH1	PA9		PE11
TIMER0_CH2	PA10		PE13
TIMER0_CH3	PA11		PE14
TIMER0_BKIN	PB12[2]	PA6	PE15
TIMER0_CH0_ON	PB13[2]	PA7	PE8
TIMER0_CH1_ON	PB14[2]	PB0	PE10
TIMER0_CH2_ON	PB15[2]	PB1	PE12

1. Remap available only for 100-pin and 144-pin packages

2. Remap not available on 36-pin package

Table 7-9. TIMER1 alternate function remapping

Alternate function	TIMER1_REMAP [1:0] = "00" (no remap)	TIMER1_REMAP [1:0] = "01" (partial remap)	TIMER1_REMAP [1: 0] = "10" (partial remap)	TIMER1_REMAP [1:0] = "11" (full remap) [2]
TIMER1_CH0/TIMER1_ETI [1]	PA0	PA15	PA0	PA15
TIMER1_CH1	PA1	PB3	PA1	PB3
TIMER1_CH2	PA2		PB10	
TIMER1_CH3	PA3		PB11	

1. TIMER1_CH0 and TIMER1_ETI share the same pin but cannot be used at the same time

2. Remap not available on 36-pin package

Table 7-10. TIMER2 alternate function remapping

Alternate function	TIMER2_REMAP [1:0] = "00" (no remap)	TIMER2_REMAP [1:0] = "10" (partial remap)	TIMER2_REMAP [1:0] = "11" (full remap) [1]
TIMER2_CH0	PA6	PB4	PC6
TIMER2_CH1	PA7	PB5	PC7

Alternate function	TIMER2_REMAP [1:0] = “00” (no remap)	TIMER2_REMAP [1:0] = “10” (partial remap)	TIMER2_REMAP [1:0] = “11” (full remap) [(1)]
TIMER2_CH2	PB0		PC8
TIMER2_CH3	PB1		PC9

1. Remap available only for 64-pin, 100-pin and 144-pin packages.

Table 7-11. TIMER3 alternate function remapping

Alternate function	TIMER3_REMAP = 0	TIMER3_REMAP = 1[(1)]
TIMER3_CH0	PB6	PD12
TIMER3_CH1	PB7	PD13
TIMER3_CH2	PB8	PD14
TIMER3_CH3	PB9	PD15

1. Remap available only for 100-pin and 144-pin packages.

Table 7-12. TIMER4 alternate function remapping [(1)]

Alternate function	TIMER4CH3_REMAP = 0	TIMER4CH3_REMAP = 1
TIMER4_CH3	TIMER4_CH3 is connected to PA3	IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose

1. Remap available only for High-density and Extra-density and Connectivity lines devices.

Table 7-13. TIMER8 alternate function remapping [(1)]

Alternate function	TIMER8_REMAP = 0	TIMER8_REMAP = 1
TIMER8_CH0	PA2	PE5
TIMER8_CH1	PA3	PE6

1. Refer to the AF remap and debug I/O configuration register 1(AFIO_ PCF1).

Table 7-14. TIMER9 alternate function remapping [(1)]

Alternate function	TIMER9_REMAP = 0	TIMER9_REMAP = 1
TIMER9_CH0	PB8	PF6

1. Refer to the AF remap and debug I/O configuration register 1 (AFIO_ PCF1).

Table 7-15. TIMER10 alternate function remapping [(1)]

Alternate function	TIMER10_REMAP = 0	TIMER10_REMAP = 1
TIMER10_CH0	PB9	PF7

1. Refer to the AF remap and debug I/O configuration register 1(AFIO_ PCF1).

Table 7-16. TIMER12 alternate function remapping [(1)]

Alternate function	TIMER12_REMAP = 0	TIMER12_REMAP = 1
TIMER12_CH0	PA6	PF8

1. Refer to the AF remap and debug I/O configuration register 1(AFIO_ PCF1).

Table 7-17. TIMER13 alternate function remapping [(1)]

Alternate function	TIMER13_REMAP = 0	TIMER13_REMAP = 1
TIMER13_CH0	PA7	PF9

1. Refer to the AF remap and debug I/O configuration register 1(AFIO_PCF1).

7.4.6. USART AF remapping

Refer to AFIO port configuration register 0 (AFIO_PCF0).

Table 7-18. USART0 alternate function remapping

Alternate function	USART0_REMAP = 0	USART0_REMAP = 1
USART0_TX	PA9	PB6
USART0_RX	PA10	PB7

Table 7-19. USART1 alternate function remapping

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1 [(1)]
USART1_CTS	PA0	PD3
USART1_RTS	PA1	PD4
USART1_TX	PA2	PD5
USART1_RX	PA3	PD6
USART1_CK	PA4	PD7

1. Remap available only 100-pin and 144-pin packages

Table 7-20. USART2 alternate function remapping

Alternate function	USART2_REMAP [1:0] = "00" (no remap)	USART2_REMAP [1:0] ="01" (partial remap) [(1)]	USART2_REMAP [1:0] ="11" (full remap) [(2)]
USART2_TX	PB10	PC10	PD8
USART2_RX	PB11	PC11	PD9
USART2_CK	PB12	PC12	PD10
USART2_CTS	PB13		PD11
USART2_RTS	PB14		PD12

1. Remap available only for 64-pin,100-pin and 144-pin packages

2. Remap available only 100-pin and 144-pin packages

7.4.7. I2C0 AF remapping

Refer to AFIO port configuration register 0 (AFIO_ PCF0).

Table 7-21. I2C0 alternate function remapping

Alternate function	I2C0_REMAP = 0	I2C0_REMAP = 1
I2C0_SCL	PB6	PB8
I2C0_SDA	PB7	PB9

7.4.8. SPI0 AF remapping

Refer to AFIO port configuration register 0 (AFIO_PCF0).

Table 7-22. SPI0 alternate function remapping

Alternate function	SPI0_REMAP = 0	SPI0_REMAP = 1
SPI0_NSS	PA4	PA15
SPI0_SCK	PA5	PB3
SPI0_MISO	PA6	PB4
SPI0_MOSI	PA7	PB5

7.4.9. SPI2/I2S2 AF remapping

Refer to AFIO port configuration register 0 (AFIO_ PCF0).

Table 7-23. SPI2/I2S2 alternate function remapping

Alternate function	SPI0_REMAP = 0	SPI0_REMAP = 1
SPI2_NSS/ I2S2_WS	PA15	PA4
SPI2_SCK/ I2S2_CK	PB3	PC10
SPI2_MISO	PB4	PC11
SPI2_MOSI/I2S2_SD	PB5	PC12

7.4.10. CAN0 AF remapping

The CAN0 signals can be mapped on Port A, Port B or Port D as shown in table below. For port D, remapping is not possible in devices delivered in 64-pin packages.

Table 7-24. CAN0 alternate function remapping

Alternate function (1)	CAN0_REMAP[1:0] =“00”	CAN0_REMAP[1:0] =“10”(3)	CAN0_REMAP[1:0] =“11”(2)
CAN0_RX	PA11	PB8	PD0
CAN0_TX	PA12	PB9	PD1

1. CAN0_RX and CAN0_TX in connectivity line devices; CAN_RX and CAN_TX in other devices with a single CAN interface.
2. This remapping is available only on 100-pin packages, when PD0 and PD1 are not remapped on OSC_IN and OSC_OUT.
3. Remap not available on 36-pin package.

7.4.11. CAN1 AF remapping

CAN1 is available in connectivity lines devices. The external signals can be remapped as show table below.

Table 7-25. CAN1 alternate function remapping

Alternate function	CAN1_REMAP = “0”	CAN1_REMAP = “1”
CAN1_RX	PB12	PB5
CAN1_TX	PB13	PB6

7.4.12. Ethernet AF remapping

Table 7-26. ENET alternate function remapping

Alternate function	ENET_REMAP = “0”	ENET_REMAP = “1”
RX_DV-CRS_DV	PA7	PD8
RXD0	PC4	PD9
RXD1	PC5	PD10
RXD2	PB0	PD11
RXD3	PB1	PD12

7.4.13. CLK pins AF remapping

The LXTAL oscillator pins OSC32_IN and OSC32_OUT can be used as general-purpose I/O PC14 and PC15 individually, when the LXTAL oscillator is off. The LXTAL has priority over the GPIOs function.

Note: 1. But when the 1.8 V domain is powered off (by entering standby mode) or when the backup domain is supplied by V_{BAT} (V_{DD} no more supplied), the PC14/PC15 GPIO functionality is lost and will be set in analog mode.

2. Refer to the note on IO usage restrictions in Section ***Battery backup domain***.

Table 7-27. OSC32 pins configuration

Alternate function	LXTAL= ON	LXTAL= OFF
PC14	OSC32_IN	PC14
PC15	OSC32_OUT	PC15

The HXTAL oscillator pins OSC_IN/OSC_OUT can be used as general-purpose I/O PD0/PD1.

Table 7-28. OSC pins configuration

Alternate function	HXTAL= ON	HXTAL = OFF
PD0	OSC_IN	PD0
PD1	OSC_OUT	PD1

7.5. Register definition

7.5.1. Port control register 0 (GPIOx_CTL0, x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

This register has to be accessed by word (32-bit).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CTL7[1:0]	MD7[1:0]	CTL6[1:0]	MD6[1:0]	CTL5[1:0]	MD5[1:0]	CTL4[1:0]	MD4[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CTL3[1:0]	MD3[1:0]	CTL2[1:0]	MD2[1:0]	CTL1[1:0]	MD1[1:0]	CTL0[1:0]	MD0[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	CTL7[1:0]	Port 7 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
29:28	MD7[1:0]	Port 7 mode bits These bits are set and cleared by software refer to MD0[1:0]description
27:26	CTL6[1:0]	Port 6 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
25:24	MD6[1:0]	Port 6 mode bits These bits are set and cleared by software refer to MD0[1:0]description
23:22	CTL5[1:0]	Port 5 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
21:20	MD5[1:0]	Port 5 mode bits These bits are set and cleared by software refer to MD0[1:0]description
19:18	CTL4[1:0]	Port 4 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
17:16	MD4[1:0]	Port 4 mode bits These bits are set and cleared by software

		refer to MD0[1:0]description
15:14	CTL3[1:0]	Port 3 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD3[1:0]	Port 3 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL2[1:0]	Port 2 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD2[1:0]	Port 2 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL1[1:0]	Port 1 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD1[1:0]	Port 1 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL0[1:0]	Pin 0 configuration bits These bits are set and cleared by software Input mode (MD[1:0] =00) 00: Analog mode 01: Floating input 10: Input with pull-up / pull-down 11: Reserved Output mode (MD[1:0] >00) 00: GPIO output with push-pull 01: GPIO output with open-drain 10: AFIO output with push-pull 11: AFIO output with open-drain
1:0	MD0[1:0]	Port 0 mode bits These bits are set and cleared by software 00: Input mode (reset state) 01: Output mode ,max speed 10MHz 10: Output mode ,max speed 2 MHz 11: Output mode ,max speed 50MHz

7.5.2. Port control register 1 (GPIOx_CTL1, x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

This register has to be accessed by word (32-bit).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CTL15[1:0]	MD15[1:0]	CTL14[1:0]	MD14[1:0]	CTL13[1:0]	MD13[1:0]	CTL12[1:0]	MD12[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
CTL11[1:0]	MD11[1:0]	CTL10[1:0]	MD10[1:0]	CTL9[1:0]	MD9[1:0]	CTL8[1:0]	MD8[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	CTL15[1:0]	Port 15 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
29:28	MD15[1:0]	Port 15 mode bits These bits are set and cleared by software refer to MD0[1:0]description
27:26	CTL14[1:0]	Port 14 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
25:24	MD14[1:0]	Port 14 mode bits These bits are set and cleared by software refer to MD0[1:0]description
23:22	CTL13[1:0]	Port 13 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
21:20	MD13[1:0]	Port 13 mode bits These bits are set and cleared by software refer to MD0[1:0]description
19:18	CTL12[1:0]	Port 12 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
17:16	MD12[1:0]	Port 12 mode bits These bits are set and cleared by software refer to MD0[1:0]description
15:14	CTL11[1:0]	Port 11 configuration bits

		These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD11[1:0]	Port 11 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL10[1:0]	Port 10 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD10[1:0]	Port 10 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL9[1:0]	Port 9 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD9[1:0]	Port 9 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL8[1:0]	Port 8 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
1:0	MD8[1:0]	Port 8 mode bits These bits are set and cleared by software refer to MD0[1:0]description

7.5.3. Port input status register (GPIOx_ISTAT, x=A..G)

Address offset: 0x08
Reset value: 0x0000 XXXX

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISTAT15	ISTAT14	ISTAT13	ISTAT12	ISTAT11	ISTAT10	ISTAT 9	ISTAT 8	ISTAT 7	ISTAT 6	ISTAT 5	ISTAT 4	ISTAT 3	ISTAT 2	ISTAT 1	ISTAT 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions

31:16	Reserved	Must be kept at reset value
15:0	ISTATy	Port input status(y=0..15) These bits are set and cleared by hardware 0: Input signal low 1: Input signal high

7.5.4. Port output control register (GPIOx_OCTL, x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCTL15	OCTL14	OCTL13	OCTL12	OCTL11	OCTL10	OCTL9	OCTL8	OCTL7	OCTL6	OCTL5	OCTL4	OCTL3	OCTL2	OCTL1	OCTL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	OCTLy	Port output control(y=0..15) These bits are set and cleared by software 0: Pin output low 1: Pin output high

7.5.5. Port bit operate register (GPIOx_BOP, x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOP15	BOP14	BOP13	BOP12	BOP11	BOP10	BOP9	BOP8	BOP7	BOP6	BOP5	BOP4	BOP3	BOP2	BOP1	BOP0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:16	CRy	Port Clear bit y(y=0..15)

		These bits are set and cleared by software 0: No action on the corresponding OCTLy bit 1: Clear the corresponding OCTLy bit to 0
15:0	BOPy	Port Set bit y(y=0..15) These bits are set and cleared by software 0: No action on the corresponding OCTLy bit 1: Set the corresponding OCTLy bit to 1

7.5.6. Port bit clear register (GPIOx_BC, x=A..G)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CRy	Port Clear bit y(y=0..15) These bits are set and cleared by software 0: No action on the corresponding OCTLy bit 1: Clear the corresponding OCTLy bit to 0

7.5.7. Port configuration lock register (GPIOx_LOCK, x=A..G)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LK15	LK14	LK13	LK12	LK11	LK10	LK9	LK8	LK7	LK6	LK5	LK4	LK3	LK2	LK1	LK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	LKK	Lock sequence key It can only be setted using the Lock Key Writing Sequence. And can always be read. 0: GPIO_LOCK register is not locked and the port configuration is not locked. 1: GPIO_LOCK register is locked until an MCU reset.. LOCK key configuration sequence Write 1→Write 0→Write 1→ Read 0→ Read 1 **Note:** The value of LK[15:0] must hold during the LOCK Key Writing sequence.
15:0	LKy	Port Lock bit y(y=0..15) These bits are set and cleared by software 0: The corresponding bit port configuration is not locked 1: The corresponding bit port configuration is locked when LKK bit is “1”

7.5.8. Event control register (AFIO_EC)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOE	PORT[2:0]			PIN[3:0]			
								rw	rw			rw			

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	EOE	Event output enable Set and cleared by software.When set the EVENTOUT Cortex output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits
6:4	PORT[2:0]	Event output port selection Set and cleared by software.Select the port used to output the Cortex EVENTOUT signal. 000: Select PORT A 001: Select PORT B 010: Select PORT C 011: Select PORT D 100: Select PORT E

3:0	PIN[3:0]	Event output pin selection Set and cleared by software. Select the pin used to output the Cortex EVENTOUT signal. 0000: Select Pin 0 0001: Select Pin 1 0010: Select Pin 2 … 1111: Select Pin 15

7.5.9. AFIO port configuration register 0 (AFIO_PCF0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

Memory map and bit definitions for Middle-density, High-density and Extra-density devices:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			SPI2_REMAP		SWJ_ CFG[2:0]			Reserved			ADC1_ETRGRER_REMAP	ADC1_ETRGINJ_REMAP	ADC0_ETRGRER_REMAP	ADC0_ETRGINJ_REMAP	TIMER4CH3_IREMAP
			rw			w					rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP [1:0]		TIMER3_REMAP	TIMER2_REMAP [1:0]		TIMER1_REMAP [1:0]		TIMER0_REMAP [1:0]		USART2_REMAP[1:0]		USART1_REMAP	USART0_REMAP	I2C0_REMAP	SPI0_REMAP
rw	rw		rw	rw		rw		rw		rw		rw	rw	rw	rw

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28	SPI2_REMAP	SPI2/I2S2 remapping This bit is set and cleared by software. 0: No remap (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5) 1: Full remap (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12) Note: This bit is available only in Extra-density devices and High-density devices.
27	Reserved	Must be kept at reset value
26:24	SWJ_CFG[2:0]	Serial wire JTAG configuration These bits are write-only (when read,the value is undefined).They are used to configure the SWJ and trace alternate function I/Os. The SWJ(Serial Wire JTAG)

		supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace.This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS/JTCK pin. 000: Full SWJ (JTAG-DP + SW-DP)Reset State 001: Full SWJ (JTAG-DP + SW-DP) but without NJTRST 010: JTAG-DP Disabled and SW-DP Enabled 100: JTAG-DP Disabled and SW-DP Disabled Other: Undefined
23:21	Reserved	Must be kept at reset value
20	ADC1_ETRGREG_REMAP	ADC 1 external trigger regular conversion remapping Set and cleared by software. The bit control the trigger input connected to ADC1 external trigger regular conversion. When this bit is reset, the ADC1 external trigger reqular conversion to EXTI11.When this bit is set, the ADC1 external event regular conversion is connected to TIM7_TRGO.
19	ADC1_ETRGINS_REMAP	ADC 1 external trigger inserted conversion remapping Set and cleared by software. The bit control the trigger input connected to ADC1 external trigger inserted conversion. When this bit is reset, the ADC1 external trigger inserted conversion to EXTI15.When this bit is set, the ADC1 external event inserted conversion is connected to TIM7_CH3.
18	ADC0_ETRGREG_REMAP	ADC 0 external trigger regular conversion remapping Set and cleared by software. The bit control the trigger input connected to ADC0 external trigger inserted conversion. When this bit is reset, the ADC0 external trigger inserted conversion to EXTI11.When this bit is set, the ADC0 external event inserted conversion is connected to TIM7_TRGO.
17	ADC0_ETRGINS_REMAP	ADC 0 external trigger inserted conversion remapping Set and cleared by software. The bit control the trigger input connected to ADC0 external trigger inserted conversion. When this bit is reset, the ADC0 external trigger inserted conversion to EXTI15.When this bit is set, the ADC0 external event inserted conversion is connected to TIM7_CH3.
16	TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping Set and cleared by software.This bit controls the TIMER4_CH3 internal mapping.When reset the timer TIMER4_CH3 is connected to PA3.When set the IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose. **Note:** This bit is available only in High-density value line devices.
15	PD01_REMAP	Port D0/Port D1 mapping on OSC_IN/OSC_OUT This bit is set and cleared by software 0: Not remap 1: PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT
14:13	CAN_REMAP [1:0]	CAN interface remapping

		These bits are set and cleared by software. 00: No remap (CAN_RX/PA11,CAN_TX/PA12) 01: Not used 10: Partial remap (CAN_RX/PB8,CAN_TX/PB9) 11: Full remap (CAN_RX/PD0,CAN_TX/PD1)
12	TIMER3_REMAP	TIMER3 remapping This bit is set and cleared by software 0: No remap (TIMER3_CH0/PB6,TIMER3_CH1/PB7,TIMER3_CH2/PB8, TIMER3_CH3/PB9) 1: Full remap (TIMER3_CH0/PD12,TIMER3_CH1/PD13,TIMER3_CH2/PD14, TIMER3_CH3/PD15)
11:10	TIMER2_ REMAP[1:0]	TIMER2 remapping These bits are set and cleared by software 00: No remap (TIMER2_CH0/PA6,TIMER2_CH1/PA7,TIMER2_CH2/PB0, TIMER2_CH3/PB1) 01: Not used 10: Partial remap (TIMER2_CH0/PB4,TIMER2_CH1/PB5,TIMER2_CH2/PB0, TIMER2_CH3/PB1) 11: Full remap (TIMER2_CH0/PC6,TIMER2_CH1/PC7,TIMER2_CH2/PC8, TIMER2_CH3/PC9)
9:8	TIMER1_REMAP [1:0]	TIMER1 remapping These bits are set and cleared by software 00: No remap (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 01: Partial remap (TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 10: Partial remap (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PB10,TIMER1_CH3/PB11) 11: Full remap(TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PB10,TIMER1_CH3/PB11)
7:6	TIMER0_REMAP [1:0]	TIMER0 remapping These bits are set and cleared by software 00: No remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9, TIMER0_CH2/PA10,TIMER0_CH3/PA11, TIMER0_BKIN/PB12, TIMER0_CH0_ON/PB13, TIMER0_CH1_ON/PB14, TIMER0_CH2_ON/PB15) 01: Partial remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9, TIMER0_CH2/PA10,TIMER0_CH3/PA11, TIMER0_BKIN/PA6, TIMER0_CH0_ON/PA7, TIMER0_CH1_ON/PB0, TIMER0_CH2_ON/PB1) 10: Not used 11: Full remap (TIMER0_ETI/PE7, TIMER0_CH0/ PE9, TIMER0_CH1/PE11, TIMER0_CH2/PE13,TIMER0_CH3/PE14, TIMER0_BKIN/PE15,

		TIMER0_CH0_ON/PE8, TIMER0_CH1_ON/PE10, TIMER0_CH2_ON/PE12)
5:4	USART2_REMAP [1:0]	USART2 remapping These bits are set and cleared by software 00: No remap (USART2_TX/PB10, USART2_RX /PB11, USART2_CK/PB12, USART2_CTS/PB13, USART2_RTS/PB14) 01: Partial remap (USART2_TX/PC10, USART2_RX /PC11, USART2_CK/PC12, USART2_CTS/PB13, USART2_RTS/PB14) 10: Not used 11: Full remap (USART2_TX/PD8, USART2_RX /PD9, USART2_CK/PD10, USART2_CTS/PD11, USART2_RTS/PD12)
3	USART1_REMAP	USART1 remapping This bit is set and cleared by software 0: No remap (USART1_CTS/PA0, USART1_RTS/PA1,USART1_TX/PA2, USART1_RX /PA3, USART1_CK/PA4) 1: Remap (USART1_CTS/PD3, USART1_RTS/PD4,USART1_TX/PD5, USART1_RX /PD6, USART1_CK/PD7)
2	USART0_REMAP	USART0 remapping This bit is set and cleared by software 0: No remap (USART0_TX/PA9, USART0_RX /PA10) 1: Remap (USART0_TX/PB6, USART0_RX /PB7)
1	I2C0_REMAP	I2C0 remapping This bit is set and cleared by software 0: No remap (I2C0_SCL/PB6, I2C0_SDA /PB7) 1: Remap (I2C0_SCL/PB8, I2C0_SDA /PB9)
0	SPI0_REMAP	SPI0 remapping This bit is set and cleared by software 0: No remap (SPI0_NSS/PA4, SPI0_SCK /PA5, SPI0_MISO /PA6, SPI0_MOSI /PA7) 1: Remap (SPI0_NSS/PA15, SPI0_SCK /PB3, SPI0_MISO /PB4, SPI0_MOSI /PB5)

Memory map and bit definitions for connectivity devices:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PTP_PPS_REMAP	TIMER1ITI1_REMAP	SPI2_REMAP	Reserved	SWJ_ CFG[2:0]			ENET_PHY_SEL	CAN1_REMAP	ENET_REMAP	Reserved				TIMER4CH3_IREMAP
	rw	rw	rw		w			rw	rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PD01_ REMAP	CAN0_REMAP [1:0]	TIMER3_ REMAP	TIMER2_ REMAP[1:0]	TIMER1_ REMAP[1:0]	TIMER0_ REMA [1:0]	USART2_ REMAP[1:0]	USART1_ REMAP	USART0_ REMAP	I2C0_ REMAP	SPI0_ REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	PTP_PPS_REMAP	Ethernet PTP PPS remapping This bit is set and cleared by software. It enables the Ethernet MAC_PPS to be output on the PB5 pin 0: PPT_PPS not output PB5 pin 1: PPT_PPS is output on PB5 pin Note: This bit is available only in connectivity line devices and is reserved otherwise.
29	TIMER1ITI1_REMAP	TIMER1 internal trigger 1 remapping These bits are set and cleared by software. It control the TIMER1_ITI1 internal mapping 0: Connect TIMER1_ITI1 internally to the Ethernet PTP output for calibration purposes 1: Connect USB OTG SOF (Start of Frame) output TIMER1_ITI1 for calibration purposes Note: This bit is available only in connectivity line devices and is reserved otherwise.
28	SPI2_REMAP	SPI2/I2S2 remapping This bit is set and cleared by software. 0: No remap (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5) 1: Full remap (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12) Note: This bit is available only in connectivity line devices and is reserved otherwise.
27	Reserved	Must be kept at reset value
26:24	SWJ_CFG[2:0]	Serial wire JTAG configuration These bits are write-only (when read,the value is undefined).They are used to configure the SWJ and trace alternate function I/Os. The SWJ(Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace.This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS/JTCK pin 000: Full SWJ(JTAG-DP +SW-DP): Reset State 001: Full SWJ(JTAG-DP +SW-DP): but without NJTRST 010: JTAG-DP Disabled and SW-DP Enabled

		100: JTAG-DP Disabled and SW-DP Disabled Other combinations: no effect
23	ENET_PHY_SEL	Ethernet MII or RMII PHY selection This bit is set and cleared by software.It configures the Ethernet MAC internally for use with an external MII or RMII PHY. 0:Configure Ethernet MAC for connection with an MII PHY 1:Configure Ethernet MAC for connection with an RMII PHY Note: This bit is available only in connectivity line devices and is reserved otherwise.
22	CAN1_REMAP	CAN1 I/O remapping This bit is set and cleared by software.It controls the CAN1_TX and CAN1_RX pins 0: No remap (CAN1_RX/PB12,CAN_TX/PB13) 1: Remap (CAN1_RX/PB5,CAN_TX/PB6) Note: This bit is available only in connectivity line devices and is reserved otherwise.
21	ENET_REMAP	Ethernet MAC I/O remapping This bit is set and cleared by software.It controls the Ethernet MAC connections with PHY 0: No remap (RX_DV-CRS_DV/PA7,RXD0/PC4,RXD1/PC5,RXD2/PB0, RXD3/PB1) 1: Remap (RX_DV-CRS_DV/PD8,RXD0/PD9,RXD1/PD10,RXD2/PD11, RXD3/PD12) Note: This bit is available only in connectivity line devices and is reserved otherwise.
20:17	Reserved	Must be kept at reset value
16	TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping Set and cleared by software.This bit controls the TIMER4_CH3 internal mapping. When reset timer TIMER4_CH3 is connected to PA3.When set the IRC40K internal clock connected to TIMER4_CH3 input for calibration purpose. 0: No remap 1: Remap
15	PD01_REMAP	Port D0/Port D1 mapping on OSC_IN/OSC_OUT This bit is set and cleared by software 0: Not remap 1: PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT
14:13	CAN0_REMAP[1:0]	CAN0 alternate interface remapping These bits are set and cleared by software 00: No remap (CAN0_RX/PA11,CAN0_TX/PA12)

		01: Not used 10: Partial remap (CAN0_RX/PB8,CAN0_TX/PB9) 11: Full remap (CAN0_RX/PD0,CAN0_TX/PD1)
12	TIMER3_REMAP	TIMER3 remapping This bit is set and cleared by software. 0: No remap (TIMER3_CH0/PB6,TIMER3_CH1/PB7,TIMER3_CH2/PB8, TIMER3_CH3/PB9) 1: Full remap (TIMER3_CH0/PD12,TIMER3_CH1/PD13, TIMER3_CH2/PD14,TIMER3_CH3/PD15)
11:10	TIMER2_REMAP [1:0]	TIMER2 remapping These bits are set and cleared by software 00: No remap (TIMER2_CH0/PA6,TIMER2_CH1/PA7,TIMER2_CH2/PB0, TIMER2_CH3/PB1) 01: Not used 10: Partial remap (TIMER2_CH0/PB4,TIMER2_CH1/PB5, TIMER2_CH2/PB0,TIMER2_CH3/PB1) 11: Full remap (TIMER2_CH0/PC6,TIMER2_CH1/PC7,TIMER2_CH2/PC8, TIMER2_CH3/PC9)
9:8	TIMER1_REMAP [1:0]	TIMER1 remapping These bits are set and cleared by software 00: No remap (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 01: Partial remap 0 (TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 10: Partial remap 1 (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PB10,TIMER1_CH3/PB11) 11: Full remap (TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PB10,TIMER1_CH3/PB11)
7:6	TIMER0_REMAP [1:0]	TIMER0 remapping These bits are set and cleared by software 00: No remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9, TIMER0_CH2/PA10,TIMER0_CH3/PA11, TIMER0_BKIN/PB12, TIMER0_CH0_ON/PB13, TIMER0_CH1_ON/PB14, TIMER0_CH2_ON/PB15) 01: Partial remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9,TIMER0_CH2/PA10,TIMER0_CH3/PA11, TIMER0_BKIN/PA6, TIMER0_CH0_ON/PA7, TIMER0_CH1_ON/PB0, TIMER0_CH2_ON/PB1) 10: Not used 11: Full remap (TIMER0_ETI/PE7, TIMER0_CH0/ PE9, TIMER0_CH1/PE11,TIMER0_CH2/PE13,TIMER0_CH3/PE14, TIMER0_BKIN/PE15, TIMER0_CH0_ON/PE8, TIMER0_CH1_ON/PE10,

		TIMER0_CH2_ON/PE12)
5:4	USART2_REMAP [1:0]	USART2 remapping These bits are set and cleared by software 00: No remap (USART2_TX/PB10, USART2_RX /PB11, USART2_CK/PB12,USART2_CTS/PB13, USART2_RTS/PB14) 01: Partial remap (USART2_TX/PC10, USART2_RX /PC11, USART2_CK/PC12,USART2_CTS/PB13, USART2_RTS/PB14) 10: Not used 11: Full remap (USART2_TX/PD8, USART2_RX /PD9, USART2_CK/PD10,USART2_CTS/PD11, USART2_RTS/PD12)
3	USART1_REMAP	USART1 remapping This bit is set and cleared by software 0: No remap (USART1_CTS/PA0, USART1_RTS/PA1,USART1_TX/PA2, USART1_RX /PA3, USART1_CK/PA4) 1: Remap (USART1_CTS/PD3, USART1_RTS/PD4,USART1_TX/PD5, USART1_RX /PD6, USART1_CK/PD7)
2	USART0_REMAP	USART0 remapping This bit is set and cleared by software 0: No remap (USART0_TX/PA9, USART0_RX /PA10) 1: Remap (USART0_TX/PB6, USART0_RX /PB7)
1	I2C0_REMAP	I2C0 remapping This bit is set and cleared by software 0: No remap (I2C0_SCL/PB6, I2C0_SDA /PB7) 1: Remap (I2C0_SCL/PB8, I2C0_SDA /PB9)
0	SPI0_REMAP	SPI0 remapping This bit is set and cleared by software 0: No remap (SPI0_NSS/PA4, SPI0_SCK /PA5, SPI0_MISO /PA6, SPI0_MOSI /PA7, SPI0_IO2 /PA2, SPI0_IO3 /PA3) 1: Remap (SPI0_NSS/PA15, SPI0_SCK /PB3, SPI0_MISO /PB4, SPI0_MOSI /PB5, SPI0_IO2 /PB6, SPI0_IO3 /PB7)

7.5.10. EXTI sources selection register 0 (AFIO_EXTISS0)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3_SS [3:0]				EXTI2_SS [3:0]				EXTI1_SS [3:0]				EXTI0_SS [3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI3_SS [3:0]	EXTI 3 sources selection 0000: PA3 pin 0001: PB3 pin 0010: PC3 pin 0011: PD3 pin 0100: PE3 pin 0101: PF3 pin 0110: PG3 pin Other configurations are reserved.
11:8	EXTI2_SS [3:0]	EXTI 2 sources selection 0000: PA2 pin 0001: PB2 pin 0010: PC2 pin 0011: PD2 pin 0100: PE2 pin 0101: PF2 pin 0110: PG2 pin Other configurations are reserved.
7:4	EXTI1_SS [3:0]	EXTI 1 sources selection 0000: PA1 pin 0001: PB1 pin 0010: PC1 pin 0011: PD1 pin 0100: PE1 pin 0101: PF1 pin 0110: PG1 pin Other configurations are reserved.
3:0	EXTI0_SS [3:0]	EXTI 0 sources selection 0000: PA0 pin 0001: PB0 pin 0010: PC0 pin 0011: PD0 pin 0100: PE0 pin 0101: PF0 pin

0110: PG0 pin

Other configurations are reserved.

7.5.11. EXTI sources selection register 1 (AFIO_EXTISS1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7_SS [3:0]				EXTI6_SS [3:0]				EXTI5_SS [3:0]				EXTI4_SS [3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI7_SS [3:0]	EXTI 7 sources selection 0000: PA7 pin 0001: PB7 pin 0010: PC7 pin 0011: PD7 pin 0100: PE7 pin 0101: PF7 pin 0110: PG7 pin Other configurations are reserved.
11:8	EXTI6_SS [3:0]	EXTI 6 sources selection 0000: PA6 pin 0001: PB6 pin 0010: PC6 pin 0011: PD6 pin 0100: PE6 pin 0101: PF6 pin 0110: PG6 pin Other configurations are reserved.
7:4	EXTI5_SS [3:0]	EXTI 5 sources selection 0000: PA5 pin 0001: PB5 pin 0010: PC5 pin 0011: PD5 pin

		0100: PE5 pin 0101: PF5 pin 0110: PG5 pin Other configurations are reserved.
3:0	EXTI4_SS [3:0]	EXTI 4 sources selection 0000: PA4 pin 0001: PB4 pin 0010: PC4 pin 0011: PD4 pin 0100: PE4 pin 0101: PF4 pin 0110: PG4 pin Other configurations are reserved.

7.5.12. EXTI sources selection register 2 (AFIO_EXTISS2)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11_SS [3:0]				EXTI10_SS [3:0]				EXTI9_SS [3:0]				EXTI8_SS [3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI11_SS [3:0]	EXTI 11 sources selection 0000: PA11 pin 0001: PB11 pin 0010: PC11 pin 0011: PD11 pin 0100: PE11 pin 0101: PF11 pin 0110: PG11 pin Other configurations are reserved.
11:8	EXTI10_SS [3:0]	EXTI 10 sources selection 0000: PA10 pin 0001: PB10 pin

		0010: PC10 pin 0011: PD10 pin 0100: PE10 pin 0101: PF10 pin 0110: PG10 pin Other configurations are reserved.
7:4	EXTI9_SS [3:0]	EXTI 9 sources selection 0000: PA9 pin 0001: PB9 pin 0010: PC9 pin 0011: PD9 pin 0100: PE9 pin 0101: PF9 pin 0110: PG9 pin Other configurations are reserved.
3:0	EXTI8_SS [3:0]	EXTI 8 sources selection 0000: PA8 pin 0001: PB8 pin 0010: PC8 pin 0011: PD8 pin 0100: PE8 pin 0101: PF8 pin 0110: PG8 pin Other configurations are reserved.

7.5.13. EXTI sources selection register 3 (AFIO_EXTISS3)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15_SS [3:0]				EXTI14_SS [3:0]				EXTI13_SS [3:0]				EXTI12_SS [3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI15_SS [3:0]	EXTI 15 sources selection

		0000: PA15 pin 0001: PB15 pin 0010: PC15 pin 0011: PD15 pin 0100: PE15 pin 0101: PF15 pin 0110: PG15 pin Other configurations are reserved.
11:8	EXTI14_SS [3:0]	EXTI 14 sources selection 0000: PA14 pin 0001: PB14 pin 0010: PC14 pin 0011: PD14 pin 0100: PE14 pin 0101: PF14 pin 0110: PG14 pin Other configurations are reserved.
7:4	EXTI13_SS [3:0]	EXTI 13 sources selection 0000: PA13 pin 0001: PB13 pin 0010: PC13 pin 0011: PD13 pin 0100: PE13 pin 0101: PF13 pin 0110: PG13 pin Other configurations are reserved.
3:0	EXTI12_SS [3:0]	EXTI 12 sources selection 0000: PA12 pin 0001: PB12 pin 0010: PC12 pin 0011: PD12 pin 0100: PE12 pin 0101: PF12 pin 0110: PG12 pin Other configurations are reserved.

7.5.14. AFIO port configuration register 1 (AFIO_PCF1)

Address offset: 0x1C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					EXMC_NADV	TIMER13_REMAP	TIMER12_REMAP	TIMER10_REMAP	TIMER9_REMAP	TIMER8_REMAP	Reserved				
					rw	rw	rw	rw	rw	rw					

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	EXMC_NADV	EXMC_NADV connect/disconnect This bit is set and cleared by software, it controls the use of optional EXMC_NADV signal. 0: The NADV signal is connected to the output(default) 1: The NADV signal is not connected. The I/O pin can be used by another peripheral.
9	TIMER13_REMAP	TIMER13 remapping This bit is set and cleared by software, it controls the mapping of the TIMER13_CH0 alternate function onto the GPIO ports 0: No remap (PA7) 1: Remap (PF9)
8	TIMER12_REMAP	TIMER12 remapping This bit is set and cleared by software, it controls the mapping of the TIMER12_CH0 alternate function onto the GPIO ports 0: No remap (PA6) 1: Remap (PF8)
7	TIMER10_REMAP	TIMER10 remapping This bit is set and cleared by software, it controls the mapping of the TIMER10_CH0 alternate function onto the GPIO ports 0: No remap (PB9) 1: Remap (PF7)
6	TIMER9_REMAP	TIMER9 remapping This bit is set and cleared by software, it controls the mapping of the TIMER9_CH0 alternate function onto the GPIO ports 0: No remap (PB8) 1: Remap (PF6)
5	TIMER8_REMAP	TIMER8 remapping This bit is set and cleared by software, it controls the mapping of the TIMER8_CH0 and TIMER8_CH1 alternate function onto the GPIO ports 0: No remap (TIMER8_CH0 on PA2 and TIMER8_CH1 on PA3)

		1: Remap (PF6) (TIMER8_CH0 on PE5 and TIMER8_CH1 on PE6)
4:0	Reserved	Must be kept at reset value

8. CRC calculation unit (CRC)

8.1. Overview

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 32 bit CRC code with fixed polynomial.

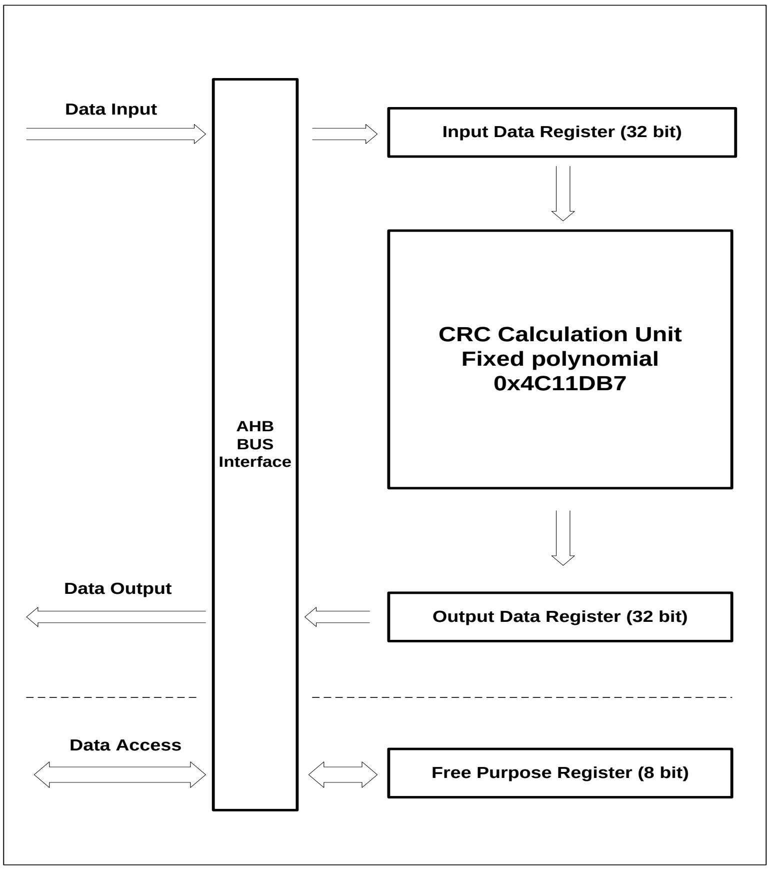
8.2. Characteristics

- 32-bit data input and 32-bit data output. Calculation period is 4 AHB clock cycles for 32-bit input data size from data entered to the calculation result available.
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices.
- Fixed polynomial: 0x4C11DB7

  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$

This 32-bit CRC polynomial is a common polynomial used in Ethernet.

Figure 8-1. Block diagram of CRC calculation unit



8.3. Function overview

- CRC calculation unit is used to calculate the 32-bit raw data, and CRC_DATA register will receive the raw data and store the calculation result.

  If the CRC_DATA register has not been cleared by software setting the CRC_CTL register, the new input raw data will be calculated based on the result of previous value of CRC_DATA.

  CRC calculation will spend 4 AHB clock cycles for 32-bit data size, during this period AHB will not be hanged because of the existence of the 32-bit input buffer.

- This module supplies an 8-bit free register CRC_FDATA.

  CRC_FDATA is unrelated to the CRC calculation, any value you write in will be read out at anytime.

8.4. Register definition

8.4.1. Data register (CRC_DATA)

Address offset: 0x00
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA [31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	DATA [31:0]	CRC calculation result bits Software writes and reads. This register is used to calculate new data, and the register can be written the new data directly. Written value cannot be read because the read value is the previous CRC calculation result.

8.4.2. Free data register (CRC_FDATA)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:8	Reserved	Keep at reset value
7:0	FDATA [7:0]	Free Data Register bits Software writes and reads. These bits are unrelated with CRC calculation. This byte can be used for any goal

by any other peripheral. The CRC_CTL register will take no effect to the byte.

8.4.3. Control register (CRC_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>RST</td></tr>
<tr><td colspan="15"></td><td>rs</td></tr>
</table>

Bits	Fields	Descriptions
31:1	Reserved	Keep at reset value
0	RST	Set this bit can reset the CRC_DATA register to the value of 0xFFFFFFFF then automatically cleared itself to 0 by hardware. This bit will take no effect to CRC_FDATA. Software writes and reads.

9. Direct memory access controller (DMA)

9.1. 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 12 channels in the DMA controller (7 for DMA0 and 5 for DMA1). Each channel is dedicated to manage memory access requests from one or more peripherals. An arbiter is implemented inside to handle the priority among DMA requests.

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.

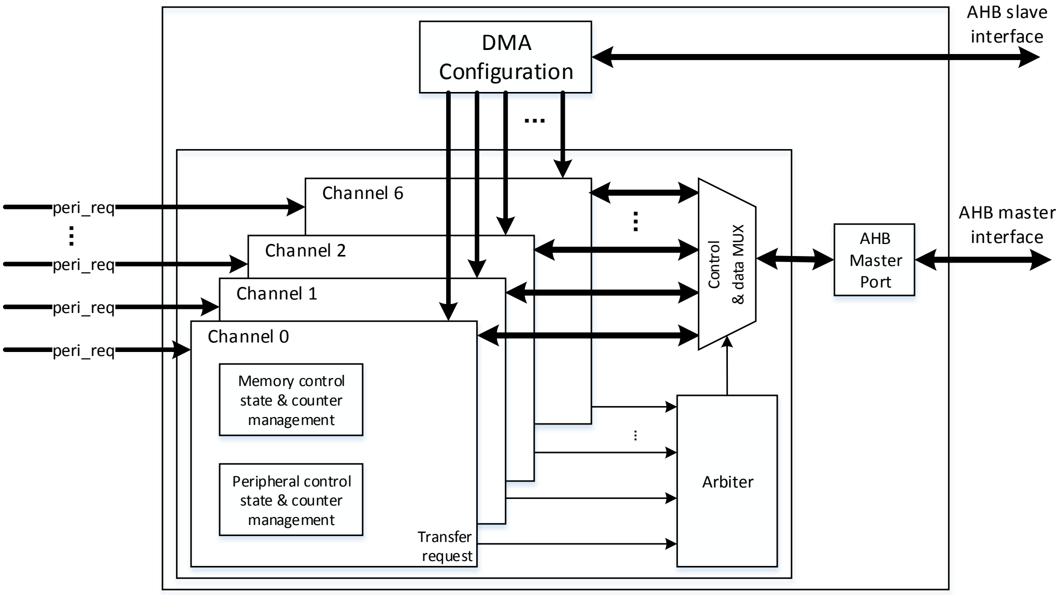
9.2. Characteristics

- Programmable length of data to be transferred, max to 65536.
- 12 channels and each channel are configurable (7 for DMA0 and 5 for DMA1).
- AHB and APB peripherals, FLASH, SRAM can be accessed as source and destination.
- Each channel is connected to fixed hardware DMA request.
- Software DMA channel priority (low, medium, high, ultra high) and hardware DMA channel priority (DMA channel 0 has the highest priority and DMA channel 6 has the lowest priority).
- Support independent 8, 16, 32-bit memory and peripheral transfer.
- Support independent fixed and increasing address generation algorithm of memory and peripheral.
- Support circular transfer mode.
- Support peripheral to memory, memory to peripheral, and memory to memory transfers.
- One separate interrupt per channel with three types of event flags.
- Support interrupt enable and clear.

Note: Only HD, XD and CL devices have DMA1 controller.

9.3. Block diagram

Figure 9-1. Block diagram of DMA



As shown in ***Figure 9-1. Block diagram of DMA***, a DMA controller consists of four main parts:

- DMA configuration through AHB slave interface
- Data transmission through two AHB master interfaces for memory access and peripheral access
- An arbiter inside to manage multiple peripheral requests coming at the same time
- Channel management to control address/data selection and data counting

9.4. Function overview

9.4.1. 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. The source and destination addresses are computed by the DMA controller based on the programmed values in the DMA_CHxPADDR, DMA_CHxMADDR, and DMA_CHxCTL registers. The DMA_CHxCNT register controls how many transfers to be transmitted on the channel. The PWIDTH and MWIDTH bits in the DMA_CHxCTL register determine how many bytes to be transmitted in a transfer.

Suppose DMA_CHxCNT is 4, and both PNAGA and MNAGA are set. The DMA transfer operations for each combination of PWIDTH and MWIDTH are shown in the following ***Table 9-1. DMA transfer operation***.

Table 9-1. DMA transfer operation

Transfer size		Transfer operations	
Source	Destination	Source	Destination
32 bits	32 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B3B2B1B0[31:0] @0x0 2: Write B7B6B5B4[31:0] @0x4 3: Write BBBAB9B8[31:0] @0x8 4: Write BFBEBDBC[31:0] @0xC
32 bits	16 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B1B0[7:0] @0x0 2: Write B5B4[7:0] @0x2 3: Write B9B8[7:0] @0x4 4: Write BDBC[7:0] @0x6
32 bits	8 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B0[7:0] @0x0 2: Write B4[7:0] @0x1 3: Write B8[7:0] @0x2 4: Write BC[7:0] @0x3
16 bits	32 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write 0000B1B0[31:0] @0x0 2: Write 0000B3B2[31:0] @0x4 3: Write 0000B5B4[31:0] @0x8 4: Write 0000B7B6[31:0] @0xC
16 bits	16 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write B1B0[15:0] @0x0 2: Write B3B2[15:0] @0x2 3: Write B5B4[15:0] @0x4 4: Write B7B6[15:0] @0x6
16 bits	8 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write B0[7:0] @0x0 2: Write B2[7:0] @0x1 3: Write B4[7:0] @0x2 4: Write B6[7:0] @0x3
8 bits	32 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1: Write 000000B0[31:0] @0x0 2: Write 000000B1[31:0] @0x4 3: Write 000000B2[31:0] @0x8 4: Write 000000B3[31:0] @0xC
8 bits	16 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1, Write 00B0[15:0] @0x0 2, Write 00B1[15:0] @0x2 3, Write 00B2[15:0] @0x4 4, Write 00B3[15:0] @0x6
8 bits	8 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1, Write B0[7:0] @0x0 2, Write B1[7:0] @0x1 3, Write B2[7:0] @0x2 4, Write B3[7:0] @0x3

The CNT bits in the DMA_CHxCNT register control how many data to be transmitted on the channel and must be configured before enable the CHEN bit in the register. During the transmission, the CNT bits indicate the remaining number of data items to be transferred.

The DMA transmission is disabled by clearing the CHEN bit in the DMA_CHxCTL register.

- If the DMA transmission is not completed when the CHEN bit is cleared, two situations may be occurred when restart this DMA channel:
  - If no register configuration operations of the channel occurs before restart the DMA channel, the DMA will continue to complete the rest of the transmission.
  - If any register configuration operations occur, the DMA will restart a new transmission.
- If the DMA transmission has been finished when clearing the CHEN bit, enable the DMA channel without any register configuration operation will not launch any DMA transfer.

9.4.2. Peripheral handshake

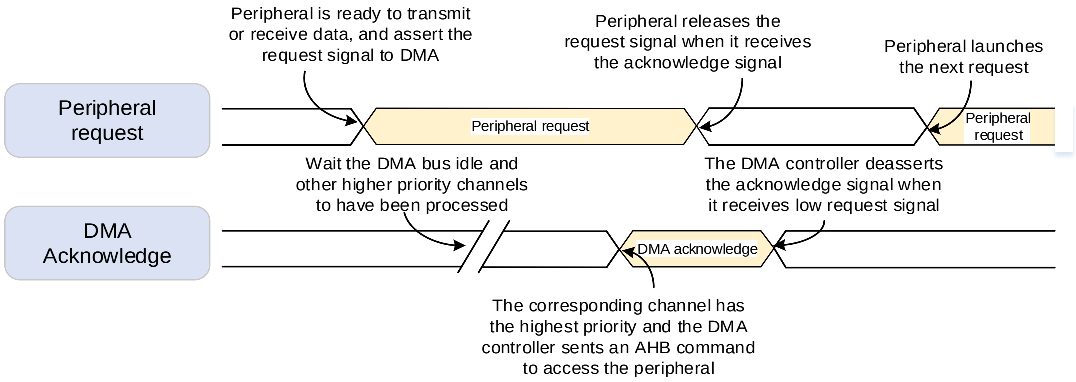
To ensure a well-organized and efficient data transfer, a handshake mechanism is introduced between the DMA and peripherals, including a request signal and a acknowledge signal:

- Request signal asserted by peripheral to DMA controller, indicating that the peripheral is ready to transmit or receive data

- Acknowledge signal responded by DMA to peripheral, indicating that the DMA controller has initiated an AHB command to access the peripheral

Figure 9-2. Handshake mechanism shows how the handshake mechanism works between the DMA controller and peripherals.

Figure 9-2. Handshake mechanism


Peripheral is ready to transmit or receive data, and assert the request signal to DMA
Peripheral releases the request signal when it receives the acknowledge signal
Peripheral launches the next request
Peripheral request
Peripheral request
Peripheral request
Wait the DMA bus idle and other higher priority channels to have been processed
The DMA controller deasserts the acknowledge signal when it receives low request signal
DMA Acknowledge
DMA acknowledge
The corresponding channel has the highest priority and the DMA controller sents an AHB command to access the peripheral

9.4.3. Arbitration

When two or more requests are received at the same time, the arbiter determines which request is served based on the priorities of channels. There are two-stage priorities, including the software priority and the hardware priority. The arbiter determines which channel is selected to respond according to the following priority rules:

- Software priority: Four levels, including low, medium, high and ultra high by configuring the PRIO bits in the DMA_CHxCTL register.

- For channels with equal software priority level, priority is given to the channel with lower channel number.

9.4.4. Address generation

Two kinds of address generation algorithm are implemented independently for memory and peripheral, including the fixed mode and the increased mode. The PNAGA and MNAGA bit in the DMA_CHxCTL register are used to configure the next address generation algorithm of peripheral and memory.

In the fixed mode, the next address is always equal to the base address configured in the base address registers (DMA_CHxPADDR, DMA_CHxMADDR).

In the increasing mode, the next address is equal to the current address plus 1 or 2 or 4, depending on the transfer data width.

9.4.5. Circular mode

Circular mode is implemented to handle continue peripheral requests (for example, ADC scan mode). The circular mode is enabled by setting the CMEN bit in the DMA_CHxCTL register.

In circular mode, the CNT bits are automatically reloaded with the pre-programmed value and the full transfer finish flag is asserted at the end of every DMA transfer. DMA can always responds the peripheral request until the CHEN bit in the DMA_CHxCTL register is cleared.

9.4.6. Memory to memory mode

The memory to memory mode is enabled by setting the M2M bit in the DMA_CHxCTL register. In this mode, the DMA channel can also work without being triggered by a request from a peripheral. The DMA channel starts transferring as soon as it is enabled by setting the CHEN bit in the DMA_CHxCTL register, and completed when the DMA_CHxCNT register reaches zero.

9.4.7. Channel configuration

When starting a new DMA transfer, it is recommended to respect the following steps:

1. Read the CHEN bit and judge whether the channel is enabled or not. If the channel is enabled, clear the CHEN bit by software. When the CHEN bit is read as ‘0’, configuring and starting a new DMA transfer is allowed.
2. Configure the M2M bit and DIR bit in the DMA_CHxCTL register to set the transfer mode.
3. Configure the CMEN bit in the DMA_CHxCTL register to enable/disable the circular mode.
4. Configure the PRIO bits in the DMA_CHxCTL register to set the channel software priority.
5. Configure the memory and peripheral transfer width, memory and peripheral address generation algorithm in the DMA_CHxCTL register.
6. Configure the enable bit for full transfer finish interrupt, half transfer finish interrupt, transfer error interrupt in the DMA_CHxCTL register.
7. Configure the DMA_CHxPADDR register for setting the peripheral base address.
8. Configure the DMA_CHxMADDR register for setting the memory base address.
9. Configure the DMA_CHxCNT register to set the total transfer data number.

10. Configure the CHEN bit with ‘1’ in the DMA_CHxCTL register to enable the channel.

9.4.8. Interrupt

Each DMA channel has a dedicated interrupt. There are three types of interrupt event, including full transfer finish, half transfer finish, and transfer error.

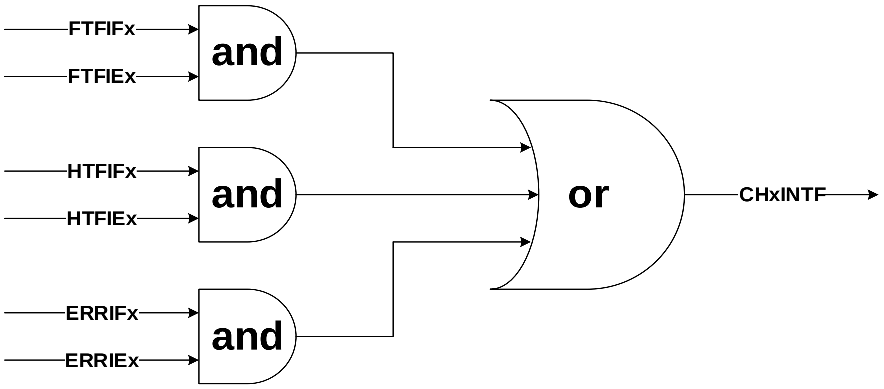
Each interrupt event has a dedicated flag bit in the DMA_INTF register, a dedicated clear bit in the DMA_INTC register, and a dedicated enable bit in the DMA_CHxCTL register. The relationship is described in the following ***Table 9-2. interrupt events***.

Table 9-2. interrupt events

Interrupt event	Flag bit	Clear bit	Enable bit
	DMA_INTF	DMA_INTC	DMA_CHxCTL
Full transfer finish	FTFIF	FTFIFC	FTFIE
Half transfer finish	HTFIF	HTFIFC	HTFIE
Transfer error	ERRIF	ERRIFC	ERRIE

The DMA interrupt logic is shown in the ***Figure 9-3. DMA interrupt logic***, an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

Figure 9-3. DMA interrupt logic



Note: “x” indicates channel number (for DMA0, x=0…6. for DMA1, x=0…4).

9.4.9. DMA request mapping

Several requests from peripherals may be mapped to one DMA channel. They are logically ORed before entering the DMA. For details, see the following ***Figure 9-4. DMA0 request mapping*** and ***Figure 9-5. DMA1 request mapping***. The request of each peripheral can be independently enabled or disabled by programming the registers of the corresponding peripheral. The user has to ensure that only one request is enabled at a time on one channel. ***Table 9-3. DMA0 requests for each channel*** lists the support request from peripheral for each channel of DMA0, and ***Table 9-4. DMA1 requests for each channel*** lists the support request from peripheral for each channel of DMA1.

Figure 9-4. DMA0 request mapping

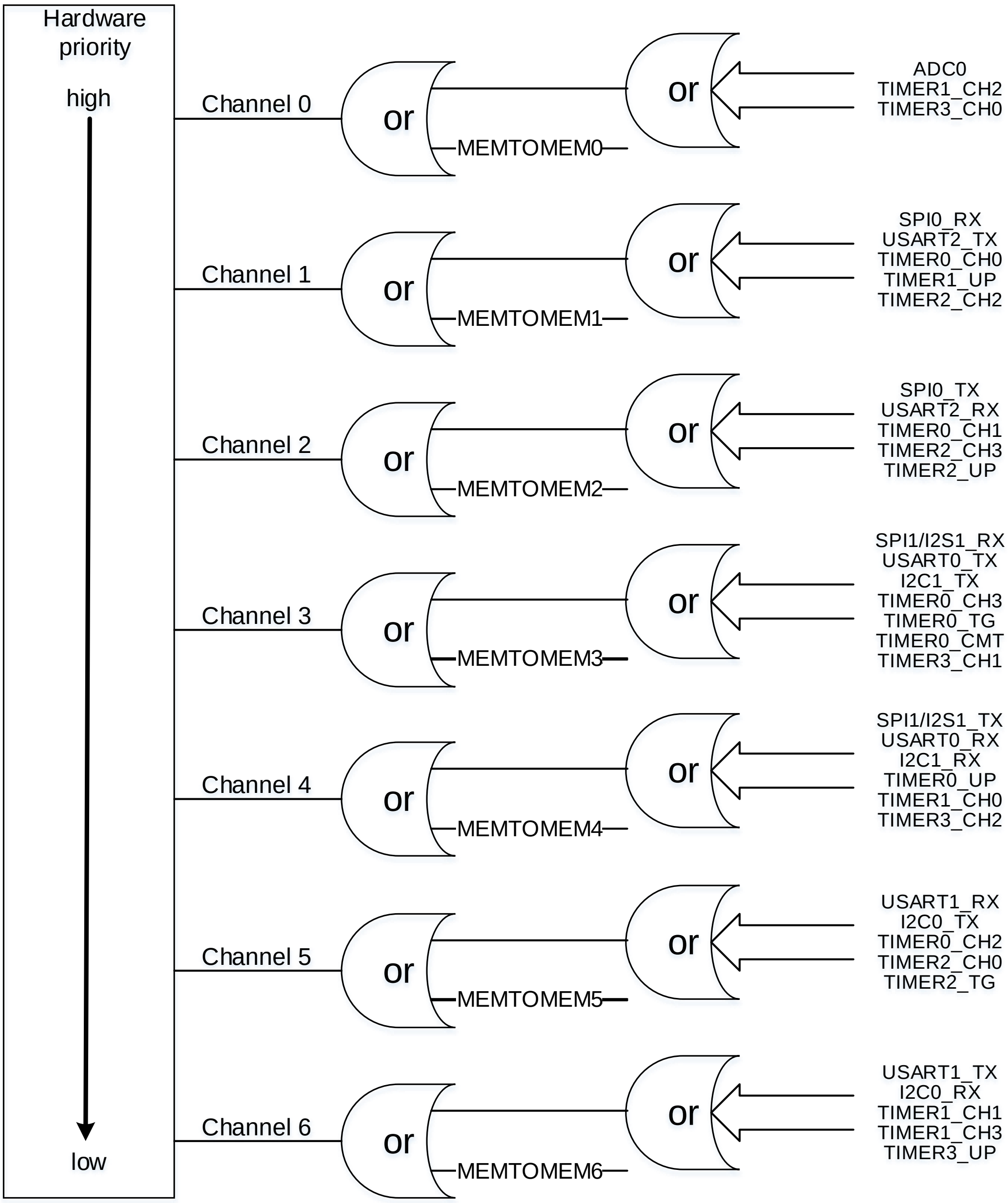


Table 9-3. DMA0 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
TIMER0	•	TIMER0_CH0	TIMER0_CH1	TIMER0_CH3 TIMER0_TG TIMER0_CMT	TIMER0_UP	TIMER0_CH2	•
TIMER1	TIMER1_CH2	TIMER1_UP	•	•	TIMER1_CH0	•	TIMER1_CH1 TIMER1_CH3
TIMER2	•	TIMER2_CH2	TIMER2_CH3	•	•	TIMER2_CH0	•

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
			TIMER2_UP			TIMER2_TG	
TIMER3	TIMER3_CH0	●	●	TIMER3_CH1	TIMER3_CH2	●	TIMER3_UP
ADC0	ADC0	●	●	●	●	●	●
SPI/I2S	●	SPI0_RX	SPI0_TX	SPI1/I2S1_RX	SPI1/I2S1_TX	●	●
USART	●	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	●	●	●	I2C1_TX	I2C1_RX	I2C0_TX	I2C0_RX

Figure 9-5. DMA1 request mapping

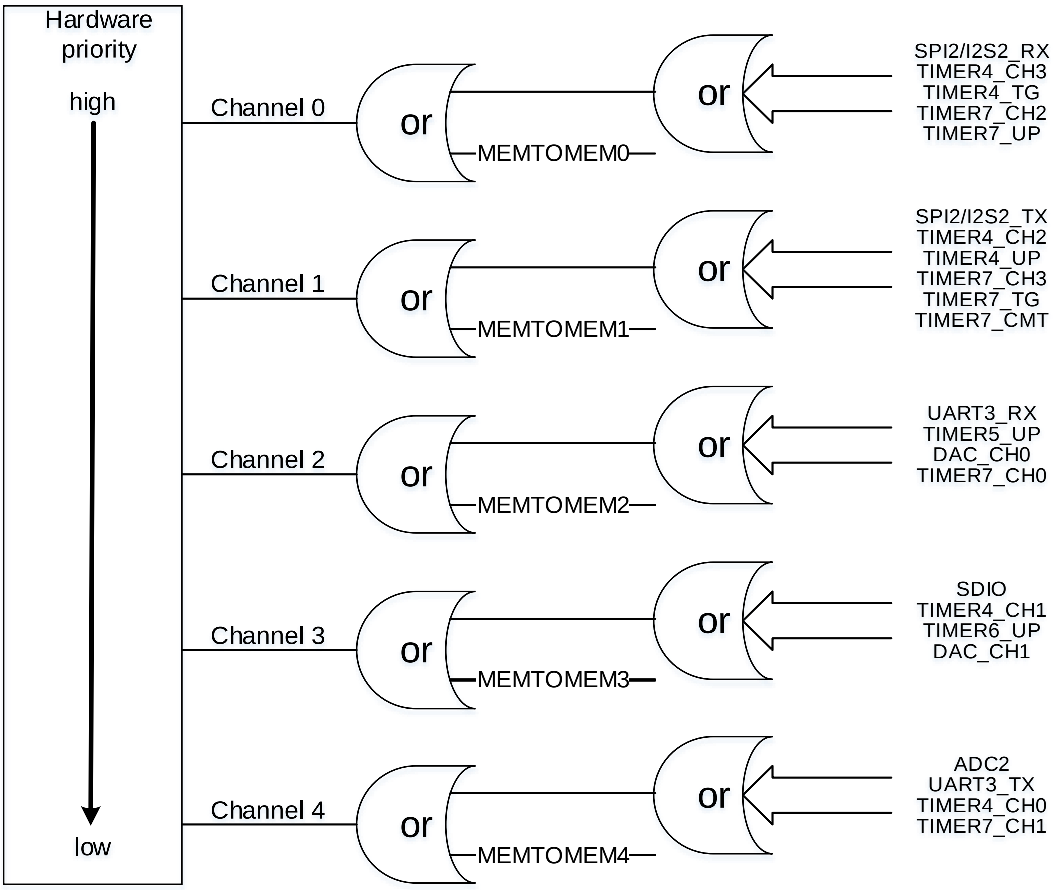


Table 9-4. DMA1 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
TIMER4	TIMER4_CH3 TIMER4_TG	TIMER4_CH2 TIMER4_UP	●	TIMER4_CH1	TIMER4_CH0
TIMER5	●	●	TIMER5_UP	●	●
TIMER6	●	●	●	TIMER6_UP	●
TIMER7	TIMER7_CH2 TIMER7_UP	TIMER7_CH3 TIMER7_TG TIMER7_CMT	TIMER7_CH0	●	TIMER7_CH1
ADC2	●	●	●	●	ADC2

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
DAC	●	●	DAC_CH0	DAC_CH1	●
SPI/I2S	SPI2/I2S2_RX	SPI2/I2S2_TX	●	●	●
USART	●	●	UART3_RX	●	UART3_TX
SDIO	●	●	●	SDIO	●

9.5. Register definition

Note: For DMA1 having 5 channels, all bits related to channel 5 and channel 6 in the following registers are not suitable for DMA1.

9.5.1. Interrupt flag register (DMA_INTF)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ERRIF6	HTFIF6	FTFIF6	GIF6	ERRIF5	HTFIF5	FTFIF5	GIF5	ERRIF4	HTFIF4	FTFIF4	GIF4
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIF3	HTFIF3	FTFIF3	GIF3	ERRIF2	HTFIF2	FTFIF2	GIF2	ERRIF1	HTFIF1	FTFIF1	GIF1	ERRIF0	HTFIF0	FTFIF0	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:28	Reserved	Keep at reset value
27/23/19/ 15/11/7/3	ERRIFx	Error flag of channel x (x=0…6) Hardware set and software cleared by configuring DMA_INTC register. 0: Transfer error has not occurred on channel x 1: Transfer error has occurred on channel x
26/22/18/ 14/10/6/2	HTFIFx	Half transfer finish flag of channel x (x=0…6) Hardware set and software cleared by configuring DMA_INTC register. 0: Half number of transfer has not finished on channel x 1: Half number of transfer has finished on channel x
25/21/17/ 13/9/5/1	FTFIFx	Full Transfer finish flag of channel x (x=0…6) Hardware set and software cleared by configuring DMA_INTC register. 0: Transfer has not finished on channel x 1: Transfer has finished on channel x
24/20/16/ 12/8/4/0	GIFx	Global interrupt flag of channel x (x=0…6) Hardware set and software cleared by configuring DMA_INTC register. 0: None of ERRIF, HTFIF or FTFIF occurs on channel x 1: At least one of ERRIF, HTFIF or FTFIF occurs on channel x

9.5.2. Interrupt flag clear register (DMA_INTC)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ERRIFC6	HTFIFC6	FTFIFC6	GIFC6	ERRIFC5	HTFIFC5	FTFIFC5	GIFC5	ERRIFC4	HTFIFC4	FTFIFC4	GIFC4
				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIFC3	HTFIFC3	FTFIFC3	GIFC3	ERRIFC2	HTFIC2	FTFIFC2	GIFC2	ERRIFC1	HTFIFC1	FTFIFC1	GIFC1	ERRIFC0	HTFIFC0	FTFIFC0	GIFC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:28	Reserved	Keep at reset value
27/23/19/ 15/11/7/3	ERRIFCx	Clear bit for error flag of channel x (x=0…6) 0: No effect 1: Clear error flag
26/22/18/ 14/10/6/2	HTFIFCx	Clear bit for half transfer finish flag of channel x (x=0…6) 0: No effect 1: Clear half transfer finish flag
25/21/17/ 13/9/5/1	FTFIFCx	Clear bit for full transfer finish flag of channel x (x=0…6) 0: No effect 1: Clear full transfer finish flag
24/20/16/ 12/8/4/0	GIFCx	Clear global interrupt flag of channel x (x=0…6) 0: No effect 1: Clear GIFx, ERRIFx, HTFIFx and FTFIFx bits in the DMA_INTF register

9.5.3. Channel x control register (DMA_CHxCTL)

x = 0...6, where x is a channel number

Address offset: 0x08 + 0x14 × x
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	M2M	PRIO[1:0]		MWIDTH[1:0]		PWIDTH[1:0]		MNAGA	PNAGA	CMEN	DIR	ERRIE	HTFIE	FTFIE	CHEN
	rw	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:15	Reserved	Keep at reset value
14	M2M	Memory to Memory Mode Software set and cleared

		0: Disable Memory to Memory Mode 1: Enable Memory to Memory mode This bit can not be written when CHEN is ‘1’.
13:12	PRIO[1:0]	Priority level Software set and cleared 00: Low 01: Medium 10: High 11: Ultra high These bits can not be written when CHEN is ‘1’.
11:10	MWIDTH[1:0]	Transfer data size of memory Software set and cleared 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can not be written when CHEN is ‘1’.
9:8	PWIDTH[1:0]	Transfer data size of peripheral Software set and cleared 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can not be written when CHEN is ‘1’.
7	MNAGA	Next address generation algorithm of memory Software set and cleared 0: Fixed address mode 1: Increasing address mode This bit can not be written when CHEN is ‘1’.
6	PNAGA	Next address generation algorithm of peripheral Software set and cleared 0: Fixed address mode 1: Increasing address mode This bit can not be written when CHEN is ‘1’.
5	CMEN	Circular mode enable Software set and cleared 0: Disable circular mode 1: Enable circular mode This bit can not be written when CHEN is ‘1’.

4	DIR	Transfer direction Software set and cleared 0: Read from peripheral and write to memory 1: Read from memory and write to peripheral This bit can not be written when CHEN is ‘1’.
3	ERRIE	Enable bit for channel error interrupt Software set and cleared 0: Disable the channel error interrupt 1: Enable the channel error interrupt
2	HTFIE	Enable bit for channel half transfer finish interrupt Software set and cleared 0:Disable channel half transfer finish interrupt 1:Enable channel half transfer finish interrupt
1	FTFIE	Enable bit for channel full transfer finish interrupt Software set and cleared 0:Disable channel full transfer finish interrupt 1:Enable channel full transfer finish interrupt
0	CHEN	Channel enable Software set and cleared 0:Disable channel 1:Enable channel

9.5.4. Channel x counter register (DMA_CHxCNT)

x = 0...6, where x is a channel number

Address offset: 0x0C + 0x14 × x

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Keep at reset value
15:0	CNT[15:0]	Transfer counter These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’.

This register indicates how many transfers remain. Once the channel is enabled, it is read-only, and decreases after each DMA transfer. If the register is zero, no transaction can be issued whether the channel is enabled or not. Once the transmission of the channel is complete, the register can be reloaded automatically by the previously programmed value if the channel is configured in circular mode.

9.5.5. Channel x peripheral base address register (DMA_CHxPADDR)

x = 0...6, where x is a channel number

Address offset: 0x10 + 0x14 × x

Reset value: 0x0000 0000

Note: Do not configure this register when channel is enabled.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">PADDR[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">PADDR[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:0	PADDR[31:0]	Peripheral base address These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’. When PWIDTH is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address. When PWIDTH is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.

9.5.6. Channel x memory base address register (DMA_CHxMADDR)

x = 0...6, where x is a channel number

Address offset: 0x14 + 0x14 × x

Reset value: 0x0000 0000

Note: Do not configure this register when channel is enabled.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">MADDR[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">MADDR[15:0]</td></tr>
</table>

rw

Bits	Fields	Descriptions
31:0	MADDR[31:0]	Memory base address These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’. When MWIDTH in the DMA_CHxCTL register is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address. When MWIDTH in the DMA_CHxCTL register is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.

10. Debug (DBG)

10.1. Overview

The GD32F10x series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the ARM CoreSightTM 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 hold unit helps debugger to debug power saving mode, TIMER, I2C, WWDGT, FWDGT and CAN. When corresponding bit is set, provide clock when in power saving mode or hold the state for TIMER, WWDGT, FWDGT, I2C or CAN.

10.2. 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).

10.2.1. 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.

10.2.2. Pin assignment

The JTAG interface provides 5-pin standard JTAG, known as JTAG clock (JTCK), JTAG mode selection (JTMS), JTAG data input (JTDI), JTAG data output (JTDO) and JTAG reset (NJTRST, active low). The serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK). The two SW pin are multiplexed with two of five JTAG pin, which is SWDIO multiplexed with JTMS, SWCLK multiplexed with JTCK. The JTDO is also used as Trace async data output (TRACESWO) when async trace enabled.

The pin assignment are:

PA15	: JTDI
PA14	: JTCK/SWCLK
PA13	: JTMS/SWDIO
PB4	: NJTRST
PB3	: JTDO

By default, 5-pin standard JTAG debug mode is chosen after reset. Users can also use JTAG function without NJTRST pin, then the PB4 can be used to other GPIO functions. (NJTRST tied to 1 by hardware). If switch to SW debug mode, the PA15/PB4/PB3 are released to other GPIO functions. If JTAG and SW not used, all 5-pin can be released to other GPIO functions. Please refer to ***GPIO pin configuration***.

10.2.3. JTAG daisy chained structure

The Cortex-M3 JTAG TAP is connected to a Boundary-Scan (BSD) JTAG TAP. The BSD JTAG IR is 5-bit width, while the Cortec-M3 JTAG IR is 4-bit width. So when JTAG in IR shift step, it first shift 5-bit BYPASS instruction (5'b 11111) for BSD JTAG, and then shift normal 4-bit instruction for Cortext-M3 JTAG. Because of the data shift under BSD JTAG BYPASS mode, adding 1 extra bit to the data chain is needed.

The BSD JTAG IDCODE is 0x790007A3.

10.2.4. Debug reset

The JTAG-DP and SW-DP register are in the power on reset domain. The System reset initializes the majority of the Cortex-M3, excluding NVIC and debug logic, (FPB, DWT, and ITM). The NJTRST reset can reset JTAG TAP controller only. So, it can perform debug feature under system reset. Such as, halt-after-reset, which is the debugger sets halt under system reset, and the core halts immediately after the system reset is released.

10.2.5. JEDEC-106 ID code

The Cortex-M3 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FF000_0xE00FFFFF.

10.3. Debug hold function overview

10.3.1. Debug support for power saving mode

When STB_HOLD bit in DBG control register (DBG_CTL) is set and entering the standby mode, the clock of AHB bus and system clock are provided by CK_IRC8M, and the debugger can debug in standby mode. When exit the standby mode, a system reset generated.

When DSLP_HOLD bit in DBG control register (DBG_CTL) is set and entering the Deep-sleep mode, the clock of AHB bus and system clock are provided by CK_IRC8M, and the debugger can debug in Deep-sleep mode.

When SLP_HOLD bit in DBG control register (DBG_CTL) 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.

10.3.2. Debug support for TIMER, I2C, WWDGT, FWDGT and CAN

When the core halted and the corresponding bit in DBG control register (DBG_CTL) is set, the following behaved.

For TIMER, the timer counters stopped and hold for debug.

For I2C, SMBUS timeout hold for debug.

For WWDGT or FWDGT, the counter clock stopped for debug.

For CAN, the receive register stopped counting for debug.

10.4. Register definition

10.4.1. ID code register (DBG_ID)

Address: 0xE004 2000

Read only

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID_CODE[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	ID_CODE[31:0]	DBG ID code register These bits read by software, These bits are unchanged constant

10.4.2. Control register (DBG_CTL)

Address offset: 0x04

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved.	TIMER10_HOLD	TIMER9_HOLD	TIMER8_HOLD	TIMER13_HOLD	TIMER12_HOLD	TIMER11_HOLD	Reserved			CAN1_HOLD	TIMER6_HOLD	TIMER5_HOLD	TIMER4_HOLD	TIMER7_HOLD	I2C1_HOLD
	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C0_HOLD	CAN0_HOLD	TIMER3_HOLD	TIMER2_HOLD	TIMER1_HOLD	TIMER0_HOLD	WWDGT_HOLD	FWDGT_HOLD	TRACE_MODE		TRACE_IOEN	Reserved		STB_HOLD	DSLP_HOLD	SLP_HOLD
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw			rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30	TIMER10_HOLD	TIMER 10 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 10 counter for debug when core halted
29	TIMER9_HOLD	TIMER 9 hold bit This bit is set and reset by software

		0: no effect 1: hold the TIMER 9 counter for debug when core halted
28	TIMER8_HOLD	TIMER 8 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 8 counter for debug when core halted
27	TIMER13_HOLD	TIMER 13 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 13 counter for debug when core halted
26	TIMER12_HOLD	TIMER 12 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 12 counter for debug when core halted
25	TIMER11_HOLD	TIMER 11 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 11 counter for debug when core halted
24:22	Reserved	Must be kept at reset value
21	CAN1_HOLD	CAN1 hold bit This bit is set and reset by software 0: no effect 1: the receive register of CAN1 stops receiving data when core halted
20	TIMER6_HOLD	TIMER6 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER6 counter for debug when core halted
19	TIMER5_HOLD	TIMER5 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER5 counter for debug when core halted
18	TIMER4_HOLD	TIMER4 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER4 counter for debug when core halted
17	TIMER7_HOLD	TIMER7 hold bit This bit is set and reset by software 0: no effect

		1: hold the TIMER7 counter for debug when core halted
16	I2C1_HOLD	I2C1 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C1 SMBUS timeout for debug when core halted
15	I2C0_HOLD	I2C0 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C0 SMBUS timeout for debug when core halted
14	CAN0_HOLD	CAN0 hold bit This bit is set and reset by software 0: no effect 1: the receive register of CAN0 stops receiving data when core halted
13	TIMER3_HOLD	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
12	TIMER2_HOLD	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
11	TIMER1_HOLD	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
10	TIMER0_HOLD	TIMER 0 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 0 counter for debug when core halted
9	WWDGT_HOLD	WWDGT hold bit This bit is set and reset by software 0: no effect 1: hold the WWDGT counter clock for debug when core halted
8	FWDGT_HOLD	FWDGT hold bit This bit is set and reset by software 0: no effect 1: hold the FWDGT counter clock for debug when core halted
7:6	TRACE_MODE[1:0]	Trace pin allocation mode

		This bit is set and reset by software 00: Trace pin used in asynchronous mode 01: Trace pin used in synchronous mode and the data length is 1 10: Trace pin used in synchronous mode and the data length is 2 11: Trace pin used in synchronous mode and the data length is 4
5	TRACE_IOEN	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	Must be kept at reset value
2	STB_HOLD	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 CK_IRC8M, a system reset generated when exit standby mode
1	DSLP_HOLD	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 CK_IRC8M
0	SLP_HOLD	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.

11. Analog-to-digital converter (ADC)

11.1. Overview

The 12-bit ADC is an analog-to-digital converter using the successive approximation method. It has 18 multiplexed channels making the ADC convert analog signals from 16 external channels, and 2 internal channels. The analog watchdog allows the application to detect whether the input voltage goes outside the user-defined higher or lower thresholds. The analog signals of the channels can be converted by the ADC in single, continuous, scan or discontinuous mode. A left-aligned or right-aligned 16-bit data register holds the output of the ADC.

11.2. Characteristics

- High performance
  —12-bit
  —ADC sampling rate: 1 MSPs for 12-bit resolution
  —Self-calibration
  —Programmable sampling time
  —Data alignment with built-in data coherency
  —DMA support

- Analog input channels
  —16 external analog inputs
  —1 channel for internal temperature sensor (V_{SENSE})
  —1 channel for internal reference voltage (V_{REFINT})

- Start-of-conversion can be initiated
  —By software
  —By hardware triggers

- Conversion modes
  — Converts a single channel or scans a sequence of channels.
  —Single mode converts selected inputs once per trigger.
  —Continuous mode converts selected inputs continuously
  —Discontinuous mode
  —SYNC mode(the device with two or more ADCs)

- Analog watchdog

- Interrupt generation：
  —at the end of regular and inserted group conversions
  —analog watchdog event

- ADC supply requirements: 2.6V to 3.6V, and typical power supply voltage is 3.3V
- ADC input range: V_{REFN} ≤V_{IN} ≤V_{REFP}

11.3. Pins and internal signals

Figure 11-1. ADC module block diagram shows the ADC block diagram and *Table 11-2. ADC pins definition* gives the ADC pin description.

Table 11-1. ADC internal signals

Internal signal name	Signal type	Description
V_{SENSE}	Input	Internal temperature sensor output voltage
V_{REFINT}	Input	Internal voltage reference output voltage

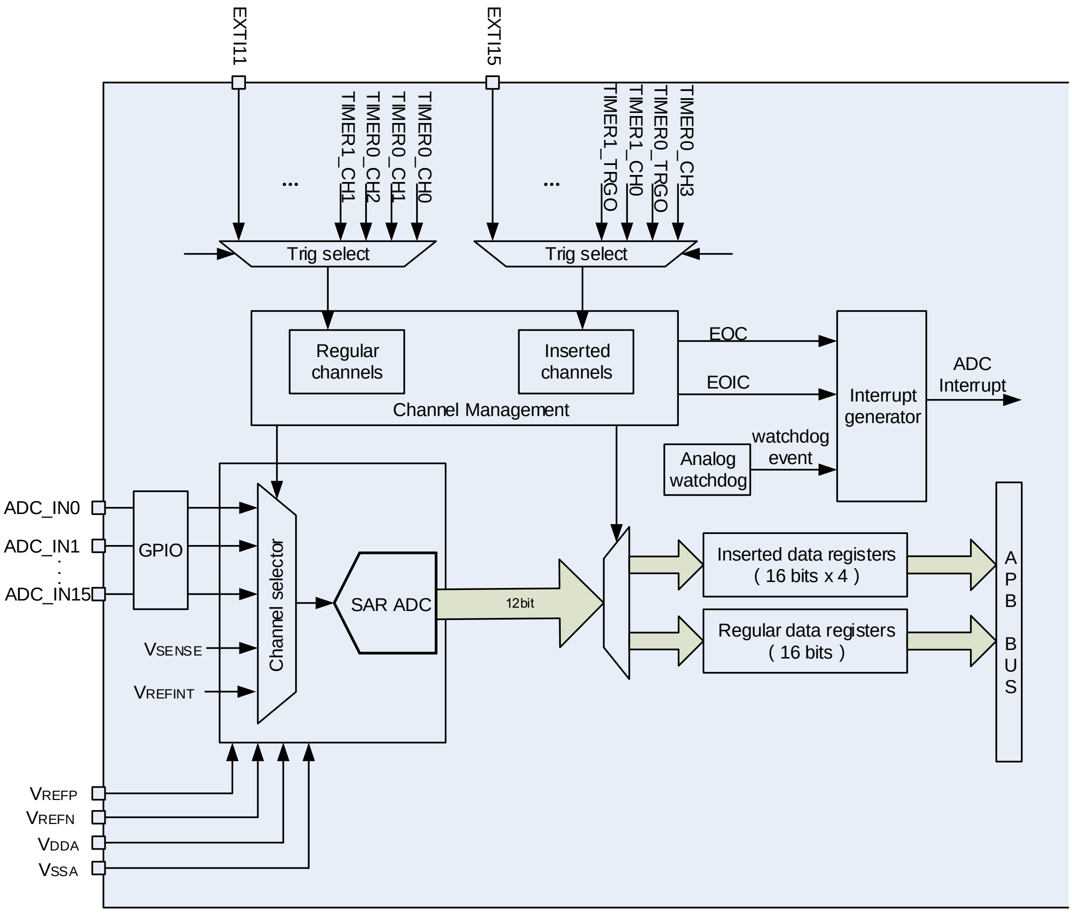
Table 11-2. ADC pins definition

Name	Signal type	Remarks
V_{DDA}	Input, analog power supply	Analog power supply equal to V_{DD} and 2.6 V ≤ V_{DDA}≤ 3.6 V
V_{SSA}	Input, analog power supply ground	Ground for analog power supply equal to V_{SS}
V_{REFP}	Input, analog reference positive	The positive reference voltage for the ADC, 2.6 V ≤ V_{REFP} ≤ V_{DDA}
V_{REFN}	Input, analog reference negative	The negative reference voltage for the ADC,V_{REFN} = V_{SSA}
ADCx_IN[15:0]	Input, Analog signals	Up to 16 external channels

Note: The ADC_IN[15:0] should set as Analog Input mode.

11.4. Functional overview

Figure 11-1. ADC module block diagram



11.4.1. Calibration (CLB)

The ADC has a foreground calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is completed. Calibration should be performed before starting A/D conversion. The calibration is initiated by software by setting bit CLB=1. CLB bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration is completed.

When the ADC operating conditions change (such as supply power voltage V_{DDA}, positive reference voltage V_{REFP}, temperature and so on), it is recommended to re-run a calibration cycle.

The internal analog calibration can be reset by setting the RSTCLB bit in ADC_CTL1 register.

Calibration software procedure:

1. Ensure that ADCON=1.

2. Delay 14 ADCCLK to wait for ADC stability.
3. Set RSTCLB (optional).
4. Set CLB=1.
5. Wait until CLB=0.

11.4.2. ADC clock

The ADCCLK clock provided by the clock controller is synchronous with the AHB and APB2 clock. The maximum frequency is 14MHz. The RCU controller has a dedicated programmable prescaler for the ADC clock.

11.4.3. ADCON switch

The ADCON bit on the ADC_CTL1 register is the enable switch of the ADC module. The ADC module will keep in reset state if this bit is 0. For power saving, when this bit is reset, the analog sub-module will be put into power-down mode.

Note: Delay no less than 20us should be added when set the ADCON.

11.4.4. Regular and inserted channel groups

The ADC supports 18 multiplexed channels and organizes the conversion results into two groups: a regular channel group and an inserted channel group.

In the regular group, a sequence of up to 16 conversions can be organized in a specific sequence. The ADC_RSQ0~ADC_RSQ2 registers specify the selected channels of the regular group. The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length.

In the inserted group, a sequence of up to 4 conversions can be organized in a specific sequence. The ADC_ISQ register specify the selected channels of the inserted group. The IL[1:0] bits in the ADC_ISQ register specify the total conversion sequence length.

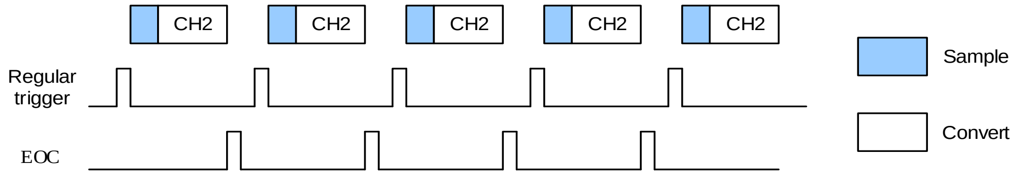
Note: If the regular group and inserted group both are used, the inserted group is advised not to use the sample time of 1.5 or 7.5.

11.4.5. Conversion modes

Single conversion mode

This mode can be running on both regular and inserted channel group. In the single conversion mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits of ADC_RSQ2 at a regular trigger or the channel specified in the ISQ3[4:0] bits of ADC_ISQ. When the ADCON has been set high, the ADC samples and converts a single channel, once the corresponding software trigger or external trigger is active.

Figure 11-2. Single conversion mode



After conversion of a single regular channel, the conversion data will be stored in the ADC_RDATA register, the EOC will be set. An interrupt will be generated if the EOCIE bit is set.

After conversion of a single inserted channel, the conversion data will be stored in the ADC_IDATA0 register, the EOC and EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set.

Software procedure for a single conversion of a regular channel:

1. Make sure the DISRC, SM in the ADC_CTL0 register and CTN bit in the ADC_CTL1 register are reset
2. Configure RSQ0 with the analog channel number
3. Configure ADC_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Read the converted in the ADC_RDATA register
8. Clear the EOC flag by writing 0 to it

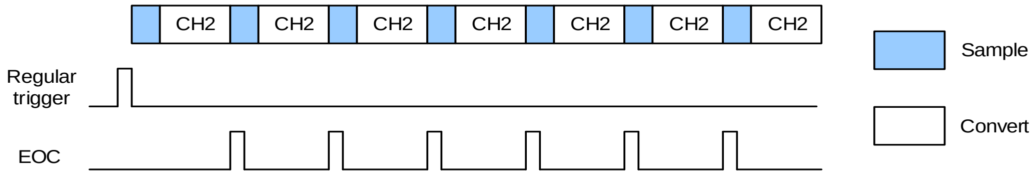
Software procedure for a single conversion of an inserted channel:

1. Make sure the DISIC, SM in the ADC_CTL0 register are reset
2. Configure ISQ3 with the analog channel number
3. Configure ADC_SAMPTx register
4. Configure ETEIC and ETSIC bits in the ADC_CTL1 register if in need
5. Set the SWICST bit, or generate an external trigger for the inserted group
6. Wait the EOC/EOIC flags to be set
7. Read the converted in the ADC_IDATA0 register
8. Clear the EOC/EOIC flags by writing 0 to them

Continuous conversion mode

This mode can be run on the regular channel group. The continuous conversion mode will be enabled when CTN bit in the ADC_CTL1 register is set. In this mode, the ADC performs conversion on the channel specified in the RSQ0[4:0]. When the ADCON has been set high, the ADC samples and converts specified channel, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA register.

Figure 11-3. Continuous conversion mode



Software procedure for continuous conversion on a regular channel:

1. Set the CTN bit in the ADC_CTL1 register
2. Configure RSQ0 with the analog channel number
3. Configure ADC_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Read the converted in the ADC_RDATA register
8. Clear the EOC flag by writing 0 to it
9. Repeat steps 6~8 as soon as the conversion is in need

To get rid of checking, DMA can be used to transfer the converted data:

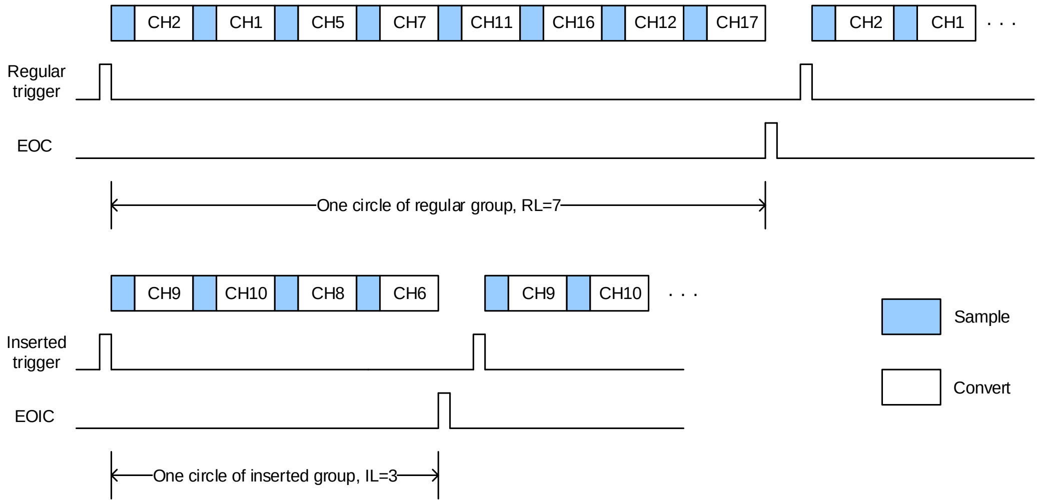
1. Set the CTN and DMA bit in the ADC_CTL1 register
2. Configure RSQ0 with the analog channel number
3. Configure ADC_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need
5. Prepare the DMA module to transfer data from the ADC_RDATA.
6. Set the SWRCST bit, or generate an external trigger for the regular group

Scan conversion mode

The scan conversion mode will be enabled when SM bit in the ADC_CTL0 register is set. In this mode, the ADC performs conversion on the channels with a specific sequence specified in the ADC_RSQ0~ADC_RSQ2 registers or ADC_ISQ register. When the ADCON has been set high, the ADC samples and converts specified channels one by one in the regular or inserted group till the end of the regular or inserted group, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA or ADC_IDATAx register. After conversion of the regular or inserted channel group, the EOC or EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set. The DMA bit in ADC_CTL1 register must be set when the regular channel group works in scan mode.

After conversion of a regular channel group, the conversion can be restarted automatically if the CTN bit in the ADC_CTL1 register is set.

Figure 11-4. Scan conversion mode, continuous disable



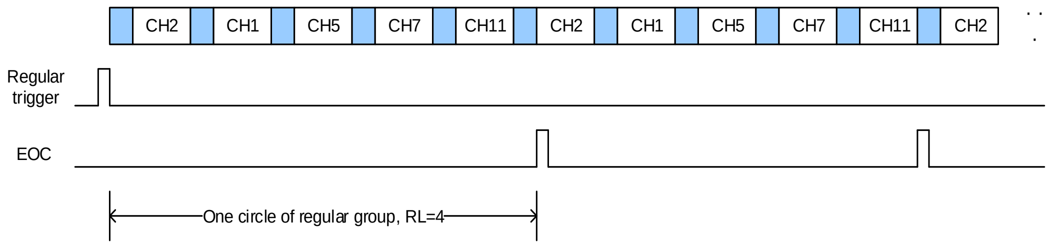
Software procedure for scan conversion on a regular channel group:

1. Set the SM bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register
2. Configure ADC_RSQx and ADC_SAMPTx registers
3. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need
4. Prepare the DMA module to transfer data from the ADC_RDATA.
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Clear the EOC flag by writing 0 to it

Software procedure for scan conversion on an inserted channel group:

1. Set the SM bit in the ADC_CTL0 register
2. Configure ADC_ISQ and ADC_SAMPTx registers
3. Configure ETEIC and ETSIC bits in the ADC_CTL1 register if in need
4. Set the SWICST bit, or generate an external trigger for the inserted group
5. Wait the EOC/EOIC flags to be set
6. Read the converted in the ADC_IDATAx register
7. Clear the EOC/EOIC flag by writing 0 to them

Figure 11-5. Scan conversion mode, continuous enable



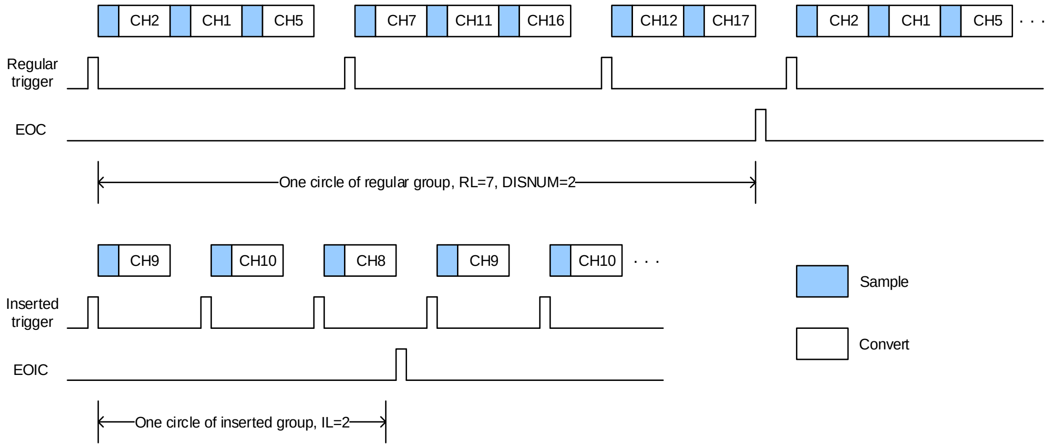
Discontinuous mode

For regular channel group, the discontinuous conversion mode will be enabled when DISRC bit in the ADC_CTL0 register is set. In this mode, the ADC performs a short sequence of n conversions (n<=8) which is a part of the sequence of conversions selected in the ADC_RSQ0~ADC_RSQ2 registers. The value of n is defined by the DISNUM[2:0] bits in the ADC_CTL0 register. When the corresponding software trigger or external trigger is active, the ADC samples and coverts the next n channels selected in the ADC_RSQ0~ADC_RSQ2 registers until all the channels in the regular sequence are done. The EOC will be set after every circle of the regular channel group. An interrupt will be generated if the EOCIE bit is set.

For inserted channel group, the discontinuous conversion mode will be enabled when DISIC bit in the ADC_CTL0 register is set. In this mode, the ADC performs one conversion which is a part of the sequence of conversions selected in the ADC_ISQ register. When the corresponding software trigger or external trigger is active, the ADC samples and coverts the next channel selected in the ADC_ISQ register until all the channels in the inserted sequence are done. The EOIC will be set after every circle of the inserted channel group. An interrupt will be generated if the EOICIE bit is set.

The regular and inserted groups cannot both work in discontinuous conversion mode. Only one group conversion can be set in discontinuous conversion mode at a time.

Figure 11-6. Discontinuous conversion mode



Software procedure for discontinuous conversion on a regular channel group:

1. Set the DISRC bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register
2. Configure DISNUM[2:0] bits in the ADC_CTL0 register
3. Configure ADC_RSQx and ADC_SAMPTx registers
4. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need
5. Prepare the DMA module to transfer data from the ADC_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an external trigger for the regular group
7. Repeat step6 if in need.
8. Wait the EOC flag to be set
9. Clear the EOC flag by writing 0 to it

Software procedure for discontinuous conversion on an inserted channel group:

1. Set the DISIC bit in the ADC_CTL0 register
2. Configure ADC_ISQ and ADC_SAMPTx registers

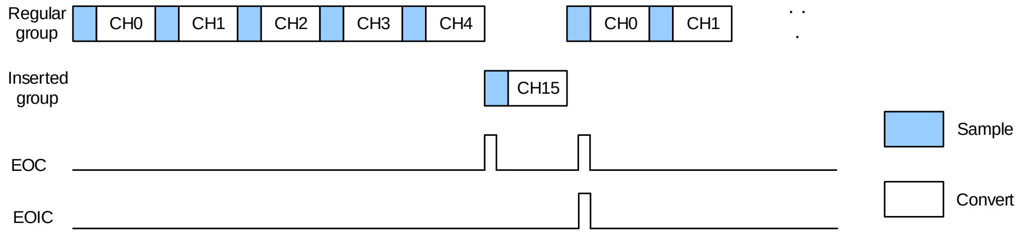
3. Configure ETEIC and ETSIC bits in the ADC_CTL1 register if in need
4. Set the SWICST bit, or generate an external trigger for the inserted group
5. Repeat step4 if in need
6. Wait the EOC/EOIC flags to be set
7. Read the converted in the ADC_IDATAx register
8. Clear the EOC/EOIC flag by writing 0 to them

11.4.6. Inserted channel management

Auto-insertion

The inserted group channels are automatically converted after the regular group channels when the ICA bit in ADC_CTL0 register is set. In this mode, external trigger on inserted channels cannot be enabled. A sequence of up to 20 conversions programmed in the ADC_RSQ0~ADC_RSQ2 and ADC_ISQ registers can be used to convert in this mode. In addition to the ICA bit, if the CTN bit is also set, regular channels followed by inserted channels are continuously converted.

Figure 11-7. Auto-insertion, CTN = 1

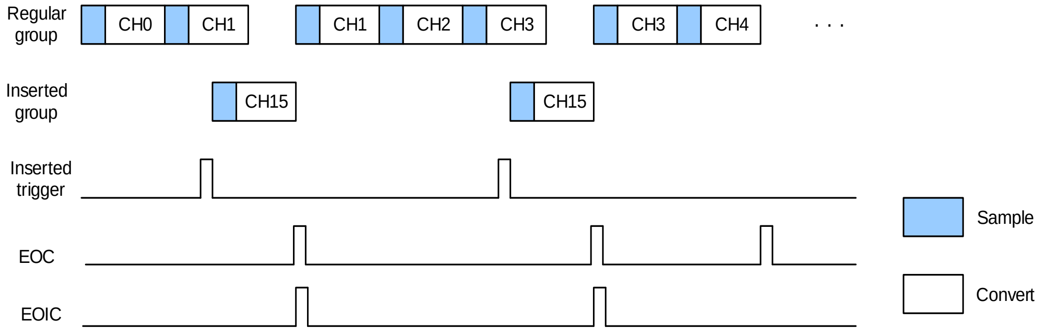


The auto insertion mode cannot be enabled when the discontinuous conversion mode is set.

Triggered insertion

If the ICA bit is cleared, the triggered insertion occurs if a software or external trigger occurs during the regular group channel conversion. In this situation, the ADC aborts from the current conversion and starts the conversion of inserted channel sequence. After the inserted channel group is done, the regular group channel conversion is resumed from the last aborted conversion.

Figure 11-8. Triggered insertion



11.4.7. Analog watchdog

The analog watchdog is enabled when the RWDEN and IWDEN bits in the ADC_CTL0 register are set for regular and inserted channel groups respectively. When the analog voltage converted by the ADC is below a low threshold or above a high threshold, the WDE bit in ADC_STAT register will be set. An interrupt will be generated if the WDEIE bit is set. The ADC_WDHT and ADC_WDLT registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold value is independent of the alignment, which is specified by the DAL bit in the ADC_CTL1 register. One or more channels, which are select by the RWDEN, IWDEN, WDSC and WDCHSEL[4:0] bits in ADC_CTL0 register, can be monitored by the analog watchdog.

11.4.8. Data alignment

The alignment of data stored after conversion can be specified by DAL bit in the ADC_CTL1 register.

After being decreased by the user-defined offset written in the ADC_IOFFx registers, the inserted group data value may be a negative value. The sign value is extended.

Figure 11-9. 12-bit Data alignment

Regular group data

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

Inserted group data

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

DAL=0

Regular group data

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

Inserted group data

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

DAL=1

11.4.9. Programmable sample time

The number of ADCCLK cycles which is used to sample the input voltage can be specified by the SPTn[2:0] bits in the ADC_SAMPT0 and ADC_SAMPT1 registers. A different sample time can be specified for each channel. For 12-bits resolution, the total conversion time is “sampling time + 12.5” ADCCLK cycles.

Example:

ADCCLK = 14MHz and sample time is 1.5 cycles, the total conversion time is “1.5+12.5” ADCCLK cycles, that means 1.0us.

Note: Reducing the sample clock and increasing the sample time can get a more stable sampled data. When the hardware is used, the input impedance outside need to be smaller.

11.4.10. External trigger

The conversion of regular or inserted group can be triggered by rising edge of external trigger inputs. The external trigger source of regular channel group is controlled by the ETSRC[2:0] bits in the ADC_CTL1 register, while the external trigger source of inserted channel group is controlled by the ETSIC[2:0] bits in the ADC_CTL1 register

ETSRC[2:0] and ETSIC[2:0] control bits are used to specify which out of 8 possible events can trigger conversion for the regular and inserted groups.

Table 11-3. External trigger for regular channels for ADC0 and ADC1

ETSRC[2:0]	Trigger Source	Trigger Type
000	TIMER0_CH0	Internal on-chip signal
001	TIMER0_CH1	
010	TIMER0_CH2	
011	TIMER1_CH1	
100	TIMER2_TRGO	
101	TIMER3_CH3	
110	EXTI11/TIMER7_TRGO	External signal
111	SWRCST	Software trigger

Table 11-4. External trigger for inserted channels for ADC0 and ADC1

ETSIC[2:0]	Trigger Source	Trigger Type
000	TIMER0_TRGO	Internal on-chip signal
001	TIMER0_CH3	
010	TIMER1_TRGO	
011	TIMER1_CH0	
100	TIMER2_CH3	
101	TIMER3_TRGO	
110	EXTI15/TIMER7_CH3	External signal
111	SWICST	Software trigger

Table 11-5. External trigger for regular channels for ADC2

ETSRC[2:0]	Trigger Source	Trigger Type
000	TIMER2_CH0	Internal on-chip signal
001	TIMER1_CH2	
010	TIMER0_CH2	
011	TIMER7_CH0	
100	TIMER7_TRGO	
101	TIMER4_CH0	
110	TIMER4_CH2	

111	SWRCST	Software trigger

Table 11-6. External trigger for inserted channels for ADC2

ETSIC[2:0]	Trigger Source	Trigger Type
000	TIMER0_TRGO	Internal on-chip signal
001	TIMER0_CH3	
010	TIMER3_CH2	
011	TIMER7_CH1	
100	TIMER7_CH3	
101	TIMER4_TRGO	
110	TIMER4_CH3	
111	SWICST	Software trigger

11.4.11. DMA request

The DMA request, which is enabled by the DMA bit of ADC_CTL1 register, is used to transfer data of regular group for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a regular channel. When this request is received, the DMA will transfer the converted data from the ADC_RDATA register to the destination location which is specified by the user.

11.4.12. Temperature sensor, and internal reference voltage V_{REFINT}

When the TSVREN bit of ADC_CTL1 register is set, the temperature sensor channel (ADC0_CH16) and V_{REFINT} channel (ADC0_CH17) is enabled. The temperature sensor can be used to measure the ambient temperature of the device. The sensor output voltage can be converted into a digital value by ADC. The sampling time for the temperature sensor is recommended to be set to at least 17.1 µs. When this sensor is not in use, it can be put in power down mode by resetting the TSVREN bit.

The output voltage of the temperature sensor changes linearly with temperature. Because there is an offset, which is up to 45 °C and varies from chip to chip due to process variation, the internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperature. When it is used to detect accurate temperature, an external temperature sensor part should be used to calibrate the offset error.

The internal voltage reference (V_{REFINT}) provides a stable (bandgap) voltage output for the ADC and Comparators. V_{REFINT} is internally connected to the ADC0_CH17 input channel.

11.5. ADC sync mode

In devices with two ADC, ADC sync mode can be used.

In ADC sync mode, the conversion starts alternately or simultaneously triggered by ADC0 master to ADC1 slave, according to the mode selected by the SYNCM[3:0] bits in ADC1_CTL0 register.

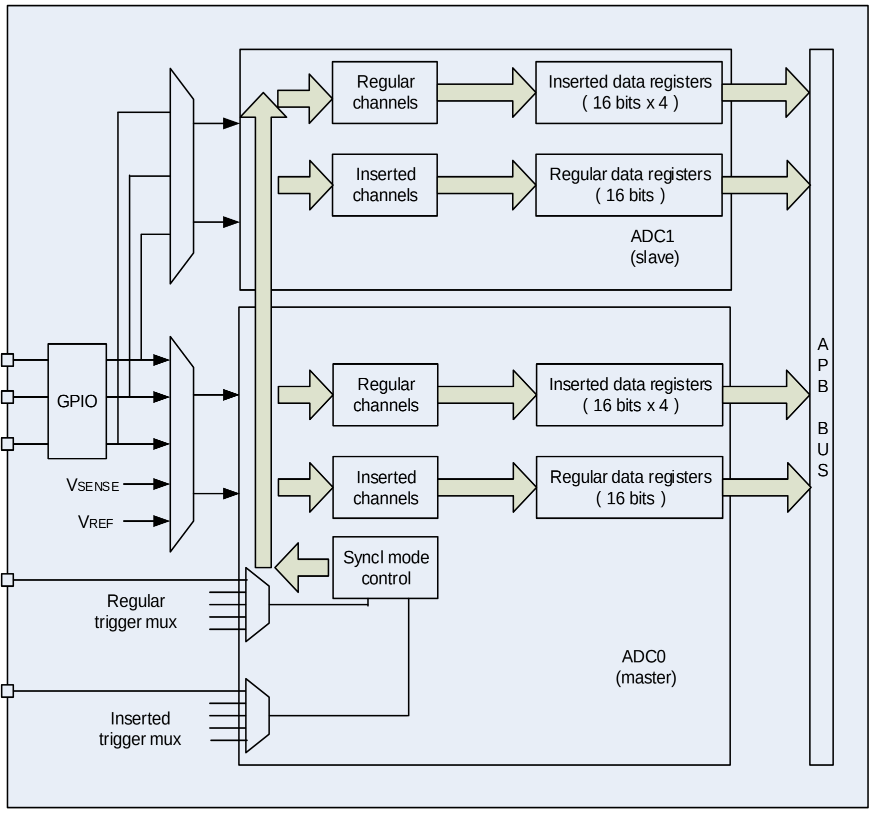
In sync mode, when configure the conversion which is triggered by an external event, the slave ADC must be configured as triggered by the software in order to prevent false triggers to start unwanted conversion. However, the external trigger must be enabled for ADC master and ADC slave.

The following modes can be configured:

- Free mode
- Regular parallel mode
- Inserted parallel mode
- Follow-up fast mode
- Follow-up slow mode
- Trigger rotation mode
- Inserted parallel mode + regular parallel mode
- Regular parallel mode + trigger rotation mode
- Inserted parallel mode + follow-up fast mode
- Inserted parallel mode + follow-up slow mode

In ADC sync mode, the DMA bit must be set even if it is not used; the converted data of ADC slave can be read from the master data register.

Figure 11-10. ADC sync block diagram



11.5.1. Free mode

In this mode, the ADC synchronization is bypassed, and each ADC works freely.

11.5.2. Regular parallel mode

This mode converts the regular channel simultaneously. The source of external trigger comes from the regular group MUX of ADC0 (selected by the ETSRC[2:0] bits in the ADC_CTL1 register). A simultaneous trigger is provided to ADC1.

At the end of conversion event on ADC0 or ADC1 an EOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC0/ADC1 regular channels are all converted. The behavior of regular parallel mode shows in the ***Figure 11-11. Regular parallel mode on 16 channels***.

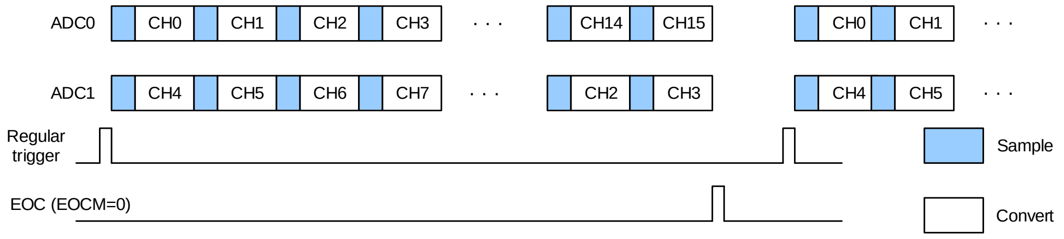
A 32-bit DMA is used, which transfers ADC_RDATA 32-bit register (the ADC_RDATA 32-bit register containing the ADC1 converted data in the upper half-word and the ADC0 converted data in the lower half-word) to SRAM.

Note:

1. Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

2. In parallel mode, exactly the same sampling time should be configured for the two channels that will be sampled simultaneously by ACD0 and ADC1.

Figure 11-11. Regular parallel mode on 16 channels



11.5.3. Inserted parallel mode

This mode converts the inserted channel simultaneously. The source of external trigger comes from the inserted group MUX of ADC0 (selected by the ETSIC[2:0] bits in the ADC_CTL1 register). A simultaneous trigger is provided to ADC1.

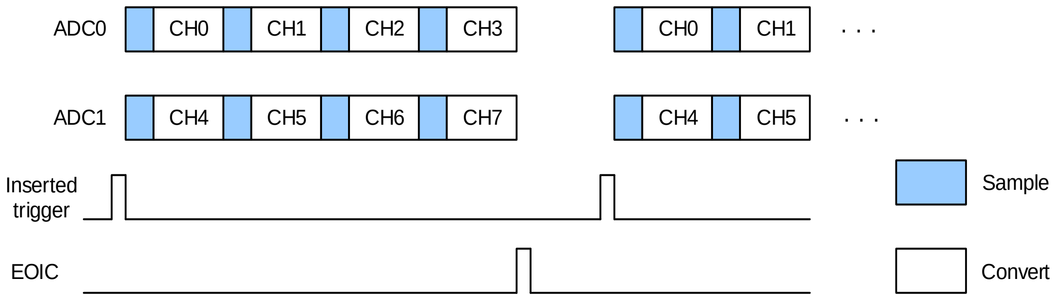
At the end of conversion event on ADC0 or ADC1, an EOIC interrupt is generated (if enabled on one of the two ADC interfaces). ADC0/ADC1 inserted channels are all converted, and the converted data is stored in the ADC_IDATAx registers of each ADC interface. The behavior of inserted parallel mode shows in the ***Figure 11-12. Inserted parallel mode on 4 channels***.

Note:

1. Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

2. In parallel mode, exactly the same sampling time should be configured for the two channels that will be sampled simultaneously by ADC0 and ADC1.

Figure 11-12. Inserted parallel mode on 4 channels



11.5.4. Follow-up fast mode

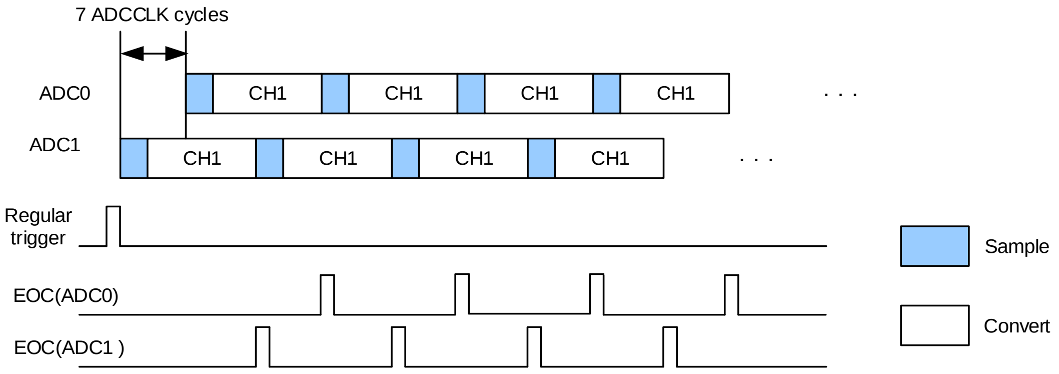
This mode can be running on the regular channel group (usually one channel). The source of external trigger comes from the regular channel MUX of ADC0 (selected by the ETSRC[2:0] bits in the ADC_CTL1 register). When the trigger occurs, ADC1 runs immediately and ADC0 runs after 7 ADC clock cycles.

If the continuous mode is enabled for both ADC0 and ADC1, the selected regular channels of both ADCs are continuously converted. The behavior of follow-up fast mode shows in the ***Figure 11-13. Follow-up fast mode on 1 channel in continuous conversion mode***.

After an EOC interrupt is generated by ADC0 in case of setting the EOCIE bit, we can use a 32-bit DMA, which transfers to SRAM the ADC_RDATA 32-bit register containing the ADC1 converted data in the upper half word and the ADC0 converted data in the lower half word.

Note: The maximum sampling time allowed is <7 ADCCLK cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.

Figure 11-13. Follow-up fast mode on 1 channel in continuous conversion mode



11.5.5. Follow-up slow mode

This mode can be running on the regular channel group (usually one channel). The source of external trigger comes from the regular channel MUX of ADC0(selected by the ETSRC[2:0] bits in the ADC_CTL1 register).When the trigger occurs, ADC1 runs immediately, ADC0 runs after 14 ADC clock cycles, after the second 14 ADC clock cycles the ADC1 runs again.

Continuous mode can't be used in this mode, because it continuously converts the regular channel. The behavior of follow-up slow mode shows in the ***Figure 11-14. Follow-up slow mode on 1 channel***.

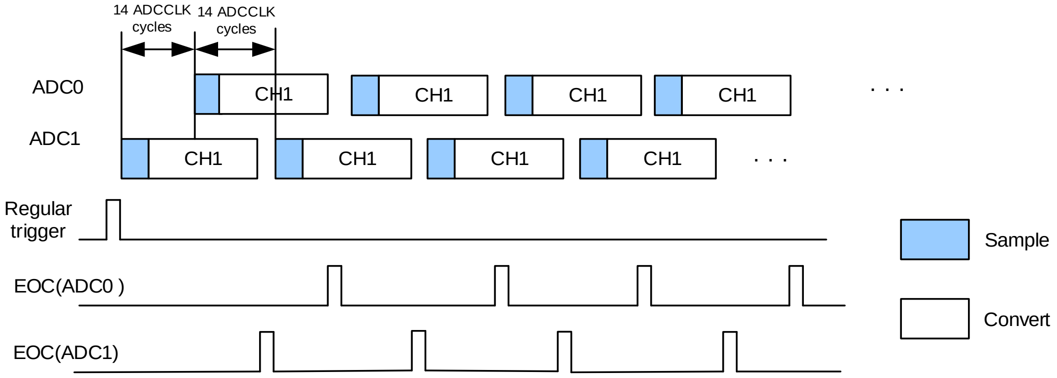
After an EOC interrupt is generated by ADC0 (if enabled through the EOCIE bit), we can use a 32-bit DMA, which transfers to SRAM the ADC_RDATA 32-bit register containing the ADC1 converted data in the upper half-word and the ADC0 converted data in the lower half-word.

Note:

1. The maximum sampling time allowed is <14 ADCCLK cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.

2. For both the fast and follow-up slow mode, we must ensure that no external trigger for inserted channel occurs.

Figure 11-14. Follow-up slow mode on 1 channel



11.5.6. Trigger rotation mode

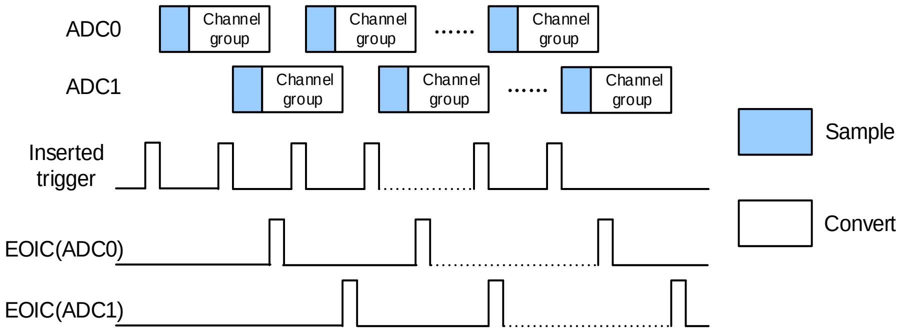
This mode can be running on the inserted channel group. The source of external trigger comes from the inserted channel MUX of ADC0 (selected by the ETSIC[2:0] bits in the ADC_CTL1 register).

When the first trigger occurs, all the inserted channels of ADC0 are converted. When the second trigger occurs, all the inserted channels of ADC1 are converted. The behavior of trigger rotation mode shows in the ***Figure 11-15. Trigger rotation: inserted channel group***.

If the EOIC interrupt of ADC0 and ADC1 are enabled, when all the channels of ADC0 or ADC1 have been converted, the corresponded interrupt occurred.

If another external trigger occurs after all inserted group channels have been converted, the trigger rotation process restarts by converting ADC0 inserted group channels.

Figure 11-15. Trigger rotation: inserted channel group



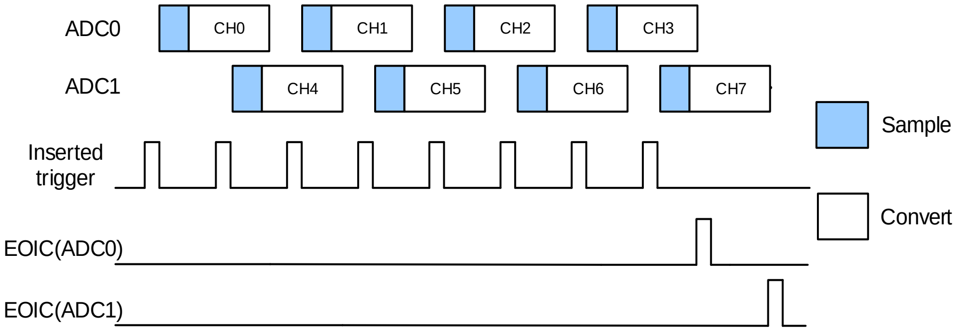
If the discontinuous mode is enabled for both ADC0 and ADC1, when the first trigger occurs, the first inserted channel in ADC0 is converted. When the second trigger occurs, the first inserted channel in ADC1 is converted. Then the second channel in ADC0, the second channel in ADC1, and so on.

The behavior of trigger rotation discontinuous mode shows in the ***Figure 11-16. Trigger rotation: inserted channels in discontinuous mode***.

If the EOIC interrupt of ADC0 and ADC1 are enabled. When all the channels of ADC0 or ADC1 have been converted, the corresponded interrupt occurred.

If another external trigger occurs after all inserted group channels have been converted then the trigger rotation process restarts.

Figure 11-16. Trigger rotation: inserted channels in discontinuous mode



11.5.7. Combined regular parallel & inserted parallel mode

In the free mode, the conversion of regular group can be interrupted by the conversion of inserted group. In the sync mode, it is also possible to interrupt parallel conversion of a regular group to insert parallel conversion of an inserted group.

Note: In combined regular parallel + inserted parallel mode, the sampling time for the two ADCs should be configured the same.

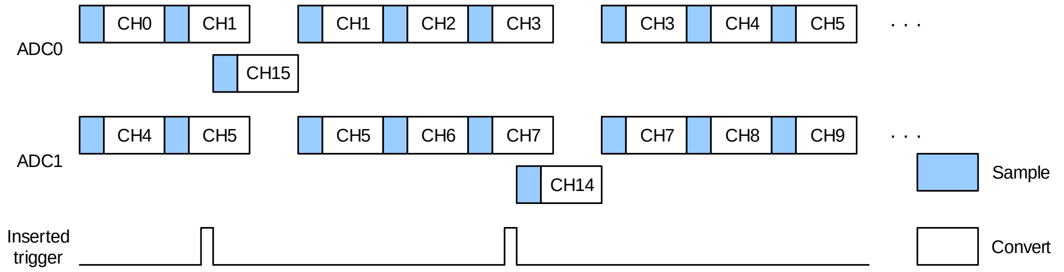
11.5.8. Combined regular parallel & trigger rotation mode

It is possible to interrupt regular group parallel conversion to start trigger rotation conversion of an inserted group. The behavior of an alternate trigger interrupt a regular parallel conversion shows in the ***Figure 11-17. Regular parallel & trigger rotation mode***.

When the inserted event occurs, the inserted rotation conversion is immediately started. If regular conversion is already running, in order to ensure synchronization after the inserted conversion, the regular conversion of both (master/slave) ADCs is stopped and resumed synchronously at the end of the inserted conversion.

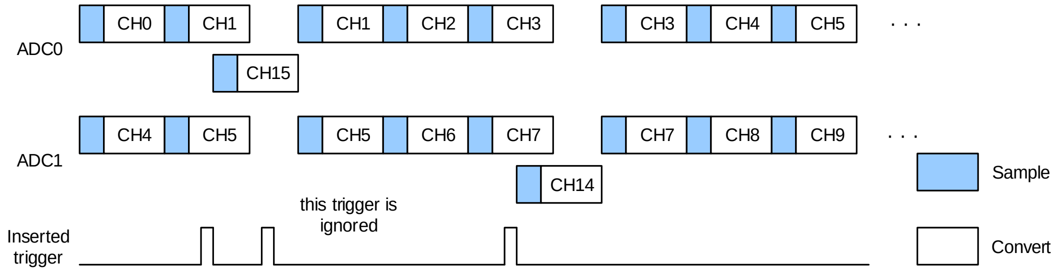
Note: In combined regular parallel + trigger rotation mode, the sampling time for the two ADCs should be configured the same.

Figure 11-17. Regular parallel & trigger rotation mode



If one inserted trigger occurs during an inserted conversion that has interrupted a regular conversion, it will be ignored. ***Figure 11-18. Trigger occurs during inserted conversion*** shows the case (the third trigger is ignored).

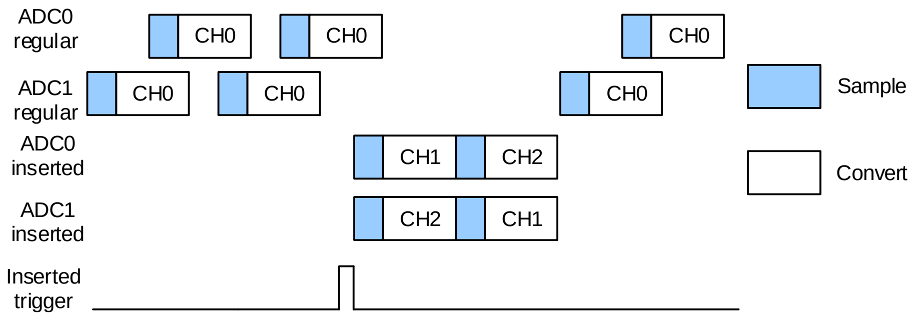
Figure 11-18. Trigger occurs during inserted conversion



11.5.9. Combined inserted parallel & follow-up mode

It is possible to interrupt a follow-up conversion (both fast and slow) with an inserted event. When the inserted trigger occurs, the follow-up conversion is interrupted and the inserted conversion starts, at the end of the inserted sequence the follow-up conversion is resumed. ***Figure 11-19 Follow-up single channel with inserted sequence CH1, CH2*** shows the behavior of this mode.

Figure 11-19 Follow-up single channel with inserted sequence CH1, CH2



11.6. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for regular and inserted groups
- The analog watchdog event

Separate interrupt enable bits are available for flexibility.

The interrupts of ADC0, ADC1 and ADC2 are mapped into the same interrupt vector ISR[18].

11.7. Register definition

11.7.1. Status register (ADC_STAT)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRC	STIC	EOIC	EOC	WDE
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value
4	STRC	Start flag of regular channel group 0: No regular channel group started 1: Regular channel group started Set by hardware when regular channel conversion starts. Cleared by software writing 0 to it.
3	STIC	Start flag of inserted channel group 0: No inserted channel group started 1: Inserted channel group started Set by hardware when inserted channel group conversion starts. Cleared by software writing 0 to it.
2	EOIC	End of inserted group conversion flag 0: No end of inserted group conversion 1: End of inserted group conversion Set by hardware at the end of all inserted group channel conversion. Cleared by software writing 0 to it.
1	EOC	End of group conversion flag 0: No end of group conversion 1: End of group conversion Set by hardware at the end of a regular or inserted group channel conversion. Cleared by software writing 0 to it or by reading the ADC_RDATA register.
0	WDE	Analog watchdog event flag 0: No analog watchdog event

1: Analog watchdog event

Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT and ADC_WDHT registers. Cleared by software writing 0 to it.

11.7.2. Control register 0 (ADC_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RWDEN	IWDEN	Reserved		SYNCM[3:0]			
								rw	rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISNUM[2:0]			DISIC	DISRC	ICA	WDSC	SM	EOICIE	WDEIE	EOCIE	WDCHSEL[4:0]				
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	RWDEN	Regular channel analog watchdog enable 0: Regular channel analog watchdog disable 1: Regular channel analog watchdog enable
22	IWDEN	Inserted channel analog watchdog enable 0: Inserted channel analog watchdog disable 1: Inserted channel analog watchdog enable
21:20	Reserved	Must be kept at reset value
19:16	SYNCM[3:0]	Sync mode selection These bits use to select the operating mode. 0000: Free mode. 0001: Combined regular parallel + inserted parallel mode 0010: Combined regular parallel + trigger rotation mode 0011: Combined inserted parallel + follow-up fast mode 0100: Combined inserted parallel + follow-up slow mode 0101: Inserted parallel mode only 0110: Regular parallel mode only 0111: Follow-up fast mode only 1000: Follow-up slow mode only 1001: Trigger rotation mode only **Note:** These bits are reserved in ADC1 and ADC2.In sync mode, the change of configuration will cause unpredictable consequences. We must disable sync mode

		before any configuration change.
15:13	DISNUM[2:0]	Number of conversions in discontinuous mode The number of channels to be converted after a trigger will be DISNUM+1
12	DISIC	Discontinuous mode on inserted channels 0: Discontinuous mode on inserted channels disable 1: Discontinuous mode on inserted channels enable
11	DISRC	Discontinuous mode on regular channels 0: Discontinuous mode on regular channels disable 1: Discontinuous mode on regular channels enable
10	ICA	Inserted channel group convert automatically 0: Inserted channel group convert automatically disable 1: Inserted channel group convert automatically enable
9	WDSC	When in scan mode, analog watchdog is effective on a single channel 0: Analog watchdog is effective on all channels 1: Analog watchdog is effective on a single channel
8	SM	Scan mode 0: scan mode disable 1: scan mode enable
7	EOICIE	Interrupt enable for EOIC 0: EOIC interrupt disable 1: EOIC interrupt enable
6	WDEIE	Interrupt enable for WDE 0: WDE interrupt disable 1: WDE interrupt enable
5	EOCIE	Interrupt enable for EOC 0: EOC interrupt disable 1: EOC interrupt enable
4:0	WDCHSEL[4:0]	Analog watchdog channel select 00000: ADC channel0 00001: ADC channel1 00010: ADC channel2 00011: ADC channel 3 00100: ADC channel 4 00101: ADC channel 5 00110: ADC channel 6 00111: ADC channel 7 01000: ADC channel 8 01001: ADC channel 9

		01010: ADC channel 10 01011: ADC channel 11 01100: ADC channel 12 01101: ADC channel 13 01110: ADC channel 14 01111: ADC channel15 10000: ADC channel16 10001: ADC channel17 Other values are reserved. **Note:** ADC0 analog inputs Channel16 and Channel17 are internally connected to the temperature sensor, and to V_{REFINT} inputs. ADC1 analog inputs Channel16, and Channel17 are internally connected to V_{SSA}. ADC2 analog inputs Channel9, Channel14, Channel15, Channel16, and Channel17 are internally connected to $V_{SSA.}$

11.7.3. Control register 1 (ADC_CTL1)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREN	SWRCST	SWICST	ETERC	ETSRC[2:0]			Reserved
								rw	rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETEIC	ETSIC[2:0]			DAL	Reserved		DMA	Reserved				RSTCLB	CLB	CTN	ADCON
rw		rw		rw			rw					rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	TSVREN	Channel 16 and 17 enable of ADC0. 0: Channel 16 and 17 of ADC0 disable 1: Channel 16 and 17 of ADC0 enable
22	SWRCST	Start on regular channel. Set 1 on this bit starts a conversion of a group of regular channels if ETSRC is 111. It is set by software and cleared by software or by hardware immediately after the conversion starts.
21	SWICST	Start on inserted channel. Set 1 on this bit starts a conversion of a group of inserted channels if ETSIC is 111. It is set by software and cleared by software or by hardware immediately after

		the conversion starts.
20	ETERC	External trigger enable for regular channel 0: External trigger for regular channel disable 1: External trigger for regular channel enable
19:17	ETSRC[2:0]	External trigger select for regular channel For ADC0 and ADC1: 000: Timer 0 CH0 001: Timer 0 CH1 010: Timer 0 CH2 011: Timer 1 CH1 100: Timer 2 TRGO 101: Timer 3 CH3 110: EXTI line 11/ Timer 7 TRGO 111: SWRCST For ADC2: 000: Timer 2 CH0 001: Timer 1 CH2 010: Timer 0 CH2 011: Timer 7 CH0 100: Timer 7 TRGO 101: Timer 4 CH0 110: Timer 4 CH2 111: SWRCST
16	Reserved	Must be kept at reset value
15	ETEIC	External trigger enable for inserted channel 0: External trigger for inserted channel disable 1: External trigger for inserted channel enable
14:12	ETSIC[2:0]	External trigger select for inserted channel For ADC0 and ADC1: 000: Timer 0 TRGO 001: Timer 0 CH3 010: Timer 1 TRGO 011: Timer 1 CH0 100: Timer 2 CH3 101: Timer 3 TRGO 110: EXTI line15/ Timer 7 CH3 111: SWICST For ADC2: 000: Timer 0 TRGO

		001: Timer 0 CH3 010: Timer 3 CH2 011: Timer 7 CH1 100: Timer 7 CH3 101: Timer 4 TRGO 110: Timer 4 CH3 111: SWICST
11	DAL	Data alignment 0: LSB alignment 1: MSB alignment
10:9	Reserved	Must be kept at reset value
8	DMA	DMA request enable. 0: DMA request disable 1: DMA request enable
7:4	Reserved	Must be kept at reset value
3	RSTCLB	Reset calibration This bit is set by software and cleared by hardware after the calibration registers are initialized. 0: Calibration register initialize done. 1: Initialize calibration register start
2	CLB	ADC calibration 0: Calibration done 1: Calibration start
1	CTN	Continuous mode 0: Continuous mode disable 1: Continuous mode enable
0	ADCON	ADC ON. The ADC will be wake up when this bit is changed from low to high and take a stabilization time. When this bit is high and “1” is written to it with other bits of this register unchanged, the conversion will start. 0: ADC disable and power down 1: ADC enable

11.7.4. Sample time register 0 (ADC_SAMPT0)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SPT17[2:0]			SPT16[2:0]			SPT15[2:1]	
								rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT15[0]	SPT14[2:0]			SPT13[2:0]			SPT12[2:0]			SPT11[2:0]			SPT10[2:0]		
rw	rw			rw			rw			rw			rw		

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:21	SPT17[2:0]	refer to SPT10[2:0] description
20:18	SPT16[2:0]	refer to SPT10[2:0] description
17:15	SPT15[2:0]	refer to SPT10[2:0] description
14:12	SPT14[2:0]	refer to SPT10[2:0] description
11:9	SPT13[2:0]	refer to SPT10[2:0] description
8:6	SPT12[2:0]	refer to SPT10[2:0] description
5:3	SPT11[2:0]	refer to SPT10[2:0] description
2:0	SPT10[2:0]	Channel sample time 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

11.7.5. Sample time register 1 (ADC_SAMPT1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SPT9[2:0]			SPT8[2:0]			SPT7[2:0]			SPT6[2:0]			SPT5[2:1]	
		rw			rw			rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT5[0]	SPT4[2:0]			SPT3[2:0]			SPT2[2:0]			SPT1[2:0]			SPT0[2:0]		
rw	rw			rw			rw			rw			rw		

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:27	SPT9[2:0]	refer to SPT0[2:0] description
26:24	SPT8[2:0]	refer to SPT0[2:0] description
23:21	SPT7[2:0]	refer to SPT0[2:0] description
20:18	SPT6[2:0]	refer to SPT0[2:0] description
17:15	SPT5[2:0]	refer to SPT0[2:0] description
14:12	SPT4[2:0]	refer to SPT0[2:0] description
11:9	SPT3[2:0]	refer to SPT0[2:0] description
8:6	SPT2[2:0]	refer to SPT0[2:0] description
5:3	SPT1[2:0]	refer to SPT0[2:0] description
2:0	SPT0[2:0]	Channel sample time 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

11.7.6. Inserted channel data offset register x (ADC_IOFFx) (x=0..3)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value

11:0	IOFF[11:0]	Data offset for inserted channel x These bits will be subtracted from the raw converted data when converting inserted channels. The conversion result can be read from in the ADC_IDATAx registers.

11.7.7. Watchdog high threshold register (ADC_WDHT)

Address offset: 0x24

Reset value: 0x0000 0FFF

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WDHT[11:0]	Analog watchdog high threshold These bits define the high threshold for the analog watchdog.

11.7.8. Watchdog low threshold register (ADC_WDLT)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WDLT[11:0]	Analog watchdog low threshold These bits define the low threshold for the analog watchdog.

11.7.9. Regular sequence register 0 (ADC_RSQ0)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RL[3:0]				RSQ15[4:1]			
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ15[0]	RSQ14[4:0]					RSQ13[4:0]					RSQ12[4:0]				
rw	rw					rw					rw				

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:20	RL[3:0]	Regular channel group length. The total number of conversion in regular group equals to RL[3:0]+1.
19:15	RSQ15[4:0]	refer to RSQ0[4:0] description
14:10	RSQ14[4:0]	refer to RSQ0[4:0] description
9:5	RSQ13[4:0]	refer to RSQ0[4:0] description
4:0	RSQ12[4:0]	refer to RSQ0[4:0] description

11.7.10. Regular sequence register 1 (ADC_RSQ1)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		RSQ11[4:0]					RSQ10[4:0]					RSQ9[4:1]			
		rw					rw					rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ9[0]	RSQ8[4:0]					RSQ7[4:0]					RSQ6[4:0]				
rw	rw					rw					rw				

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:25	RSQ11[4:0]	refer to RSQ0[4:0] description

24:20	RSQ10[4:0]	refer to RSQ0[4:0] description
19:15	RSQ9[4:0]	refer to RSQ0[4:0] description
14:10	RSQ8[4:0]	refer to RSQ0[4:0] description
9:5	RSQ7[4:0]	refer to RSQ0[4:0] description
4:0	RSQ6[4:0]	refer to RSQ0[4:0] description

11.7.11. Regular sequence register 2 (ADC_RSQ2)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		RSQ5[4:0]					RSQ4[4:0]					RSQ3[4:1]			
		rw					rw					rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ3[0]	RSQ2[4:0]					RSQ1[4:0]					RSQ0[4:0]				
rw	rw					rw					rw				

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:25	RSQ5[4:0]	refer to RSQ0[4:0] description
24:20	RSQ4[4:0]	refer to RSQ0[4:0] description
19:15	RSQ3[4:0]	refer to RSQ0[4:0] description
14:10	RSQ2[4:0]	refer to RSQ0[4:0] description
9:5	RSQ1[4:0]	refer to RSQ0[4:0] description
4:0	RSQ0[4:0]	The channel number (0..17) is written to these bits to select a channel as the nth conversion in the regular channel group.

11.7.12. Inserted sequence register (ADC_ISQ)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										IL[1:0]		ISQ3[4:1]			

rw rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISQ3[0]	ISQ2[4:0]					ISQ1[4:0]					ISQ0[4:0]				
rw	rw					rw					rw				

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21:20	IL[1:0]	Inserted channel group length. The total number of conversion in Inserted group equals to IL[1:0] + 1.
19:15	ISQ3[4:0]	refer to ISQ0[4:0] description
14:10	ISQ2[4:0]	refer to ISQ0[4:0] description
9:5	ISQ1[4:0]	refer to ISQ0[4:0] description
4:0	ISQ0[4:0]	The channel number (0..17) is written to these bits to select a channel at the nth conversion in the inserted channel group. Unlike the regular conversion sequence, the inserted channels are converted starting from (4 - IL[1:0] - 1), if IL[1:0] length is less than 4. IL Insert channel order 3 ISQ0 >> ISQ1 >> ISQ2 >> ISQ3 2 ISQ1 >> ISQ2 >> ISQ3 1 ISQ2 >> ISQ3 0 ISQ3

11.7.13. Inserted data register x (ADC_IDATAx) (x= 0..3)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	IDATAn[15:0]	Inserted number n conversion data These bits contain the number n conversion result, which is read only.

11.7.14. Regular data register (ADC_RDATA)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">ADC1RDTR[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RDATA[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:16	ADC1RDTR[15:0]	ADC1 regular channel data In ADC0: In sync mode, these bits contain the regular data of ADC1. In ADC1 and ADC2: these bits are not used.
15:0	RDATA[15:0]	Regular channel data These bits contain the conversion result from regular channel, which is read only.

12. Digital-to-analog converter (DAC)

12.1. Overview

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The digital data can be configured in 8-bit or 12-bit mode, left-aligned or right-aligned mode. DMA can be used to update the digital data on external triggers. The output voltage can be optionally buffered for higher drive capability.

The two DACs can work independently or concurrently.

12.2. Characteristics

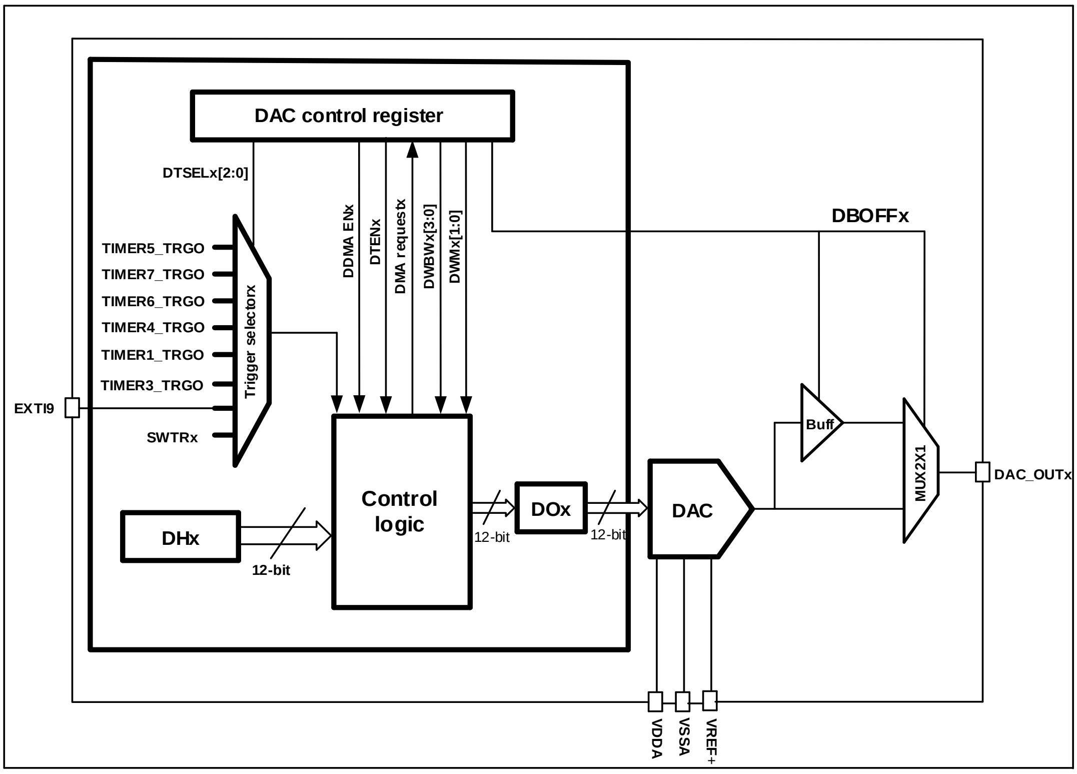
DAC’s main features are as follows:

- 12-bit resolution. Left or right data alignment.
- DMA capability for each channel.
- Conversion update synchronously.
- Conversion triggered by external triggers.
- Configurable internal buffer.
- Input voltage reference, V_{REF+}.
- Noise wave generation (LFSR noise mode and Triangle noise mode).
- Two DACs in concurrent mode.

Figure 12-1. DAC block diagram shows the block diagram of DAC and ***Table 12-1. DAC***

pins gives the pin description.

Figure 12-1. DAC block diagram



Note: In connectivity line devices, the TIMER7_TRGO trigger is replaced by TIMER2_TRGO.

Table 12-1. DAC pins

Name	Description	Signal type
V_{DDA}	Analog power supply	Input, analog supply
V_{SSA}	Ground for analog power supply	Input, analog supply ground
V_{REF+}	Positive reference voltage for the DAC, 2.4 V ≤ V_{REF+} ≤ V_{DDA}	Input, analog positive reference
DAC_OUTx	DACx analog output	Analog output signal

The GPIO pins (PA4 for DAC0, PA5 for DAC1) should be configured to analog mode before enable the DAC module.

12.3. Function overview

12.3.1. DAC enable

The DACs can be powered on by setting the DENx bit in the DAC_CTL register. A t_{WAKEUP} time is needed to startup the analog DAC submodule.

12.3.2. DAC output buffer

For the concern of reducing output impedance, and driving external loads without an external operational amplifier, an output buffer is integrated inside each DAC module.

The output buffer, which is turned on by default to reduce the output impedance and improve the driving capability, can be turned off by setting the DBOFFx bits in the DAC_CTL register.

12.3.3. DAC data configuration

The 12-bit DAC holding data (DACx_DH) can be configured by writing any one of the DACx_R12DH, DACx_L12DH and DACx_R8DH registers. When the data is loaded by DACx_R8DH register, only the MSB 8 bits are configurable, the LSB 4 bits are forced to 4’b0000.

12.3.4. DAC trigger

The DAC external trigger is enabled by setting the DTENx bits in the DAC_CTL register. The DAC external triggers are selected by the DTSELx bits in the DAC_CTL register.

Table 12-2. External triggers of DAC

DTSELx[2:0]	Trigger Source	Trigger Type
000	TIMER5_TRGO	Internal on-chip signal
001	TIMERa2_TRGO in connectivity line devices; TIMER7_TRGO in other type devices	
010	TIMER6_TRGO	
011	TIMER4_TRGO	
100	TIMER1_TRGO	
101	TIMER3_TRGO	
110	EXTI9	External signal
111	SWTRIG	Software trigger

The TIMERx_TRGO signals are generated from the timers, while the software trigger can be generated by setting the SWTRx bits in the DAC_SWT register.

12.3.5. DAC conversion

If the external trigger is enabled by setting the DTENx bit in DAC_CTL register, the DAC holding data is transferred to the DAC output data (DACx_DO) register at the selected trigger events. Otherwise, when the external trigger is disabled, the transfer is performed automatically.

When the DAC holding data (DACx_DH) is loaded into the DACx_DO register, after the time

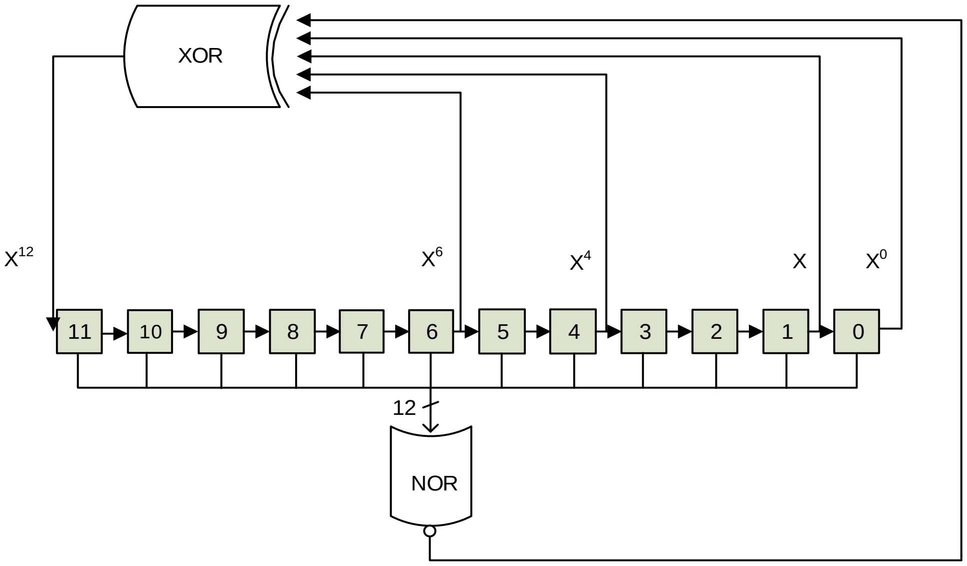
$t_{SETTLING}$, the analog output is valid, and the value of $t_{SETTLING}$ is related to the power supply voltage and the analog output load.

12.3.6. DAC noise wave

There are two methods of adding noise wave to the DAC output data: LFSR noise wave and Triangle wave. The noise wave mode can be selected by the DWMx bits in the DAC_CTL register. The amplitude of the noise can be configured by the DAC noise wave bit width (DWBWx) bits in the DAC_CTL register.

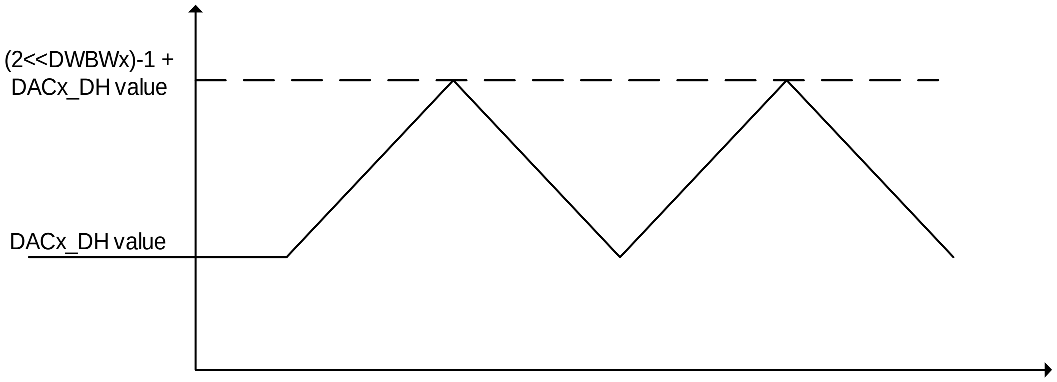
There is a Linear Feedback Shift Register (LFSR) in the DAC control logic. In the LFSR noise mode, the LFSR noise signal is added to the DACx_DH value. When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register.

Figure 12-2. DAC LFSR algorithm



In the triangle noise mode, a triangle signal is added to the DACx_DH value. The minimum value of the triangle signal is 0, while the maximum value of the triangle signal is (2<<DWBWx)-1.

Figure 12-3. DAC triangle noise wave



12.3.7. DAC output voltage

The analog output voltage on the DAC pin is determined by the following equation:

$$\mathrm{DAC}_{\mathrm{output}} = \mathrm{V}_{\mathrm{REF+}} * \mathrm{DAC\_DO}/4096 \qquad (12\text{-}1)$$

The digital input is linearly converted to an analog output voltage, its range is 0 to V_{REF+}.

12.3.8. DMA request

When the external trigger is enabled, the DMA request is enabled by setting the DDMAENx bits of the DAC_CTL register. A DAC DMA request will be generated when an external hardware trigger (not a software trigger) occurs.

12.3.9. DAC concurrent conversion

When the two DACs work at the same time, for maximum bus bandwidth utilization in specific applications, two DACs can be configured in concurrent mode. In concurrent mode, two DACs data transfer (DACx_DH to DACx_DO) will be at the same time.

There are three concurrent registers that can be used to load the DACx_DH value: DACC_R8DH, DACC_R12DH and DACC_L12DH. You just need to access a unique register to realize driving both DACs at the same time.

When external trigger is enabled, both DTENx bits should be set. DTSEL0 and DTSEL1 bits should be configured with the same value.

When DMA is enabled, only one of the DDMAENx bits should be set.

The noise mode and noise bit width can be configured either the same or different, depending on the usage.

12.4. Register definition

12.4.1. Control register (DAC_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			DDMAEN1	DWBW1[3:0]				DWM1[1:0]		DTSEL1[2:0]			DTEN1	DBOFF1	DEN1
			rw	rw				rw		rw			rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			DDMAEN0	DWBW0[3:0]				DWM0[1:0]		DTSEL0[2:0]			DTEN0	DBOFF0	DEN0
			rw	rw				rw		rw			rw	rw	rw

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28	DDMAEN1	DAC1 DMA enable 0: DAC1 DMA mode disabled 1: DAC1 DMA mode enabled
27:24	DWBW1[3:0]	DAC1 noise wave bit width These bits specify bit width of the noise wave signal of DAC1. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is ((2<<(n-1))-1) in triangle noise mode, where n is the bit width of wave. 0000: The bit width of the wave signal is 1 0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9 1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11 ⩾1011: The bit width of the wave signal is 12
23:22	DWM1[1:0]	DAC1 noise wave mode These bits specify the mode selection of the noise wave signal of DAC1 when external trigger of DAC1 is enabled (DTEN1=1). 00: wave disabled 01: LFSR noise mode

		1x: Triangle noise mode
21:19	DTSEL1[2:0]	DAC1 trigger selection These bits select the external trigger of DAC1 when DTEN1=1. 000: Timer 5 TRGO 001: Timer 2 TRGO (connectivity line devices); Timer 7 TRGO (other type devices) 010: Timer 6 TRGO 011: Timer 4 TRGO 100: Timer 1 TRGO 101: Timer 3 TRGO 110: EXTI line 9 111: Software trigger
18	DTEN1	DAC1 trigger enable 0: DAC1 trigger disabled 1: DAC1 trigger enabled
17	DBOFF1	DAC1 output buffer turn off 0: DAC1 output buffer turn on to reduce the output impedance and improve the driving capability 1: DAC1 output buffer turn off
16	DEN1	DAC1 enable 0: DAC1 disabled 1: DAC1 enabled
15:13	Reserved	Must be kept at reset value
12	DDMAEN0	DAC0 DMA enable 0: DAC0 DMA mode disabled 1: DAC0 DMA mode enabled
11:8	DWBW0[3:0]	DAC0 noise wave bit width These bits specify bit width of the noise wave signal of DAC0. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is ((2<<(n-1))-1) in triangle noise mode, where n is the bit width of wave. 0000: The bit width of the wave signal is 1 0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9

		1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11 ⩾1011: The bit width of the wave signal is 12
7:6	DWM0[1:0]	DAC0 noise wave mode These bits specify the mode selection of the noise wave signal of DAC0 when external trigger of DAC0 is enabled (DTEN0=1). 00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode
5:3	DTSEL0[2:0]	DAC0 trigger selection These bits select the external trigger of DAC0 when DTEN0=1. 000: Timer 5 TRGO 001: Timer 2 TRGO (connectivity line devices); Timer 7 TRGO (other type devices) 010: Timer 6 TRGO 011: Timer 4 TRGO 100: Timer 1 TRGO 101: Timer 3 TRGO 110: EXTI line 9 111: Software trigger
2	DTEN0	DAC0 trigger enable 0: DAC0 trigger disabled 1: DAC0 trigger enabled
1	DBOFF0	DAC0 output buffer turn off 0: DAC0 output buffer turn on to reduce the output impedance and improve the driving capability 1: DAC0 output buffer turn off
0	DEN0	DAC0 enable 0: DAC0 disabled 1: DAC0 enabled

12.4.2. Software trigger register (DAC_SWT)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTR1	SWTR0
														w	w

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	SWTR1	DAC1 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled
0	SWTR0	DAC0 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled

12.4.3. DAC0 12-bit right-aligned data holding register (DAC0_R12DH)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC0_DH[11:0]											
				rw											

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

12.4.4. DAC0 12-bit left-aligned data holding register (DAC0_L12DH)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

DAC0_DH[11:0]	Reserved
rw	

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value

12.4.5. DAC0 8-bit right-aligned data holding register (DAC0_R8DH)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DAC0_DH[7:0]							
								rw							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	DAC0_DH[7:0]	DAC0 8-bit right-aligned data These bits specify the MSB 8 bits of the data that is to be converted by DAC0.

12.4.6. DAC1 12-bit right-aligned data holding register (DAC1_R12DH)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC1_DH[11:0]											
				rw											

Bits	Fields	Descriptions

31:12	Reserved	Must be kept at reset value
11:0	DAC1_DH[11:0]	DAC1 12-bit right-aligned data These bits specify the data that is to be converted by DAC1.

12.4.7. DAC1 12-bit left-aligned data holding register (DAC1_L12DH)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">DAC1_DH[11:0]</td><td colspan="4">Reserved</td></tr>
<tr><td colspan="12">rw</td><td colspan="4"></td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:4	DAC1_DH[11:0]	DAC1 12-bit left-aligned data These bits specify the data that is to be converted by DAC1.
3:0	Reserved	Must be kept at reset value

12.4.8. DAC1 8-bit right-aligned data holding register (DAC1_R8DH)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">DAC1_DH[7:0]</td></tr>
<tr><td colspan="8"></td><td colspan="8">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	DAC1_DH[7:0]	DAC1 8-bit right-aligned data These bits specify the MSB bits of the data that is to be converted by DAC1.

12.4.9. DAC concurrent mode 12-bit right-aligned data holding register (DACC_R12DH)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DAC1_DH[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC0_DH[11:0]											
				rw											

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:16	DAC1_DH[11:0]	DAC1 12-bit right-aligned data These bits specify the data that is to be converted by DAC1.
15:12	Reserved	Must be kept at reset value
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

12.4.10. DAC concurrent mode 12-bit left-aligned data holding register (DACC_L12DH)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAC1_DH[11:0]												Reserved			
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC0_DH[11:0]												Reserved			
rw															

Bits	Fields	Descriptions
31:20	DAC1_DH[11:0]	DAC1 12-bit left-aligned data

		These bits specify the data that is to be converted by DAC1.
19:16	Reserved	Must be kept at reset value
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value

12.4.11. DAC concurrent mode 8-bit right-aligned data holding register (DACC_R8DH)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC1_DH [7:0]								DAC0_DH [7:0]							
rw								rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:8	DAC1_DH[7:0]	DAC1 8-bit right-aligned data These bits specify the MSB 8-bit of the data that is to be converted by DAC1.
7:0	DAC0_DH[7:0]	DAC0 8-bit right-aligned data These bits specify the MSB 8-bit of the data that is to be converted by DAC0.

12.4.12. DAC0 data output register (DAC0_DO)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC0_DO [11:0]											

r

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC0_DO [11:0]	DAC0 data output These bits, which are read only, reflect the data that is being converted by DAC0.

12.4.13. DAC1 data output register (DAC1_DO)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC1_DO [11:0]											
				r											

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC1_DO [11:0]	DAC1 data output These bits, which are read only, reflect the data that is being converted by DAC1.

13. Watchdog timer (WDGT)

The watchdog timer (WDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. There are two watchdog timer peripherals in the chip: free watchdog timer (FWDGT) and window watchdog timer (WWDGT). They offer a combination of a high safety level, flexibility of use and timing accuracy. Both watchdog timers are offered to resolve malfunctions of software.

The watchdog timer will generate a reset (or an interrupt in window watchdog timer) when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

13.1. Free watchdog timer (FWDGT)

13.1.1. Overview

The free watchdog timer (FWDGT) has free clock source (IRC40K). Thereupon the FWDGT 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.

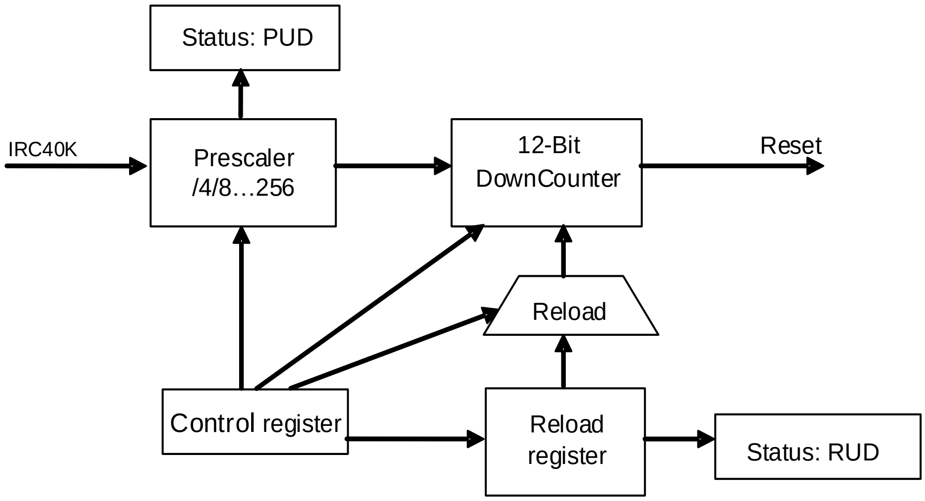
13.1.2. Characteristics

- Free-running 12-bit downcounter.
- Reset when the downcounter reaches 0, if the watchdog is enabled.
- Free clock source, FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog bit, automatically start the FWDGT at power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

13.1.3. Function overview

The free watchdog consists of an 8-stage prescaler and a 12-bit down-counter. Refer to the figure below for the functional block of the free watchdog module.

Figure 13-1. Free watchdog block diagram



The free watchdog is enabled by writing the value 0xCCCC in the control register (FWDGT_CTL), 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 FWDGT_CTL register at anytime. The reload value comes from the FWDGT_RLD 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 FWDGT_PSC register and the FWDGT_RLD register are write-protected. Before writing these registers, the software should write the value 0x5555 to the FWDGT_CTL register. These registers will be protected again by writing any other value to the FWDGT_CTL register. When an update operation of the prescaler register (FWDGT_PSC) or the reload value register (FWDGT_RLD) is on going, the status bits in the FWDGT_STAT register are set.

If the FWDGT_HOLD bit in DBG module is cleared, the FWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

Table 13-1. Min/max FWDGT timeout period at 40 kHz (IRC40K)

Prescaler divider	PSC[2:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFF
1/4	000	0.1	409.6
1/8	001	0.2	819.2
1/16	010	0.4	1638.4
1/32	011	0.8	3276.8
1/64	100	1.6	6553.6
1/128	101	3.2	13107.2

Prescaler divider	PSC[2:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFF
1/256	110 or 111	6.4	26214.4

The FWDGT timeout can be more accurate by calibrating the IRC40K.

Note：

- All the 10X devices. When after the execution of dog reload operation, if the MCU needs enter the deepsleep/standby mode immediately, (more than 3) IRC40K clock interval must be inserted in the middle of reload and stop/standby mode commands by software setting.
- For all the 101 devices and the 103 devices with flash no more than 128K, when software finished the executing operation of FWDGT, if the MCU needs enter the deepsleep/standby mode immediately, it is at least 100 us interval left between the two instructions.
- For all the 101 devices and the 103 devices with flash no more than 128K, if you need access to the MCU debug mode, recommend to use hardware watchdog, or enable watchdog again after exit debug mode by software setting.

13.1.4. Register definition

Control register (FWDGT_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CMD[15:0]</td></tr>
<tr><td colspan="16">w</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMD[15:0]	Write only. Several different fuctions are realized by writing these bits with different values: 0x5555: Disable the FWDGT_PSC and FWDGT_RLD write protection 0xCCCC: Start the free watchdog counter. When the counter reduces to 0, the free watchdog generates a reset 0xAAAA: Reload the counter

Prescaler register (FWDGT_PSC)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">Reserved</td><td colspan="3">PSC[2:0]</td></tr>
<tr><td colspan="13"></td><td colspan="3">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	PSC[2:0]	Free watchdog timer prescaler selection. Write 0x5555 in the FWDGT_CTL register before writing these bits. During a write operation to this register, the PUD bit in the

FWDGT_STAT register is set and the value read from this register is invalid.

000: 1/4

001: 1/8

010: 1/16

011: 1/32

100: 1/64

101: 1/128

110: 1/256

111: 1/256

If several prescaler values are used by the application, it is mandatory to wait until PUD bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PUD is reset before continuing code execution except in case of low-power mode entry.

Reload register (FWDGT_RLD)

Address offset: 0x08

Reset value: 0x0000 0FFF

This register can be accessed by half-word (16-bit) or word (32-bit) access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	RLD[11:0]	Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT counter with the RLD value. These bits are write-protected. Write 0x5555 in the FWDGT_CTL register before writing these bits. During a write operation to this register, the RUD bit in the FWDGT_STAT 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 RUD bit is reset before changing the reload value. However, after updating the reload value it is not necessary to wait until RUD is reset before continuing code execution except in case of low-power mode entry.

Status register (FWDGT_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit) access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RUD	PUD
														r	r

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	RUD	Free watchdog timer counter reload value update During a write operation to FWDGT_RLD register, this bit is set and the value read from FWDGT_RLD register is invalid. This bit is reset by hardware after the update operation of FWDGT_RLD register.
0	PUD	Free watchdog timer prescaler value update During a write operation to FWDGT_PSC register, this bit is set and the value read from FWDGT_PSC register is invalid. This bit is reset by hardware after the update operation of FWDGT_PSC register.

13.2. Window watchdog timer (WWDGT)

13.2.1. Overview

The window watchdog timer (WWDGT) 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 APB1 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

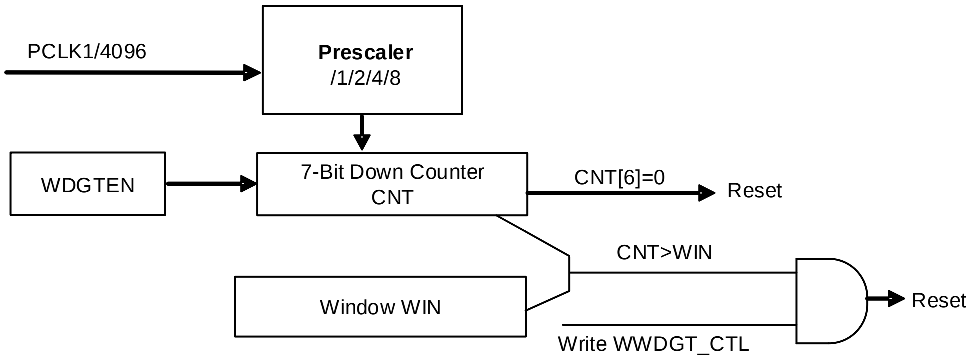
13.2.2. Characteristics

- Programmable free-running 7-bit downcounter.
- Generate reset in two conditions when WWDGT 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.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

13.2.3. Function overview

If the window watchdog timer is enabled (set the WDGTEN bit in the WWDGT_CTL), 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 13-2. Window watchdog timer block diagram



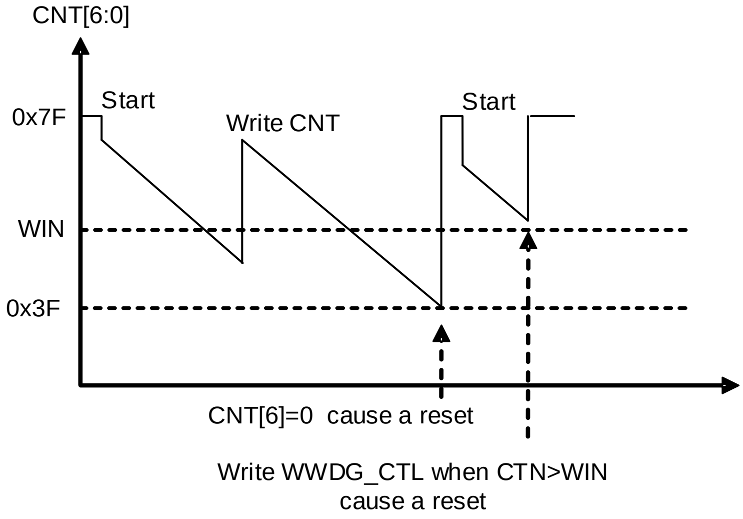
The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDGTEN bit in the WWDGT_CTL register. Whenever window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, it implies that 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 APB1 clock and the prescaler (PSC[1:0] bits in the WWDGT_CFG register).

The WIN[6:0] bits in the configuration register (WWDGT_CFG) 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 WWDGT_CFG 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 WWDGT will never generate a WWDGT reset but can be used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDGT_STAT register.

Figure 13-3. Window watchdog timing diagram



Calculate the WWDGT timeout by using the formula below.

$$\mathrm{t_{WWDGT}} = \mathrm{t_{PCLK1}} \times 4096 \times 2^{PSC} \times (\,CNT[5:0] + 1) \quad (ms) \qquad (13\text{-}1)$$

where:

t_{WWDGT}: WWDGT timeout

t_{PCLK1}: APB1 clock period measured in ms

Refer to the table below for the minimum and maximum values of the t_{WWDGT}.

Table 13-2. Min/max timeout value at 54 MHz (f_{PCLK1})

Prescaler divider	PSC[1:0]	Min timeout value CNT[6:0] =0x40	Max timeout value CNT[6:0]=0x7F
1/1	00	75.8 μs	4.8 ms
1/2	01	151.7 μs	9.7 ms
1/4	10	303.4 μs	19.4 ms
1/8	11	606.8 μs	38.8 ms

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

13.2.4. Register definition

Control register (WWDGT_CTL)

Address offset: 0x00

Reset value: 0x0000 007F

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGTEN	CNT[6:0]						
								rs	rw						

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	WDGTEN	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[6:0]	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.

Configuration register (WWDGT_CFG)

Address offset: 0x04

Reset value: 0x0000 007F

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWIE	PSC[1:0]		WIN[6:0]						
						rs	rw		rw						

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	EWIE	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 RCU WWDGT software reset. A write operation of ‘0’ has no effect.
8:7	PSC[1:0]	Prescaler. The time base of the watchdog timer counter 00: (PCLK1 / 4096) / 1 01: (PCLK1 / 4096) / 2 10: (PCLK1 / 4096) / 4 11: (PCLK1 / 4096) / 8
6:0	WIN[6:0]	The Window value. A reset occurs if the watchdog counter (CNT bits in WWDGT_CTL) is written when the value of the watchdog counter is greater than the Window value.

Status register (WWDGT_STAT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rw

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EWIF	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 (EWIE in WWDGT_CFG is cleared). This bit is cleared by writing 0 to it. There is no effect when writing 1 to it.

14. Real-time Clock (RTC)

14.1. 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.

14.2. 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) HXTAL clock divided by 128
  B) LXTAL oscillator clock
  C) IRC40K oscillator clock
- Maskable interrupt source:
  A) Alarm interrupt
  B) Second interrupt
  C) Overflow interrupt

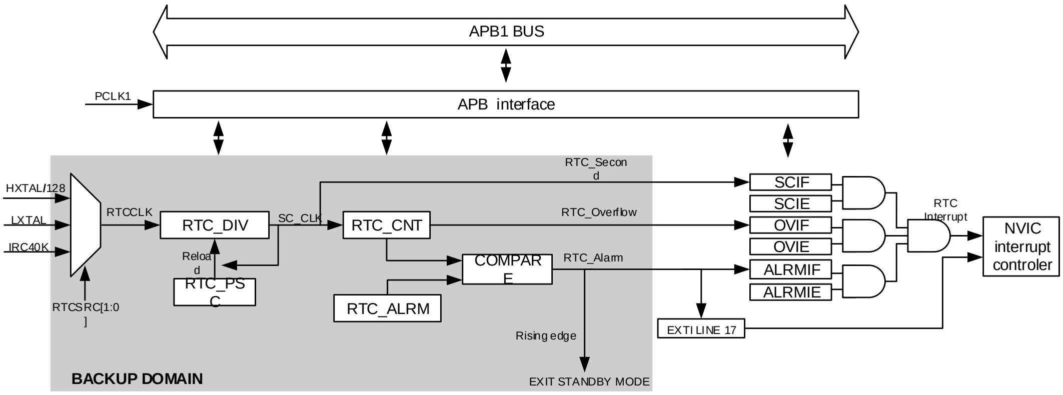
14.3. Function overview

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_INTEN 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_INTEN register, the RTC will generate an alarm interrupt when the system time equals to the alarm time (stored in the RTC_ALRMH/L register),

Figure 14-1. Block diagram of RTC



14.3.1. RTC reset

The APB interface and the RTC_INTEN 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:

1. Set the PMUEN and BKPIEN bits in the RCU_APB1EN register to enable the power and backup interface clocks.

2. Enable access to the backup registers and RTC by setting the BKPWEN bit in the (PMU_CTL).

14.3.2. 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 RSYNF 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.

14.3.3. 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 CMF bit in the RTC_CTL 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 LWOFF bit in the RTC_CTL 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 LWOFF bit in the RTC_CTL register sets to '1';
B) Enter Configuration mode by setting the CMF bit in the RTC_CTL register;
C) Write to the RTC registers;
D) Exit Configuration mode by clearing the CMF bit in the RTC_CTL register;
E) Wait until the value of LWOFF bit in the RTC_CTL register sets to '1'.

14.3.4. RTC flag assertion

Before the update of the RTC Counter, the RTC second interrupt flag (SCIF) 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 (ALRMIF) is asserted on the last RTCCLK cycle.

Before the counter equals to 0x0, the RTC Overflow interrupt flag (OVIF) 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 SCIF bit to be set in the RTC Control register.

Figure 14-2. RTC second and alarm waveform example (RTC_PSC = 3, RTC_ALRM = 2)

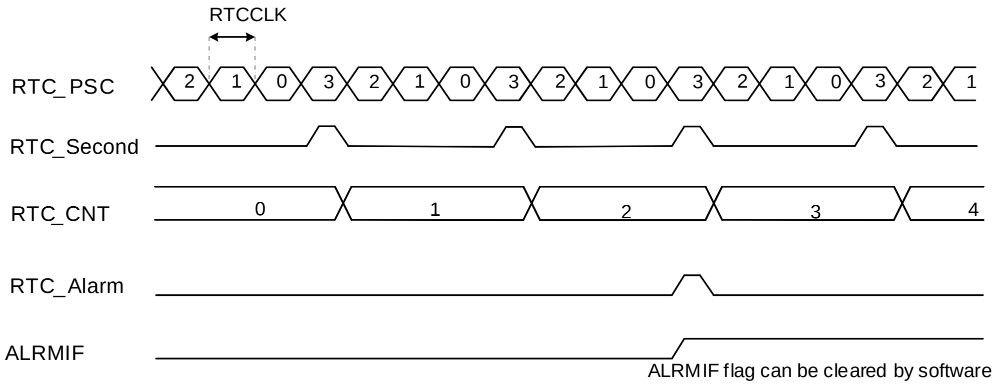
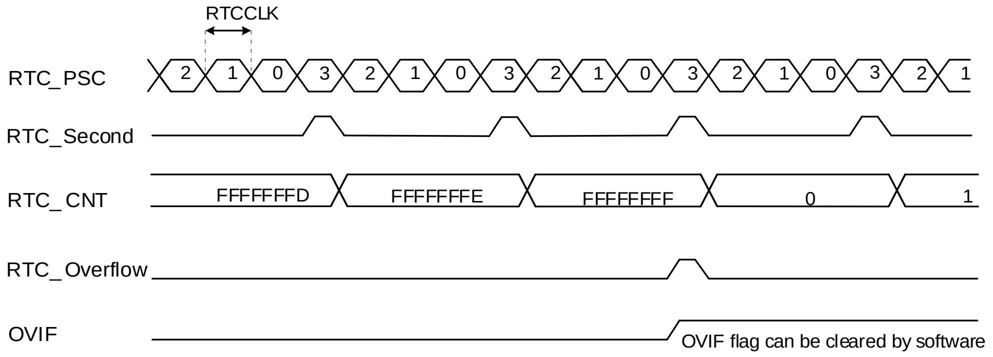


Figure 14-3. RTC second and overflow waveform example (RTC_PSC= 3)



14.4. Register definition

14.4.1. RTC interrupt enable register(RTC_INTEN)

Address offset : 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													OVIE	ALRMIE	SCIE
													rw	rw	rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	OVIE	Overflow interrupt enable 0: Disable overflow interrupt 1: Enable overflow interrupt
1	ALRMIE	Alarm interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
0	SCIE	Second interrupt enable 0: Disable second interrupt . 1: Enable second interrupt

14.4.2. RTC control register(RTC_CTL)

Address offset : 0x04

Reset value :0x0020

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										LWOFF	CMF	RSYNF	OVIF	ALRMIF	SCIF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions

31:6	Reserved	Must be kept at reset value
5	LWOFF	Last write operation finished flag 0: Last write operation on RTC registers did not finished. 1: Last write operation on RTC registers finished.
4	CMF	Configuration mode flag 0: Exit configuration mode. 1: Enter configuration mode.
3	RSYNF	Registers synchronized flag 0: Registers not yet synchronized with the APB1 clock. 1: Registers synchronized with the APB1 clock.
2	OVIF	Overflow interrupt flag 0: Overflow event not detected 1: Overflow event detected. An interrupt will occur if the OVIE bit is set in RTC_INTEN.
1	ALRMIF	Alarm interrupt flag 0: Alarm event not detected 1: Alarm event detected. An interrupt named RTC global interrupt will occur if the ALRMIE bit is set in RTC_INTEN. And another interrupt named the RTC Alarm interrupt will occur if the EXTI 17 is enabled in interrupt mode.
0	SCIF	Second interrupt flag 0: Second event not detected. 1: Second event detected. An interrupt will occur if the SCIE bit is set in RTC_INTEN. Set by hardware when the divider reloads the value in RTC_PSCH/L, thus incrementing the RTC counter.

14.4.3. RTC prescaler high register (RTC_PSCH)

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PSC[19:16]			
												w			

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	PSC[19:16]	RTC prescaler value high

14.4.4. RTC prescaler low register(RTC_PSCL)

Address offset: 0x0C

Reset value: 0x8000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">PSC[15:0]</td></tr>
<tr><td colspan="16">w</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	RTC prescaler value low The frequency of SC_CLK is the RTCCLK frequency divided by (PSC[19:0]+1).

14.4.5. RTC divider high register (RTC_DIVH)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">DIV[19:16]</td></tr>
<tr><td colspan="12"></td><td colspan="4">r</td></tr>
</table>

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	DIV[19:16]	RTC divider value high

14.4.6. RTC divider low register (RTC_DIVL)

Address offset: 0x14

Reset value: 0x8000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DIV[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	DIV[15:0]	RTC divider value low The RTC divider register is reloaded by hardware when the RTC prescaler or RTC counter register updated.

14.4.7. RTC counter high register (RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CNT[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[31:16]	RTC counter value high

14.4.8. RTC counter low register (RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CNT[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	RTC counter value low

14.4.9. RTC alarm high register (RTC_ALRMH)

Address offset: 0x20

Reset value: 0xFFFF

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALRM[31:16]															
w															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	ALRM[31:16]	RTC alarm value high

14.4.10. RTC alarm low register (RTC_ALRML)

Address offset: 0x24

Reset value: 0xFFFF

This register can be accessed by half-word (16-bit) or word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALRM[15:0]															
w															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	ALRM[15:0]	RTC alarm value low

15. TIMER

Table 15-1. Timers (TIMERx) are divided into five sorts

TIMER	TIMER0/7	TIMER1/2/3/4	TIMER8/11	TIMER9/10/12/13	TIMER5/6
TYPE	Advanced	General-L0	General-L1	General-L2	Basic
Prescaler	16-bit	16-bit	16-bit	16-bit	16-bit
Counter	16-bit	16-bit	16-bit	16-bit	16-bit
Count mode	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP ONLY
Repetition	●	×	×	×	×
CH Capture/ Compare	4	4	2	1	0
Complementary & Dead-time	●	×	×	×	×
Break	●	×	×	×	×
Single Pulse	●	●	●	×	●
Quadrature Decoder	●	●	×	×	×
Slave Controller	●	●	●	×	×
Inter connection	●(1)	●(2)	●(3)	×	TRGO TO DAC
DMA	●	●	×	×	●(4)
Debug Mode	●	●	●	●	●

(1)	TIMER0	**ITI0:** TIMER4_TRGO	**ITI1:** TIMER1_TRGO	**ITI2:** TIMER2_TRGO	**ITI3:** TIMER3_TRGO
	TIMER7	**ITI0:** TIMER0_TRGO	**ITI1:** TIMER1_TRGO	**ITI2:** TIMER3_TRGO	**ITI3:** TIMER4_TRGO
(2)	TIMER1	**ITI0:** TIMER0_TRGO	**ITI1:** refer to note (5)	**ITI2:** TIMER2_TRGO	**ITI3:** TIMER3_TRGO
	TIMER2	**ITI0:** TIMER0_TRGO	**ITI1:** TIMER1_TRGO	**ITI2:** TIMER4_TRGO	**ITI3:** TIMER3_TRGO
	TIMER3	**ITI0:** TIMER0_TRGO	**ITI1:** TIMER1_TRGO	**ITI2:** TIMER2_TRGO	**ITI3:** TIMER7_TRGO
	TIMER4	**ITI0:** TIMER1_TRGO	**ITI1:** TIMER2_TRGO	**ITI2:** TIMER3_TRGO	**ITI3:** TIMER7_TRGO
(3)	TIMER8	**ITI0:** TIMER1_TRGO	**ITI1:** TIMER2_TRGO	**ITI2:** TIMER9_TRGO	**ITI3:** TIMER10_ TRGO
	TIMER11	**ITI0:** TIMER3_TRGO	**ITI1:** TIMER4_TRGO	**ITI2:** TIMER12_TRGO	**ITI3:** TIMER13_ TRGO

(4) Only update events will generate DMA request. Note that TIMER5/6 do not have DMA configuration registers.

(5) In connectivity line devices, the source of TIMER1 ITI1 is decided by TIMER1ITI1_REMAP in ***AFIO port configuration register 0 (AFIO_PCF0)***;

In non-connectivity line devices, the source of TIMER1 ITI1 is internally connected to TIMER7_TRGO;

15.1. Advanced timer (TIMERx, x=0, 7)

15.1.1. Overview

The advanced timer module (Timer0 & Timer7) 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 16-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 issuitable for motor control applications.

Timer and timer are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

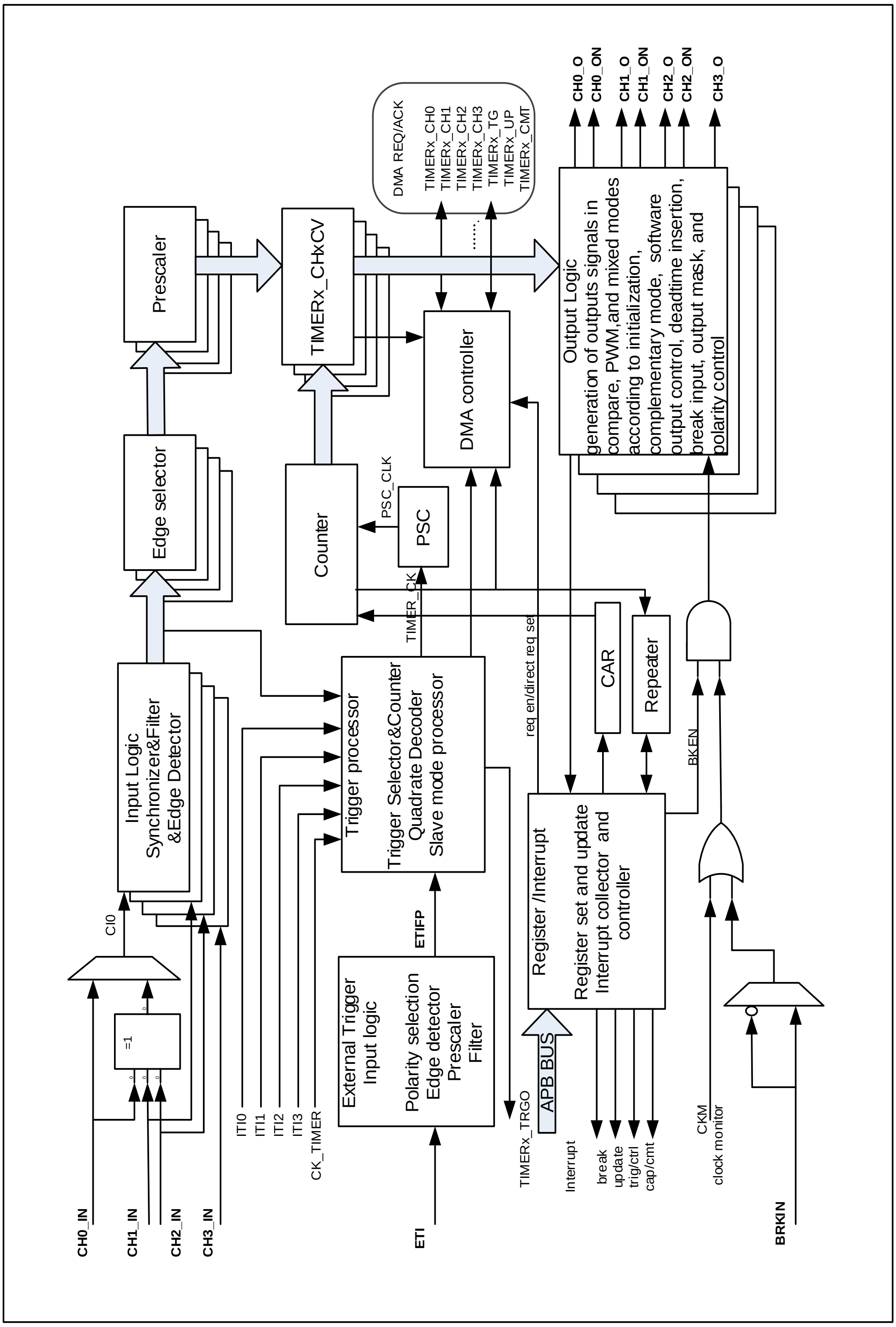
15.1.2. Characteristics

- Total channel num: 4.
- Counter width: 16 bit.
- Source of counter clock is selectable:
  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
- 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, and 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.

15.1.3. Block diagram

Figure 15-1. Advanced timer block diagram provides details of the internal configuration of the advanced timer.

Figure 15-1. Advanced timer block diagram



15.1.4. Function overview

Clock selection

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

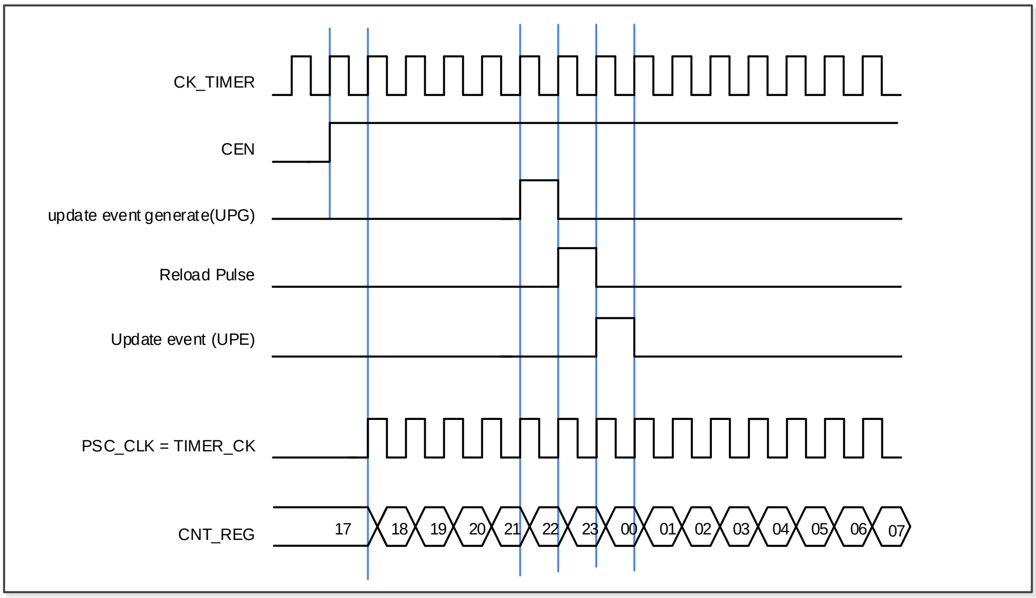
- SMC [2:0] == 3’b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when the slave mode is disabled (SMC [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 RCU.

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

Figure 15-2. Normal mode, internal clock divided by 1



- SMC [2:0] == 3’b111 (external clock mode 0). 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 TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

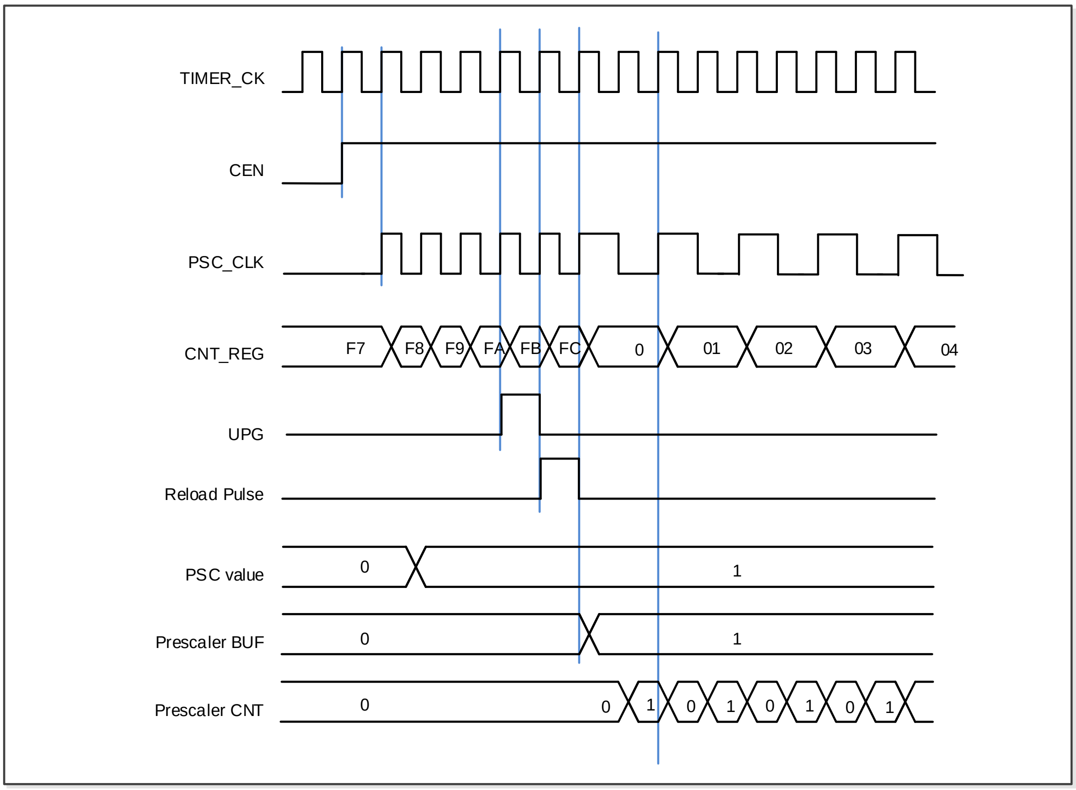
- SMC1== 1'b1 (external clock mode 1). External input is selected as timer clock source (ETI)

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 ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

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 (TIMERx_PSC) which can be changed on the go but is taken into account at the next update event.

Figure 15-3. Counter timing diagram with prescaler division change from 1 to 2 (PSC value change from 0 to 1)



Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR 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 (TIMERx_CREP+1) times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

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

If set the UPDIS bit in TIMERx_CTL0 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 TIMERx_CAR=0x63.

Figure 15-4. Up-counter timechart, PSC=0/1

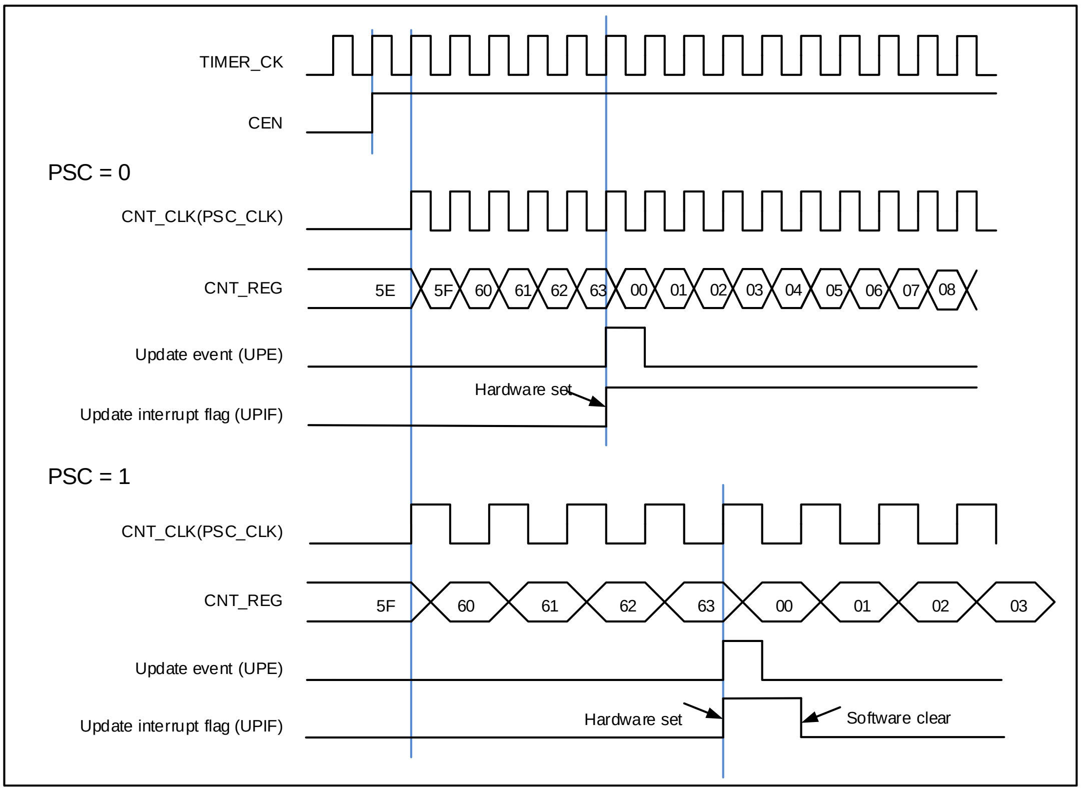
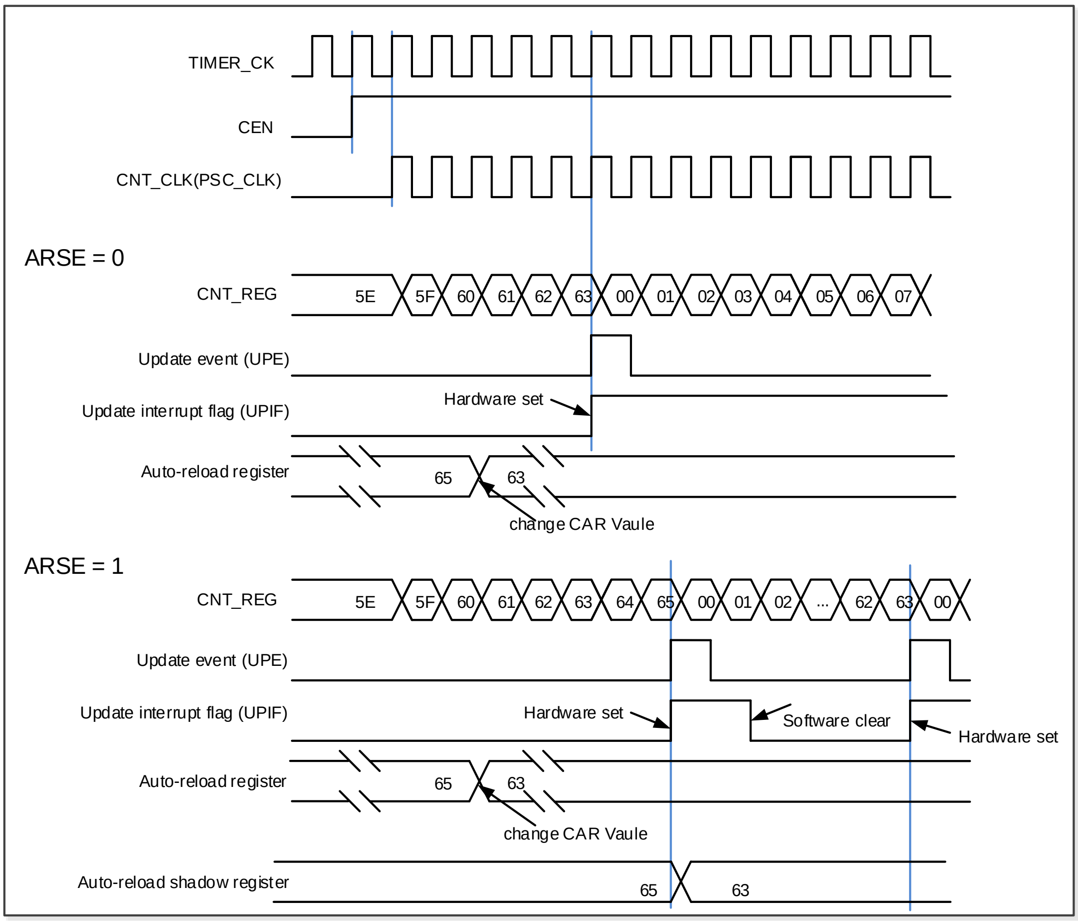


Figure 15-5. Up-counter timechart, change TIMERx_CAR on the go



Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR 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 (TIMERx_CREP+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down-counting mode.

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

If set the UPDIS bit in TIMERx_CTL0 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 TIMERx_CAR=0x63.

Figure 15-6. Down-counter timechart, PSC=0/1

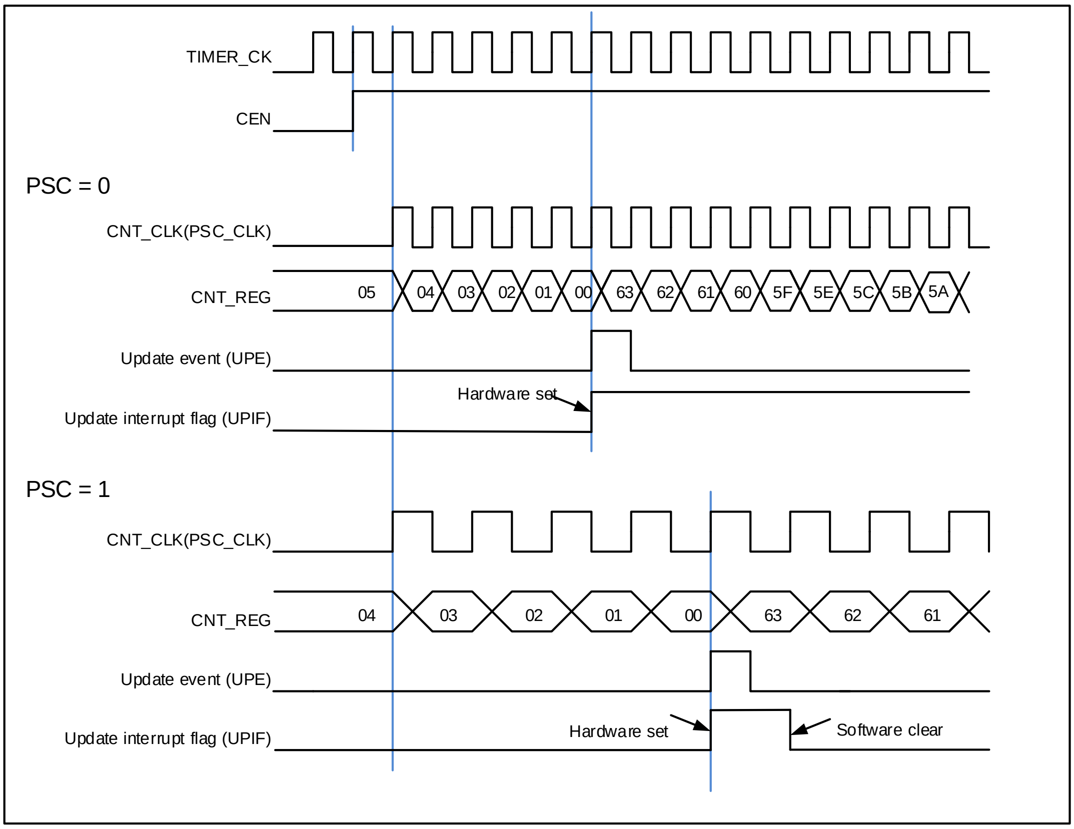
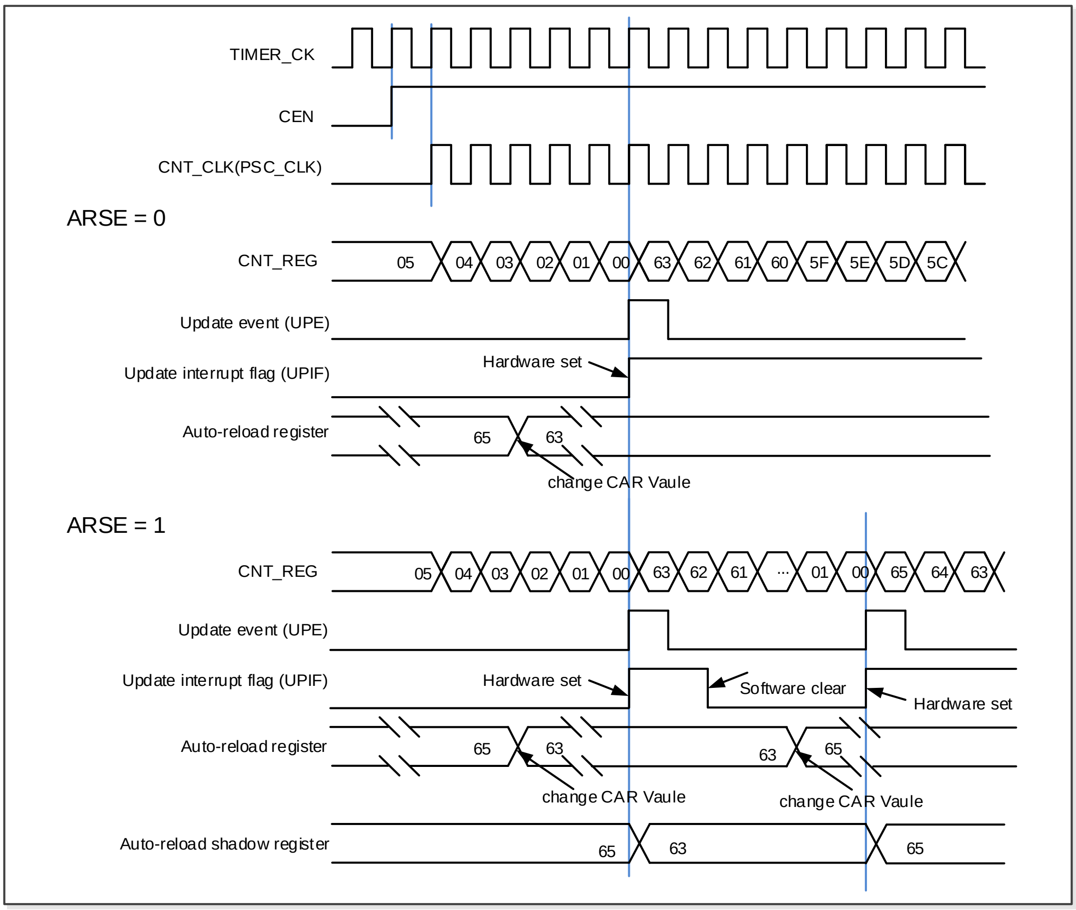


Figure 15-7. Down-counter timechart, change TIMERx_CAR on the go



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 TIMERx_CTL0 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 UPG bit in the TIMERx_SWEVG 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 UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 15-8. Center-aligned counter timechart***.

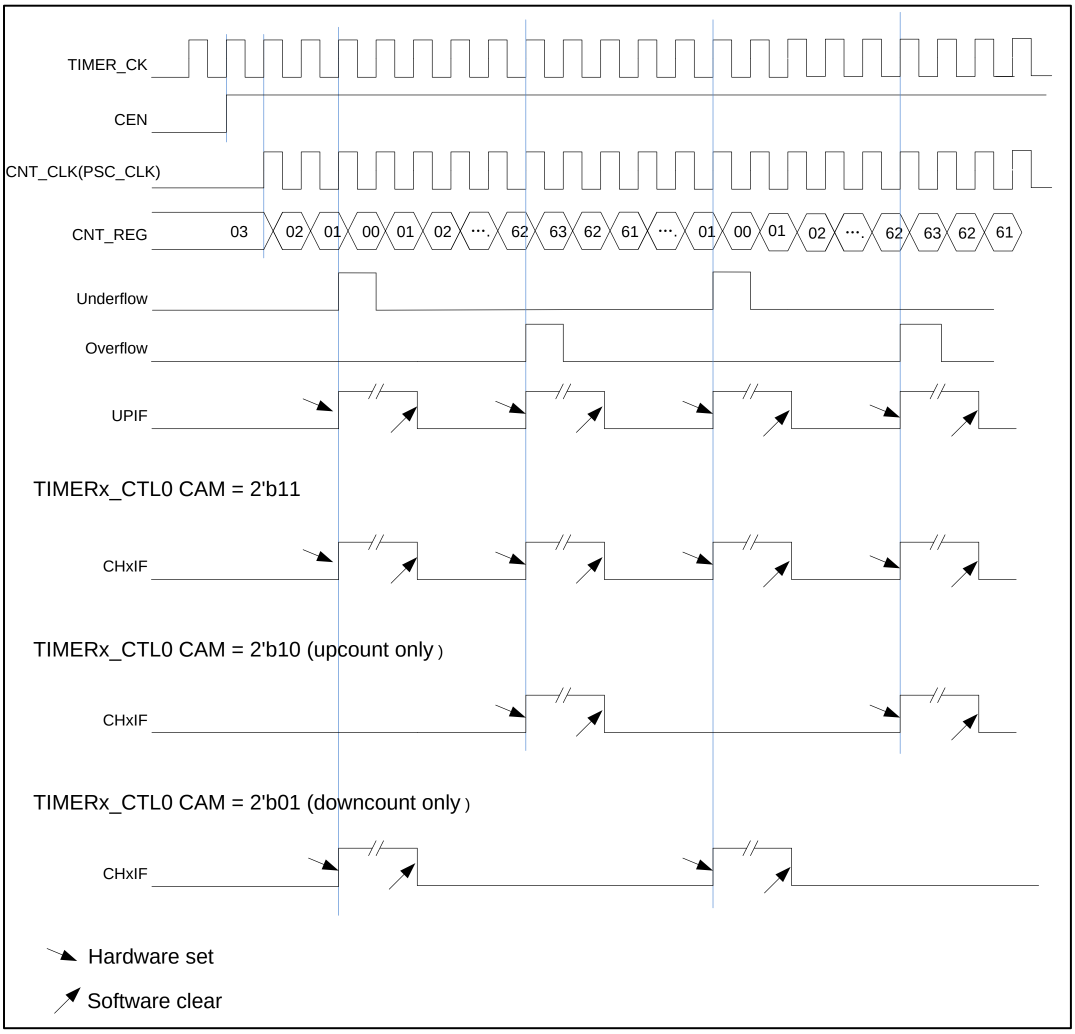
If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register,

prescaler register) are updated.

Figure 15-8. Center-aligned counter timechart show some examples of the counter behavior when TIMERx_CAR=0x63. TIMERx_PSC=0x0

Figure 15-8. Center-aligned counter timechart



Counter repetition

Counter Repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in TIMERx_CREP 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 UPG bit in the TIMERx_SWEVG register will reload the content of CREP in TIMERx_CREP register and generator an update event.

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

Figure 15-9. Repetition timecart for center-aligned counter

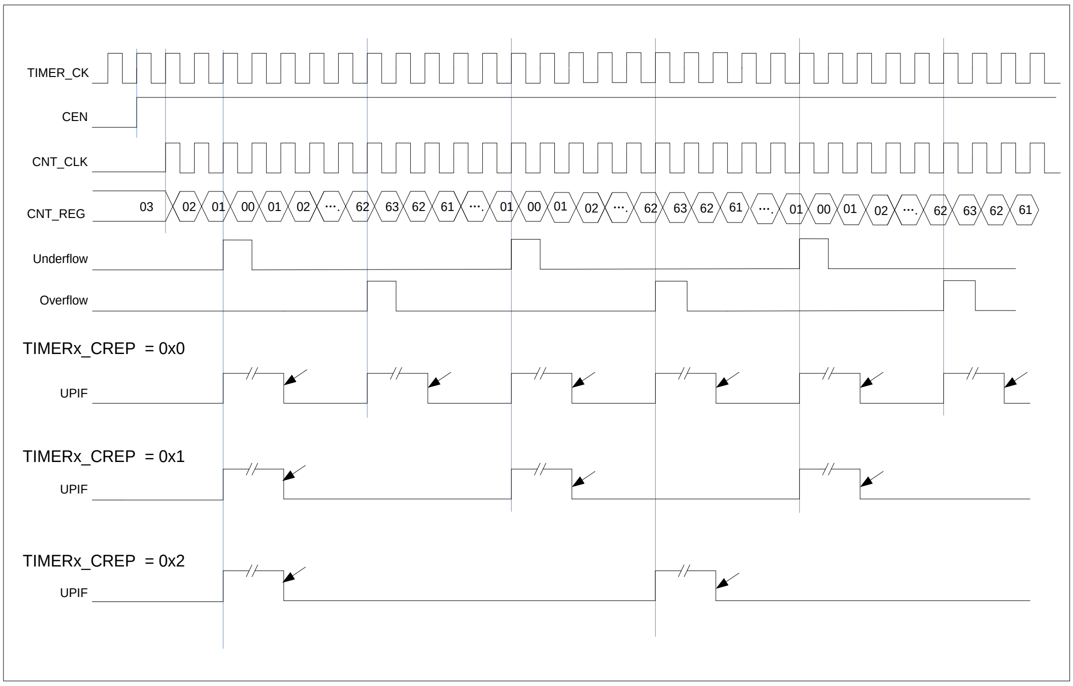


Figure 15-10. Repetition timechart for up-counter

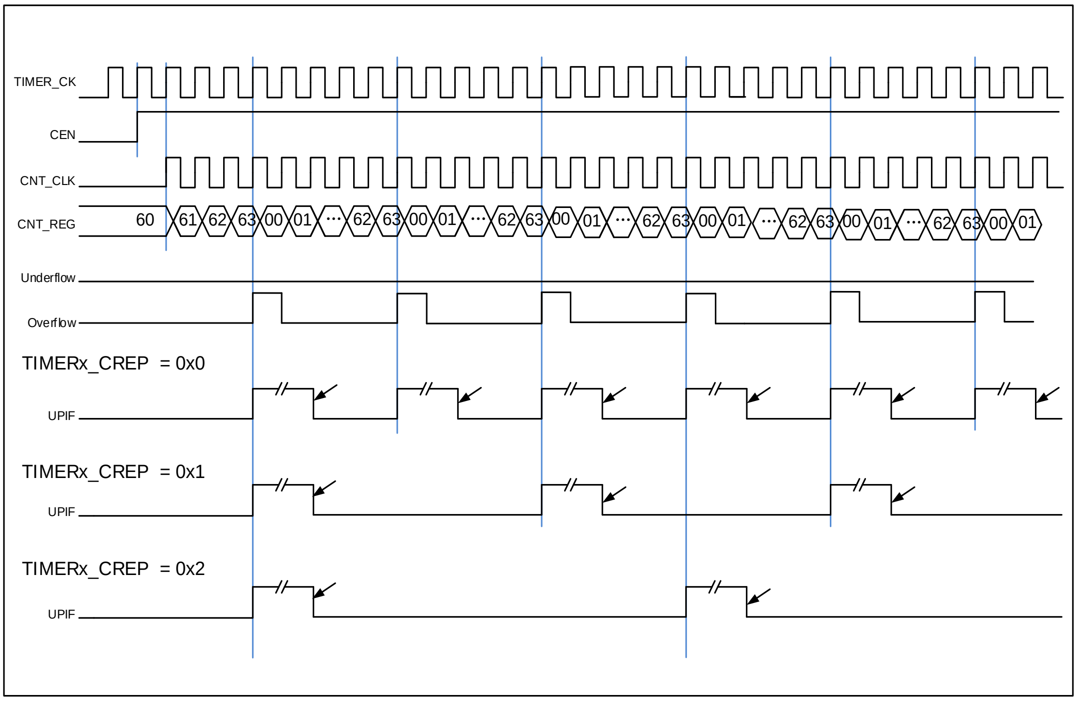
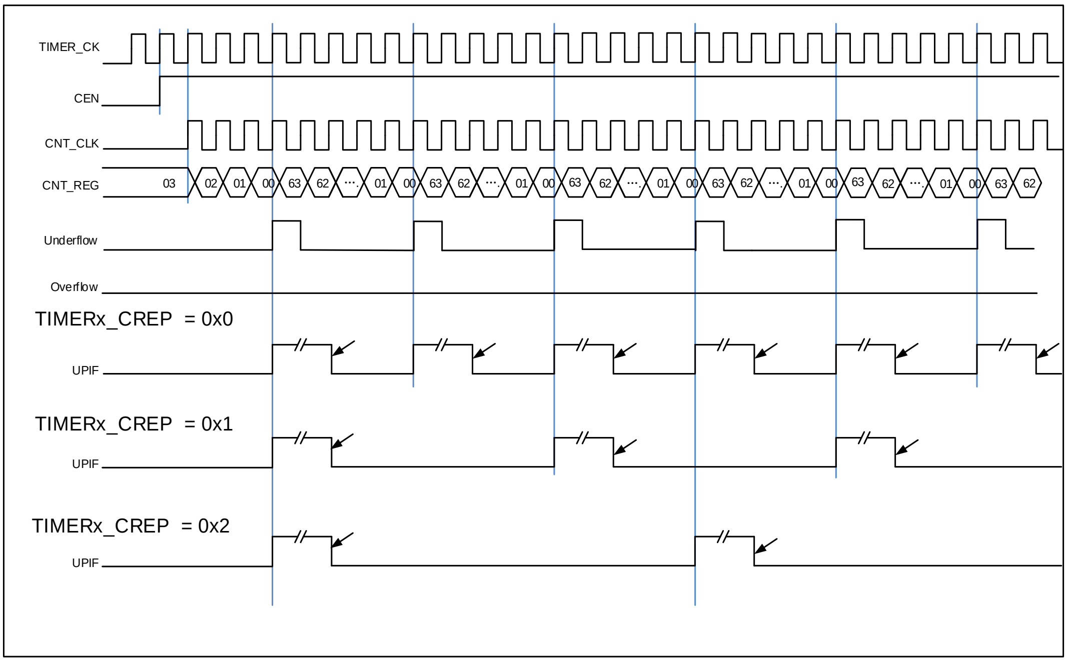


Figure 15-11. Repetition timechart for down-counter



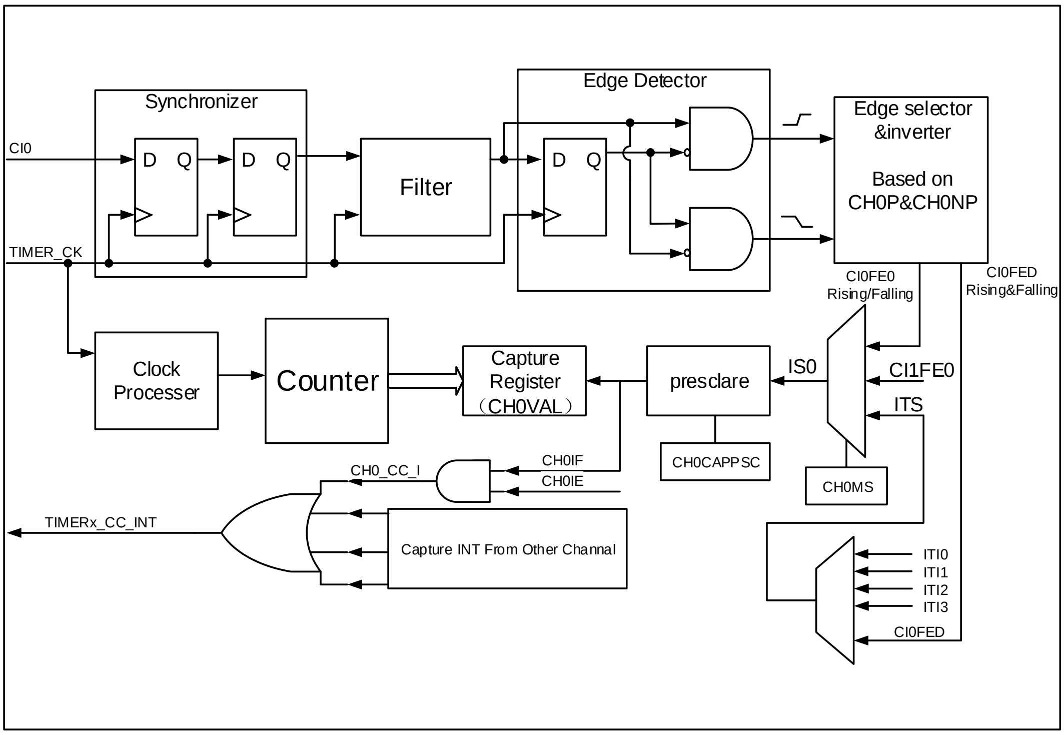
Capture/compare channels

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 TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 15-12. Input capture logic



One of channels’ input signals (CIx) can be chosen from the TIMERx_CHx signal or the Excusive-OR function of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the channel input signal (CIx) 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 CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

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

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

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

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

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

Step4: Interrupt enable. (CHxIE and CHxDEN in TIMERx_DMAINTEN)

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

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: when you wanted input signal is got, TIMERx_CHxCV will be set by counter’s value.

And CHxIF is asserted. If the CHxIF has been high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN

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

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2’b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2’b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERx_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Output compare mode**

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CHxDEN = 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 CHxCOMSEN
* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
* Select the active high polarity by CHxP/CHxNP
* Enable the output by CHxEN

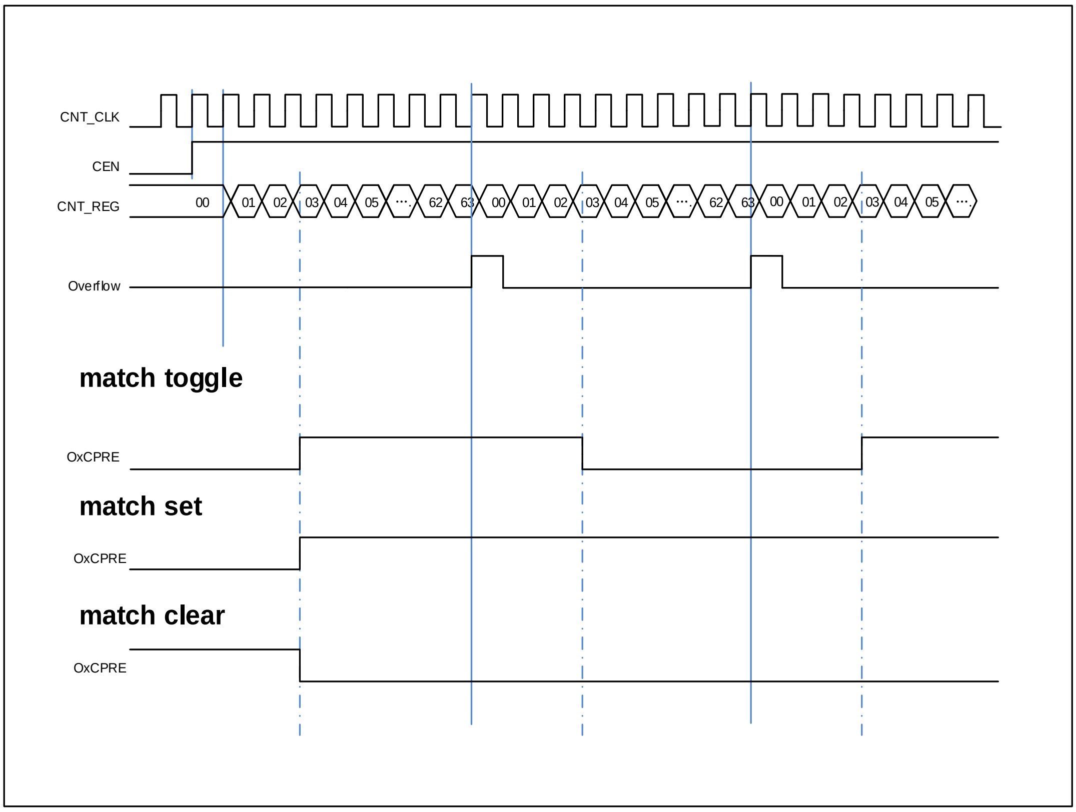
Step3: Interrupt/DMA-request enables configuration by CHxIE/CHxDEN.

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV
About the CHxVAL; you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-13. Output-compare under three modes



PWM mode

In the output PWM mode (by setting the CHxCOMCTL bits to 3’b110 (PWM mode0) or to 3’b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV 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 TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV. ***Figure 15-14. EAPWM timechart*** shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is by 2*TIMERx_CHxCV. ***Figure 15-15. CAPWM timechart*** shows the CAPWM output and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL = 3’b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL = 3’b110).

Figure 15-14. EAPWM timechart

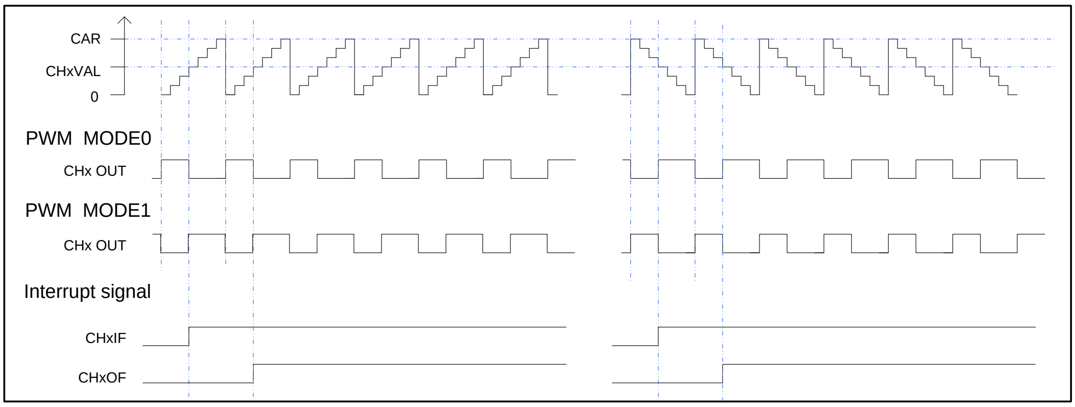
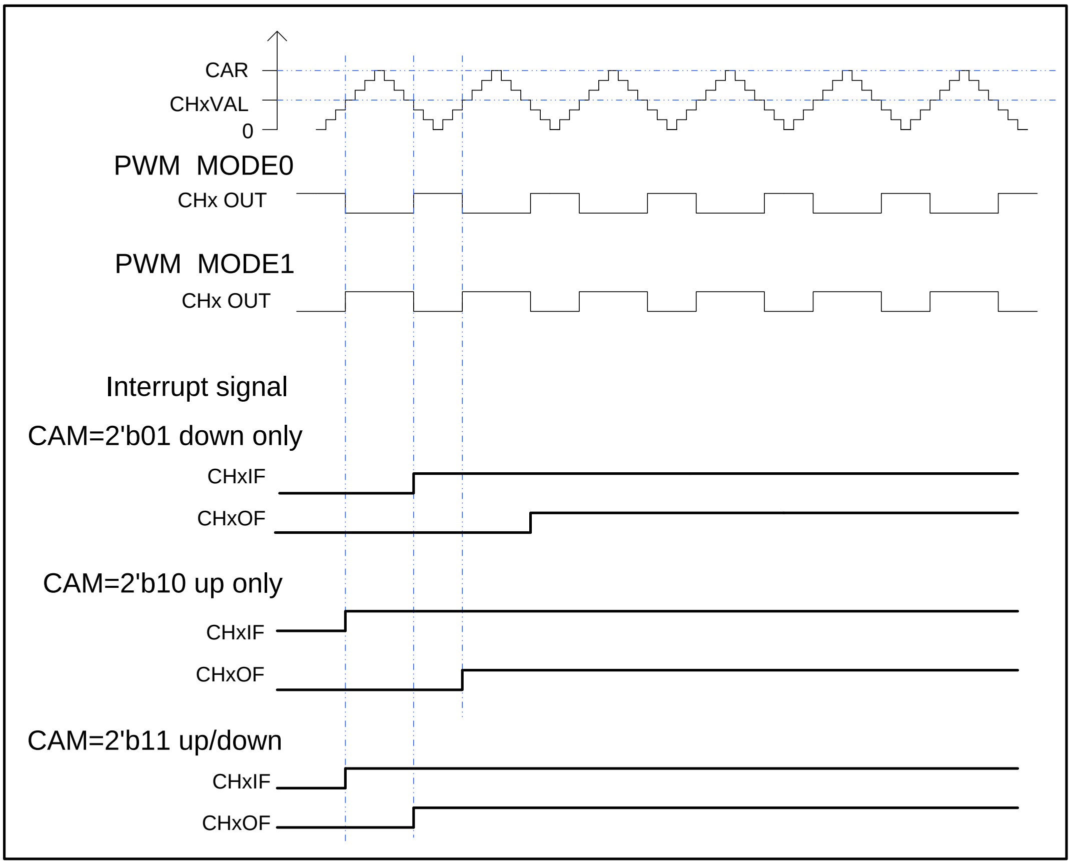


Figure 15-15. CAPWM timechart



Channel output reference signal

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

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL 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 TIMERx_CHxCV 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 CHxCOMCTL 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 TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFP signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

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 TIMERx 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 CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx_CCHP and TIMERx_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register.

Table 15-2. Complementary outputs controlled by parameters

Complementary Parameters					Output Status	
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable. If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
			1	0		
				1		
		1	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP 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⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = LOW CHx_ON output disable.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON=(!OxCPRE)⊕CHxNP CHx_ON output enable
	1		0	0	CHx_O = CHxP CHx_O output disable.	CHx_ON = CHxNP CHx_ON output disable.
				1	CHx_O = CHxP CHx_O output enable	CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = CHxNP CHx_ON output enable.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON=(!OxCPRE)⊕CHxNP CHx_ON output enable.

Dead time insertion

The dead time insertion is enabled when both CHxEN and CHxNEN are 1’b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for all channels expect for channel 3. The detail about the delay time, refer to the register TIMERx_CCHP.

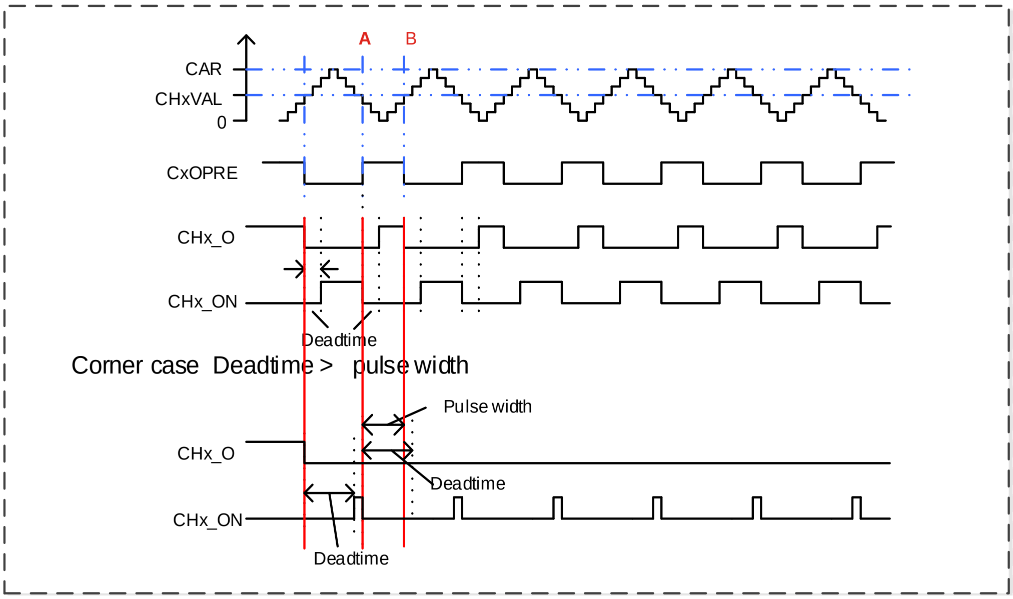
The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the ***Figure 15-16. Complementary output with dead-time insertion*** 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 = CHxVAL) 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 15-16. Complementary output with dead-time insertion***)

Figure 15-16. Complementary output with dead-time insertion



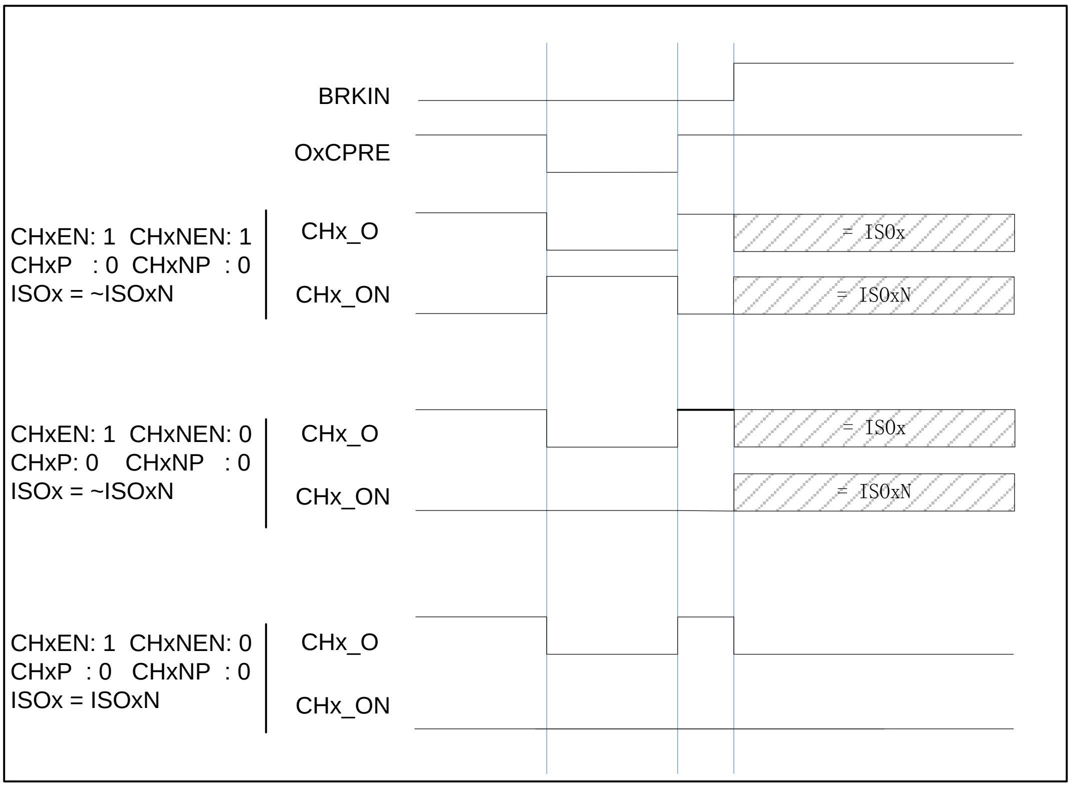
Break function

In this function, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx_CCHP.

When a break occurs, the POEN 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 TIMERx_CTL1 register as soon as POEN is 0. If IOS 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 BRKIF bit in the TIMERx_INTF register is set. If BRKIE is 1, an interrupt generated.

Figure 15-17. Output behavior in response to a break (The break high active)



Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMERx_CH0 and TIMERx_CH1 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 CI0 only, CI1 only or both CI0 and CI1, the selection mode by setting the SMC [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 TIMERx_CAR register before the counter starts to count.

Table 15-3. Counting direction versus encoder signals

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-
CI1 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
CI0 and CI1 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down
	CI0FE0=Low	X	X	Down	Up

Note:"-" means "no counting"; "X" means impossible.

Figure 15-18. Example of counter operation in encoder interface mode

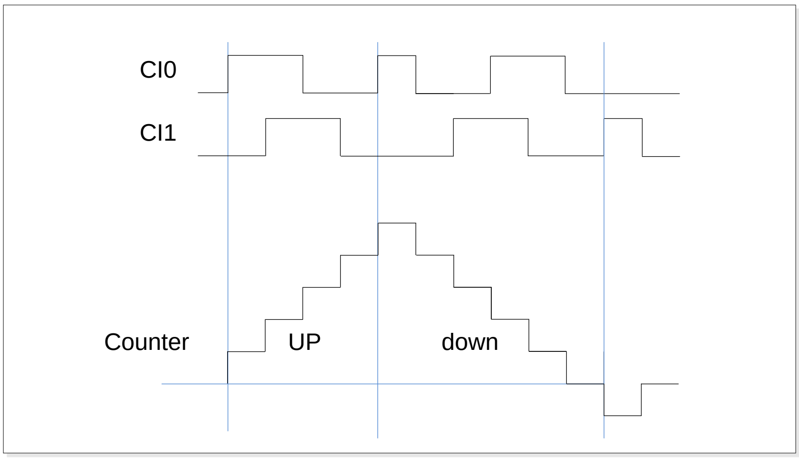
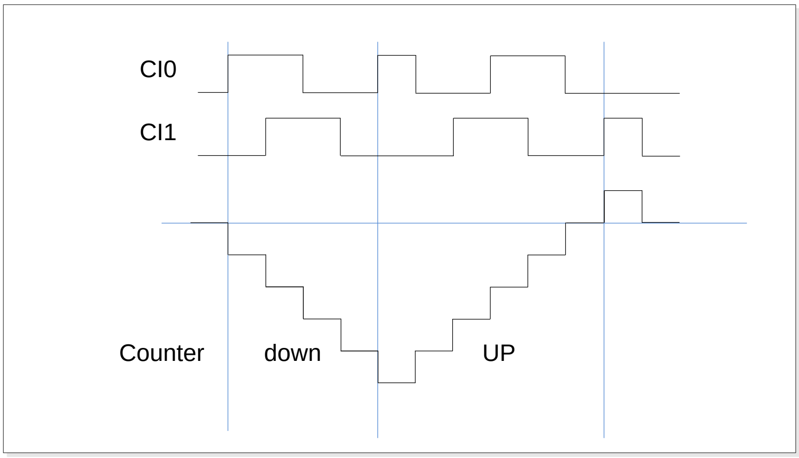


Figure 15-19. Example of encoder interface mode with CI0FE0 polarity inverted



Hall sensor function

Hall sensor is generally used to control BLDC Motor; advanced timer can support this function.

Figure 15-20. Hall sensor is used to BLDC motor show how to connect. And we can see we need two timers. First TIMER_in（Advanced/GeneralL0 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/GeneralL0 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 (TIMER0) -> TIMER_out (TIMER7 ITI0)

TIMER_in (TIMER1) -> TIMER_out (TIMER0 ITI1)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI0S, then, each of input signal change will make the CI0 toggle. CH0VAL will record the value of counter at that moment.
- Enable ITIx connected to commutation function directly by setting CCUC and CCSE.
- Configuration PWM parameter based on your request.

Figure 15-20. Hall sensor is used to BLDC motor

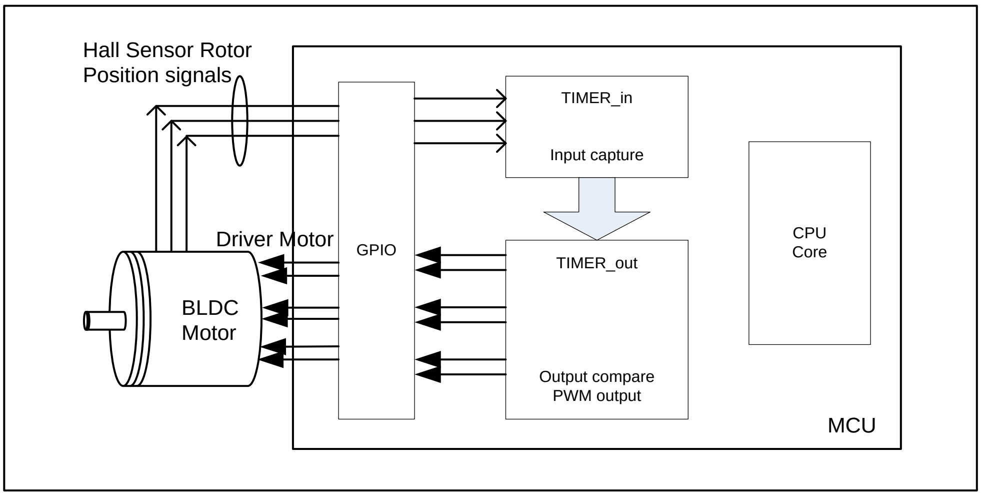
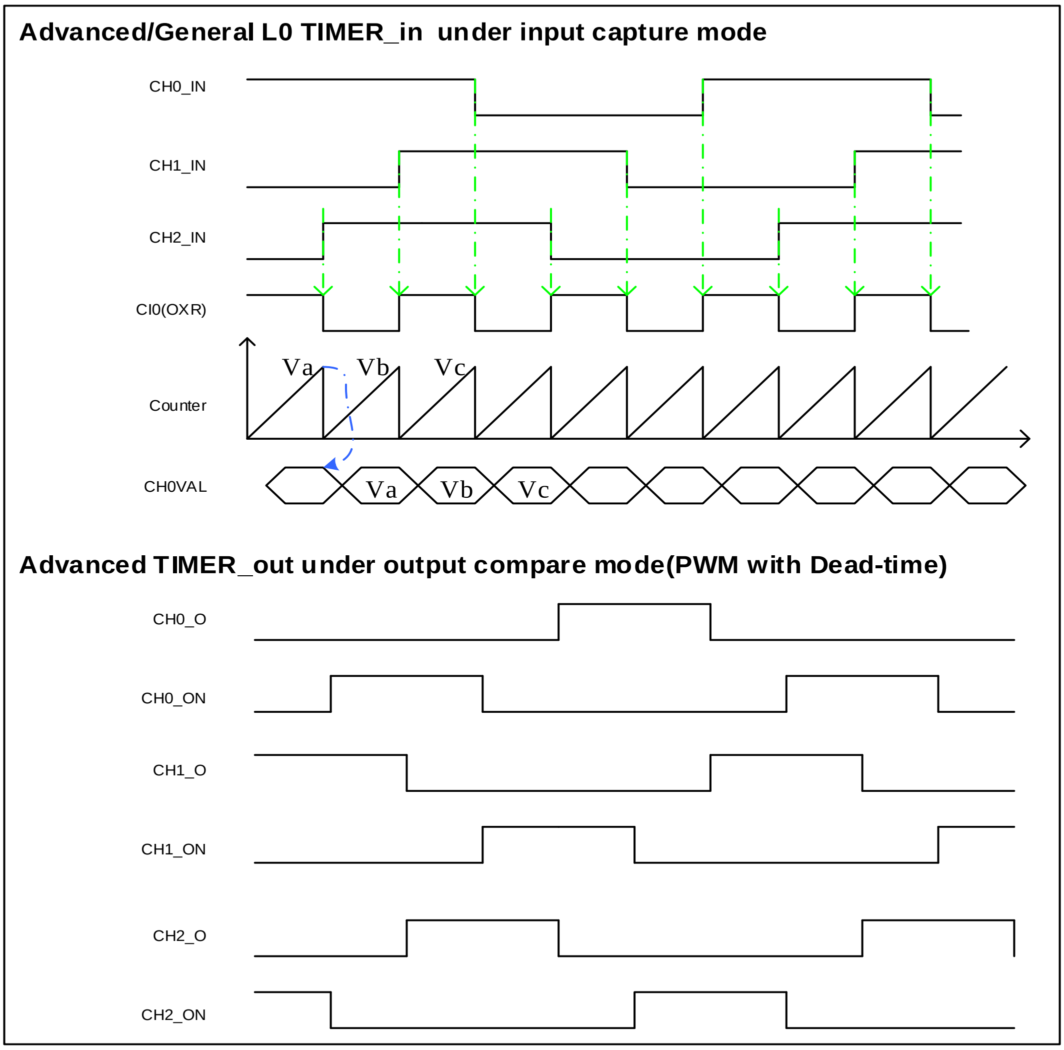


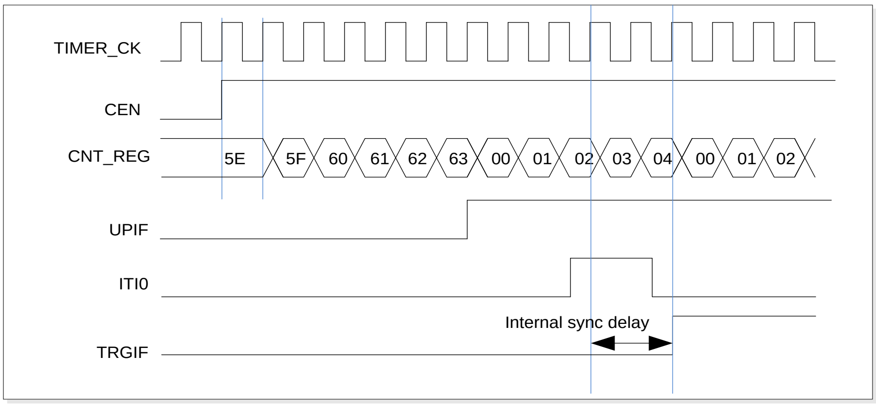
Figure 15-21. Hall sensor timing between two timers

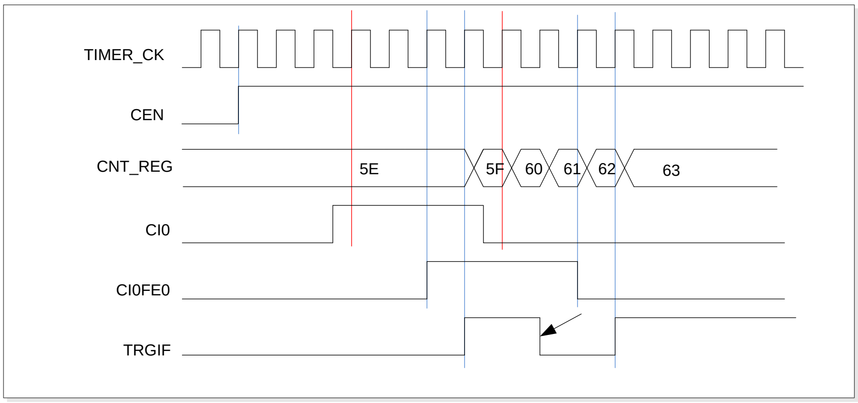

Advanced/General L0 TIMER_in under input capture mode
CH0_IN
CH1_IN
CH2_IN
CI0(OXR)
Counter
Va
Vb
Vc
CH0VAL
Va
Vb
Vc
Advanced TIMER_out under output compare mode(PWM with Dead-time)
CH0_O
CH0_ON
CH1_O
CH1_ON
CH2_O
CH2_ON

Slave controller

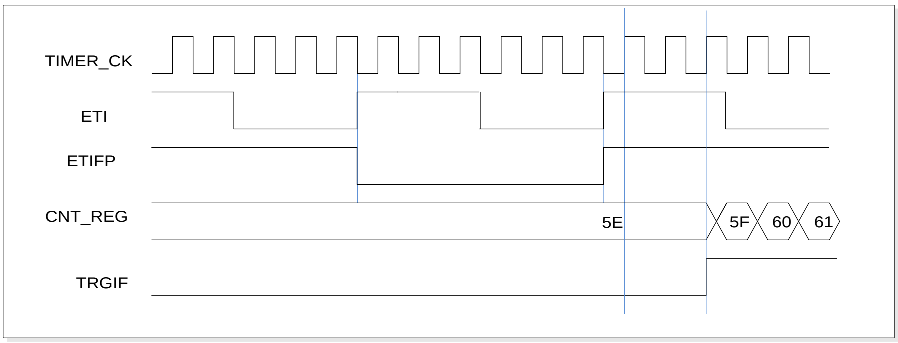
The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

Table 15-4. Slave mode example table

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000 ITI0 is the selection.	- For ITI0, no polarity selector can be used.	- For the ITI0, no filter and prescaler can be used.
	Figure 15-22. Restart mode  TIMER_CK CEN CNT_REG 5E 5F 60 61 62 63 00 01 02 03 04 00 01 02 UPIF ITI0 Internal sync delay TRGIF			
Exam2	Pause mode The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101 CI0FE0 is the selection.	TI0S=0.（Non-xor） [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	Figure 15-23. Pause mode  TIMER_CK CEN CNT_REG 5E 5F 60 61 62 63 CI0 CI0FE0 TRGIF			

Exam3	Event mode
	The counter will start to count when a rising trigger input.
		TRGS[2:0]=3'b111
		ETIF is the selection.
			ETP = 0 no polarity change.	ETPSC = 1, divided by 2.
				ETFC = 0 , no filter

	Figure 15-24. Event mode  TIMER_CK ETI ETIFP CNT_REG 5E 5F 60 61 TRGIF			

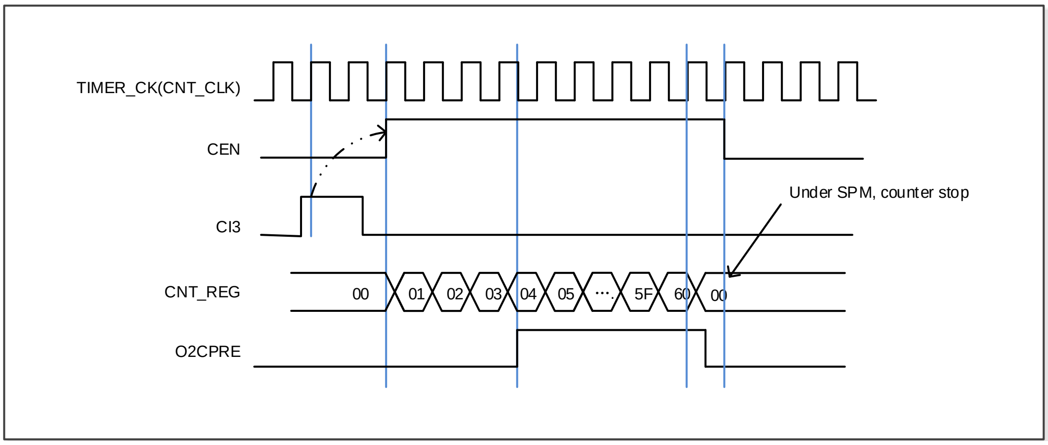
Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

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 TIMERx_CTL0 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 TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 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 CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

Figure 15-25. Single pulse mode, TIMERx_CHxCV = 0x04, TIMERx_CAR=0x60

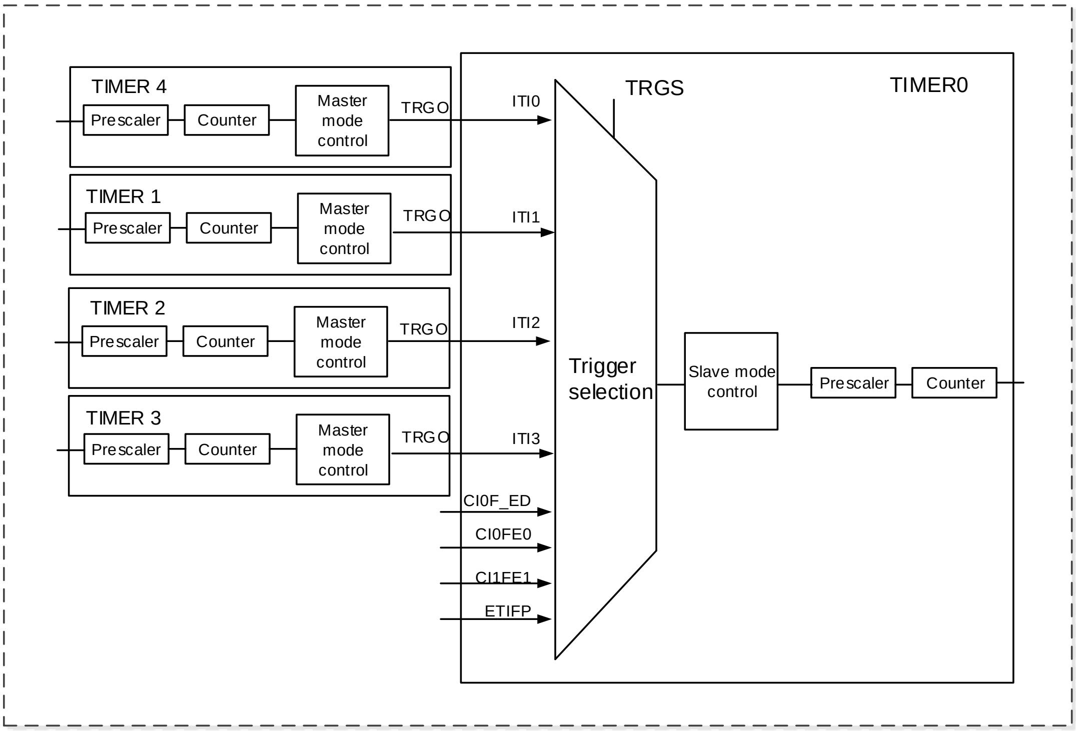


Timers interconnection

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 15-26. Timer0 master/slave mode timer example shows the timer0 trigger selection when it is configured in slave mode.

Figure 15-26. Timer0 master/slave mode timer example



Other interconnection examples:

- Timer 2 as prescaler for timer 0

We configure Timer2 as a prescaler for Timer 0. Refer to ***Figure 15-26. Timer0 master/slave mode timer example*** for connections. Do as bellow:

1. Configure Timer2 in master mode and select its update event (UPE) as trigger output (MMC=3'b010 in the TIMER2_CTL1 register). Then timer2 drives a periodic signal on each counter overflow.
2. Configure the Timer2 period (TIMER2_CAR registers).
3. Select the Timer0 input trigger source from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
4. Configure Timer0 in external clock mode 0 (SMC=3'b111 in TIMERx_SMCFG register).
5. Start Timer0 by writing '1 in the CEN bit (TIMER0_CTL0 register).
6. Start Timer2 by writing '1 in the CEN bit (TIMER2_CTL0 register).

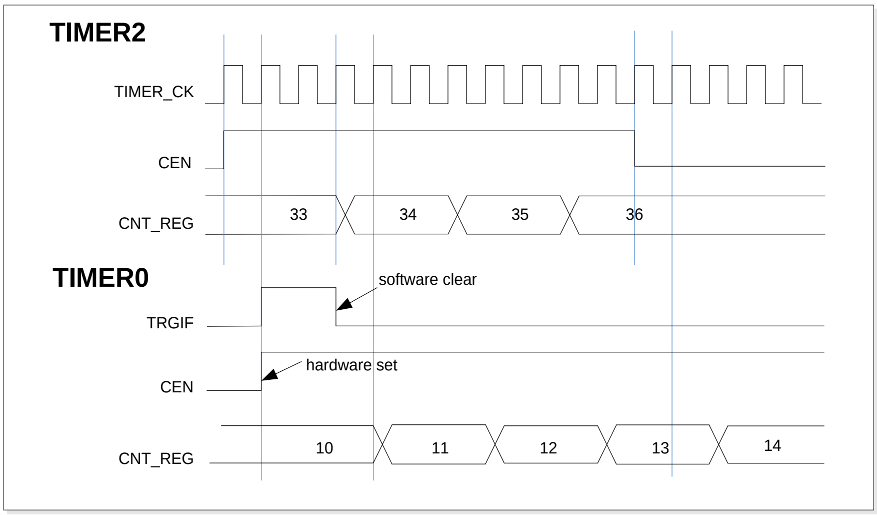
- Start Timer0 with Timer2's Enable/Update signal

In this example, we enable Timer0 with the enable output of Timer2. Refer to ***Figure 15-27. Triggering TIMER0 with enable signal of TIMER2***. Timer0 starts counting from its current value on the divided internal clock after trigger by Timer2 enable output.

When Timer0 receives the trigger signal, its CEN bit is set automatically and the counter counts until we disable timer0. 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). Timer0's SMC is set as event mode, so Timer0 can not be disabled by Timer2's disable signal. Do as follow:

1. Configure Timer2 master mode to send its enable signal as trigger output(MMC=3'b001 in the TIMER2_CTL1 register)
2. Configure Timer0 to select the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
3. Configure Timer0 in event mode (SMC=3'b 110 in TIMERx_SMCFG register).
4. Start Timer2 by writing 1 in the CEN bit (TIMER2_CTL0 register).

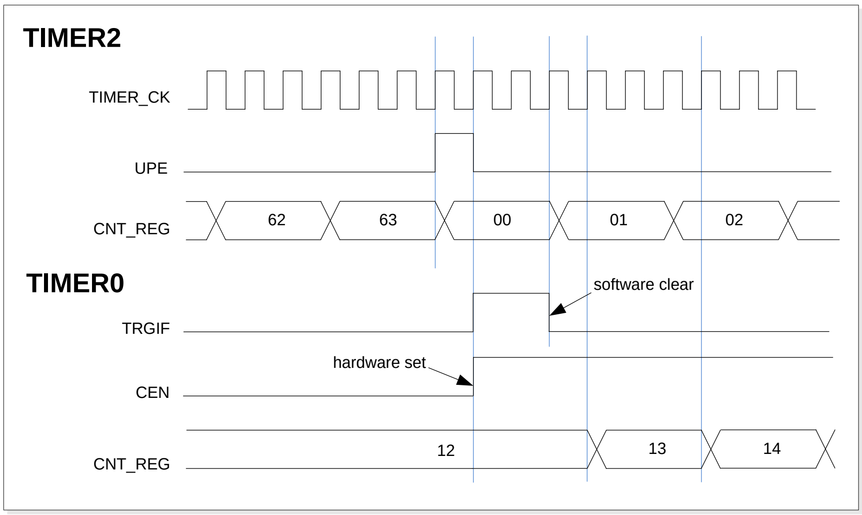
Figure 15-27. Triggering TIMER0 with enable signal of TIMER2



In this example, we also can use update Event as trigger source instead of enable signal. Refer to ***Figure 15-28. Triggering TIMER0 with update signal of TIMER2***. Do as follow:

1. Configure Timer2 in master mode and send its update event (UPE) as trigger output (MMC=3'b010 in the TIMER2_CTL1 register).
2. Configure the Timer2 period (TIMER2_CARL registers).
3. Configure Timer0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
4. Configure Timer0 in event mode (SMC=3'b110 in TIMERx_SMCFG register).
5. Start Timer2 by writing '1 in the CEN bit (TIMER2_CTL0 register).

Figure 15-28. Triggering TIMER0 with update signal of TIMER2

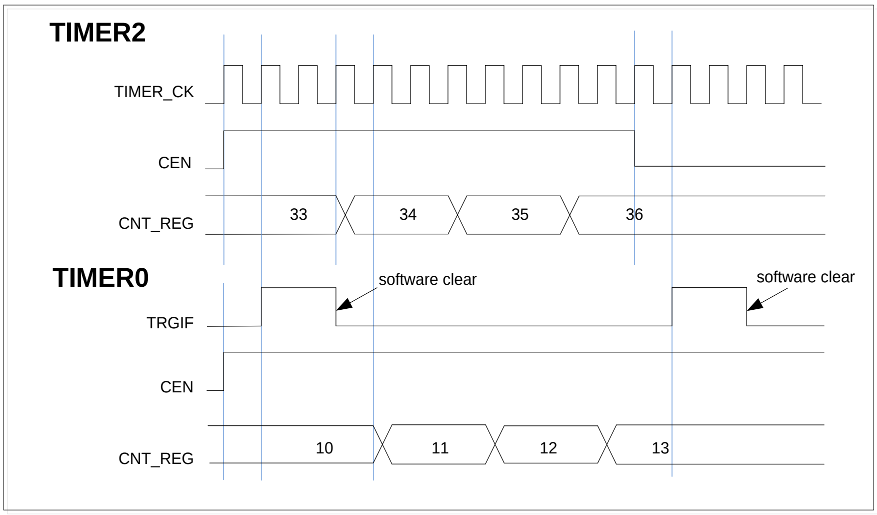


- Enable Timer0 count with Timer2's enable/O0CPRE signal

In this example, we control the enable of Timer0 with the enable output of Timer2 .Refer to ***Figure 15-29. Pause TIMER0 with enable signal of TIMER2***. Timer0 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 (fCNT_CLK = fPCLK /3). Timer0's SMC is set as pause mode, so Timer0 can be enabled/disabled by Timer2's enable/disable signal. Do as follow:

1. Configure Timer2 input master mode and output enable signal as trigger output (MMC=3'b001 in the TIMER2_CTL1 register).
2. Configure Timer0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
3. Configure Timer0 in pause mode (SMC=3'b101 in TIMERx_SMCFG register).
4. Enable Timer0 by writing '1 in the CEN bit (TIMER0_CTL0 register)
5. Start Timer2 by writing '1 in the CEN bit (TIMER2_CTL0 register).
6. Stop Timer2 by writing '0 in the CEN bit (TIMER2_CTL0 register).

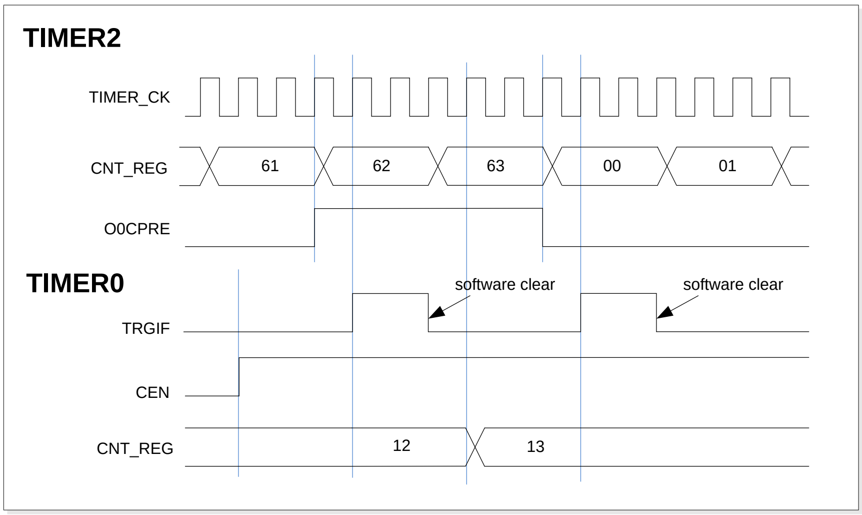
Figure 15-29. Pause TIMER0 with enable signal of TIMER2



In this example, we also can use O0CPRE as trigger source instead of enable signal output. Do as follow:

1. Configure Timer2 in master mode and output 0 Compare Prepare signal (O0CPRE) as trigger output (MMS=3'b100 in the TIMER2_CTL1 register).
2. Configure the Timer2 O0CPRE waveform (TIMER2_CHCTL0 register).
3. Configure Timer0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
4. Configure Timer0 in pause mode (SMC=3'b101 in TIMERx_SMCFG register).
5. Enable Timer0 by writing '1 in the CEN bit (TIMER0_CTL0 register).
6. Start Timer2 by writing '1 in the CEN bit (TIMER2_CTL0 register).

Figure 15-30. Pause TIMER0 with O0CPREF signal of Timer2



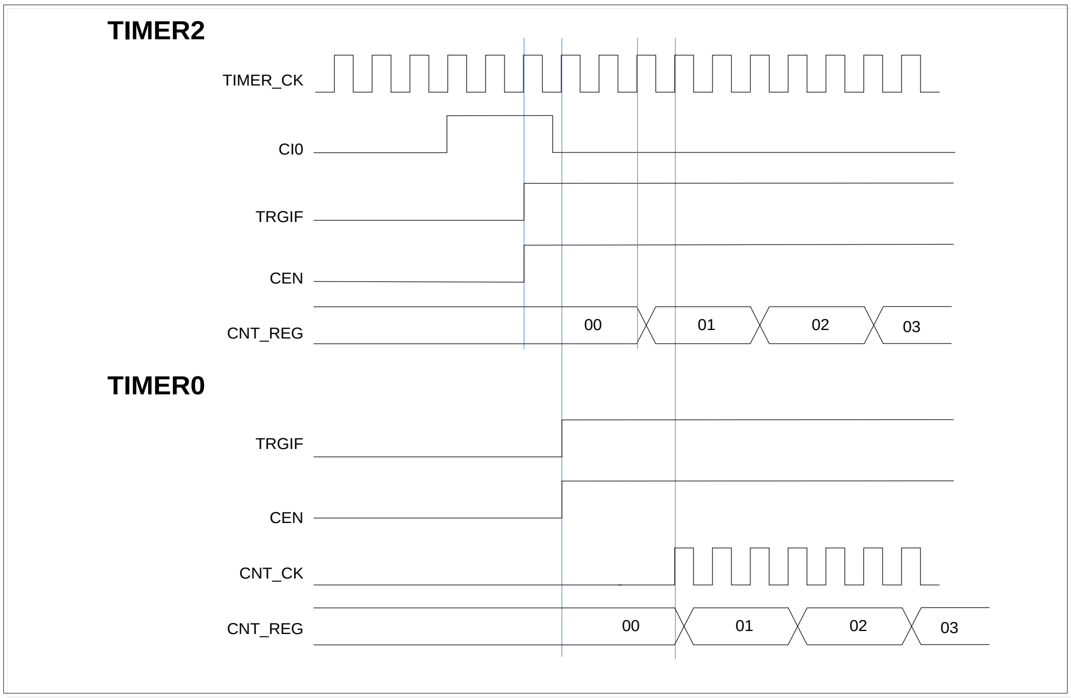
- Using an external trigger to start 2 timers synchronously

We configure the start of Timer0 triggered by the enable signal of Timer2, and Timer2 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, Timer2 must be configured in Master/Slave mode. Do as follow:

1. Configure Timer2 in slave mode to get the input trigger from CI0 (TRGS=3'b100 in the TIMER2_SMCFG register).
2. Configure Timer2 in event mode (SMC=3'b110 in the TIMER2_SMCFG register).
3. Configure the Timer2 in Master/Slave mode by writing MSM=1 (TIMER2_SMCFG register).
4. Configure Timer0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
5. Configure Timer0 in event mode (SMC=3'b110 in the TIMER0_SMCFG register).

When a rising edge occurs on Timer2's CI0, two timer's counters start counting synchronously on the internal clock and both TRGIF flags are set.

Figure 15-31. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input



Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register

TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer’s DMA request is finished. While if TIMERx_DMATC is not 0, such as 3(4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer’s registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

Timer debug mode

When the Cortex™-M3 halted, and the TIMERx_HOLD configuration bit in DBG_CTL register is set to 1, the TIMERx counter stops.

15.1.5. Register definition

Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	SPM	UPS	UPDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV 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	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in

		TIMERx_CHCTL0 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	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	SPM	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	UPS	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 UPG 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	UPDIS	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 UPG 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	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.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ISO3	ISO2N	ISO2	ISO1N	ISO1	ISO0N	ISO0	TI0S	MMC[2:0]			DMAS	CCUC	Reserved	CCSE
	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw		rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	ISO3	Idle state of channel 3 output Refer to ISO0 bit
13	ISO2N	Idle state of channel 2 complementary output Refer to ISO0N bit
12	ISO2	Idle state of channel 2 output Refer to ISO0 bit
11	ISO1N	Idle state of channel 1 complementary output Refer to ISO0N bit
10	ISO1	Idle state of channel 1 output Refer to ISO0 bit
9	ISO0N	Idle state of channel 0 complementary output 0: When POEN bit is reset, CH0_ON is set low. 1: When POEN bit is reset, CH0_ON is set high This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
8	ISO0	Idle state of channel 0 output 0: When POEN bit is reset, CH0_O is set low. 1: When POEN bit is reset, CH0_O is set high The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
7	TI0S	Channel 0 trigger input selection 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	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 UPG bit in the TIMERx_SWEVG 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 O0CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO
3	DMAS	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	CCUC	Commutation control shadow register update control When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below: 0: The shadow registers update by when CMTG bit is set. 1: The shadow registers update by when CMTG 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	Must be kept at reset value.
0	CCSE	Commutation control shadow enable 0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled. 1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL 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.

Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	SMC1	ETPSC[1:0]		ETFC[3:0]				MSM	TRGS[2:0]			Reserved	SMC[2:0]		
rw	rw	rw		rw				rw	rw				rw		

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	ETP	External trigger polarity This bit specifies the polarity of ETI signal 0: ETI is active at high level or rising edge. 1: ETI is active at low level or falling edge.
14	SMC1	Part of SMC for enable External clock mode1. In external clock mode 1, the counter is clocked by any active edge on the ETIFP signal. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. It is possible to simultaneously use external clock mode 1 with the restart mode, pause mode or event mode. But the TRGS bits must not be 3'b111 in this case. The external clock input will be ETIFP if external clock mode 0 and external clock mode 1 are enabled at the same time. Note: External clock mode 0 enable is in this register's SMC bit-filed.
13:12	ETPSC[1:0]	External trigger prescaler The frequency of external trigger signal ETIFP 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 ETIFP frequency. 00: Prescaler disable 01: ETIFP frequency will be divided by 2 10: ETIFP frequency will be divided by 4 11: ETIFP frequency will be divided by 8
11:8	ETFC[3:0]	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 ETIFP signal and the length of the digital filter applied to ETIFP.

		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	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	TRGS[2:0]	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 (ITI0) 001: Internal trigger input 1 (ITI1) 010: Internal trigger input 2 (ITI2) 011: Internal trigger input 3 (ITI3) 100: CI0 edge flag (CI0F_ED) 101: channel 0 input Filtered output (CI0FE0) 110: channel 1 input Filtered output (CI1FE1) 111: External trigger input filter output(ETIFP) These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	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 CI1FE1 edge, while the direction depends on CI0FE0 level. 010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the

		direction depends on CI1FE1 level. 011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 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 cannot be disabled by the slave mode controller. 111: External clock mode 0. The counter counts on the rising edges of the selected trigger.

DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	CMTDEN	CH3DEN	CH2DEN	CH1DEN	CH0DEN	UPDEN	BRKIE	TRGIE	CMTIE	CH3IE	CH2IE	CH1IE	CH0IE	UPIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13	CMTDEN	Commutation DMA request enable 0: disabled 1: enabled
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable

		0: disabled 1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	BRKIE	Break interrupt enable 0: disabled 1: enabled
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5	CMTIE	commutation interrupt enable 0: disabled 1: enabled
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CH3OF	CH2OF	CH1OF	CH0OF	Reserved	BRKIF	TRGIF	CMTIF	CH3IF	CH2IF	CH1IF	CH0IF	UPIF
			rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8	Reserved	Must be kept at reset value.
7	BRKIF	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	TRGIF	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	CMTIF	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	CH3IF	Channel 3 ‘s capture/compare interrupt flag Refer to CH0IF description
3	CH2IF	Channel 2 ‘s capture/compare interrupt flag Refer to CH0IF description
2	CH1IF	Channel 1 ‘s capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 ‘s capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. 0: No Channel 0 interrupt occurred 1: Channel 0 interrupt occurred
0	UPIF	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

Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BRKG	TRGG	CMTG	CH3G	CH2G	CH1G	CH0G	UPG
								w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	BRKG	Break event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event

6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	CMTG	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 (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1). 0: No affect 1: Generate channel's c/c control update event
4	CH3G	Channel 3's capture or compare event generation Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	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

Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>CH1COM
CEN</td><td colspan="3">CH1COMCTL[2:0]</td><td>CH1COM
SEN</td><td>CH1COM
FEN</td><td colspan="2" rowspan="2">CH1MS[1:0]</td><td>CH0COM
CEN</td><td colspan="3">CH0COMCTL[2:0]</td><td>CH0COM
SEN</td><td>CH0COM
FEN</td><td colspan="2" rowspan="2">CH0MS[1:0]</td></tr>
<tr><td colspan="4">CH1CAPFLT[3:0]</td><td colspan="2">CH1CAPPSC[1:0]</td><td colspan="4">CH0CAPFLT[3:0]</td><td colspan="2">CH0CAPPSC[1:0]</td></tr>
<tr><td colspan="4">rw</td><td colspan="2">rw</td><td colspan="2">rw</td><td colspan="4">rw</td><td colspan="2">rw</td><td colspan="2">rw</td></tr>
</table>

Output compare mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description
9:8	CH1MS[1:0]	Channel 1 mode 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IS1 is connected to CI1FE1 10: Channel 1 is configured as input, IS1 is connected to CI0FE1 11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.

		000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced low level. 101: Force high. O0CPRE is forced high level. 110: PWM mode0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high. 111: PWM mode1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low. When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).
3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH0COMFEN	Channel 0 output compare 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 CH0_O is set to the compare level independently from the result of the comparison. 0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 10: Channel 0 is configured as input, IS0 is connected to CI1FE0 11: Channel 0 is configured as input, IS0 is connected to ITS, This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture 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 CI0 input signal and the length of the digital filter applied to CI0. 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	CH0CAPPSC[1:0]	Channel 0 input capture prescaler

		This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 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	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>CH3COM CEN</td><td colspan="3">CH3COMCTL[2:0]</td><td>CH3COM SEN</td><td>CH3COM FEN</td><td colspan="2" rowspan="2">CH3MS[1:0]</td><td>CH2COM CEN</td><td colspan="3">CH2COMCTL[2:0]</td><td>CH2COM SEN</td><td>CH2COM FEN</td><td colspan="2" rowspan="2">CH2MS[1:0]</td></tr>
<tr><td colspan="4">CH3CAPFLT[3:0]</td><td colspan="2">CH3CAPPSC[1:0]</td><td colspan="4">CH2CAPFLT[3:0]</td><td colspan="2">CH2CAPPSC[1:0]</td></tr>
<tr><td colspan="4">rw</td><td colspan="2">rw</td><td colspan="2">rw</td><td colspan="4">rw</td><td colspan="2">rw</td><td colspan="2">rw</td></tr>
</table>

Output compare mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable Refer to CH0COMSEN description
9:8	CH3MS[1:0]	Channel 3 mode 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. (CH3EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 3 is configured as output 01: Channel 3 is configured as input, IS3 is connected to CI3FE3

		10: Channel 3 is configured as input, IS3 is connected to CI2FE3 11: Channel 3 is configured as input, IS3 is connected to ITS, This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, the O2CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 2 output compare clear disable 1: Channel 2 output compare clear enable
6:4	CH2COMCTL[2:0]	Channel 2 compare output control This bit-field controls the behavior of the output reference signal O2CPRE which drives CH2_O and CH2_ON. O2CPRE is active high, while CH2_O and CH2_ON active level depends on CH2P and CH2NP bits. 000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT. 001: Set the channel output. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter matches the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced low level. 101: Force high. O2CPRE is forced high level. 110: PWM mode 0. When counting up, O2CPRE is high as long as the counter is smaller than TIMERx_CH2CV else low. When counting down, O2CPRE is low as long as the counter is larger than TIMERx_CH2CV else high. 111: PWM mode 1. When counting up, O2CPRE is low as long as the counter is smaller than TIMERx_CH2CV else high. When counting down, O2CPRE is high as long as the counter is larger than TIMERx_CH2CV else low. When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE).
3	CH2COMSEN	Channel 2 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled. 0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable The PWM mode can be used without validating the shadow register only in single

		pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH2COMFEN	Channel 2 output compare 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 CH2_O is set to the compare level independently from the result of the comparison. 0: Channel 2 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH2_O output is 5 clock cycles. 1: Channel 2 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH2_O output is 3 clock cycles.
1:0	CH2MS[1:0]	Channel 2 I/O mode 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. (CH2EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 2 is configured as output 01: Channel 2 is configured as input, IS2 is connected to CI2FE2 10: Channel 2 is configured as input, IS2 is connected to CI3FE2 11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture 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 CI2 input signal and the length of the digital filter applied to CI2. 0000: Filter disable, 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	CH2CAPPSC[1:0]	Channel 2 input capture prescaler This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 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	CH2MS[1:0]	Channel 2 mode selection Same as Output compare mode

Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	CH3P	Channel 3 capture/compare function polarity Refer to CH0P description

12	CH3EN	Channel 3 capture/compare function enable Refer to CH0EN description
11	CH2NP	Channel 2 complementary output polarity Refer to CH0NP description
10	CH2NEN	Channel 2 complementary output enable Refer to CH0NEN description
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable Refer to CH0EN description
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	CH1NEN	Channel 1 complementary output enable Refer to CH0NEN description
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 active high 1: Channel 0 active low This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10 and CH0MS [1:0] bit-filed in TIMERx_CHCTL0 register is 00.
2	CH0NEN	Channel 0 complementary output enable When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0. 0: Channel 0 complementary output disabled 1: Channel 0 complementary output enabled
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.

		0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted. 1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is inverted. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 0 disabled 1: Channel 0 enabled

Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

PSC[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	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.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CARL[15:0]	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.

Counter repetition register (TIMERx_CREP)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Reserved	CREP[7:0]
	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	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.

Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH0VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 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 0 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.

Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH1VAL[15:0]	Capture or compare value of channel1 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.

Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH2VAL[15:0]	Capture or compare value of channel 2 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.

Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 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.

Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POEN	OAEN	BRKP	BRKEN	ROS	IOS	PROT[1:0]		DTCFG[7:0]							
rw	rw	rw	rw	rw	rw	rw		rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	POEN	Primary output enable This bit s set by software or automatically by hardware depending on the OAEN 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 (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set. 0: Channel outputs are disabled or forced to idle state. 1: Channel outputs are enabled.
14	OAEN	Output automatic enable This bit specifies whether the POEN bit can be set automatically by hardware.

		0: POEN can be not set by hardware. 1: POEN can be set by hardware automatically at the next update event, if the break input is not active. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
13	BRKP	Break polarity This bit specifies the polarity of the BRKIN input signal. 0: BRKIN input active low 1; BRKIN input active high
12	BRKEN	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 This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
11	ROS	Run mode off-state configure When POEN 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 POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.
10	IOS	Idle mode off-state configure When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode. 0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is reset, he channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.
9:8	PROT[1:0]	Complementary register protect control This bit-filed specifies the write protection property of registers. 00: protect disable. No write protection. 01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected. 10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode)

		and the ROS/IOS bits in TIMERx_CCHP register are writing protected. 11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is configured in output) are writing protected. This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.
7:0	DTCFG[7:0]	Dead time configure This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow: DTCFG [7:5] =3’b0xx: DTvalue = DTCFG [7:0] * t_{DT}, t_{DT} = t_{DTS}. DTCFG [7:5] =3’b10x: DTvalue = (64+DTCFG [5:0]) * t_{DT}, t_{DT} = t_{DTS} * 2. DTCFG [7:5] =3’b110: DTvalue = (32+DTCFG [4:0]) * t_{DT}, t_{DT} = t_{DTS} * 8. DTCFG [7:5] =3’b111: DTvalue = (32+DTCFG [4:0]) * t_{DT}, t_{DT} = t_{DTS} * 16. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed is defined the number of DMA will access(R/W) the register of TIMERx_DMATB 5’b0_0000: 1 time transfer 5’b0_0001: 2 times transfer … 5’b1_0001: 18 times transfer
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address

This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.

5’b0_0000: TIMERx_CTL0

5’b0_0001: TIMERx_CTL1

…

5’b1_0010: TIMERx_DMACFG

In a word: Start Address = TIMERx_CTL0 + DMASAR*4

DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

15.2. General level0 timer (TIMERx, x=1, 2, 3, 4)

15.2.1. Overview

The general level0 timer module (Timer1, 2, 3, 4) is a four-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 time reference is a 16-bit or 32-bit counter that can be used as an unsigned counter.

In addition, the general level0 timers can be programmed and be used to count or time external events that drive other timers.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

15.2.2. Characteristics

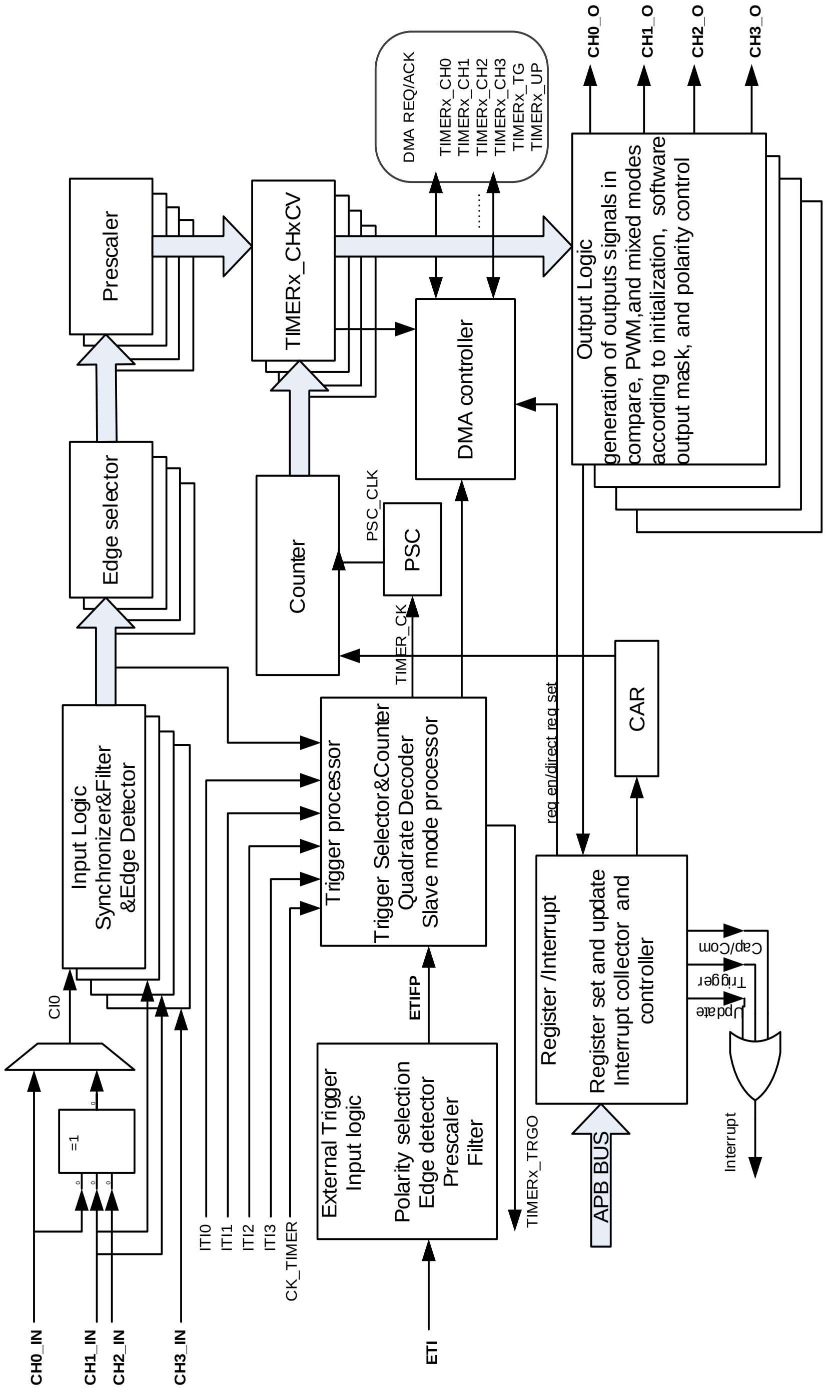
- Total channel num: 4.
- Counter width: 16bit.
- Source of count clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit. 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, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

15.2.3. Block diagram

Figure 15-32. General Level 0 timer block diagram provides details on the internal

configuration of the general level0 timer.

Figure 15-32. General Level 0 timer block diagram



15.2.4. Function overview

Clock selection

The general level0 TIMER has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

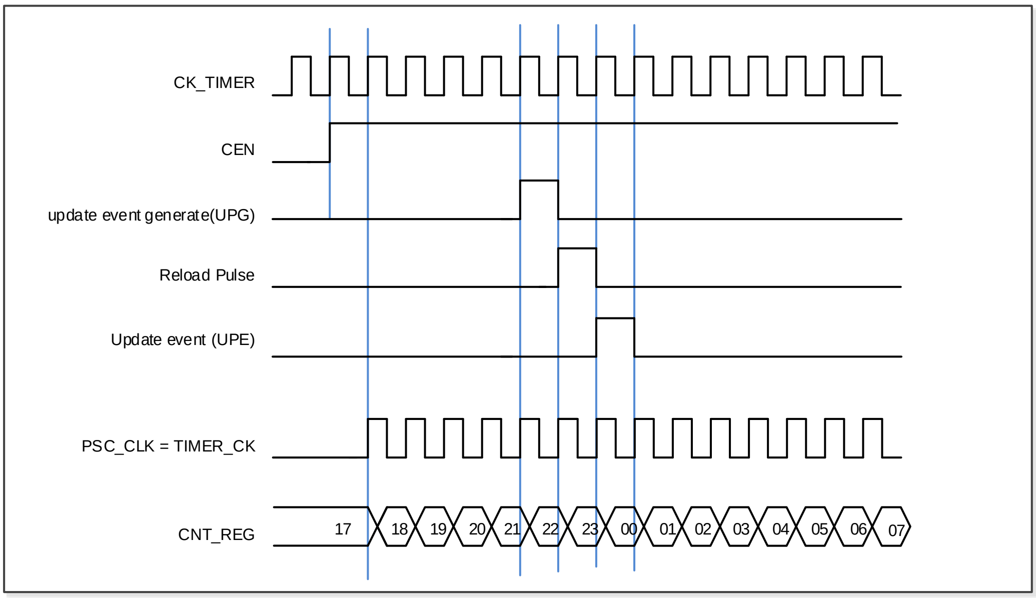
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the slave mode is disabled (SMC [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, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU.

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

Figure 15-33. Normal mode, internal clock divided by 1



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin source

The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CI0/TIMERx_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

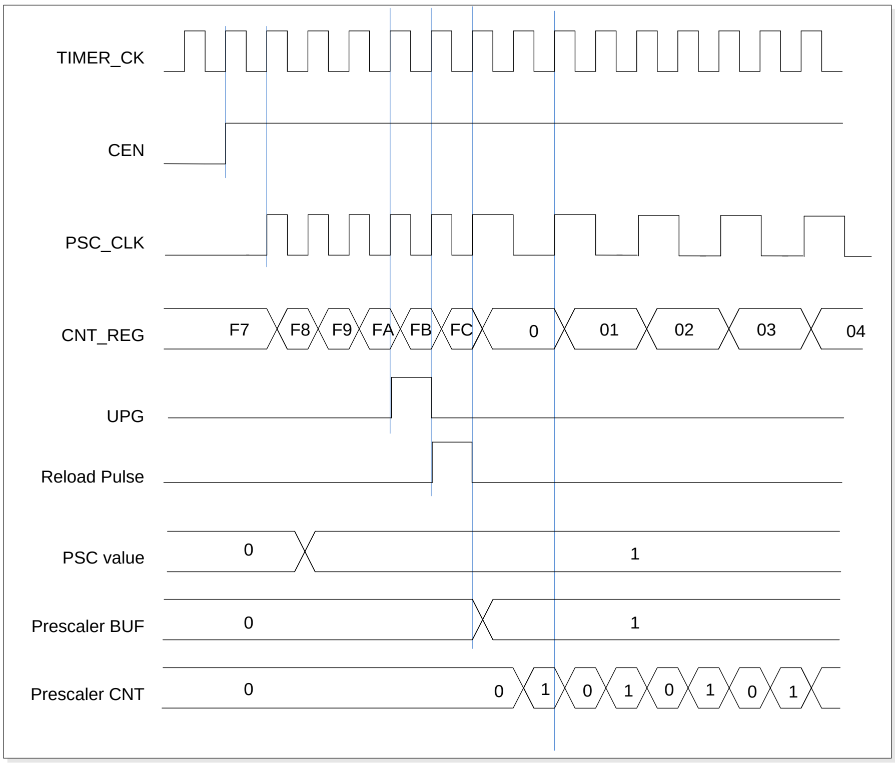
- SMC1== 1’b1 (external clock mode 1). External input pin source (ETI)

The TIMER_CK, driven counter’s prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the clock source is selected to come from the ETI signal, the trigger controller including the edge detection circuitry will generate a clock pulse during each ETI signal rising edge to clock the counter prescaler.

Prescaler

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

Figure 15-34. Counter timing diagram with prescaler division change from 1 to 2



Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, 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 TIMERx_CAR=0x63.

Figure 15-35. Up-counter timechart, PSC=0/1

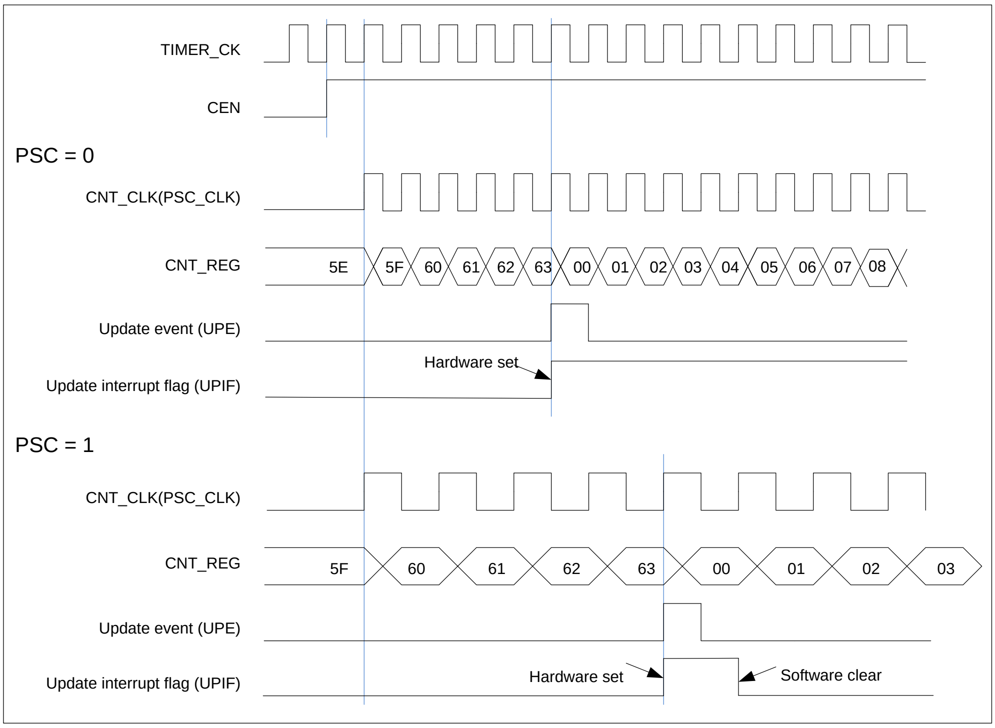
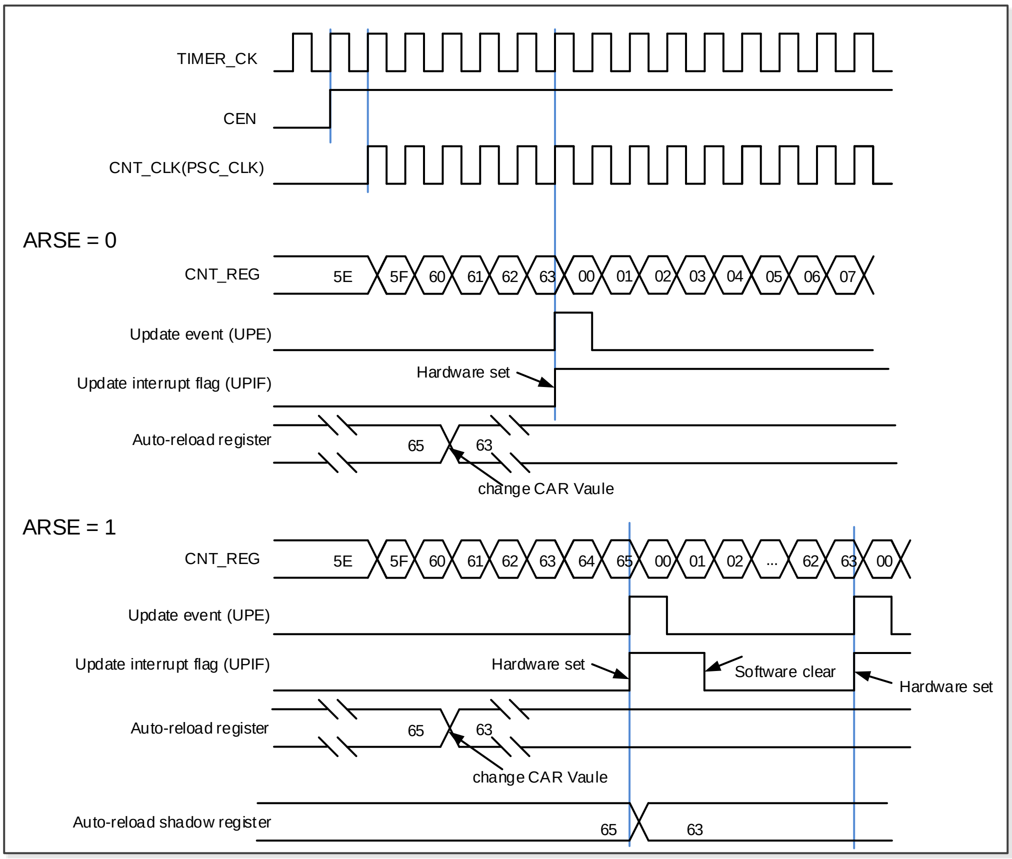


Figure 15-36. Up-counter timechart, change TIMERx_CAR on the go.



Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR 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 was generated after the number (TIMERx_CREP+1) of underflow. Else the update event is generated at each counter underflow. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down-counting mode.

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

If the UPDIS bit in TIMERx_CTL0 register is set, 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 frequencies when TIMERx_CAR=0x63.

Figure 15-37. Down-counter timechart, PSC=0/1

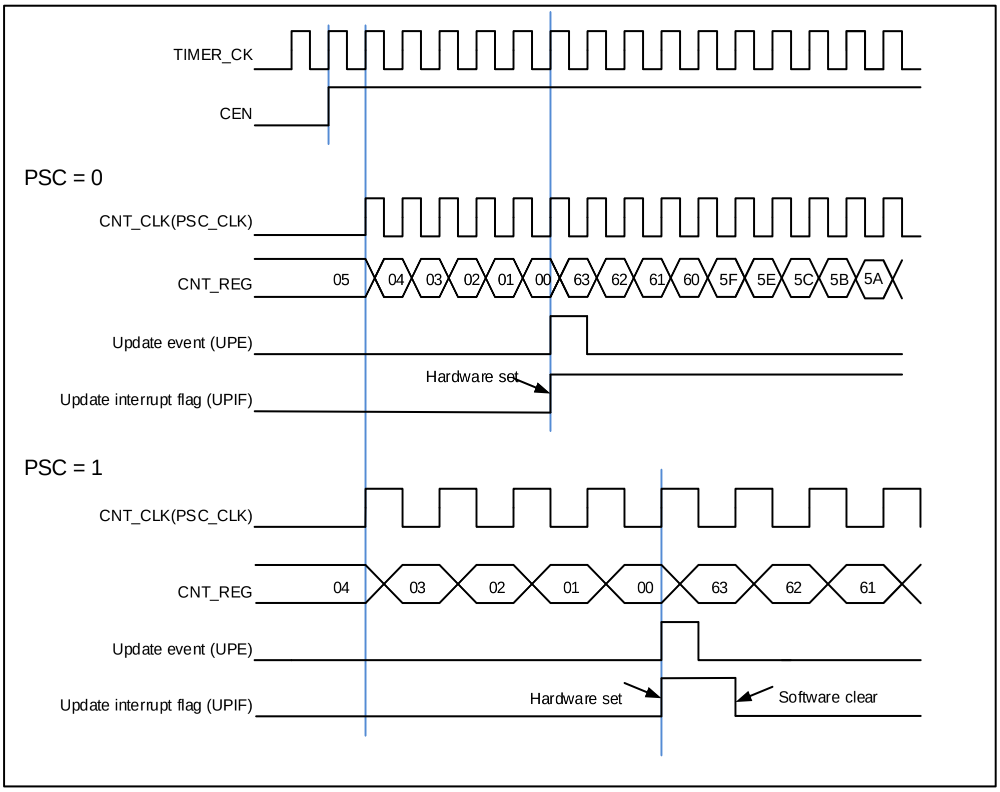
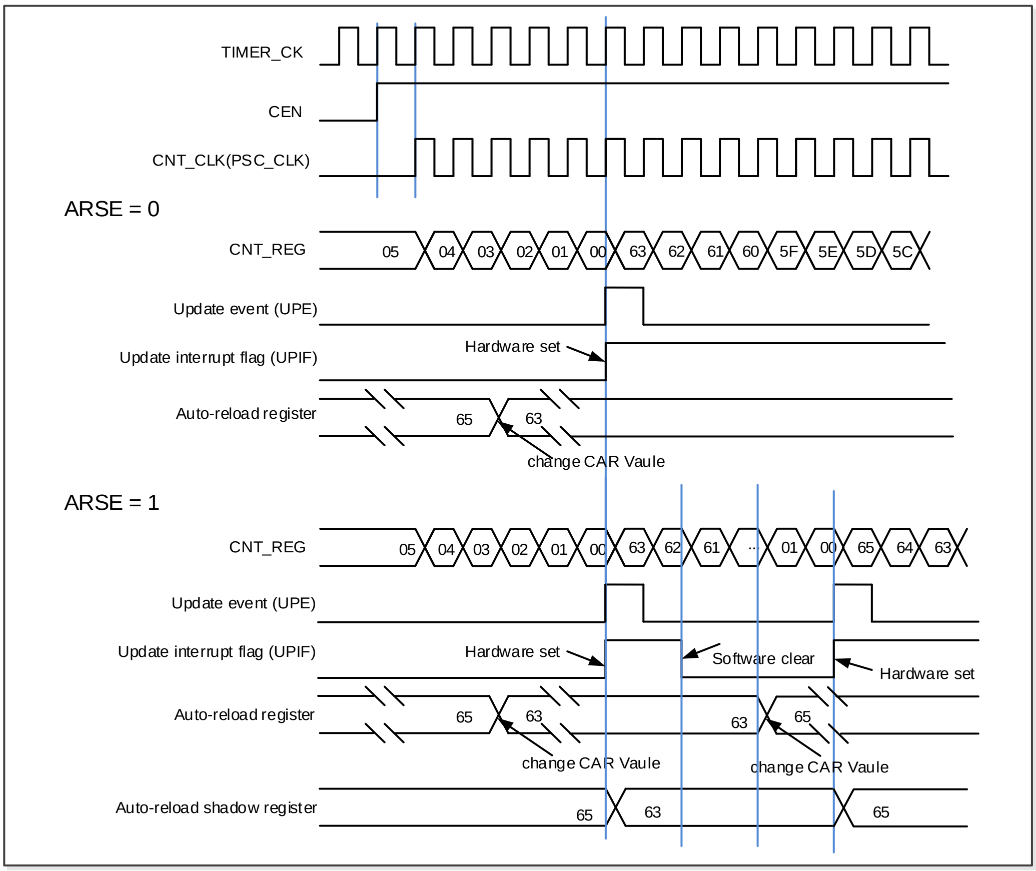


Figure 15-38. Down-counter timechart, change TIMERx_CAR on the go.



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 mode and generates an underflow event when the counter counts to 1 in the down-counting mode. The counting direction bit DIR in the TIMERx_CTL0 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 UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 irrespective of whether the counter is counting up or down in the center-align counting mode and generates an update event.

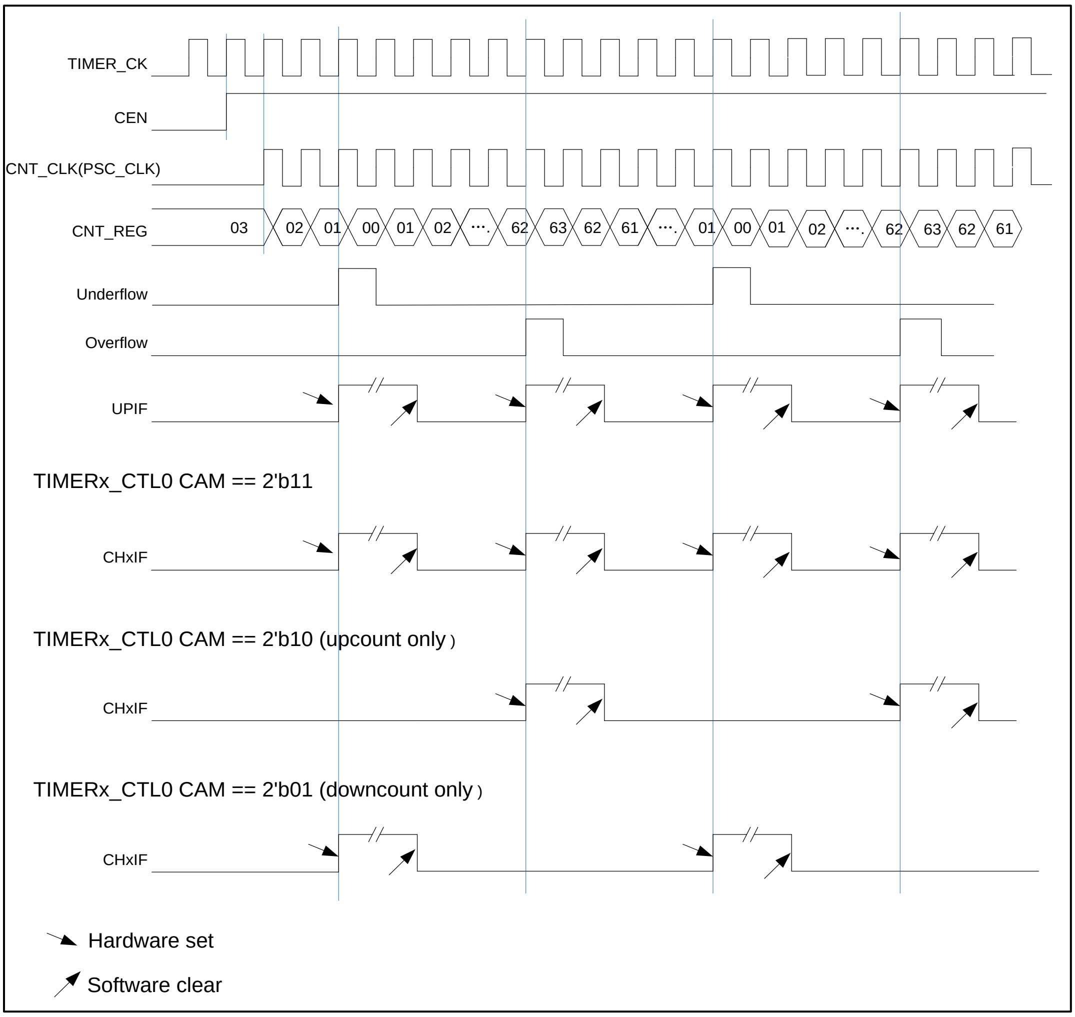
The UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 15-39. Center-aligned counter timechart***.

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

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

Figure 15-39. Center-aligned counter timechart shows some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x63, TIMERx_PSC=0x0.

Figure 15-39. Center-aligned counter timechart



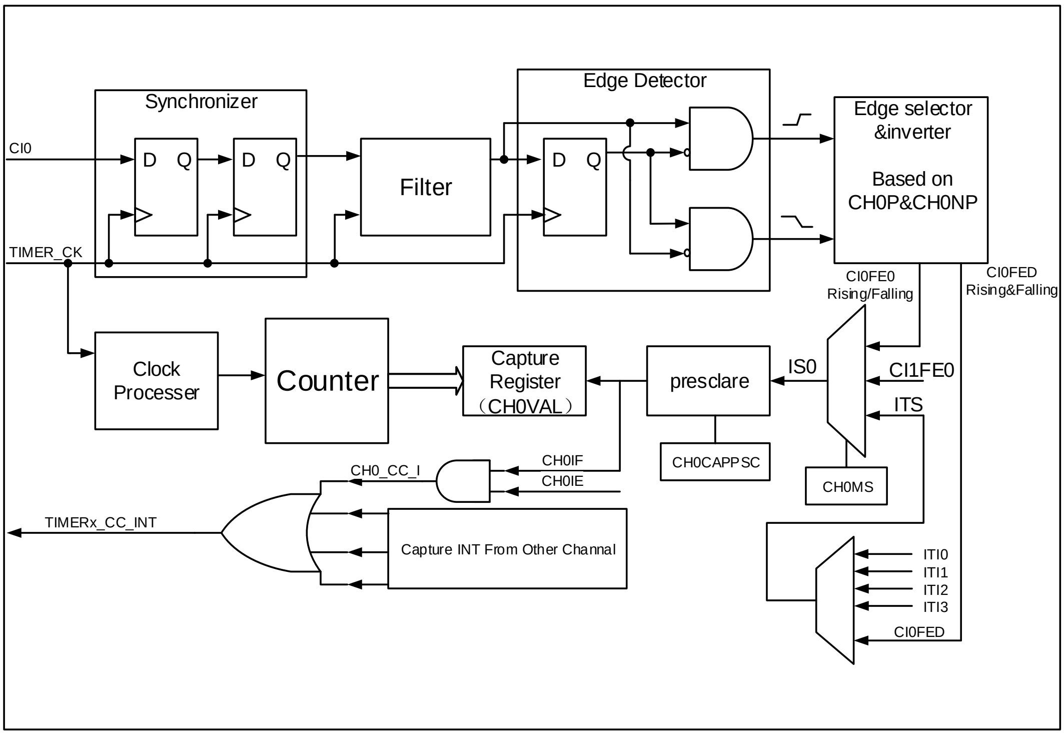
Capture/compare channels

The general level0 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 TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 15-40. Input capture logic



One of channels' input signals (CIx) can be chosen from the TIMERx_CHx signal or the Excusive-OR function of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the channel input signal (CIx) 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 fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

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

Step1: Filter Configuration. (CHxCAPFLT in TIMERx_CHCTL0)

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

Step2: Edge Selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source Selection. (CHxMS in TIMERx_CHCTL0)

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

Step4: Interrupt enable. (CHxIE and CHxDEN in TIMERx_DMAINTEN)

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

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN

Direct generation: If you want to generate a DMA request or interrupt, you can set CHxG by software directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Output compare mode**

In Output Compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. when the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=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 CHxCOMSEN
* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
* Select the active high polarity by CHxP/CHxNP
* Enable the output by CHxEN

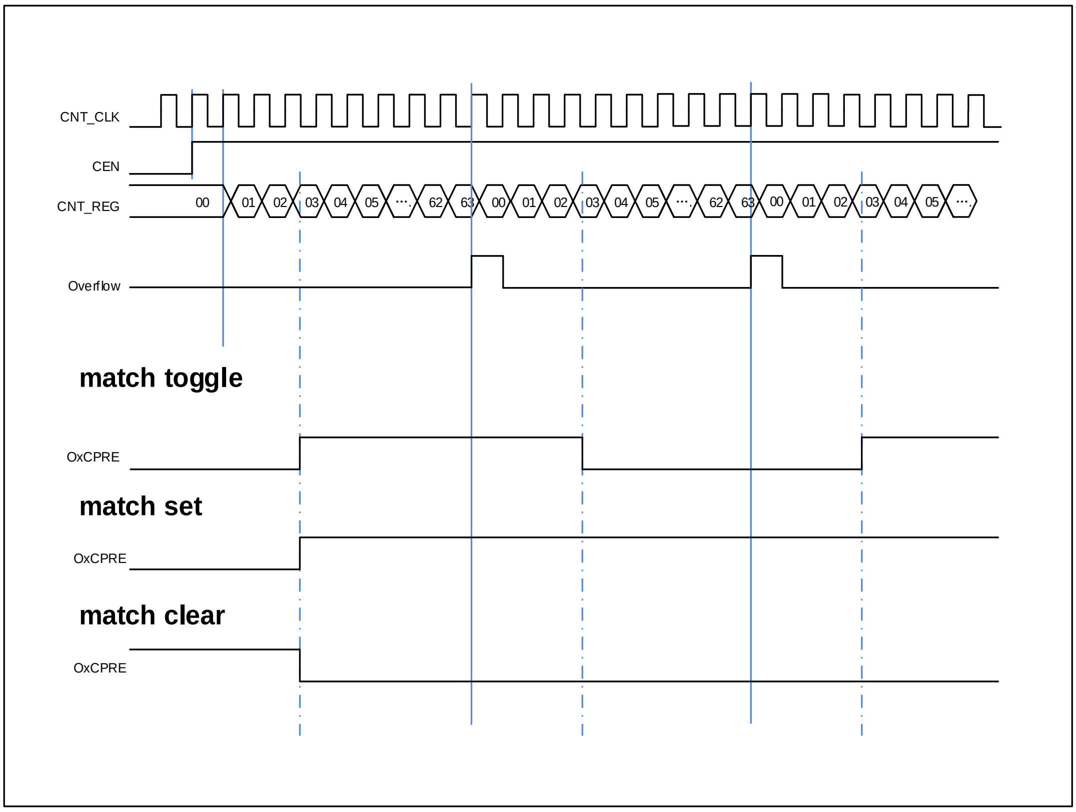
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.
About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-41. Output-compare under three modes



PWM mode

In the output PWM mode (by setting the CHxCOMCTL bits to 3’b110 (PWM mode0) or to 3’b 111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

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

The EAPWM period is determined by TIMERx_CAR and duty cycle is by TIMERx_CHxCV. *Figure 15-42. EAPWM timechart* shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is determined by 2*TIMERx_CHxCV. *Figure 15-43. CAPWM timechart* _Hlk454890020shows the CAPWM output and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3’b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3’b110).

Figure 15-42. EAPWM timechart

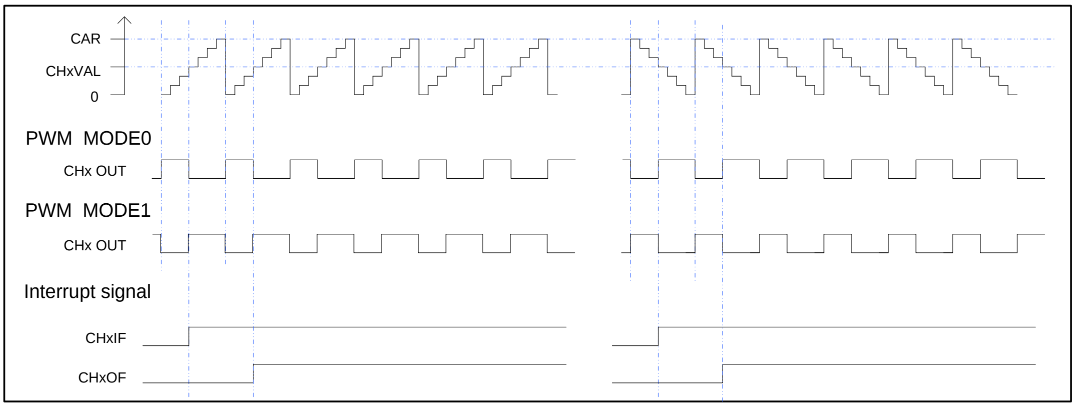
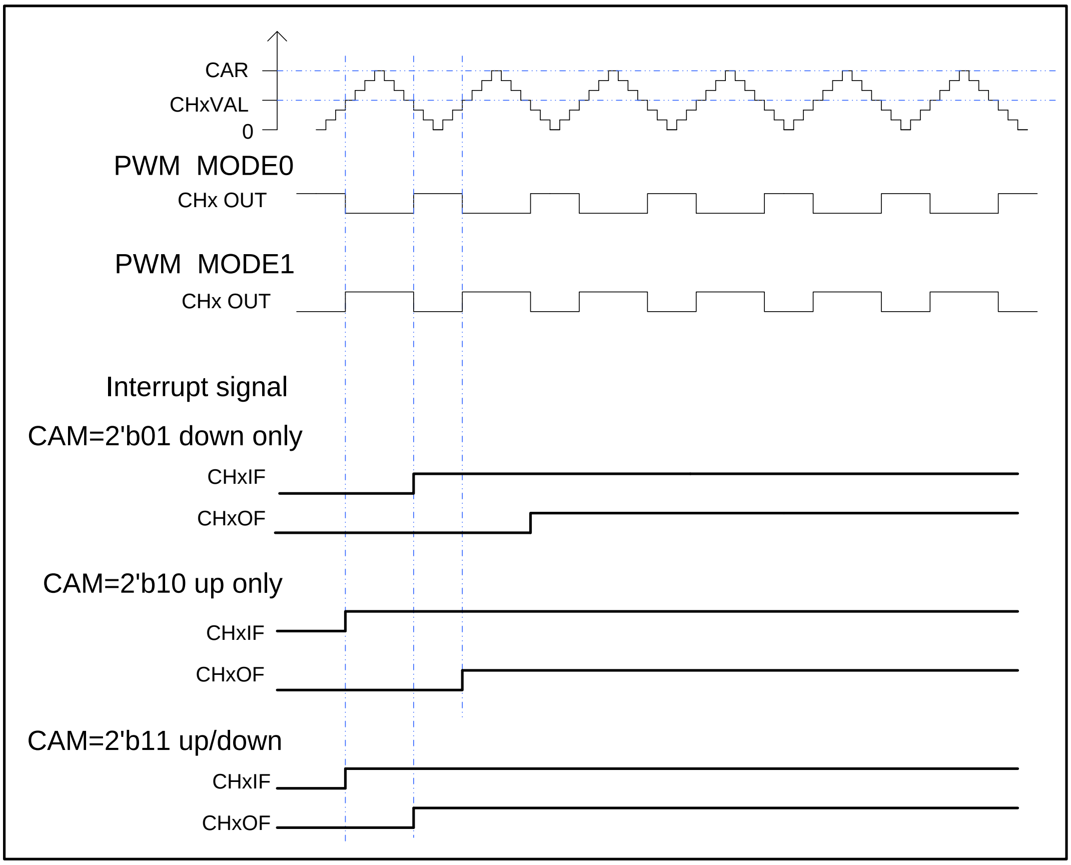


Figure 15-43. CAPWM timechart



Channel output reference signal

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

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL 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 TIMERx_CHxCV 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 CHxCOMCTL 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 TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFP signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMERx_CH0 and TIMERx_CH1 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 CI0 only, CI1 only or both CI0 and CI1, the selection made by setting the SMC [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 TIMERx_CAR register before the counter starts to count.

Table 15-5. Counting direction versus encoder signals

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-
CI1 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
CI0 and CI1 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down
	CI0FE0=Low	X	X	Down	Up

Note:"-" means "no counting"; "X" means impossible.

Figure 15-44. Example of counter operation in encoder interface mode

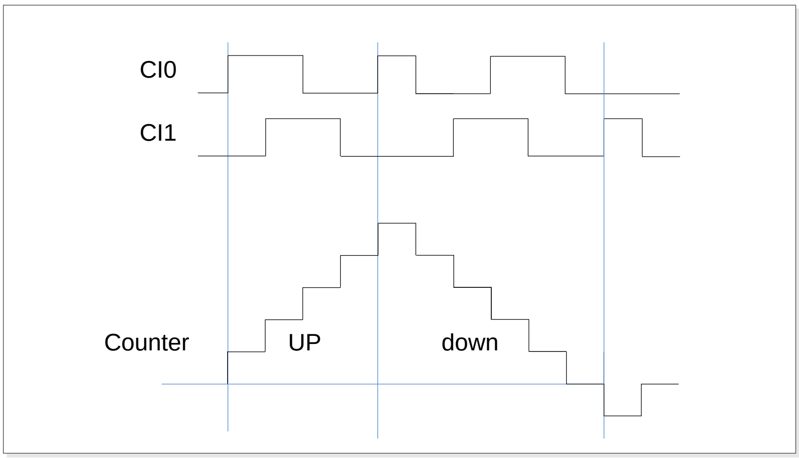
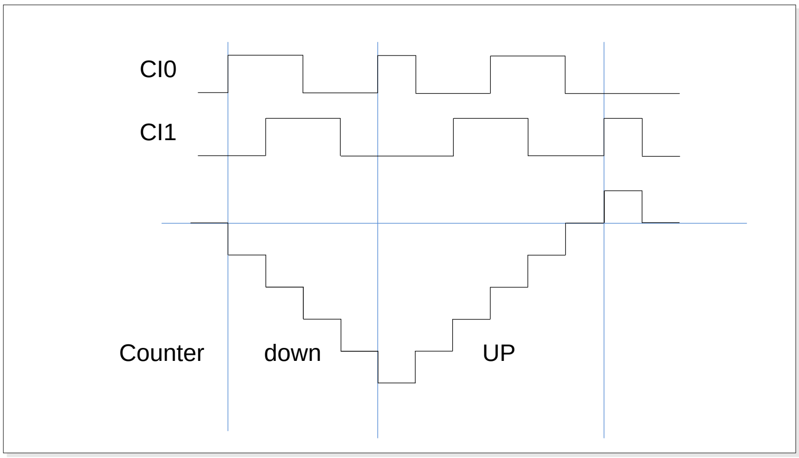


Figure 15-45. Example of encoder interface mode with CI0FE0 polarity inverted

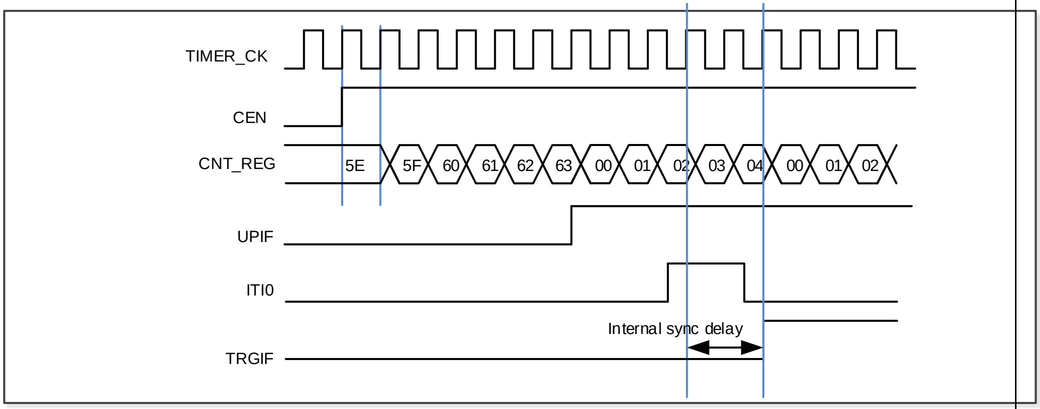
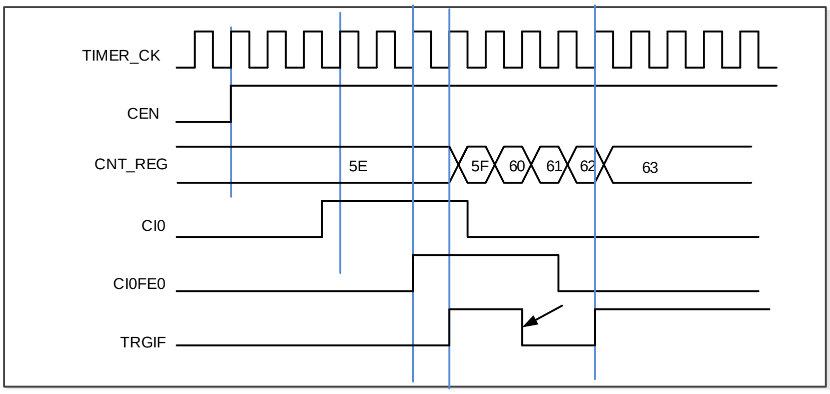


Slave controller

The TIMERx 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 SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

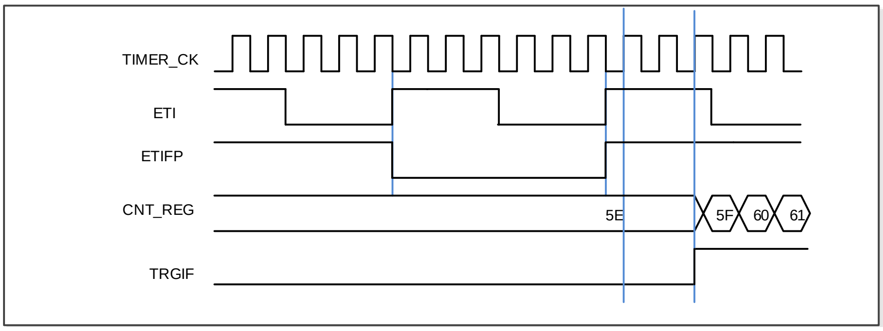
Table 15-6. Slave controller examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
		101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3’b000 ITI0 is the selection.	- For ITI0, no polarity selector can be used.	- For the ITI0, no filter and prescaler can be used.
	Figure 15-46. Restart mode  TIMER_CK CEN CNT_REG 5E 5F 60 61 62 63 00 01 02 03 04 00 01 02 UPIF ITI0 Internal sync delay TRGIF			
Exam2	Pause mode The counter can be paused when the trigger input is low.	TRGS[2:0]=3’b101 CI0FE0 is the selection.	TI0S=0.（Non-xor）[CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.
	Figure 15-47. Pause mode  TIMER_CK CEN CNT_REG 5E 5F 60 61 62 63 CI0 CI0FE0 TRGIF			

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler

Exam3	Event mode
	The counter will start to count when a rising trigger input.
		TRGS[2:0]=3'b111
		ETIF is the selection.
			ETP = 0 no polarity change.	ETPSC = 1, divided by 2.
				ETFC = 0 , no filter

	Figure 15-48. Event mode  TIMER_CK ETI ETIFP CNT_REG 5E 5F 60 61 TRGIF			

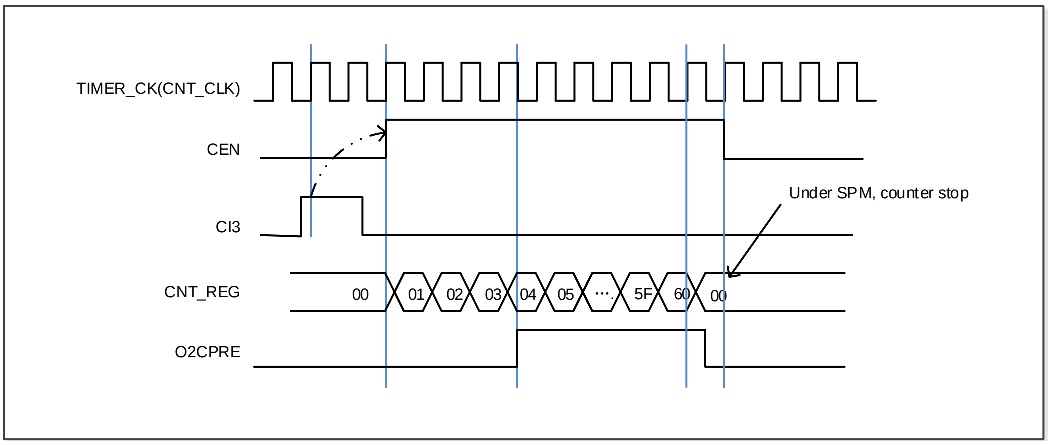
Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

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 TIMERx_CTL0 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 TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 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 CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

Figure 15-49. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60



Timers interconnection

Refer to ***Advanced timer (TIMERx, x=0, 7)***.

Timer DMA mode

Timer’s DMA mode is the function that configures timer’s register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB; Of course, you have to enable a DMA request which will be asserted by some internal interrupt event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer’s DMA request is finished. While if TIMERx_DMATC is not 0, such as 3(4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer’s registers DMASAR+0x4, DMASAR+0x8, DMASAR+0xc at the next 3 accesses to TIMERx_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

Timer debug mode

When the Cortex™-M3 halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

15.2.5. Register definition

Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	SPM	UPS	UPDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV 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	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in

		TIMERx_CHCTL0 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	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	SPM	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	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	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 UPG 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	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.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI0S	MMC[2:0]			DMAS	Reserved		
								rw	rw			rw			

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	TI0S	Channel 0 trigger input selection 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	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 UPG bit in the TIMERx_SWEVG 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 TIMERx_EN 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. 100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO
3	DMAS	DMA request source selection

Bits	Fields	Descriptions
		0: DMA request of channel x is sent when channel x event occurs. 1: DMA request of channel x is sent when update event occurs.
2:0	Reserved	Must be kept at reset value.

Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	SMC1	ETPSC[1:0]		ETFC[3:0]				MSM	TRGS[2:0]			Reserved	SMC[2:0]		
rw	rw	rw		rw				rw	rw				rw		

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	ETP	External trigger polarity This bit specifies the polarity of ETI signal 0: ETI is active at high level or rising edge. 1: ETI is active at low level or falling edge.
14	SMC1	Part of SMC for enable External clock mode1. In external clock mode 1, the counter is clocked by any active edge on the ETIFP signal. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. It is possible to simultaneously use external clock mode 1 with the restart mode, pause mode or event mode. But the TRGS bits must not be 3'b111 in this case. The external clock input will be ETIFP if external clock mode 0 and external clock mode 1 are enabled at the same time. Note: External clock mode 0 enable is in this register's SMC bit-filed.
13:12	ETPSC[1:0]	External trigger prescaler The frequency of external trigger signal ETIFP 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 ETIFP frequency. 00: Prescaler disable 01: ETIFP frequency will be divided by 2 10: ETIFP frequency will be divided by 4

		11: ETIFP frequency will be divided by 8
11:8	ETFC[3:0]	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 ETIFP signal and the length of the digital filter applied to ETIFP. 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	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	TRGS[2:0]	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 (ITI0) 001: Internal trigger input 1 (ITI1) 010: Internal trigger input 2 (ITI2) 011: Internal trigger input 3 (ITI3) 100: CI0 edge flag (CI0F_ED) 101: channel 0 input Filtered output (CI0FE0) 110: channel 1 input Filtered output (CI1FE1) 111: External trigger input filter output(ETIFP) These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.

2:0	SMC[2:0]	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 CI1FE1 edge, while the direction depends on CI0FE0 level. 010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. 011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 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 cannot be disabled by the slave mode controller. 111: External clock mode0. The counter counts on the rising edges of the selected trigger.

DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	Reserved	CH3DEN	CH2DEN	CH1DEN	CH0DEN	UPDEN	Reserved	TRGIE	Reserved	CH3IE	CH2IE	CH1IE	CH0IE	UPIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13	Reserved	Must be kept at reset value.
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable

		0: disabled 1: enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5	Reserved	Must be kept at reset value.
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CH3OF	CH2OF	CH1OF	CH0OF	Reserved		TRGIF	Reserved	CH3IF	CH3IF	CH1IF	CH0IF	UPIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	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	Reserved	Must be kept at reset value.
4	CH3IF	Channel 3 ‘s capture/compare interrupt enable Refer to CH0IF description
3	CH2IF	Channel 2 ‘s capture/compare interrupt enable Refer to CH0IF description
2	CH1IF	Channel 1 ‘s capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 ‘s capture/compare interrupt flag

		This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 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	UPIF	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

Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TRGG	Reserved	CH3G	CH2G	CH1G	CH0G	UPG
									w		w	w	w	w	w

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	Reserved	Must be kept at reset value.
4	CH3G	Channel 3's capture or compare event generation Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation

		This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	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

Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COMCEN	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		CH0COMCEN	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	CH0MS[1:0]	
CH1CAPFLT[3:0]				CH1CAPPSC[1:0]				CH0CAPFLT[3:0]				CH0CAPPSC[1:0]			
rw				rw		rw		rw				rw		rw	

Output compare mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description

9:8	CH1MS[1:0]	Channel 1 mode 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IS1 is connected to CI0FE1 10: Channel 1 is configured as input, IS1 is connected to CI1FE1 11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced low level. 101: Force high. O0CPRE is forced high level. 110: PWM mode0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high. 111: PWM mode1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low. When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).
3	CH0COMSEN	Channel 0 compare output shadow enable

		When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH0COMFEN	Channel 0 output compare 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 CH0_O is set to the compare level independently from the result of the comparison. 0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 10: Channel 0 is configured as input, IS0 is connected to CI1FE0 11: Channel 0 is configured as input, IS0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture 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 CI0 input signal and the length of the digital filter applied to CI0. 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	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 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	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3COMCEN	CH3COMCTL[2:0]			CH3COMSEN	CH3COMFEN	CH3MS[1:0]		CH2COMCEN	CH2COMCTL[2:0]			CH2COMSEN	CH2COMFEN	CH2MS[1:0]	
CH3CAPFLT[3:0]				CH3CAPPSC[1:0]				CH2CAPFLT[3:0]				CH2CAPPSC[1:0]			
rw				rw		rw		rw				rw		rw	

Output compare mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable Refer to CH0COMSEN description
9:8	CH3MS[1:0]	Channel 3 mode 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. (CH3EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 3 is configured as output 01: Channel 3 is configured as input, IS3 is connected to CI2FE3 10: Channel 3 is configured as input, IS3 is connected to CI3FE3 11: Channel 3 is configured as input, IS3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, the O2CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 2 output compare clear disable 1: Channel 2 output compare clear enable
6:4	CH2COMCTL[2:0]	Channel 2 compare output control This bit-field controls the behavior of the output reference signal O2CPRE which drives CH2_O and CH2_ON. O2CPRE is active high, while CH2_O and CH2_ON active level depends on CH2P and CH2NP bits. 000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT. 001: Set the channel output. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter matches the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx_CH2CV.

		100: Force low. O2CPRE is forced low level. 101: Force high. O2CPRE is forced high level. 110: PWM mode 0. When counting up, O2CPRE is high as long as the counter is smaller than TIMERx_CH2CV else low. When counting down, O2CPRE is low as long as the counter is larger than TIMERx_CH2CV else high. 111: PWM mode 1. When counting up, O2CPRE is low as long as the counter is smaller than TIMERx_CH2CV else high. When counting down, O2CPRE is high as long as the counter is larger than TIMERx_CH2CV else low. When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE).
3	CH2COMSEN	Channel 2 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled. 0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH2COMFEN	Channel 2 output compare 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 CH2_O is set to the compare level independently from the result of the comparison. 0: Channel 2 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH2_O output is 5 clock cycles. 1: Channel 2 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH2_O output is 3 clock cycles.
1:0	CH2MS[1:0]	Channel 2 I/O mode 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. (CH2EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 2 is configured as output 01: Channel 2 is configured as input, IS2 is connected to CI2FE2 10: Channel 2 is configured as input, IS2 is connected to CI3FE2 11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG

register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture 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 CI2 input signal and the length of the digital filter applied to CI2. 0000: Filter disable, 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	CH2CAPPSC[1:0]	Channel 2 input capture prescaler This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 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	CH2MS[1:0]	Channel 2 mode selection

Same as output compare mode

Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH3P	CH3EN	Reserved		CH2P	CH2EN	Reserved		CH1P	CH1EN	Reserved		CH0P	CH0EN
		rw	rw			rw	rw			rw	rw			rw	rw

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	CH3P	Channel 3 capture/compare function polarity Refer to CH0P description
12	CH3EN	Channel 3 capture/compare function enable Refer to CH0EN description
11:10	Reserved	Must be kept at reset value
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable Refer to CH0EN description
7:6	Reserved	Must be kept at reset value
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3:2	Reserved	Must be kept at reset value
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal

		polarity. 0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted. 1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is inverted.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 0 disabled 1: Channel 0 enabled

Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	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.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CARL[15:0]	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.

Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH0VAL[15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 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 0 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.

Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CH1VAL[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH1VAL[15:0]	Capture or compare value of channel1 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.

Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH2VAL[15:0]	Capture or compare value of channel 2 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.

Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 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.

DMA configuration register (TIMERx_DMACFG)

Address offset: 0x48
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed is defined the number of DMA will access(R/W) the register of TIMERx_DMATB 5’b0_0000: 1 time transfer 5’b0_0001: 2 times transfer … 5’b1_0001: 18 times transfer
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4. 5’b0_0000: TIMERx_CTL0 5’b0_0001: TIMERx_CTL1 … 5’b1_0010: TIMERx_DMACFG In a word: Start Address = TIMERx_CTL0 + DMASAR*4

DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

15.3. General level1 timer (TIMERx, x=8, 11)

15.3.1. Overview

The general level1 timer module (Timer8, 11) is a two-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level1 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level1 timers can be programmed and be used to count or time external events that drive other Timers.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

The general level1 timer module (Timer8, 11) is available only in the GD32F10x_XD devices.

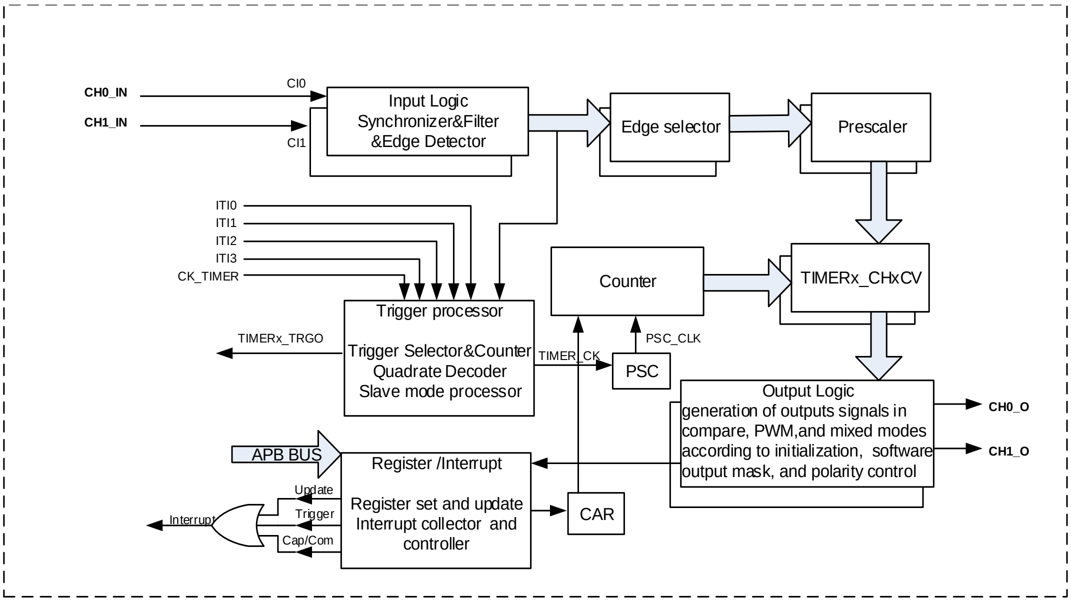
15.3.2. Characteristics

- Total channel num: 2.
- Counter width: 16bit.
- Source of count clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Programmable prescaler: 16 bit. 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 on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

15.3.3. Block diagram

Figure 15-50. General level1 timer block diagram provides details on the internal configuration of the general level1 timer.

Figure 15-50. General level1 timer block diagram



15.3.4. Function overview

Clock selection

The general level1 TIMER has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

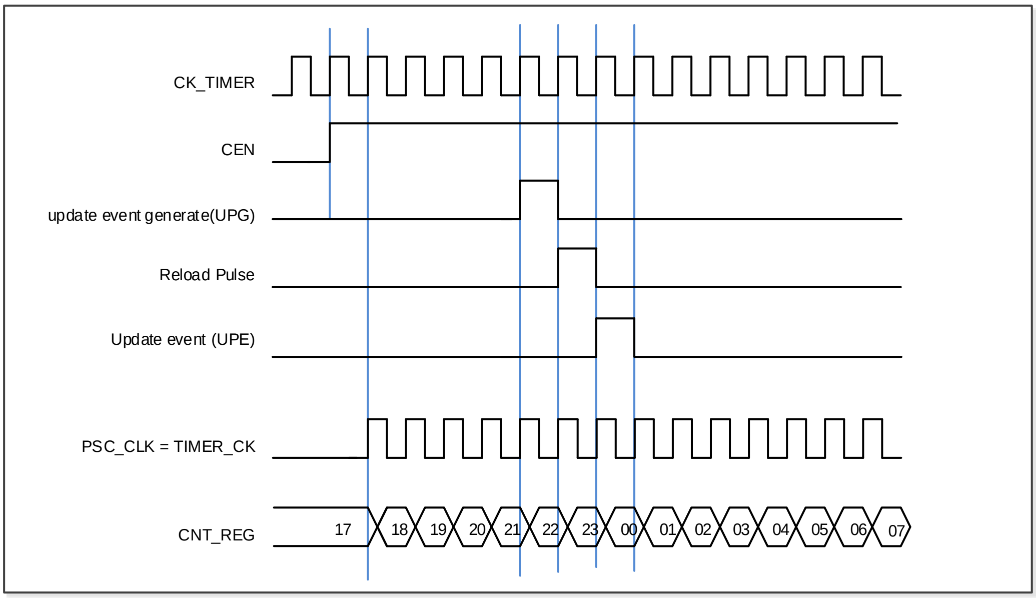
- SMC [2:0] == 3’b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the slave mode is disabled (SMC [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, driven counter’s prescaler to count, is equal to CK_TIMER which is from RCU.

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

Figure 15-51. Normal mode, internal clock divided by 1



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin source

The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CI0/TIMERx_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

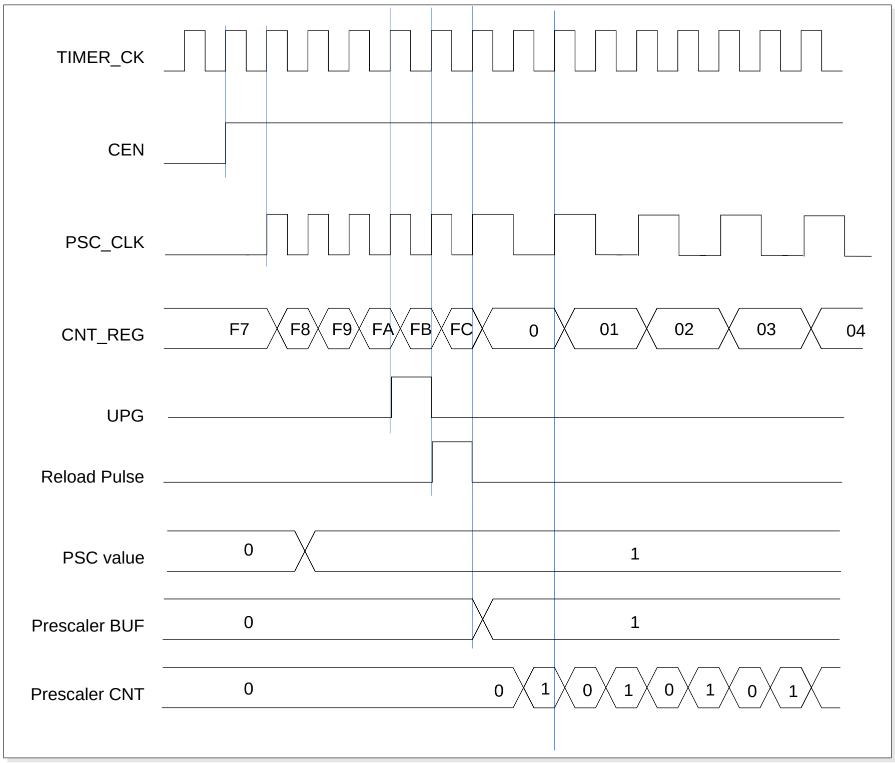
- SMC1== 1'b1 (external clock mode 1). External input pin source (ETI)

The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the clock source is selected to come from the ETI signal, the Trigger Controller including the edge detection circuitry will generate a clock pulse during each ETI signal rising edge to clock the counter prescaler.

Prescaler

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

Figure 15-52. Counter timing diagram with prescaler division change from 1 to 2



Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, 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 TIMERx_CAR=0x63.

Figure 15-53. Up-counter timechart, PSC=0/1

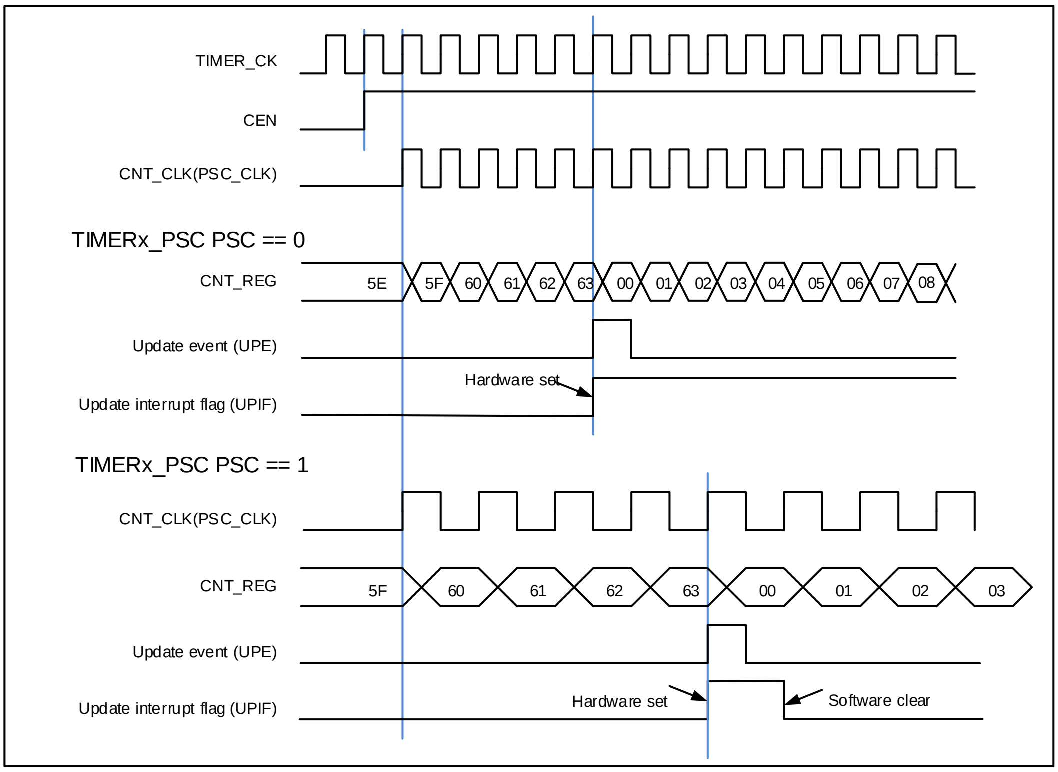
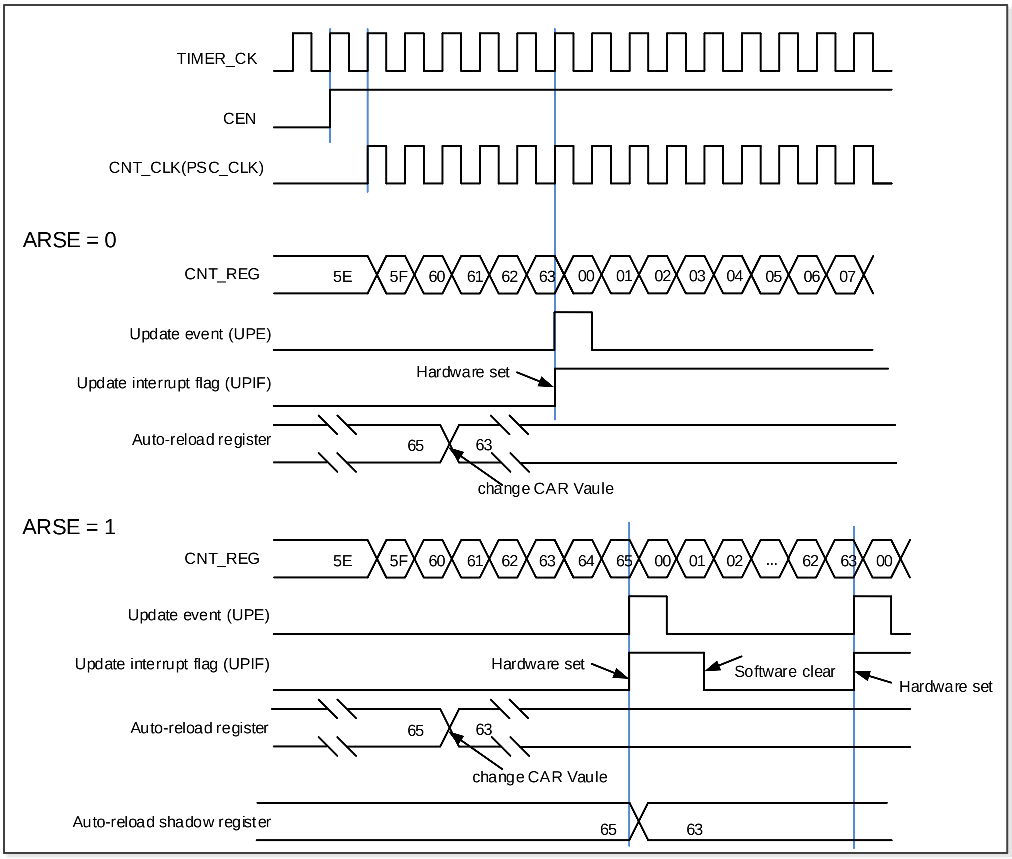


Figure 15-54. Up-counter timechart, change TIMERx_CAR on the go.



Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR 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 (TIMERx_CREP+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down-counting mode.

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

If set the UPDIS bit in TIMERx_CTL0 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 TIMERx_CAR=0x63.

Figure 15-55. Down-counter timechart, PSC=0/1

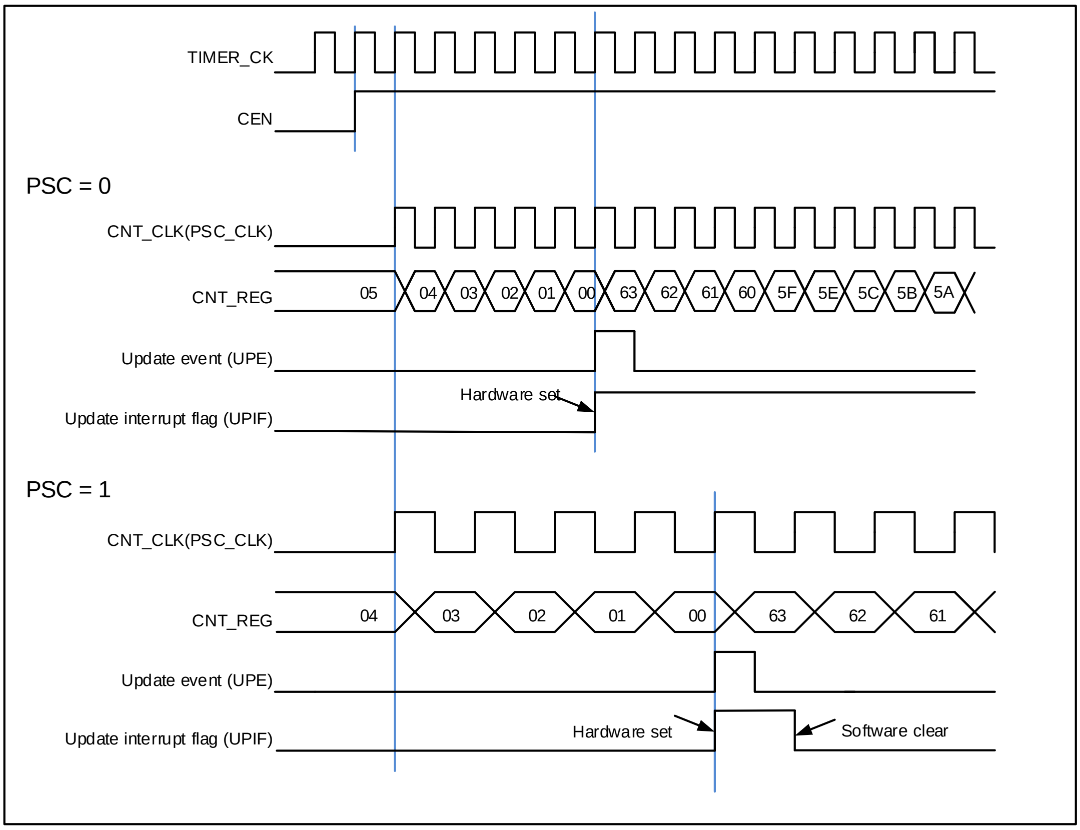
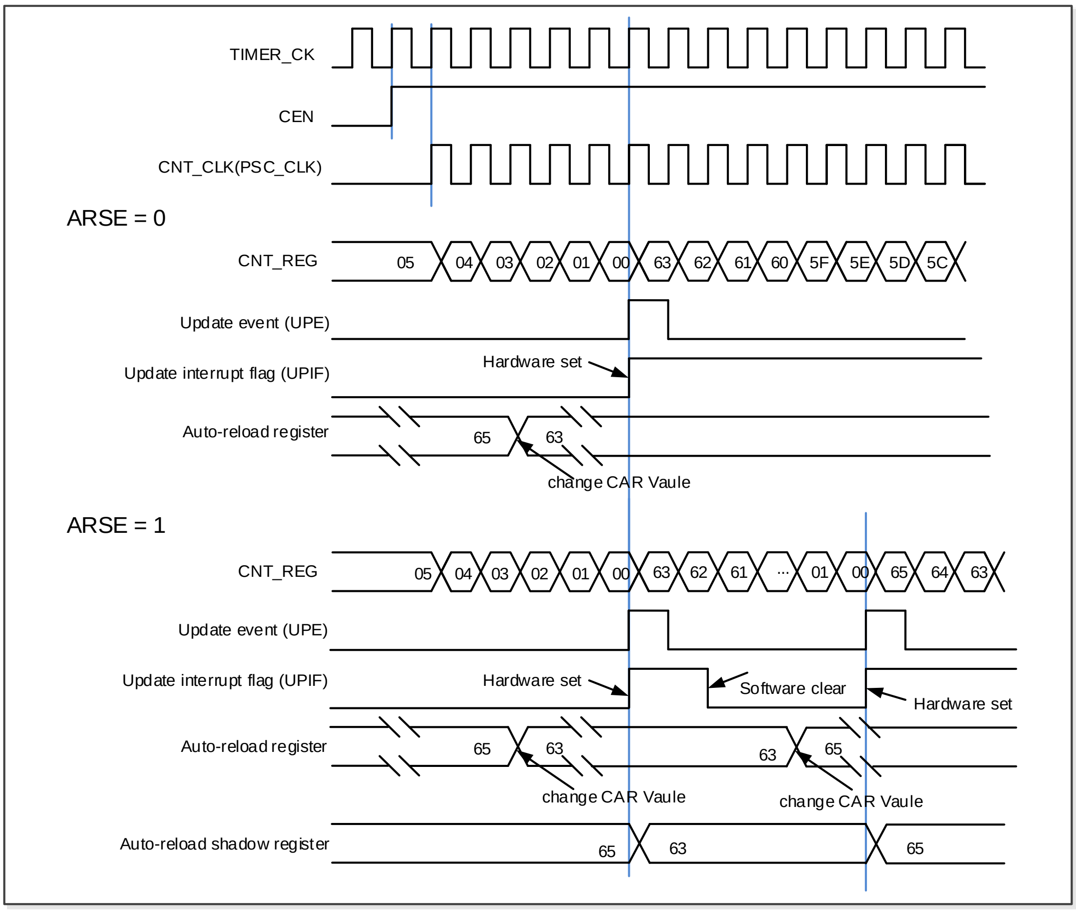


Figure 15-56. Down-counter timechart, change TIMERx_CAR on the go



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 TIMERx_CTL0 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 UPG bit in the TIMERx_SWEVG 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 UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 15-57. Center-aligned counter timechart***.

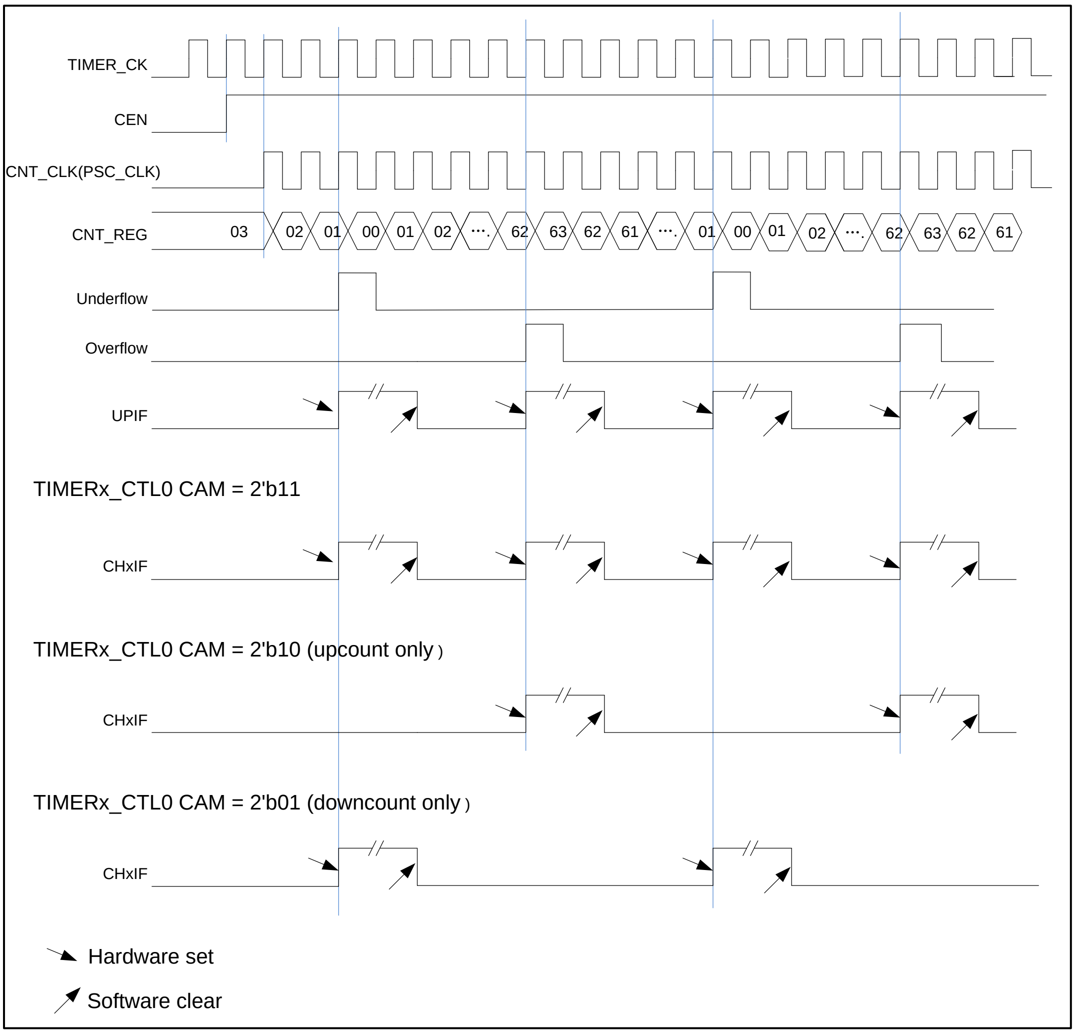
If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register,

prescaler register) are updated.

Figure 15-57. Center-aligned counter timechart show some examples of the counter behavior when TIMERx_CAR=0x63. TIMERx_PSC=0x0

Figure 15-57. Center-aligned counter timechart



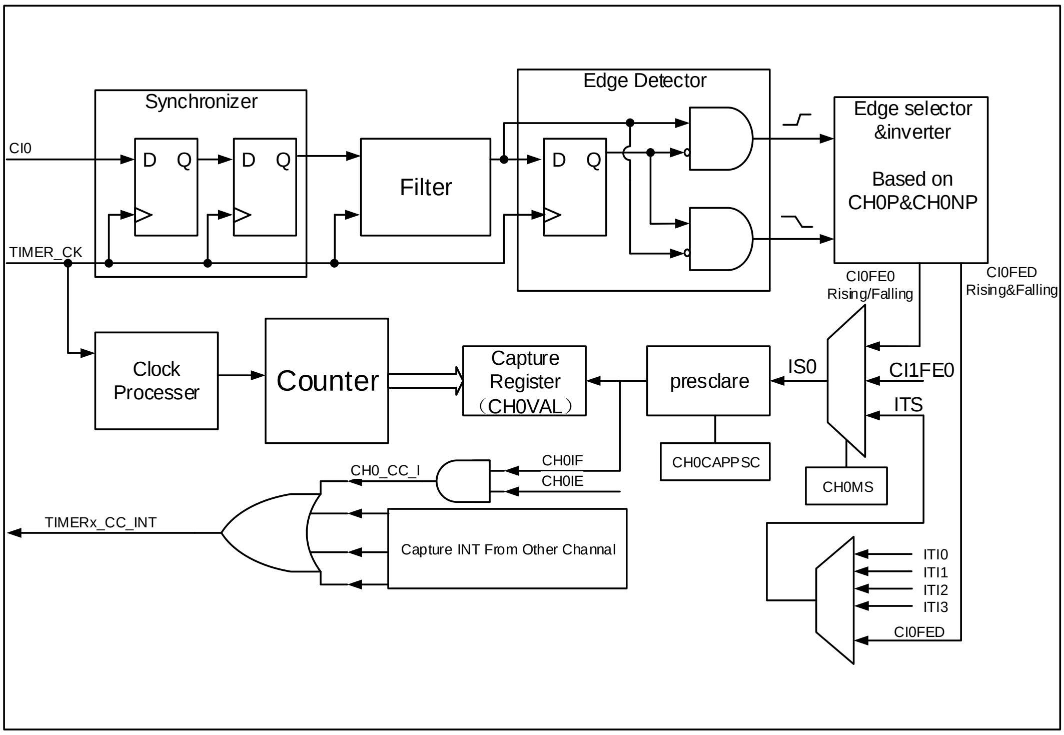
Capture/compare channels

The general level1 timer has two 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 TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 15-58. Input capture logic



First, the channel input signal (CIx) 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 fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

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

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

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

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

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

Step4: Interrupt enable. (CHxIE and CHxDEN in TIMERx_DMAINTEN)

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

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN

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

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Output compare mode**

In Output Compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. when the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=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 CHxCOMSEN

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

* Select the active high polarity by CHxP/CHxNP

* Enable the output by CHxEN

Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

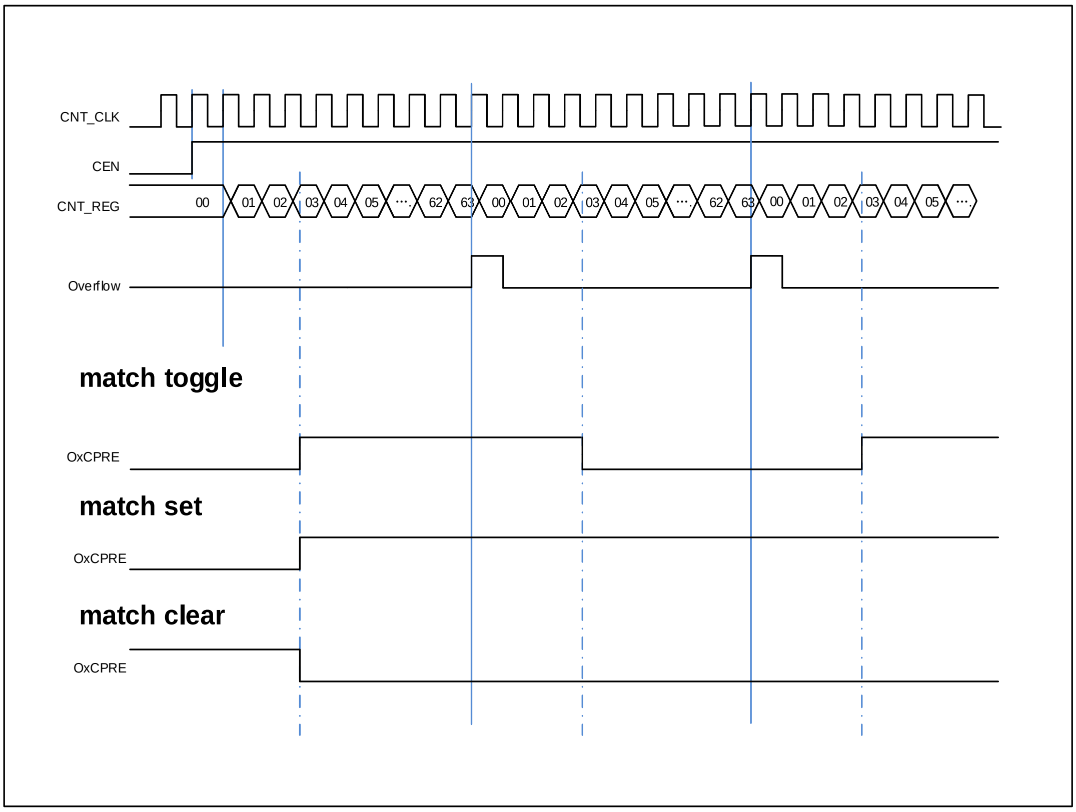
Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-59. Output-compare under three modes



PWM mode

In the output PWM mode (by setting the CHxCOMCTL bits to 3’b110 (PWM mode0) or to 3’b 111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

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

The EAPWM period is determined by TIMERx_CAR and duty cycle is by TIMERx_CHxCV. ***Figure 15-60. EAPWM timechart*** shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is determined by 2*TIMERx_CHxCV. ***Figure 15-61. CAPWM timechart*** shows the CAPWM output and interrupt waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3’b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3’b110).

Figure 15-60. EAPWM timechart

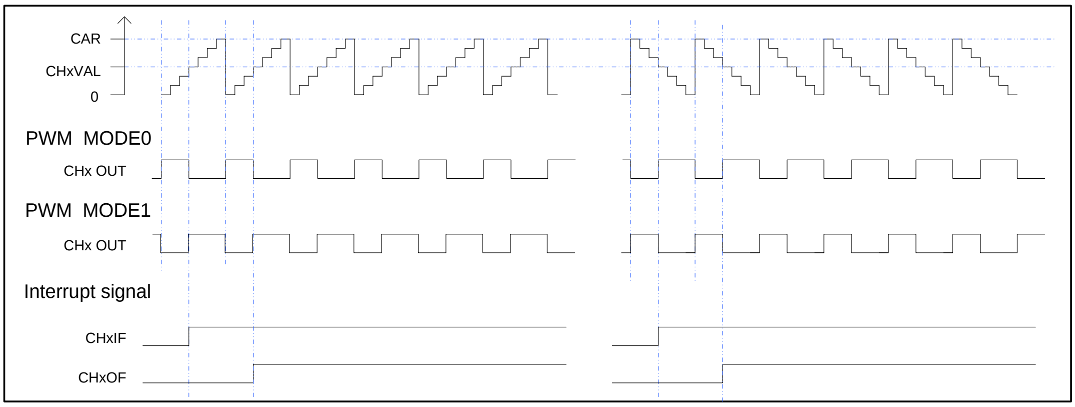
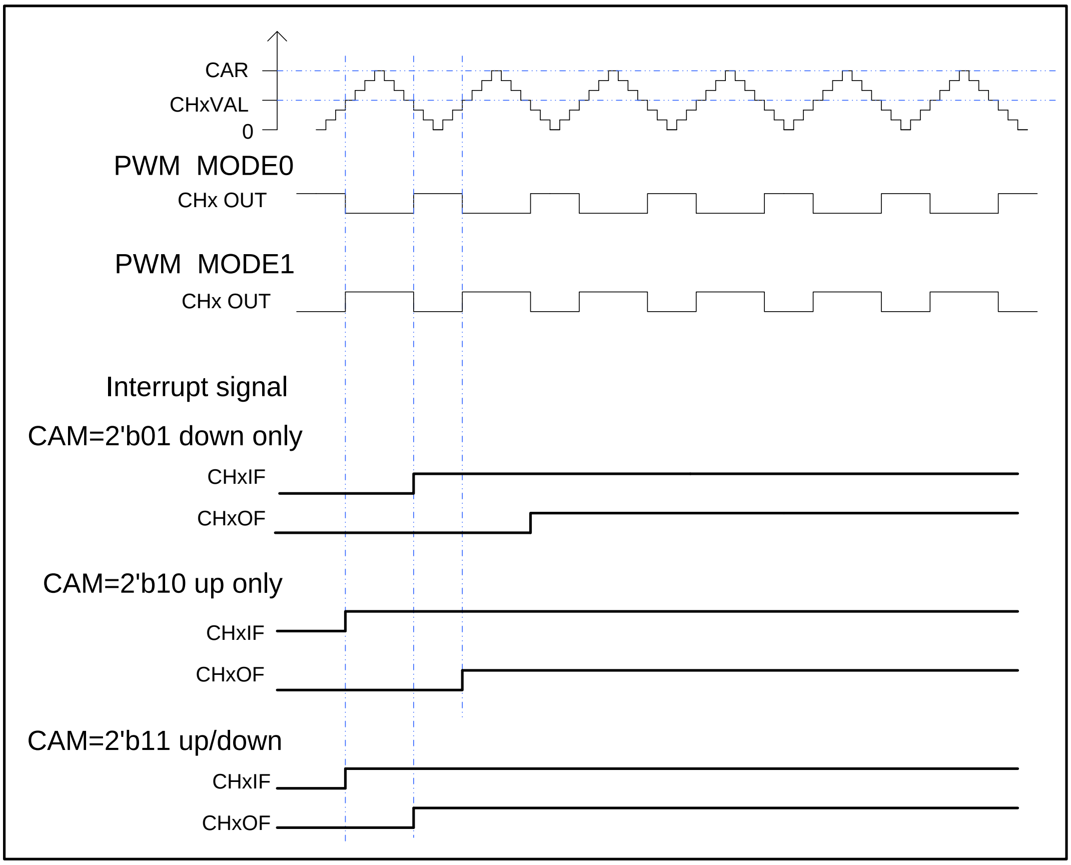


Figure 15-61. CAPWM timechart



Channel output reference signal

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

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL 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 TIMERx_CHxCV 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 CHxCOMCTL 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 TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFP signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Slave controller

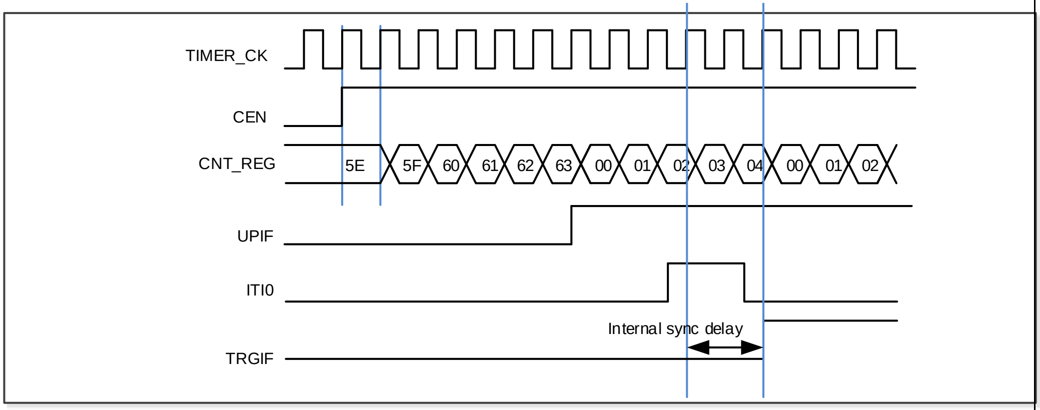
The TIMERx can be synchronized with a trigger in several modes including the Restart mode, the Pause mode and the Event mode which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

Table 15-7. Slave controller examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3’b000 ITI0 is the selection.	- For ITI0, no polarity selector can be used.	- For the ITI0, no filter and prescaler can be used.

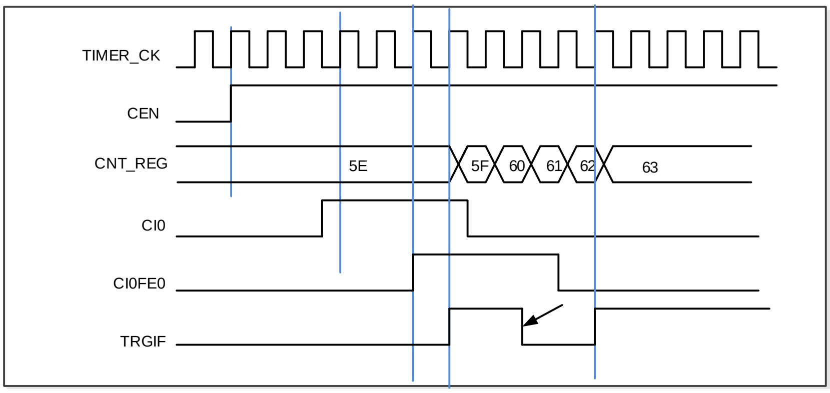
<table>
<tr><th></th><th>Mode Selection</th><th>Source Selection</th><th>Polarity Selection</th><th>Filter and Prescaler</th></tr>
<tr><td></td><td colspan="4">

Figure 15-62. Restart mode

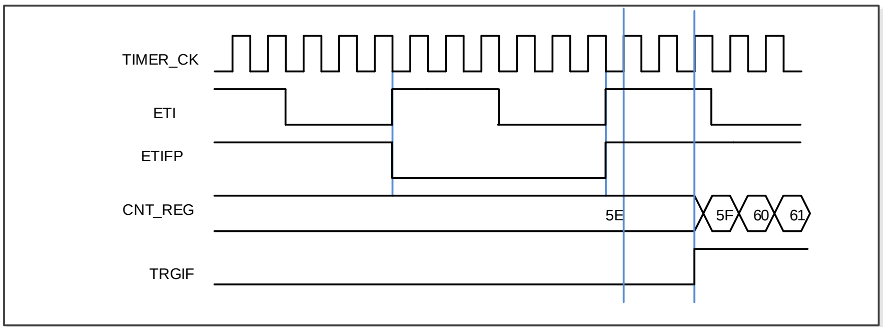


</td></tr>
<tr><td><b>Exam2</b></td><td>Pause mode
The counter can be paused when the trigger input is low.</td><td>TRGS[2:0]=3’b101
CI0FE0 is the selection.</td><td>TI0S=0.（Non-xor）[CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.</td><td>Filter is bypass in this example.</td></tr>
<tr><td></td><td colspan="4">

Figure 15-63. Pause mode



</td></tr>
<tr><td><b>Exam3</b></td><td>Event mode
The counter will start to count when a rising trigger input.</td><td>TRGS[2:0]=3’b111
ETIF is the selection.</td><td>ETP = 0 no polarity change.</td><td>ETPSC = 1, divided by 2.
ETFC = 0 , no filter</td></tr>
</table>

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	Figure 15-64. Event mode  TIMER_CK ETI ETIFP CNT_REG 5E 5F 60 61 TRGIF			

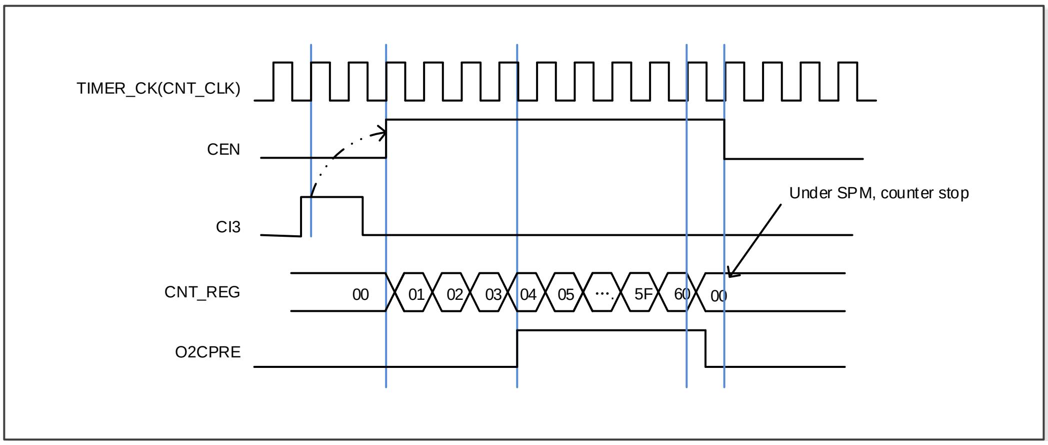
Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

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 TIMERx_CTL0 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 TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 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 CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

Figure 15-65. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60



Timers interconnection

Refer to ***Advanced timer (TIMERx, x=0, 7)***.

Timer debug mode

When the Cortex™-M3 halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

15.3.5. Register definition

Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	SPM	UPS	UPDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV 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	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in

		TIMERx_CHCTL0 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	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	SPM	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	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	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 UPG 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	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.

Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MSM	TRGS[2:0]			Reserved	SMC[2:0]		
								rw	rw				rw		

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	MSM	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	TRGS[2:0]	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 (ITI0) 001: Internal trigger input 1 (ITI1) 010: Internal trigger input 2 (ITI2) 011: Internal trigger input 3 (ITI3) 100: CI0 edge flag (CI0F_ED) 101: channel 0 input Filtered output (CI0FE0) 110: channel 1 input Filtered output (CI1FE1) 111: External trigger input filter output(ETIFP) These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	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 CI1FE1 edge, while the direction depends on CI0FE0 level. 010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. 011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 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 cannot be disabled by the slave mode controller.

111: External clock mode0. The counter counts on the rising edges of the selected trigger.

Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TRGIE	Reserved			CH1IE	CH0IE	UPIE
									rw				rw	rw	rw

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5:3	Reserved	Must be kept at reset value.
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CH1OF	CH0OF	Reserved		TRGIF	Reserved			CH1IF	CH0IF	UPIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	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.
5:3	Reserved	Must be kept at reset value.
2	CH1IF	Channel 1 ‘s capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 ‘s capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 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	UPIF	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

Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TRGG	Reserved.			CH1G	CH0G	UPG
									w				w	w	w

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5:3	Reserved	Must be kept at reset value.
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event

1: Generate an update event

Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COM CEN	CH1COMCTL[2:0]			CH1COM SEN	CH1COM FEN	CH1MS[1:0]		CH0COM CEN	CH0COMCTL[2:0]			CH0COM SEN	CH0COM FEN	CH0MS[1:0]	
CH1CAPFLT[3:0]				CH1CAPPSC[1:0]				CH0CAPFLT[3:0]				CH0CAPPSC[1:0]			
rw				rw		rw		rw				rw		rw	

Output compare mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description
9:8	CH1MS[1:0]	Channel 1 mode 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IS1 is connected to CI0FE1 10: Channel 1 is configured as input, IS1 is connected to CI1FE1 11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.

		0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced low level. 101: Force high. O0CPRE is forced high level. 110: PWM mode0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high. 111: PWM mode1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low. When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).
3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH0COMFEN	Channel 0 output compare 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 CH0_O is set to the compare level independently from the

		result of the comparison. 0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 10: Channel 0 is configured as input, IS0 is connected to CI1FE0 11: Channel 0 is configured as input, IS0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture 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 CI0 input signal and the length of the digital filter applied to CI0. 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	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 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	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CH1P	CH1EN	Reserved		CH0P	CH0EN
										rw	rw			rw	rw

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH1EN description
3:2	Reserved	Must be kept at reset value
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low

		When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. 0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted. 1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is inverted.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 0 disabled 1: Channel 0 enabled

Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

PSC[15:0]
rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	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.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CARL[15:0]	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.

Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

CH0VAL[15:0]
rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 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 0 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.

Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH1VAL[15:0]	Capture or compare value of channel1 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.

15.4. General level2 timer (TIMERx, x=9, 10, 12, 13)

15.4.1. Overview

The general level2 timer module (Timer9, 10, 12, 13) is a one-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level2 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level2 timers can be programmed and be used to count or time external events that drive other Timers.

The general level2 timer module (Timer9, 10, 12, 13) is available only in the GD32F10x_XD devices.

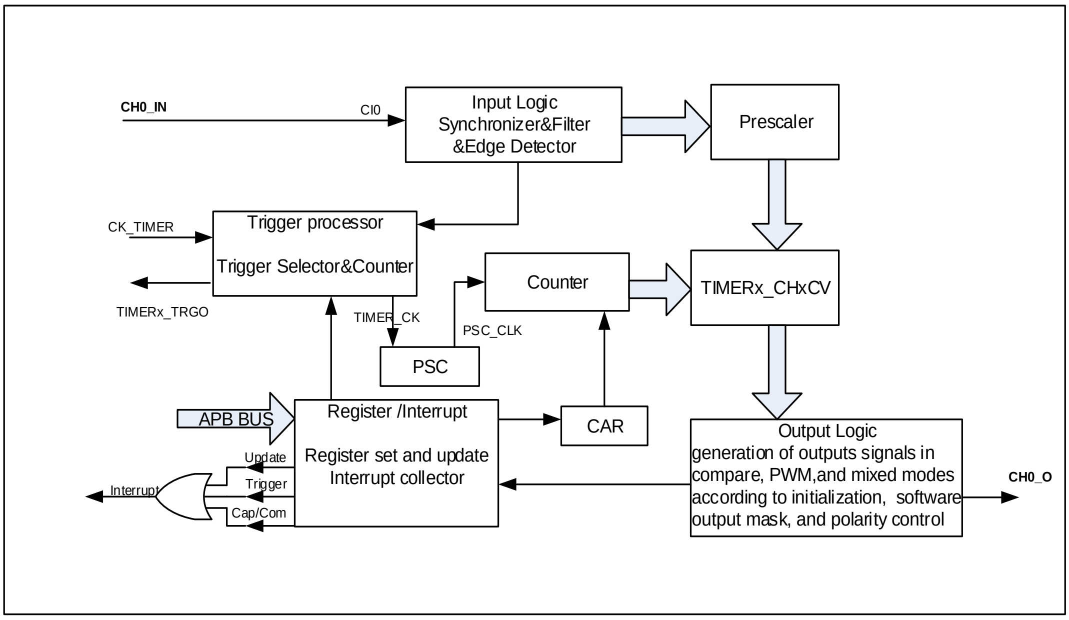
15.4.2. Characteristics

- Total channel num: 1.
- Counter width: 16bit.
- Source of count clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:
  Input capture mode, output compare mode and programmable PWM mode.
- Auto-reload function.
- Interrupt output on: update, trigger event, and compare/capture event.

15.4.3. Block diagram

Figure 15-66. General level2 timer block diagram provides details on the internal configuration of the general level2 timer.

Figure 15-66. General level2 timer block diagram



15.4.4. Function overview

Clock selection

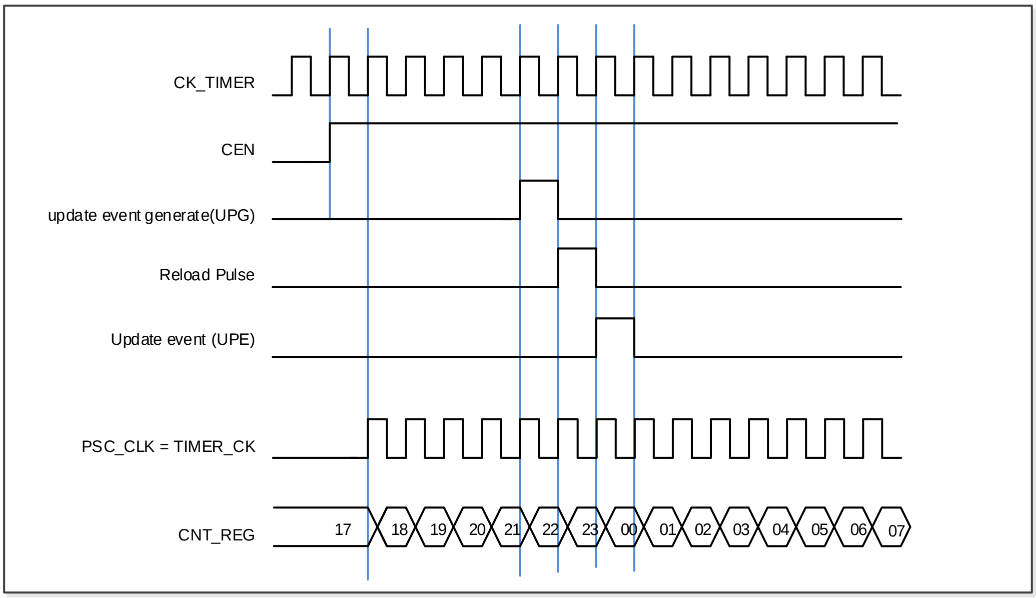
The general level2 TIMER can only being clocked by the CK_TIMER.

- Internal timer clock CK_TIMER which is from module RCU

The general level2 TIMER has only one clock source which is the internal CK_TIMER, used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

The TIMER_CK, driven counter’s prescaler to count, is equal to CK_TIMER which is from RCU

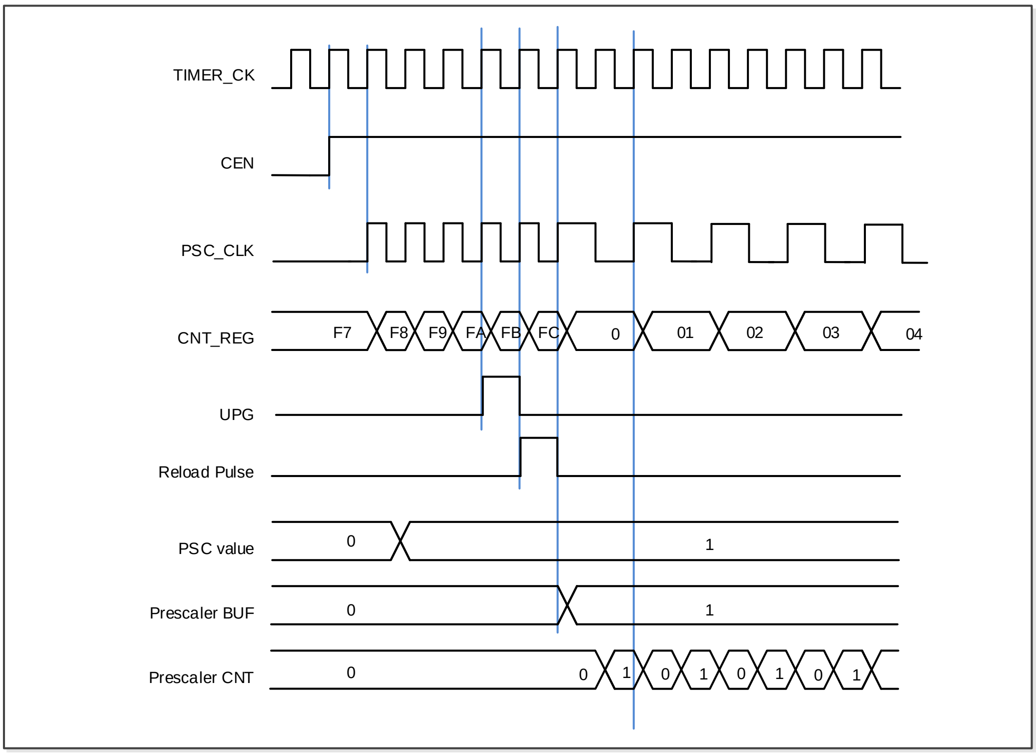
Figure 15-67. Normal mode, internal clock divided by 1



Prescaler

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

Figure 15-68. Counter timing diagram with prescaler division change from 1 to 2



Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, 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 TIMERx_CAR=0x63.

Figure 15-69. Up-counter timechart, PSC=0/1

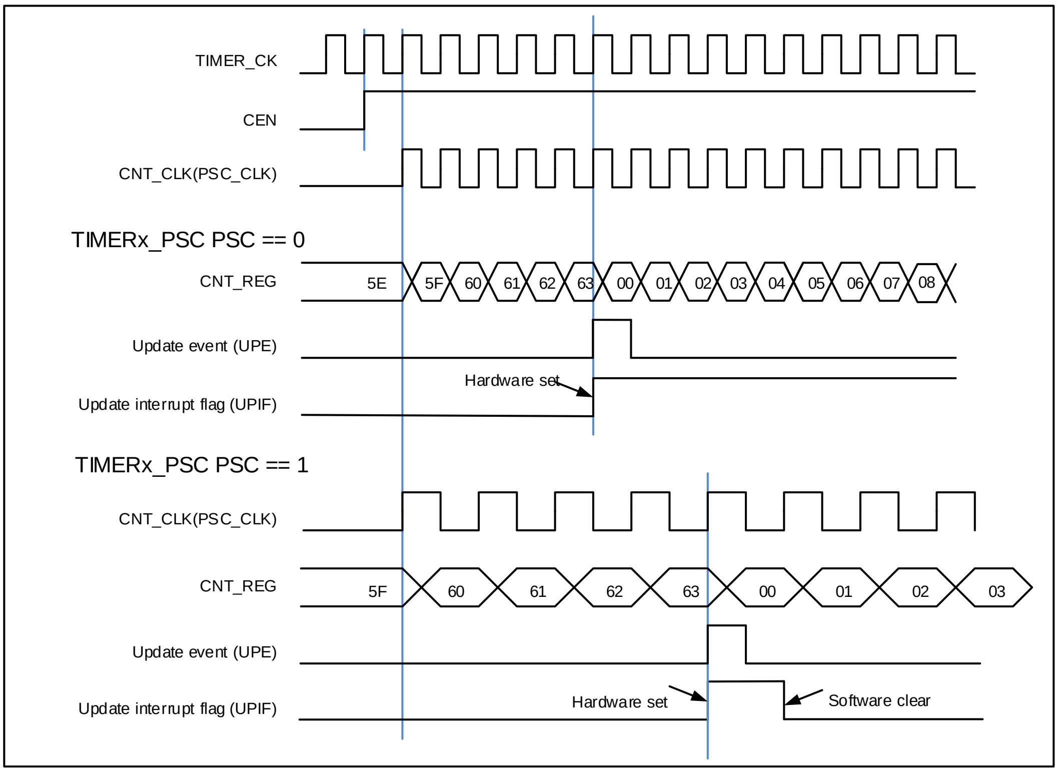
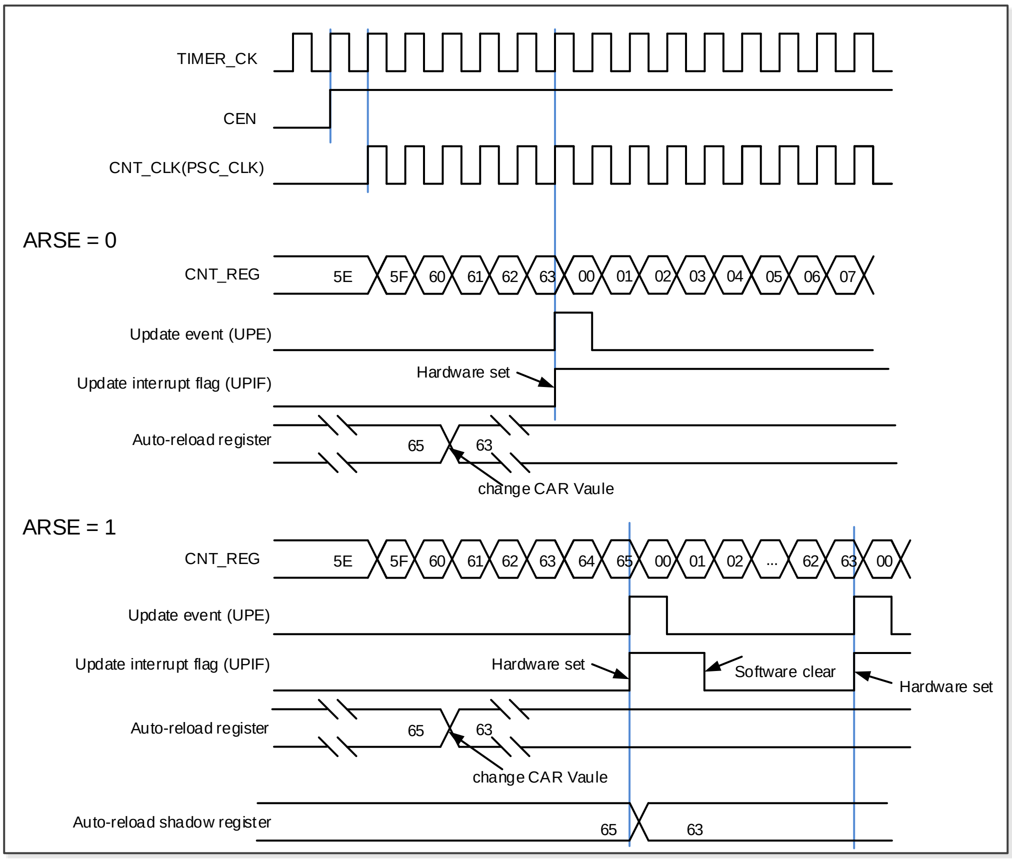


Figure 15-70. Up-counter timechart, change TIMERx_CAR on the go



Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR 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 (TIMERx_CREP+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down-counting mode.

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

If set the UPDIS bit in TIMERx_CTL0 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 TIMERx_CAR=0x63.

Figure 15-71. Down-counter timechart, PSC=0/1

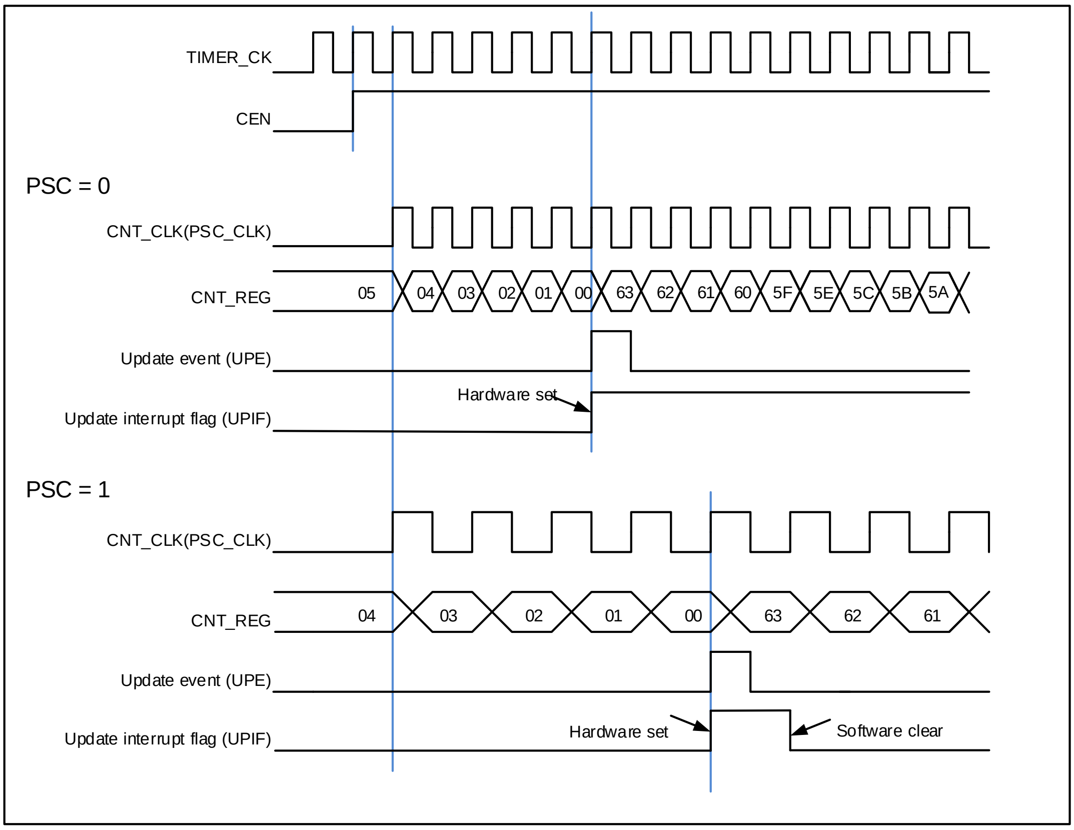
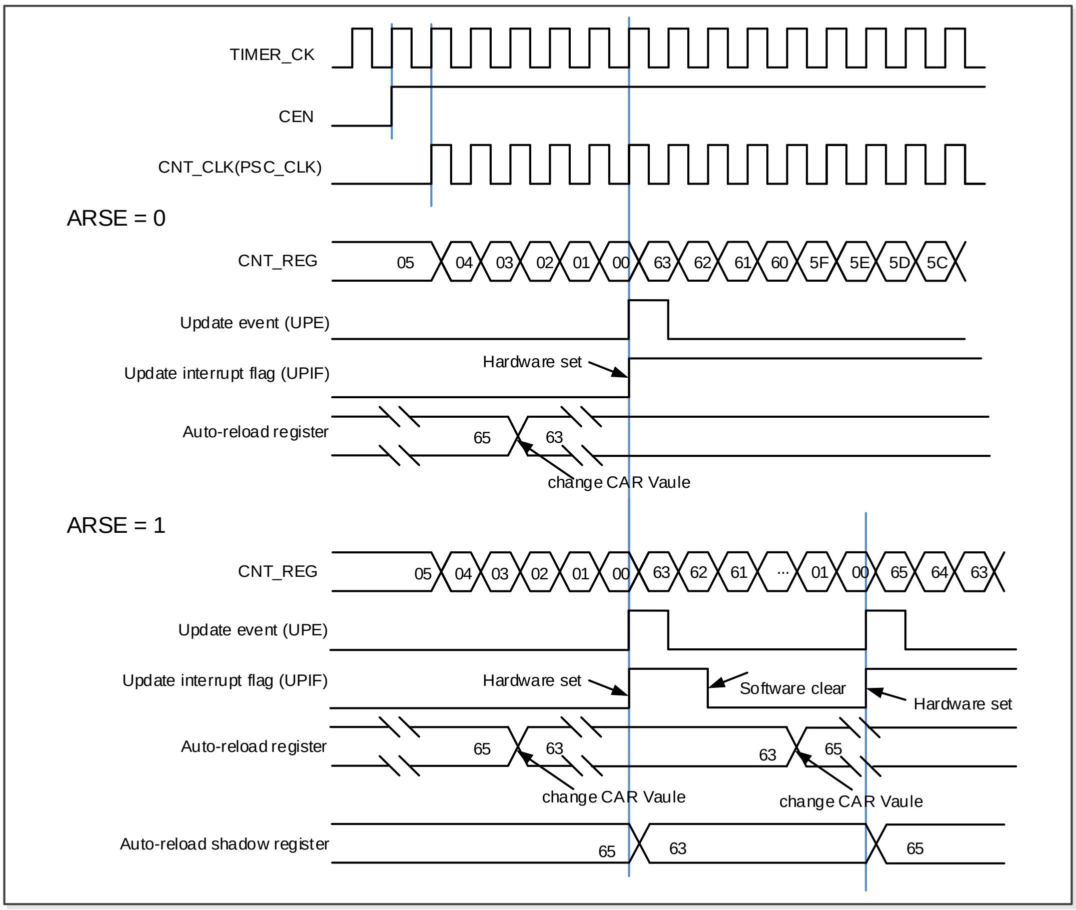


Figure 15-72. Down-counter timechart, change TIMERx_CAR on the go



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 TIMERx_CTL0 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 UPG bit in the TIMERx_SWEVG 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 UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 15-73. Center-aligned counter timechart***.

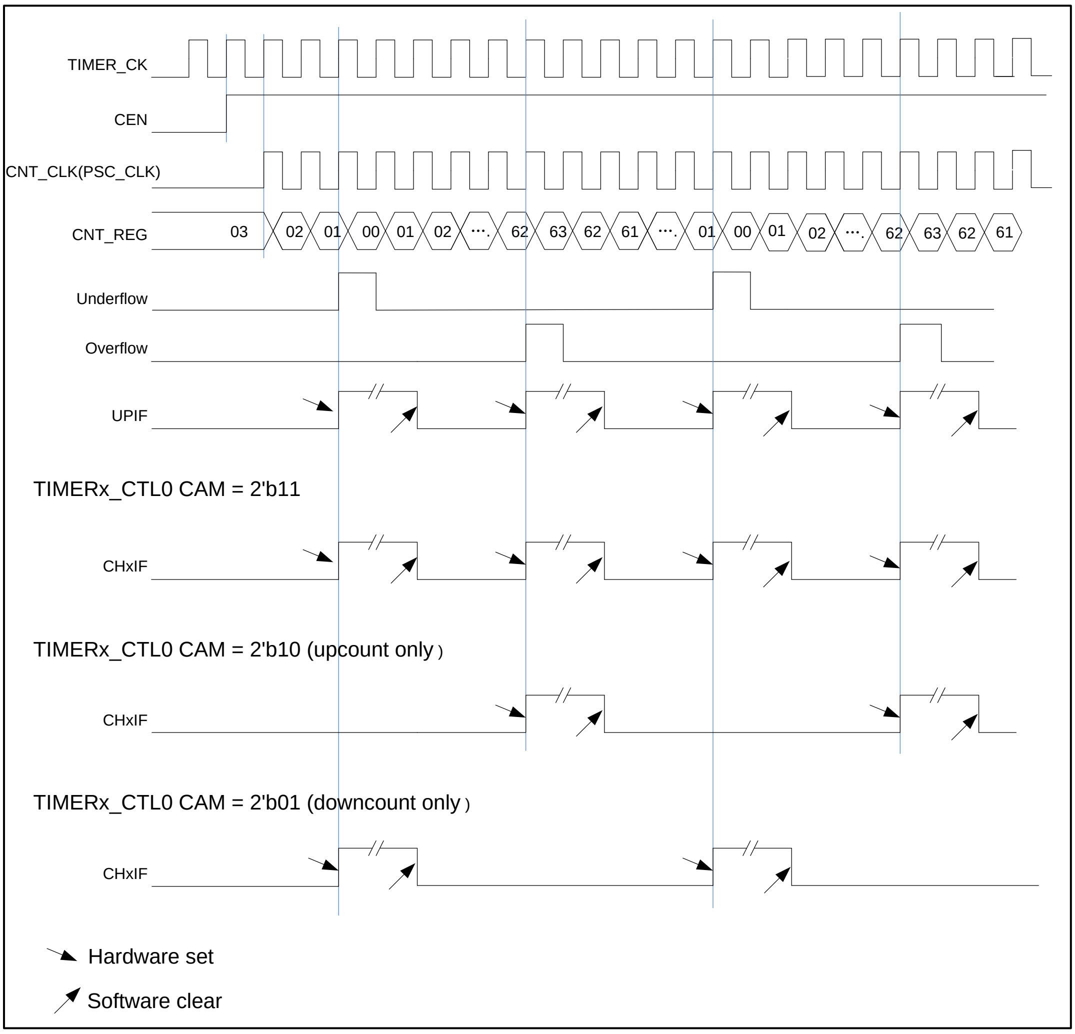
If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register,

prescaler register) are updated.

Figure 15-73. Center-aligned counter timechart show some examples of the counter behavior when TIMERx_CAR=0x63. TIMERx_PSC=0x0

Figure 15-73. Center-aligned counter timechart



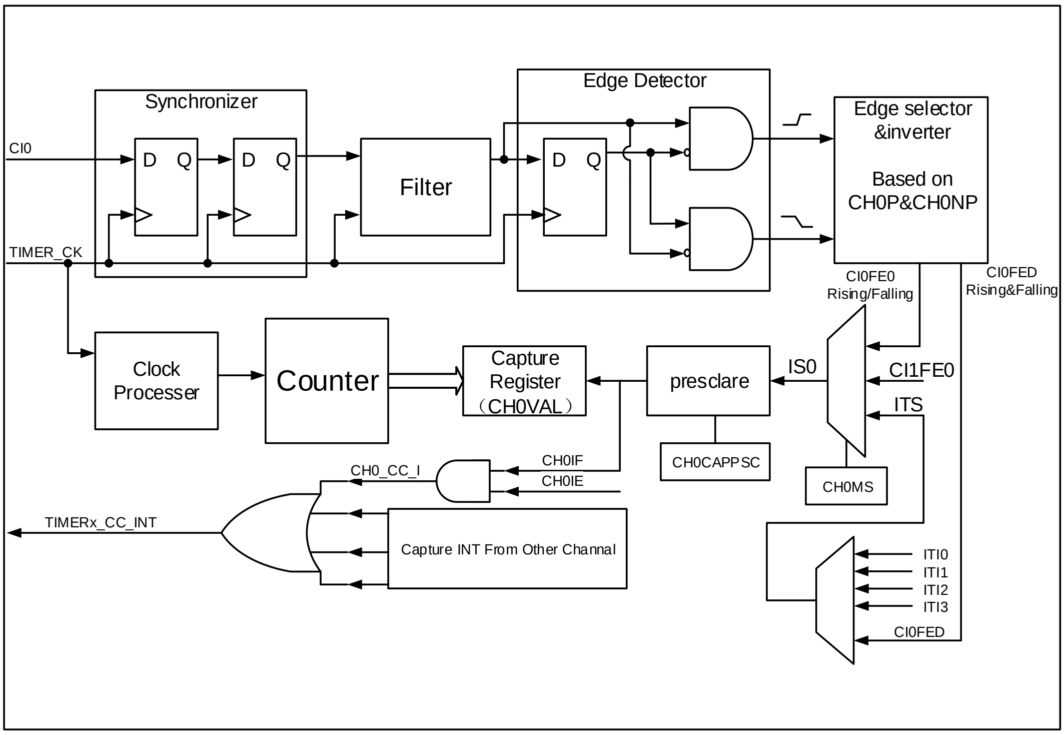
Capture/compare channels

The general level2 timer has one independent channel 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 TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 15-74. Input capture logic



First, the channel input signal (CIx) 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 fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

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

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

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

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

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

Step4: Interrupt enable. (CHxIE in TIMERx_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt.

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt will be asserted based on the your configuration of CHxIE in TIMERx_DMAINTEN

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

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2’b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2’b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Output compare mode**

In Output Compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. when the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 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 CHxCOMSEN
* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
* Select the active high polarity by CHxP/CHxNP
* Enable the output by CHxEN

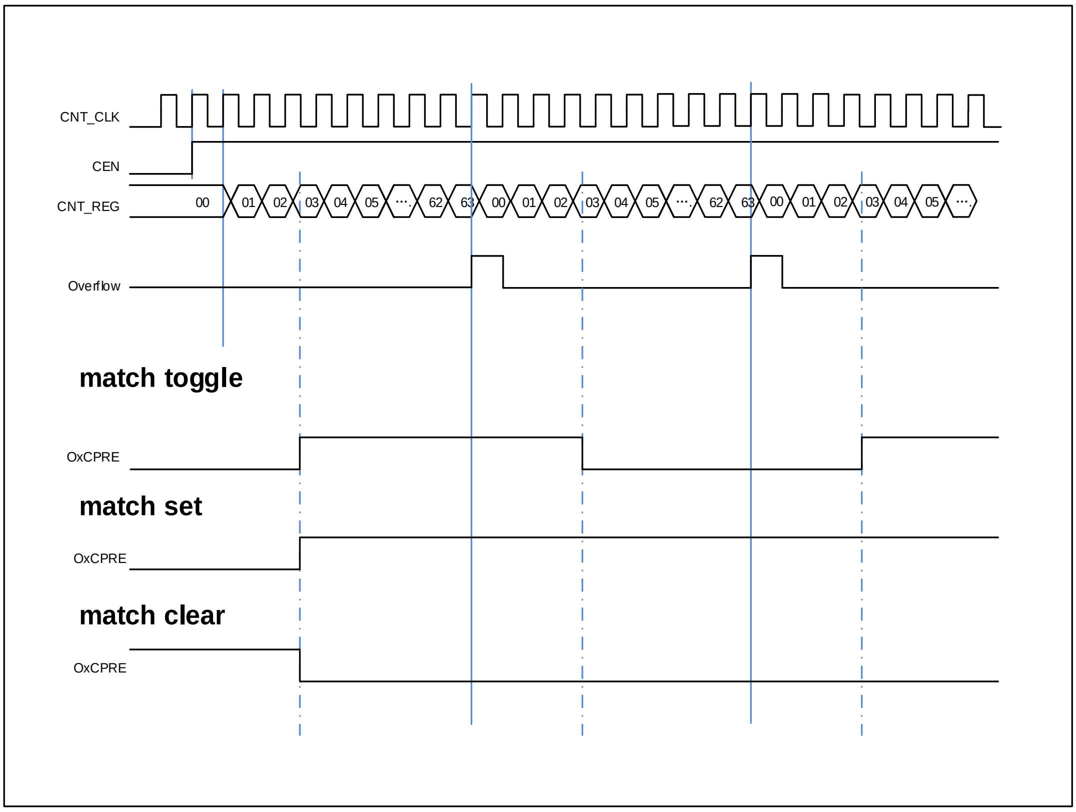
Step3: Interrupt/DMA-request enables configuration by CHxIE

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.
About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-75. Output-compare under three modes



Channel output reference signal

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

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL 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 TIMERx_CHxCV 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 CHxCOMCTL 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 TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFP signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Timer debug mode

When the Cortex™-M3 halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

15.4.5. Register definition

Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	Reserved	UPS	UPDIS	CEN
						rw		rw	rw		rw		rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV 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	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in TIMERx_CHCTL0 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 (CHxMS=00 in

		TIMERx_CHCTL0 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	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	Reserved	Must be kept at reset value
2	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	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 UPG 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	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.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6:4	MMC[2:0]	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 UPG bit in the TIMERx_SWEVG 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 TIMERx_EN 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. 100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO
3:0	Reserved	Must be kept at reset value.

Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CH0IE	UPIE
														rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CH0OF	Reserved.							CH0IF	UPIF
						rc_w0								rc_w0	rc_w0

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:2	Reserved	Must be kept at reset value.
1	CH0IF	Channel 0 ‘s capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input

		mode, this flag is set when a capture event occurs. When channel 0 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	UPIF	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

Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CH0G	UPG
														w	w

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved.								Reserved	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	CH0MS[1:0]	
								CH0CAPFLT[3:0]				CH0CAPPSC[1:0]			
									rw			rw		rw	

Output compare mode:

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced low level. 101: Force high. O0CPRE is forced high level. 110: PWM mode0. When counting up, O0CPRE is high as long as the counter is smaller than TIMERx_CH0CV else low. When counting down, O0CPRE is low as long as the counter is larger than TIMERx_CH0CV else high. 111: PWM mode1. When counting up, O0CPRE is low as long as the counter is smaller than TIMERx_CH0CV else high. When counting down, O0CPRE is high as long as the counter is larger than TIMERx_CH0CV else low. When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “Timing mode” mode to “PWM” mode or when the result of the comparison changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).

3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH0COMFEN	Channel 0 output compare 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 CH0_O is set to the compare level independently from the result of the comparison. 0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 10: Channel 0 is configured as input, IS0 is connected to CI1FE0 11: Channel 0 is configured as input, IS0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:4	CH0CAPFLT[3:0]	Channel 0 input capture 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 CI0 input signal and the length of the digital filter applied to CI0. 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	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 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	CH0MS[1:0]	Channel 0 mode selection Same as output compare mode

Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CH0P	CH0EN
														rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	CH0P	Channel 0 capture/compare polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high

		1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. 0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted. 1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is inverted.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in input mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in output mode, setting this bit enables the capture event in channel0. 0: Channel 0 disabled 1: Channel 0 enabled

Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

PSC[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	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.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CARL[15:0]	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.

Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

CH0VAL[15:0]
rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 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 0 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.

15.5. Basic timer (TIMERx, x=5, 6)

15.5.1. Overview

The basic timer module (Timer5, 6) reference is a 16-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate DMA request and TRGO to DAC.

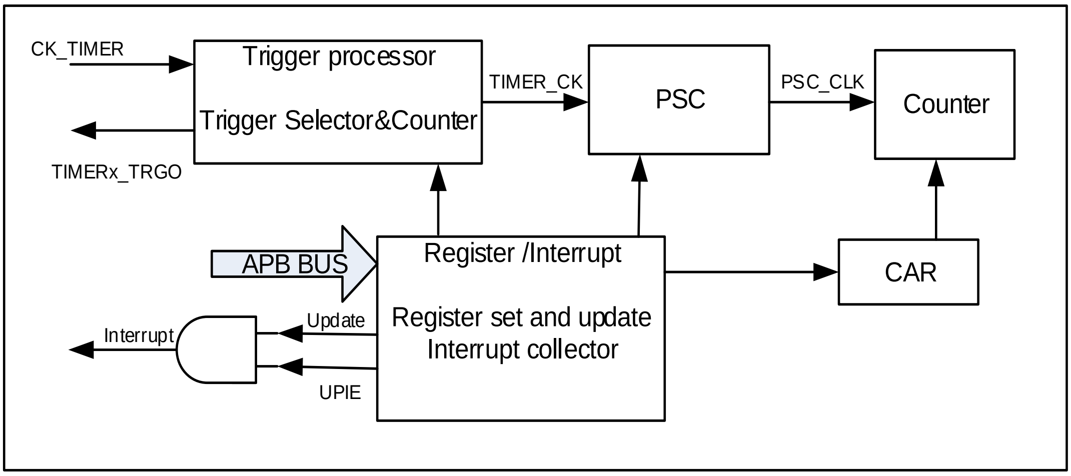
15.5.2. Characteristics

- Counter width: 16bit.
- Source of count clock is internal clock only.
- Multiple counter modes: count up.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Single pulse mode is supported.
- Auto-reload function.
- Interrupt output or DMA request on update event.

15.5.3. Block diagram

Figure 15-76. Basic timer block diagram provides details on the internal configuration of the basic timer.

Figure 15-76. Basic timer block diagram



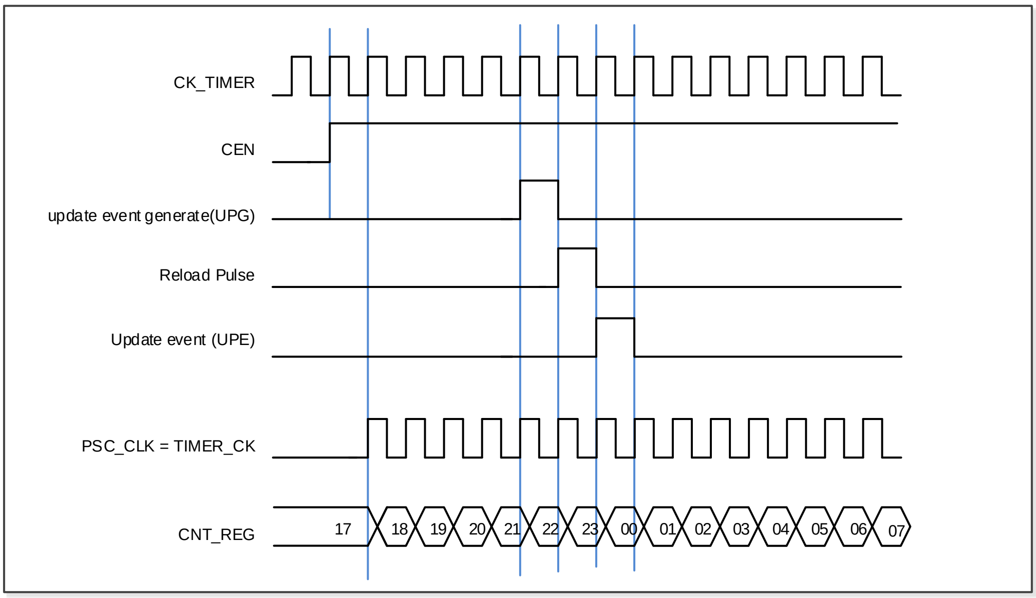
15.5.4. Function overview

Clock selection

The basic TIMER can only being clocked by the internal timer clock CK_TIMER, which is from the source named CK_TIMER in RCU

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

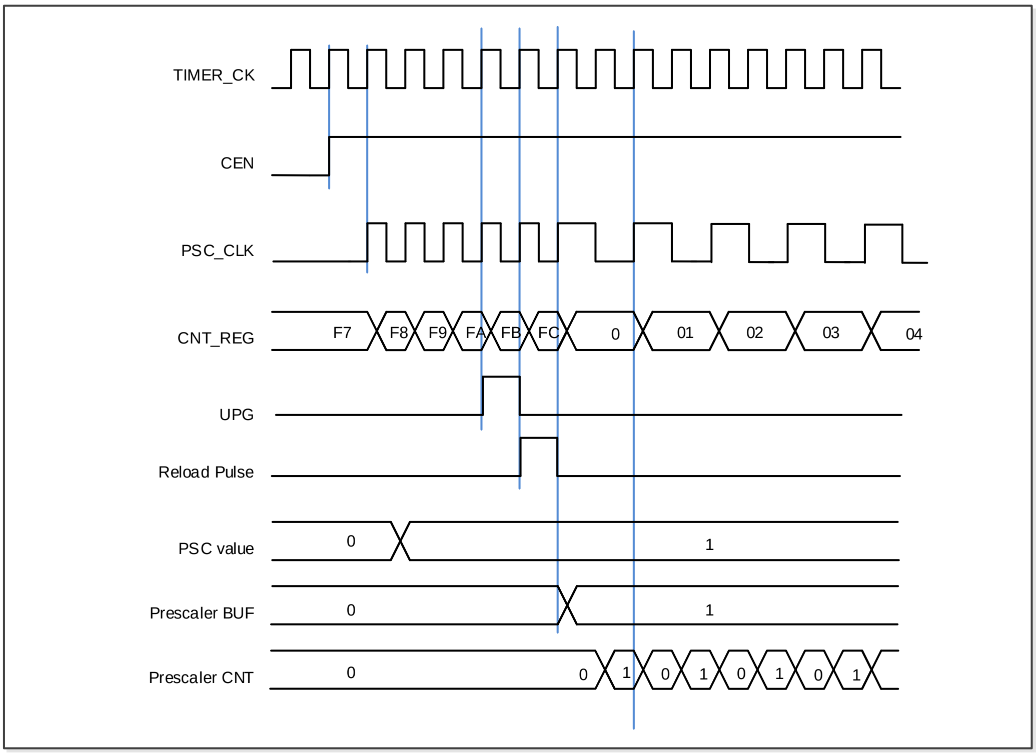
Figure 15-77. Normal mode, internal clock divided by 1



Prescaler

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

Figure 15-78. Counter timing diagram with prescaler division change from 1 to 2



Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0.The update event is generated at each counter overflow.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 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 TIMERx_CAR=0x63.

Figure 15-79. Up-counter timechart, PSC=0/1

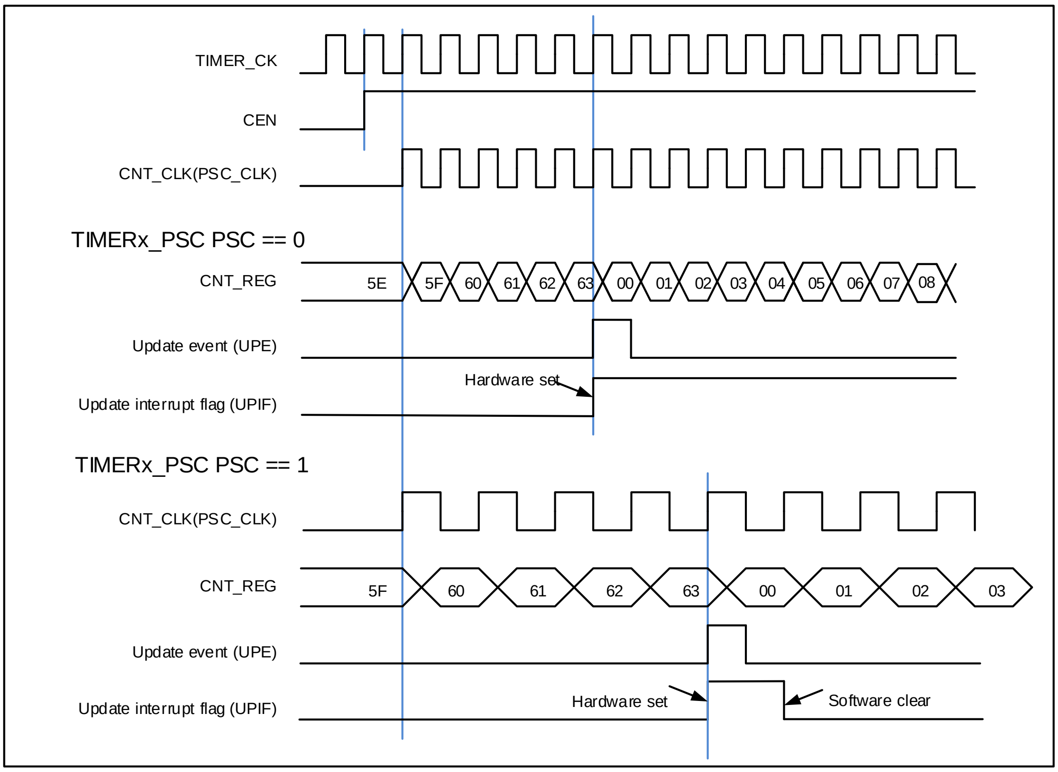
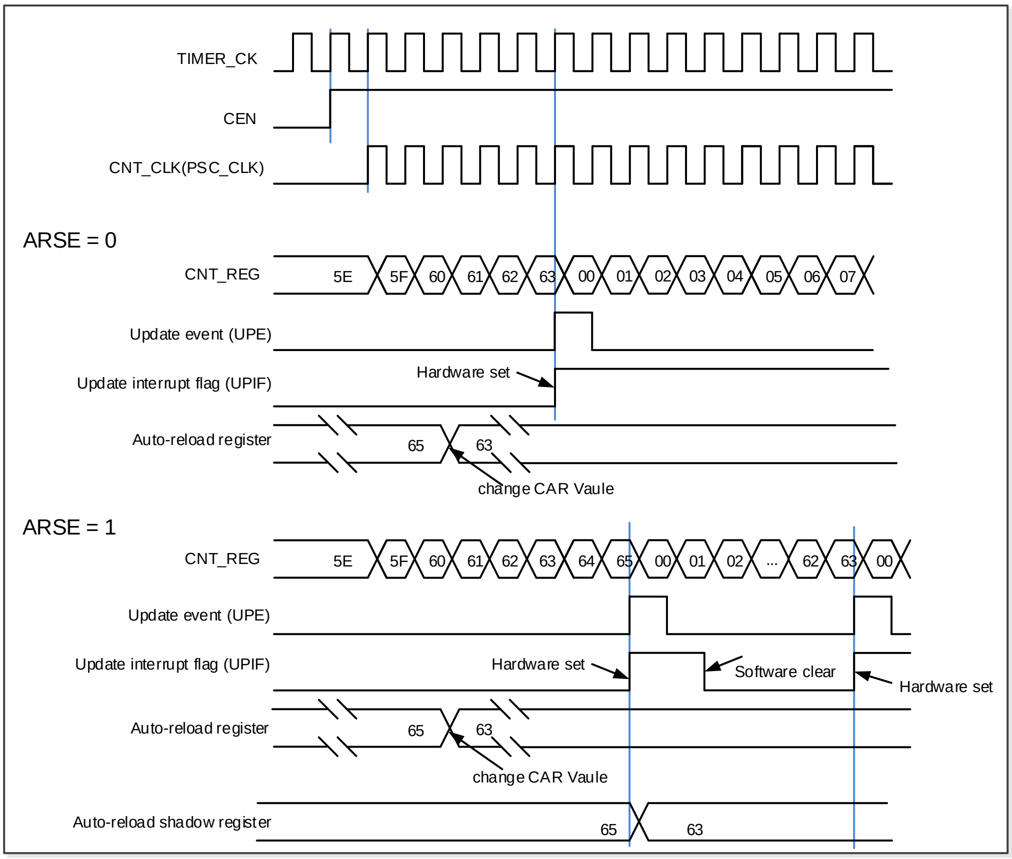


Figure 15-80. Up-counter timechart, change TIMERx_CAR on the go



Timer debug mode

When the Cortex™-M3 halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

15.5.5. Register definition

Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARSE	Reserved			SPM	UPS	UPDIS	CEN
								rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value
3	SPM	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	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: – The UPG bit is set – The counter generates an overflow or underflow event – The slave mode controller generates an update event. 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	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 UPG 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	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.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6:4	MMC[2:0]	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 UPG bit in the TIMERx_SWEVG 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 TIMERx_EN 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.
3:0	Reserved	Must be kept at reset value.

Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							UPDEN	Reserved							UPIE
							rw								rw

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7:1	Reserved	Must be kept at reset value.
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UPIF
															rc_w0

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	UPIF	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

Software event generation register (TIMERx_SWEVG)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UPG
															w

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	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.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CARL[15:0]	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. Universal synchronous/asynchronous receiver /transmitter (USART)

16.1. Overview

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. Data frames can be transferred in full duplex or half duplex mode, synchronously or asynchronously through this interface. A programmable baud rate generator divides the peripheral clock (PCLK1 or PCLK2) to produce a dedicated baud rate lock for the USART transmitter and receiver.

Besides the standard asynchronous receiver and transmitter mode, the USART implements several other types of serial data exchange modes, such as IrDA (infrared data association) SIR mode, smartcard mode, LIN (local interconnection network) mode, half-duplex mode and synchronous mode. It also supports multiprocessor communication mode, and hardware flow control protocol (CTS/RTS). The data frame can be transferred from LSB or MSB bit.

The USART supports DMA function for high-speed data communication, except UART4.

16.2. Characteristics

- NRZ standard format
- Asynchronous, full duplex communication
- Programmable baud-rate generator
  - Divided from the peripheral clocks, PCLK2 for USART0, PCLK1 for USART1/2 and UART3/4.
  - Oversampling by 16
  - Maximum speed up to 6.75 MBits/s (PCLK2 108M and oversampling by 16)
- Fully programmable serial interface characteristics:
  - Even, odd or no-parity bit generation/detection
  - A data word length can be 8 or 9 bits
  - 0.5, 1, 1.5 or 2 stop bit generation
- Transmitter and Receiver can be enabled separately
- Hardware flow control protocol (CTS/RTS)
- DMA request for data buffer access
- LIN Break generation and detection
- IrDA Support
- Synchronous mode and transmitter clock output for synchronous transmission
- ISO 7816-3 compliant smartcard interface
  - Character mode (T=0)
- Multiprocessor communication

  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle frame or address match detection
- Various status flags:
  - Flags for transfer detection: Receive buffer not empty (RBNE), Transmit buffer empty (TBE), transfer complete (TC).
  - Flags for error detection: overrun error (ORERR), noise error (NERR), frame error (FERR) and parity error (PERR)
  - Flag for hardware flow control: CTS changes (CTSF)
  - Flag for LIN mode: LIN break detected (LBDF)
  - Flag for multiprocessor communication: IDLE frame detected (IDLEF)
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set

While USART0/1/2 is fully implemented, UART3/4 is only partially implemented with the following features not supported.

- Smartcard mode
- Synchronous mode
- Hardware flow control protocol (CTS/RTS)

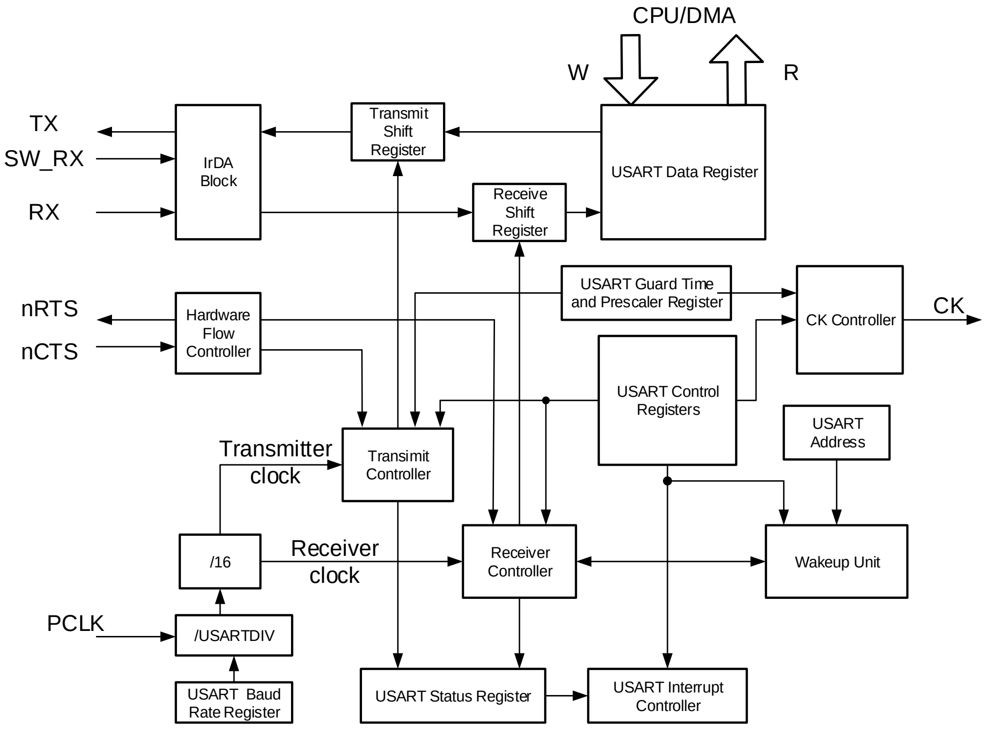
16.3. Function overview

The interface is externally connected to another device by the main pins listed as following.

Table 16-1. USART important pins description

Pin	Type	Description
RX	Input	Receive Data
TX	Output I/O (single-wire/Smartcard mode)	Transmit Data. High level when enabled but nothing to be transmitted
CK	Output	Serial clock for synchronous communication
nCTS	Input	Clear to send in hardware flow control mode
nRTS	Output	Request to send in hardware flow control mode

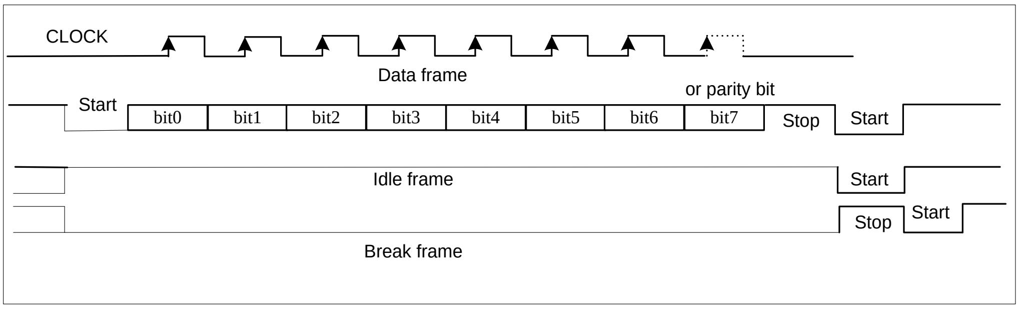
Figure 16-1. USART module block diagram



16.3.1. USART frame format

The USART frame starts with a start bit and ends up with a number of stop bits. The length of the data frame is configured by the WL bit in the USART_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit of in USART_CTL0 register. When the WL bit is reset, the parity bit is the 7th bit. When the WL bit is set, the parity bit is the 8th bit. The method of calculating the parity bit is selected by the PM bit in USART_CTL0 register.

Figure 16-2. USART character frame (8 bits data and 1 stop bit)



In transmission and reception, the number of stop bits can be configured by the STB[1:0] bits in the USART_CTL1 register.

Table 16-2. Stop bits configuration

STB[1:0]	stop bit length (bit)	usage description
00	1	default value
01	0.5	Smartcard mode for receiving

STB[1:0]	stop bit length (bit)	usage description
10	2	normal USART and single-wire modes
11	1.5	Smartcard mode for transmitting and receiving

In an idle frame, all the frame bits are logic 1. The frame length is equal to the normal USART frame.

A break frame is configured number of low bits followed by the configured number of stop bits. The transfer speed of a USART frame depends on the frequency of the PCLK and the configuration of the baud rate generator.

16.3.2. Baud rate generation

The baud-rate divider is a 16-bit number consisting of a 12-bit integer and a 4-bit fractional part. The number formed by these two values is used by the baud rate generator to determine the bit period. Having a fractional baud-rate divider allows the USART to generate all the standard baud rates.

The baud-rate divider (USARTDIV) has the following relationship to the peripheral clock:

$$\mathrm{USARTDIV} = \frac{\mathrm{PCLK}}{16 \times \mathrm{Baud\ Rate}} \tag{16-1}$$

The peripheral clock is PCLK2 for USART0 and PCLK1 for USART1/2 and UART3/4. The peripheral clock must be enabled through the clock control unit before enabling the USART.

16.3.3. USART transmitter

If the transmit enable bit (TEN) in USART_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shifts out the transmit data frame through the TX pin. Clock pulses can be output through the CK pin.

After the TEN bit is set, an idle frame will be sent. The TEN bit should not be reset while the transmission is ongoing.

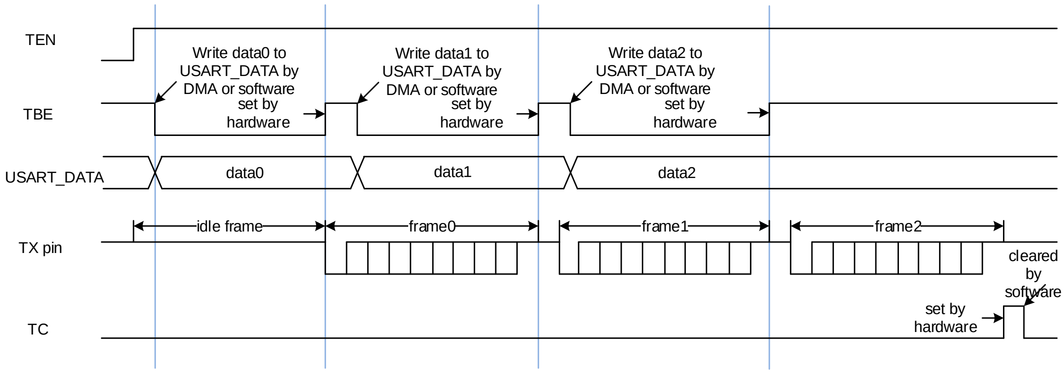
After power on, the TBE bit is high by default. Data can be written to the USART_DATA when the TBE bit of the USART_STAT register is asserted. The TBE bit is cleared by writing to the USART_DATA register and will be set by hardware after the data is put into the transmit shift register. If a data is written to the USART_DATA register while a transmission is ongoing, it will be firstly stored in the transmit buffer, and transferred to the transmit shift register after the current transmission is done. If a data is written to the USART_DATA register while no transmission is ongoing, the TBE bit will be cleared and set soon, because the data will be transferred to the transmit shift register immediately.

If a frame is transmitted and the TBE bit is asserted, the TC bit of the USART_STAT register will be set. An interrupt is generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

The USART transmit procedure is shown in ***Figure 16-3. USART transmit procedure***. The software can follow this flow:

1. Set the UEN bit in USART_CTL0 to enable the USART.
2. Write the WL bit in USART_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART_CTL1 to configure the number of stop bits.
4. Enable DMA (DENT bit) in USART_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART_BAUD.
6. Set the TEN bit in USART_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to in the USART_DATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

Figure 16-3. USART transmit procedure



It is necessary to wait for the TC bit asserted before disabling the USART or entering the power saving mode. This bit can be cleared by a software sequence: reading the USART_STAT register and then writing the USART_DATA register. If the multibuffer communication is selected (DENT=1), this bit can also be cleared by writing 0 to it directly.

16.3.4. USART receiver

After power on, the USART receiver can be enabled by the follow procedure:

1. Set the UEN bit in USART_CTL0 to enable the USART.
2. Write the WL bit in USART_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART_CTL1.
4. Enable DMA (DENR bit) in USART_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART_BAUD.
6. Set the REN bit in USART_CTL0.

After being enabled, the receiver receives a bit stream after a valid start pulse has been detected. Detection on noisy error, parity error, frame error and overrun error is performed during the reception of a frame.

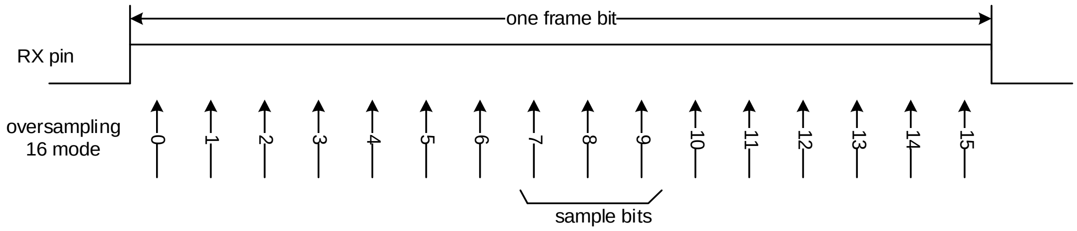
When a frame is received, the RBNE bit in USART_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register. The status bits of the received are stored in the USART_STAT register.

The software can get the received data by reading the USART_DATA register directly, or through DMA. The RBNE status is cleared by a read operation on the USART_DATA register, whatever it is performed by software directly, or through DMA.

The REN bit should not be disabled when reception is ongoing, or the current frame will be lost.

By default, the receiver gets three samples to evaluate the value of a frame bit. While in the oversampling 16 mode, the 7th, 8th, and 9th samples are used. If two or more samples of a frame bit is 0, the frame bit is confirmed as a 0, else 1. If the three samples of any bit of a frame are not the same, whatever it is a start bit, data bit, parity bit or stop bit, a noisy error (NERR) status will be generated for the frame. An interrupt is generated, if the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set.

Figure 16-4. Oversampling method of a receive frame bit



If the parity check function is enabled by setting the PCEN bit in the USART_CTL0 register, the receiver calculates the expected parity value while receiving a frame. The received parity bit will be compared with this expected value. If they are not the same, the parity error (PERR) bit in USART_STAT register will be set. An interrupt is generated, if the PERRIE bit in USART_CTL0 register is set.

If the RX pin is evaluated as 0 during a stop bit, the frame error (FERR) bit in USART_STAT register will be set. An interrupt is generated, If the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set.

When a frame is received, if the RBNE bit is not cleared yet, the last frame will not be stored in the receive data buffer. The overrun error (ORERR) bit in USART_STAT register will be set. An interrupt is generated, if the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set, or if the RBNEIE is set.

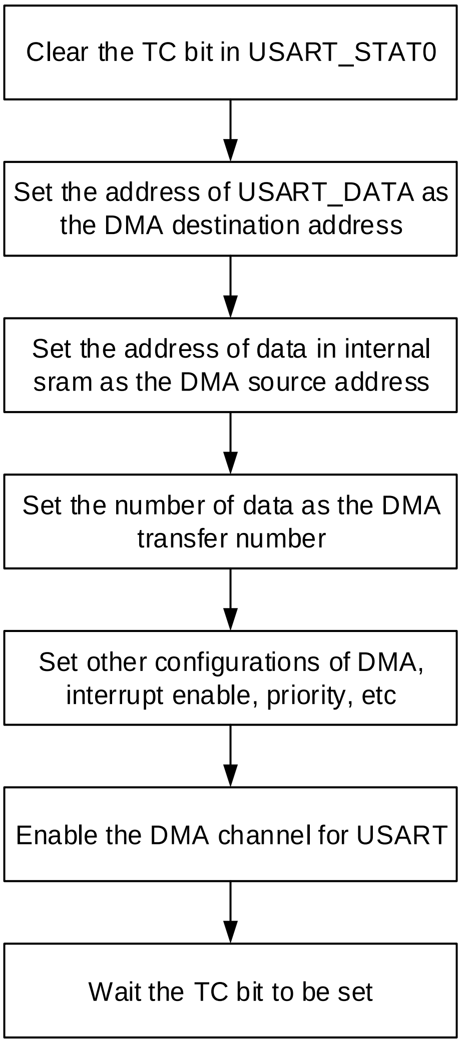
The RBNE, NERR, PERR, FERR and ORERR flags of a reception are always set at the same time. If the receive DMA is not enabled, software can check NERR, PERR, FERR and ORERR flags when serving the RBNE interrupt.

16.3.5. Use DMA for data buffer access

To reduce the burden of the processor, DMA can be used to access the transmitting and receiving data buffer. The DENT bit in USART_CTL2 is used to enable the DMA transmission, and the DENR bit in USART_CTL2 is used to enable the DMA reception.

When DMA is used for USART transmission, DMA transfers data from internal sram to the transmit data buffer of the USART. The configuration step is shown in ***Figure 16-5. Configuration step when using DMA for USART transmission***.

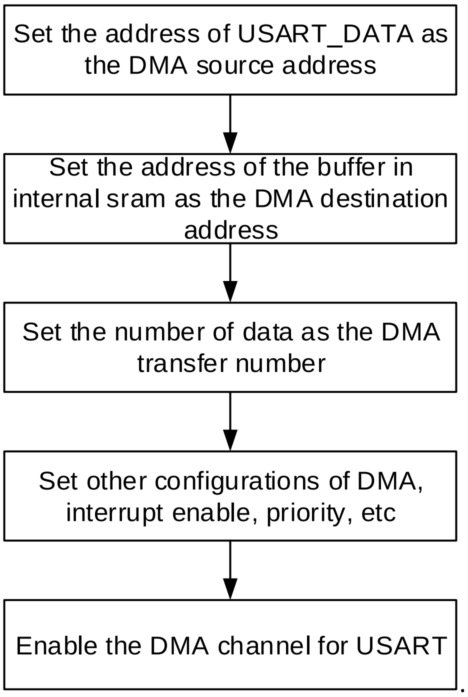
Figure 16-5. Configuration step when using DMA for USART transmission


Clear the TC bit in USART_STAT0
Set the address of USART_DATA as the DMA destination address
Set the address of data in internal sram as the DMA source address
Set the number of data as the DMA transfer number
Set other configurations of DMA, interrupt enable, priority, etc
Enable the DMA channel for USART
Wait the TC bit to be set

After all of the data frames are transmitted, the TC bit in USART_STAT is set. An interrupt occurs if the TCIE bit in USART_CTL0 is set.

When DMA is used for USART reception, DMA transfers data from the receive data buffer of the USART to the internal sram. The configuration step is shown in ***Figure 16-6. Configuration step when using DMA for USART reception***. If the ERRIE bit in USART_CTL2 is set, interrupts can be generated by the Error status bits (FERR, ORERR and NERR) in USART_STAT.

Figure 16-6. Configuration step when using DMA for USART reception

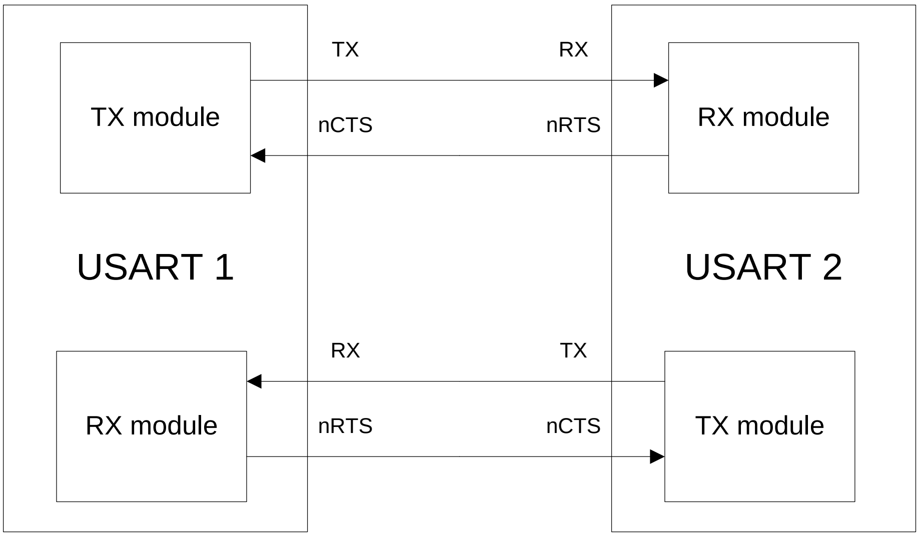

Set the address of USART_DATA as the DMA source address
Set the address of the buffer in internal sram as the DMA destination address
Set the number of data as the DMA transfer number
Set other configurations of DMA, interrupt enable, priority, etc
Enable the DMA channel for USART

When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt can be generated in the DMA module.

16.3.6. Hardware flow control

The hardware flow control function is realized by the nCTS and nRTS pins. The RTS flow control is enabled by writing ‘1’ to the RTSEN bit in USART_CTL2 and the CTS flow control is enabled by writing ‘1’ to the CTSEN bit in USART_CTL2.

Figure 16-7. Hardware flow control between two USARTs



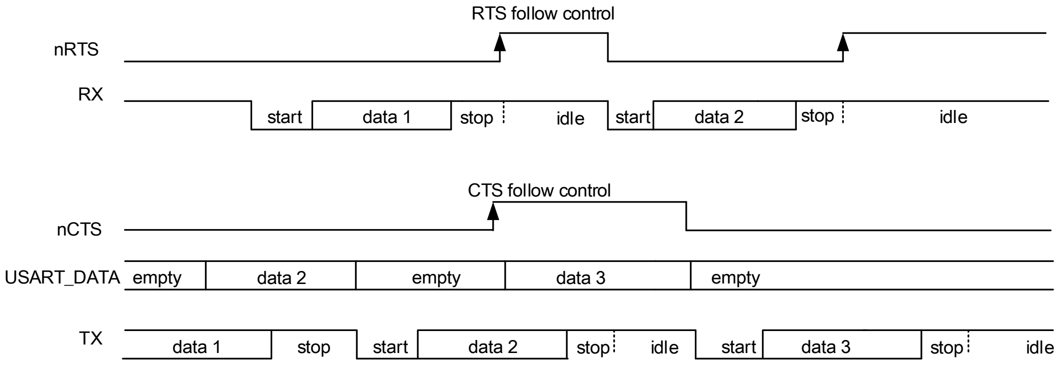
RTS flow control

The USART receiver outputs the nRTS, which reflects the status of the receive buffer. When data frame is received, the nRTS signal goes high to prevent the transmitter from sending next frame. The nRTS signal keeps high when the receive buffer is full, and can be cleared by reading the USART_DATA register.

CTS flow control

The USART transmitter monitors the nCTS input pin to decide if a data frame can be transmitted. If the TBE bit in USART_STAT is ‘0’ and the nCTS signal is low, the transmitter transmits the data frame. When the nCTS signal goes high during a transmission, the transmitter stops after the current transmission is accomplished.

Figure 16-8. Hardware flow control



If the CTS flow control is enabled, the CTSF bit in USART_STAT is set when the nCTS pin toggles. An interrupt is generated if the CTSIE bit in USART_CTL2 is set.

16.3.7. Multi-processor communication

In multiprocessor communication, several USARTs are connected as a network. It will be a big burden for a device to monitor all of the messages on the RX pin. To reduce the burden of a device, software can put an USART module into a mute mode by setting the RWU bit in USART_CTL0 register.

If a USART is in mute mode, all of the receive status bits cannot be set. Software can wake up the USART by resetting the RWU bit.

The USART can also be wake up by hardware by one of the two methods: idle frame method and address match method.

The idle frame wake up method is selected by default. When an idle frame is detected on the RX pin, the hardware clears the RWU bit and exits the mute mode. When wake up at an idle frame, the IDLEF bit in USART_STAT is not set.

When the WM bit of in USART_CTL0 register is set, the MSB bit of a frame is detected as the address flag. If the address flag is high, the frame is treated as an address frame. If the address flag is low, the frame is treated as a data frame. If the LSB 4 bits of an address frame are the same as the ADDR[3:0] bits in the USART_CTL1 register, the hardware clears the RWU bit and exits the mute mode. The RBNE bit is set for the frame that wakes up the USART. The status bits are available in the USART_STAT register. If the LSB 4 bits of an address frame differ from the ADDR[3:0] bits in the USART_CTL1 register, the hardware sets the RWU bit and enters mute mode automatically. In this situation, the RBNE bit is not set.

If the address match method is selected, the receiver does not check the parity value of an address frame by default. If the PCEN bit in USART_CTL0 is set, the MSB bit will be checked as the parity bit, and the bit preceding the MSB bit is detected as the address flag.

16.3.8. LIN mode

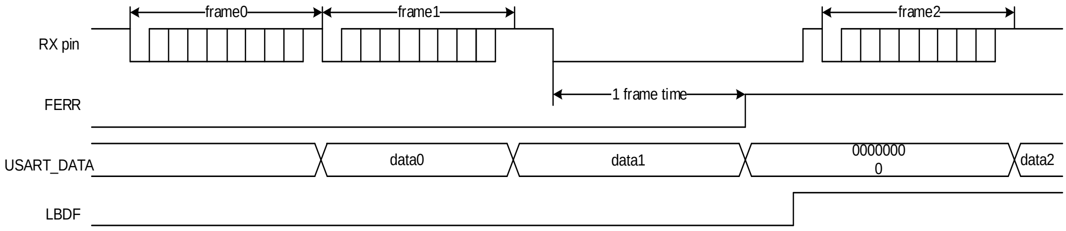
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, STB[1:0] bits in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be reset in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. The data bits length must be 8. When the SBKCMD bit in USART_CTL0 is set, the USART transmits continuous 13 ‘0’ bits, following by 1 stop bit.

The break detection function is totally independent from the normal USART receiver. So a break frame can be detected during the idle state or during a frame. The expected length of a break frame can be selected by LBLEN in USART_CTL1. When the RX pin is detected at low state for a time that is equal to or longer than the expected break frame length (10 bits when LBLEN=0, or 11 bits when LBLEN=1), the LBDF in USART_STAT is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

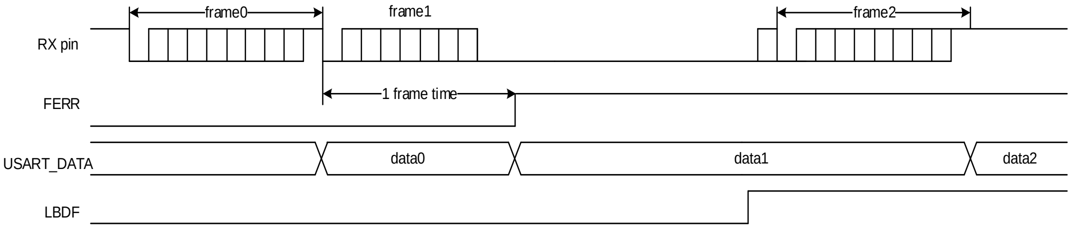
As shown in ***Figure 16-9. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all ‘0’ frame, with an asserted FERR status.

Figure 16-9. Break frame occurs during idle state



As shown in ***Figure 16-10. Break frame occurs during a frame***, if a break frame occurs during a frame on the RX pin, the FERR status will be asserted for the current frame.

Figure 16-10. Break frame occurs during a frame



16.3.9. Synchronous mode

The USART can be used for full-duplex synchronous serial communications only in master mode, by setting the CKEN bit in USART_CTL1. The LMEN bit in USART_CTL1 and SCEN, HDEN, IREN bits in USART_CTL2 should be reset in synchronous mode. The CK pin is the synchronous USART transmitter clock output, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the start bit and stop bit transmission. The CLEN bit in USART_CTL1 can be used to determine whether the clock is output or not during the last (address flag) bit transmission. The CPH bit in USART_CTL1 can be used to determine whether data is captured on the first or the second clock edge. The CPL bit in USART_CTL1 can be used to configure the clock polarity in the USART synchronous idle state.

The CPL, CPH and CLEN bits in USART_CTL1 determine the waveform on the CK pin. Software can only change them when the USART is disabled (UEN=0).

If the REN bit in USART_CTL0 is set, the receiver works differently from the normal USART reception method. The receiver samples the data on the capture edge of the CK pin without any oversampling.

Figure 16-11. Example of USART in synchronous mode

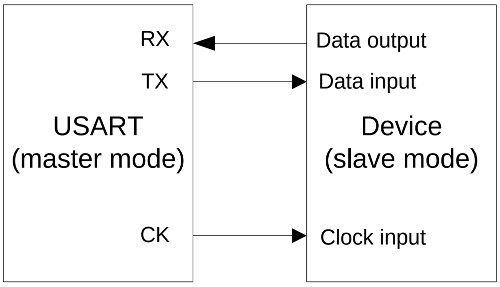
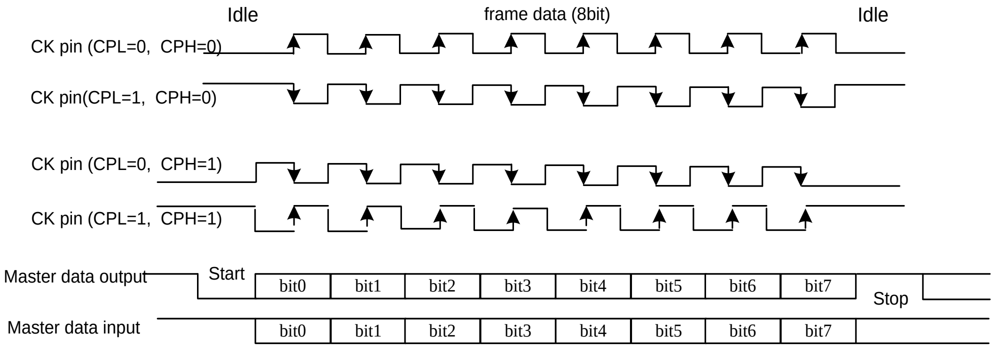


Figure 16-12. 8-bit format USART synchronous waveform (CLEN=1)

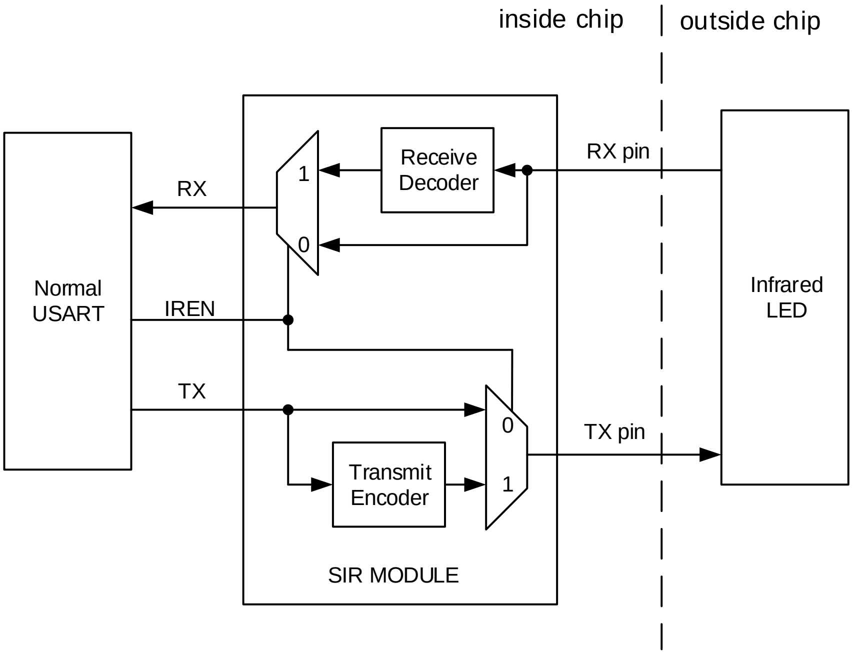


16.3.10. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART_CTL2. The LMEN, STB[1:0], CKEN bits in USART_CTL1 and HDEN, SCEN bits in USART_CTL2 should be reset in IrDA mode.

In IrDA mode, the USART transmission data frame is modulated in the SIR transmit encoder and transmitted to the infrared LED through the TX pin. The SIR receive decoder receives the modulated signal from the infrared LED through the RX pin, and puts the demodulated data frame to the USART receiver. The baud rate should not be larger than 115200 for the encoder.

Figure 16-13. IrDA SIR ENDEC module

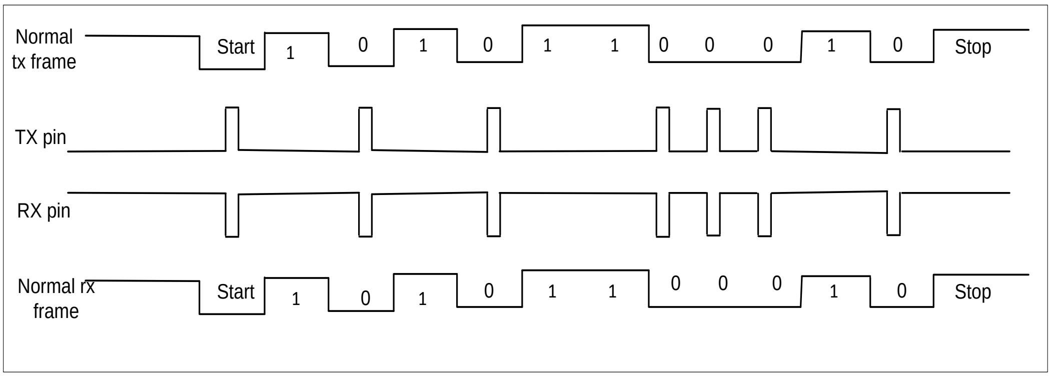


In IrDA mode, the polarity of the TX and RX pins is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic '1', while an infrared light pulse on the IrDA pins (a Return to Zero signal) represents the logic '0'.

The pulse width should be 3/16 of a bit period. The IrDA could not detect any pulse if the pulse width is less than 1 PSC clock. While it can detect a pulse by chance if the pulse width is greater than 1 but smaller than 2 times PSC clock.

Because the IrDA is a half-duplex protocol, the transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

Figure 16-14. IrDA data modulation



The SIR submodule can work in low power mode by setting the IRLP bit in USART_CTL2. The transmit encoder is driven by a low speed clock, which is divided from the PCLK. The divide ratio is configured by the PSC[7:0] bits in USART_GP register. The pulse width on the TX pin is 3 cycles of this low speed clock. The receiver decoder works in the same manner as the normal IrDA mode.

16.3.11. Half-duplex communication mode

The half-duplex communication mode is enabled by setting the HDEN bit in USART_CTL2. The LMEN, CKEN bits in USART_CTL1 and SCEN, IREN bits in USART_CTL2 should be reset in half-duplex communication mode.

In the half-duplex mode the receive line is internally connected to the TX pin, and the RX pin is no longer used. The TX pin should be configured as output open drain mode. The software should make sure the transmission and reception process never conflict each other.

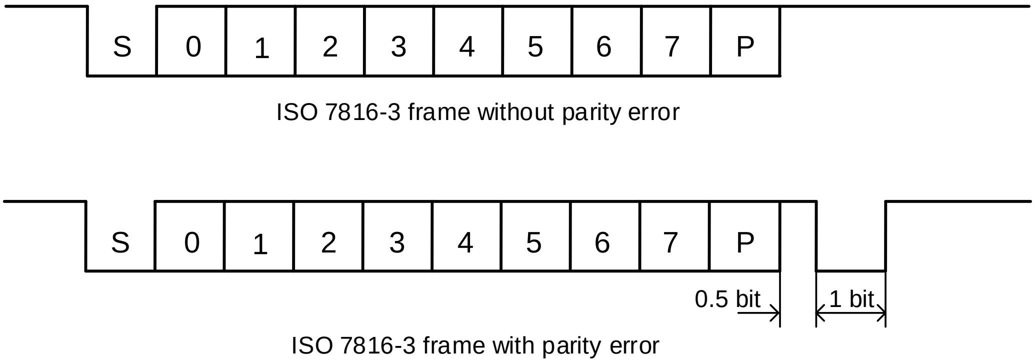
16.3.12. Smartcard (ISO7816-3) mode

The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. The smartcard mode is enabled by setting the SCEN bit in USART_CTL2. The LMEN bit in USART_CTL1 and HDEN, IREN bits in USART_CTL2 should be reset in smartcard mode.

A clock is provided to the external smart card through the CK pin after the CKEN bit is set. The clock is divided from the PCLK. The divide ratio is configured by the PSC[4:0] bits in USART_GP register. The CK pin only provides a clock source to the smart card.

The smartcard mode is a half-duplex communication protocol. When connected to a smartcard, the TX pin must be configured as open drain, and an external pull-up resistor will be needed, which drives a bidirectional line that is also driven by the smartcard. The data frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits. The 0.5 stop bit may be configured for a receiver.

Figure 16-15. ISO7816-3 frame format


S
0
1
2
3
4
5
6
7
P
ISO 7816-3 frame without parity error
S
0
1
2
3
4
5
6
7
P
0.5 bit
1 bit
ISO 7816-3 frame with parity error

Character (T=0) mode

Comparing to the timing in normal operation, the transmission time from transmit shift register to the TX pin is delayed by half baud clock, and the TC flag assertion time is delayed by a guard time that is configured by the GUAT[7:0] bits in USART_GP. In Smartcard mode, the internal guard time counter starts count up after the stop bits of the last data frame, and the GUAT[7:0] bits should be configured as the character guard time (CGT) in ISO7816-3 protocol minus 12. The TC status is forced reset while the guard time counter is counting up. When the counter reaches the programmed value TC is asserted high.

During USART transmission, if a parity error event is detected, the smartcard may NACK the current frame by pulling down the TX pin during the last 1 bit time of the stop bits. The USART can automatically resend data according to the protocol. The USART will not take the NACK signal as the start bit.

During USART reception, if the parity error is detected in the current frame, the TX pin is pulled low during the last 1 bit time of the stop bits. This signal is the NACK signal to smart card. Then a frame error occurs in smart card side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smart card can resend the data. The NACK signal is enabled by setting the NKEN bit in USART_CTL2.

The idle frame and break frame are not applied for the Smartcard mode.

16.3.13. USART interrupts

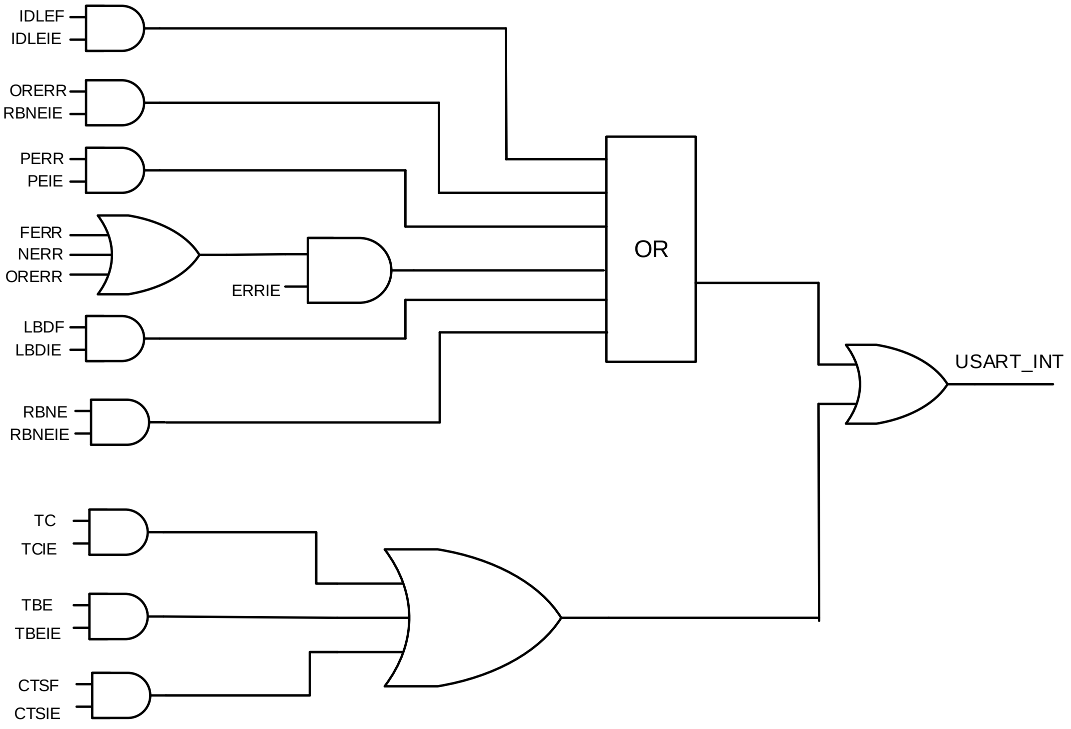
The USART interrupt events and flags are listed in the table below.

Table 16-3. USART interrupt requests

Interrupt event	Event flag	Control register	Enable Control bit
Transmit data buffer empty	TBE	USART_CTL0	TBEIE
CTS toggled flag	CTSF	USART_CTL2	CTSIE
Transmission complete	TC	USART_CTL0	TCIE
Received buff not empty	RBNE	USART_CTL0	RBNEIE
Overrun error	ORERR		
Idle frame	IDLEF	USART_CTL0	IDLEIE
Parity error	PERR	USART_CTL0	PERRIE
Break detected flag in LIN mode	LBDF	USART_CTL1	LBDIE
Reception Errors (Noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	USART_CTL2	ERRIE

All of the interrupt events are ORed together before being sent to the interrupt controller, so the USART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine.

Figure 16-16. USART interrupt mapping diagram



16.4. Register definition

16.4.1. Status register (USART_STAT)

Address offset: 0x00

Reset value: 0x0000 00C0

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTSF	LBDF	TBE	TC	RBNE	IDLEF	ORERR	NERR	FERR	PERR
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bits	Fields	Descriptions
31:10	Reserved	Must keep the reset value
9	CTSF	CTS change flag If CTSEN bit in USART_CTL2 is set, this bit is set by hardware when the nCTS input toggles. An interrupt occurs if the CTSIE bit in USART_CTL2 is set. Software can clear this bit by writing 0 to it. 0: The status of the nCTS line does not change 1: The status of the nCTS line has changed This bit is not available for UART3/4.
8	LBDF	LIN break detected flag If LMEN bit in USART_CTL1 is set, this bit is set by hardware when LIN break is detected. An interrupt occurs if the LBDIE bit in USART_CTL1 is set. Software can clear this bit by writing 0 to it. 0: The USART does not detect a LIN Break 1: The USART has detected a LIN Break
7	TBE	Transmit data buffer empty This bit is set after power on or when the transmit data has been transferred to the transmit shift register. An interrupt occurs if the TBEIE bit in USART_CTL0 is set. This bit is cleared when the software writes transmit data to the USART_DATA register. 0: Transmit data buffer is not empty 1: Transmit data buffer is empty
6	TC	Transmission complete This bit is set after power on. If the TBE bit has been set, this bit is set when the transmission of current data is complete. An interrupt occurs if the TCIE bit in

		USART_CTL0 is set. Software can clear this bit by writing 0 to it. 0: Transmission of current data is not complete 1: Transmission of current data is complete
5	RBNE	Read data buffer not empty This bit is set when the read data buffer is filled with a data frame, which has been received through the receive shift register. An interrupt occurs if the RBNEIE bit in USART_CTL0 is set. Software can clear this bit by writing 0 to it or by reading the USART_DATA register. 0: Read data buffer is empty 1: Read data buffer is not empty
4	IDLEF	IDLE frame detected flag This bit is set when the RX pin has been detected in idle state for a frame time. An interrupt occurs if the IDLEIE bit in USART_CTL0 is set. Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one. 0: The USART module does not detect an IDLE frame 1: The USART module has detected an IDLE frame
3	ORERR	Overrun error This bit is set if the RBNE is not cleared and a new data frame is received through the receive shift register. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one. 0: The USART does not detect a overrun error 1: The USART has detected a overrun error
2	NERR	Noise error flag This bit is set if the USART detects noise on the RX pin when receiving a frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one. 0: The USART does not detect a noise error 1: The USART has detected a noise error
1	FERR	Frame error flag This bit is set when the RX pin is detected low during the stop bits of a receive frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one. 0: The USART does not detect a framing error 1: The USART has detected a framing error

0	PERR	Parity error flag This bit is set when the parity bit of a receive frame does not match the expected parity value. An interrupt occurs if the PERRIE bit in USART_CTL0 is set. Software can clear this bit in the sequence: read the USART_STAT register, and then read or write the USART_DATA register. 0: The USART does not detect a parity error 1: The USART has detected a parity error

16.4.2. Data register (USART_DATA)

Offset: 0x04

Reset value: Undefined

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:9	Reserved	Must keep the reset value
8:0	DATA[8:0]	Transmit or read data value Software can write these bits to update the transmit data or read these bits to get the receive data. If the parity check function is enabled, when transmit data is written to this register, the MSB bit (bit 7 or bit 8 depending on the WL bit in USART_CTL0) will be replaced by the parity bit.

16.4.3. Baud rate register (USART_BAUD)

Address offset: 0x08

Reset value: 0x0000 0000

The software must not write this register when the USART is enabled (UEN=1).
This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTDIV [11:0]												FRADIV[3:0]			

rw rw

Bits	Fields	Descriptions
31:16	Reserved	Must keep the reset value
15:4	INTDIV[11:0]	Integer part of baud-rate divider
3:0	FRADIV[3:0]	Fraction part of baud-rate divider

16.4.4. Control register 0 (USART_CTL0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UEN	WL	WM	PCEN	PM	PERRIE	TBEIE	TCIE	RBNEIE	IDLEIE	TEN	REN	RWU	SBKCMD
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:14	Reserved	Must keep the reset value
13	UEN	USART enable 0: USART disabled 1: USART enabled
12	WL	Word length 0: 8 Data bits 1: 9 Data bits
11	WM	Wakeup method in mute mode 0: wake up by idle frame 1: wake up by address match
10	PCEN	Parity check function enable 0: Parity check function disabled 1: Parity check function enabled
9	PM	Parity mode 0: Even parity 1: Odd parity
8	PERRIE	Parity error interrupt enable If this bit is set, an interrupt occurs when the PERR bit in USART_STAT is set.

		0: Parity error interrupt is disabled 1: Parity error interrupt is enabled
7	TBEIE	Transmitter buffer empty interrupt enable If this bit is set, an interrupt occurs when the TBE bit in USART_STAT is set. 0: Transmitter buffer empty interrupt is disabled 1: Transmitter buffer empty interrupt is enabled
6	TCIE	Transmission complete interrupt enable If this bit is set, an interrupt occurs when the TC bit in USART_STAT is set. 0: Transmission complete interrupt is disabled 1: Transmission complete interrupt is enabled
5	RBNEIE	Read data buffer not empty interrupt and overrun error interrupt enable If this bit is set, an interrupt occurs when the RBNE bit or the ORERR bit in USART_STAT is set. 0: Read data register not empty interrupt and overrun error interrupt disabled 1: Read data register not empty interrupt and overrun error interrupt enabled
4	IDLEIE	IDLE line detected interrupt enable If this bit is set, an interrupt occurs when the IDLEF bit in USART_STAT is set. 0: IDLE line detected interrupt disabled 1: IDLE line detected interrupt enabled
3	TEN	Transmitter enable 0: Transmitter is disabled 1: Transmitter is enabled
2	REN	Receiver enable 0: Receiver is disabled 1: Receiver is enabled
1	RWU	Receiver wakeup from mute mode. Software can set this bit to make the USART work in mute mode and reset this bit to wake up the USART. In wake up by idle frame mode (WM=0), this bit can be reset by hardware when an idle frame has been detected. In wake up by address match mode (WM=1), this bit can be reset by hardware when receiving an address match frame or set by hardware when receiving an address mismatch frame. 0: Receiver in active mode 1: Receiver in mute mode
0	SBKCMD	Send break command Software can set this to send a break frame. Hardware resets this bit automatically when the break frame has been transmitted. 0: Do not transmit a break frame

Bits	Fields	Descriptions
		1: Transmit a break frame

16.4.5. Control register 1 (USART_CTL1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LMEN	STB[1:0]		CKEN	CPL	CPH	CLEN	Reserved	LBDIE	LBLEN	Reserved	ADDR[3:0]			
	rw	rw		rw	rw	rw	rw		rw	rw		rw			

Bits	Fields	Descriptions
31:15	Reserved	Must keep the reset value
14	LMEN	LIN mode enable 0: LIN mode disabled 1: LIN mode enabled This bit field cannot be written when the USART is enabled (UEN=1).
13:12	STB[1:0]	STOP bits length 00: 1 Stop bit 01: 0.5 Stop bit 10: 2 Stop bits 11: 1.5 Stop bit This bit field cannot be written when the USART is enabled (UEN=1). Only 1 stop bit and 2 stop bits are available for UART3/4.
11	CKEN	CK pin enable 0: CK pin disabled 1: CK pin enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
10	CPL	CK polarity This bit specifies the polarity of the CK pin in synchronous mode. 0: The CK pin is in low state when the USART is in idle state 1: The CK pin is in high state when the USART is in idle state This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
9	CPH	CK phase

		This bit specifies the phase of the CK pin in synchronous mode. 0: The capture edge of the LSB bit is the first edge of CK pin 1: The capture edge of the LSB bit is the second edge of CK pin This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
8	CLEN	CK Length This bit specifies the length of the CK signal in synchronous mode. 0: There are 7 CK pulses for an 8 bit frame and 8 CK pulses for a 9 bit frame 1: There are 8 CK pulses for an 8 bit frame and 9 CK pulses for a 9 bit frame This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
7	Reserved	Must keep the reset value
6	LBDIE	LIN break detected interrupt enable If this bit is set, an interrupt occurs when the LBDF bit in USART_STAT is set. 0: LIN break detected interrupt is disabled 1: LIN break detected interrupt is enabled
5	LBLEN	LIN break frame length This bit specifies the length of a LIN break frame. 0: 10 bit 1: 11 bit This bit field cannot be written when the USART is enabled (UEN=1).
4	Reserved	Must keep the reset value
3:0	ADDR[3:0]	Address of the USART In wake up by address match mode (WM=1), the USART enters mute mode when the LSB 4 bits of a received frame do not equal the ADDR[3:0] bits, and wakes up when the LSB 4 bits of a received frame equal the ADDR[3:0] bits.

16.4.6. Control register 2 (USART_CTL2)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSEN	RTSEN	DENT	DENR	SCEN	NKEN	HDEN	IRLP	IREN	ERRIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:11	Reserved	Must keep the reset value
10	CTSIE	CTS interrupt enable If this bit is set, an interrupt occurs when the CTSF bit in USART_STAT is set. 0: CTS interrupt is disabled 1: CTS interrupt is enabled This bit is reserved for UART3/4.
9	CTSEN	CTS enable This bit enables the CTS hardware flow control function. 0: CTS hardware flow control disabled 1: CTS hardware flow control enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
8	RTSEN	RTS enable This bit enables the RTS hardware flow control function. 0: RTS hardware flow control disabled 1: RTS hardware flow control enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
7	DENT	DMA request enable for transmission 0: DMA request is disabled for transmission 1: DMA request is enabled for transmission
6	DENR	DMA request enable for reception 0: DMA request is disabled for reception 1: DMA request is enabled for reception
5	SCEN	Smartcard mode enable This bit enables the smartcard work mode. 0: Smartcard Mode disabled 1: Smartcard Mode enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
4	NKEN	NACK enable in Smartcard mode This bit enables the NACK transmission when parity error occurs in smartcard mode. 0: Disable NACK transmission 1: Enable NACK transmission This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
3	HDEN	Half-duplex enable

		This bit enables the half-duplex USART mode. 0: Half duplex mode is disabled 1: Half duplex mode is enabled This bit field cannot be written when the USART is enabled (UEN=1).
2	IRLP	IrDA low-power This bit selects low-power mode of IrDA mode. 0: Normal mode 1: Low-power mode This bit field cannot be written when the USART is enabled (UEN=1).
1	IREN	IrDA mode enable This bit enables the IrDA mode of USART. 0: IrDA disabled 1: IrDA enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
0	ERRIE	Error interrupt enable When DMA request for reception is enabled (DENR=1), if this bit is set, an interrupt occurs when any one of the FERR, ORERR and NERR bits in USART_STAT is set. 0: Error interrupt disabled 1: Error interrupt enabled

16.4.7. Guard time and prescaler register (USART_GP)

Address offset: 0x18

Reset value: 0x0000 0000

This bit field cannot be written when the USART is enabled (UEN=1).

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must keep the reset value.
15:8	GUAT[7:0]	Guard time value in Smartcard mode TC flag assertion time is delayed by GUAT[7:0] baud clock cycles. This bit field cannot be written when the USART is enabled (UEN=1).

		These bits are not available for UART3/4.
7:0	PSC[7:0]	When the USART IrDA low-power mode is enabled, these bits specify the division factor that is used to divide the peripheral clock (PCLK1/PCLK2) to generate the low-power frequency. 00000000: Reserved - never program this value 00000001: divides by 1 00000010: divides by 2 ... 11111111: divides by 255 When the USART works in IrDA normal mode, these bits must be set to 00000001. When the USART smartcard mode is enabled, the PSC [4:0] bits specify the division factor that is used to divide the peripheral clock (APB1/APB2) to generate the smartcard clock (CK). The actual division factor is twice as the PSC [4:0] value. 00000: Reserved - never program this value 00001: divides by 2 00010: divides by 4 ... 11111: divides by 62 The PSC [7:5] bits are reserved in smartcard mode. This bit field cannot be written when the USART is enabled (UEN=1).

17. Inter-integrated circuit interface (I2C)

17.1. 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 and fast-mode as well as CRC calculation and checking, 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.

17.2. Characteristics

- Parallel-bus to I2C-bus protocol converter and interface.
- Both master and slave functions with the same interface.
- Bi-directional data transfer between master and slave.
- Supports 7-bit and 10-bit addressing and general call addressing.
- Multi-master capability.
- Supports standard-mode (up to 100 kHz) and fast-mode (up to 400 kHz).
- Configurable SCL stretching in slave mode.
- Supports DMA mode.
- SMBus 2.0 and PMBus compatible.
- 2 Interrupts: one for successful byte transmission and the other for error event.
- Optional PEC (packet error checking) generation and check.

17.3. Function overview

Figure 17-1. I2C module block diagram below provides details on the internal configuration of the I2C interface.

Figure 17-1. I2C module block diagram

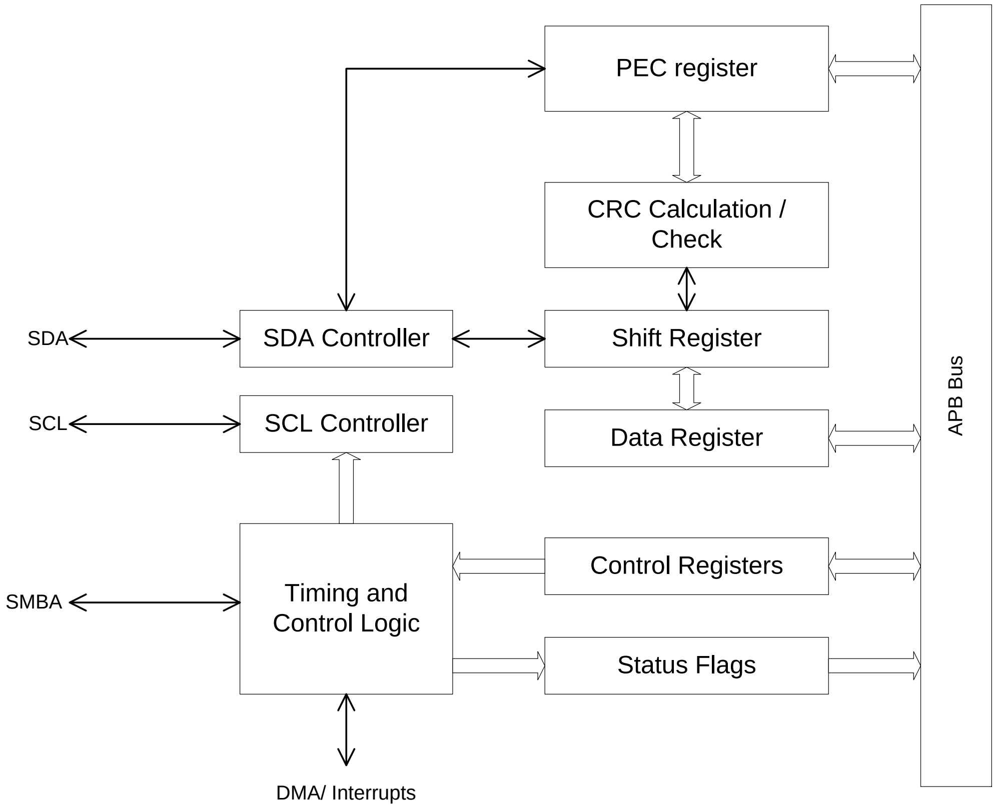


Table 17-1. Definition of I2C-bus terminology (refer to the I2C specification of philips semiconductors)

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master
Multi-master	more than one master can attempt to control the bus at the same time without corrupting the message
Synchronization	procedure to synchronize the clock signals of two or more devices
Arbitration	procedure to ensure that, if more than one master tries to control the bus simultaneously, only one is allowed to do so and the winning master’s message is not corrupted

17.3.1. SDA and SCL lines

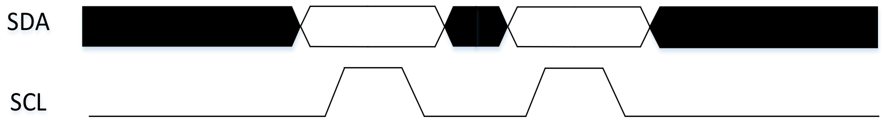
The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus.

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collect to perform the wired-AND function. Data on the I2C-bus can be transferred at rates of up to 100 kbit/s in the standard-mode and up to 400 kbit/s in the fast-mode. Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the voltage levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of V_{DD}.

17.3.2. Data validation

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see ***Figure 17-2. Data validation)***. One clock pulse is generated for each data bit transferred.

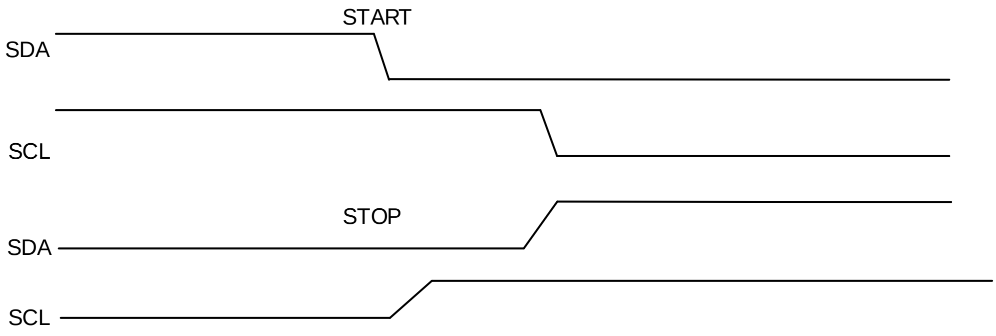
Figure 17-2. Data validation



17.3.3. START and STOP condition

All transactions begin with a START (S) and are terminated by a STOP (P) (see ***Figure 17-3. START and STOP condition***). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

Figure 17-3. START and STOP condition

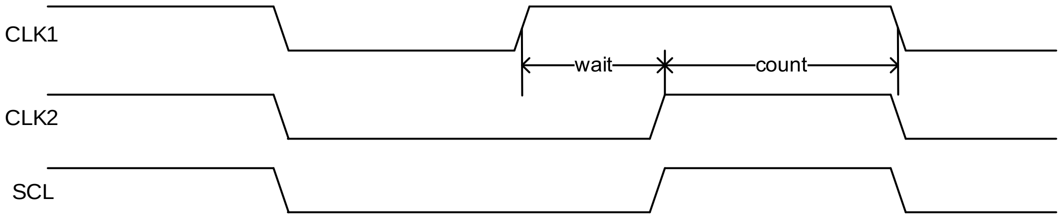


17.3.4. Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which master takes control of the bus and complete its transmission. This is done by clock synchronization and bus arbitration. In a single master system, clock synchronization and bus arbitration are unnecessary.

Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the masters concerned to start counting off their LOW period and, once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see ***Figure 17-4. Clock synchronization***). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the master with the longest LOW period. Masters with shorter LOW periods enter a HIGH wait-state during this time.

Figure 17-4. Clock synchronization



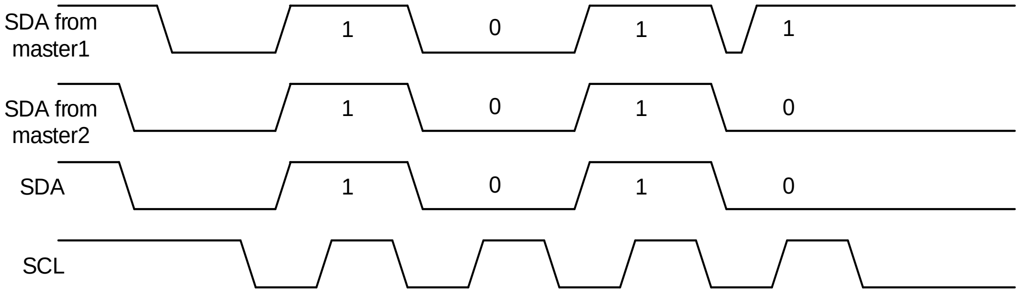
17.3.5. Arbitration

Arbitration, like synchronization, is part of the protocol where more than one master is used in the system. Slaves are not involved in the arbitration procedure.

A master may start a transfer only if the bus is free. Two masters may generate a START condition within the minimum hold time of the START condition which results in a valid START condition on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see whether the SDA level matches what it has sent. This process may take many bits. Two masters can evencomplete an entire transaction without error, as long as the transmissions are identical. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, then the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.

Figure 17-5. SDA Line arbitration



17.3.6. I2C communication flow

Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device.

An I2C slave will continue to detect addresses after a START condition on I2C bus and compare the detected address with its slave address which is programmable by software. Once the two addresses match, the I2C slave will send an ACK to the I2C bus and responses to the following command on I2C bus: transmitting or receiving the desired data. Additionally, if General Call is enabled by software, the I2C slave always responses to a General Call Address (0x00). The I2C block support both 7-bit and 10-bit address modes.

An I2C master always initiates or end a transfer using START or STOP condition and it's also responsible for SCL clock generation.

Figure 17-6. I2C communication flow with 7-bit address

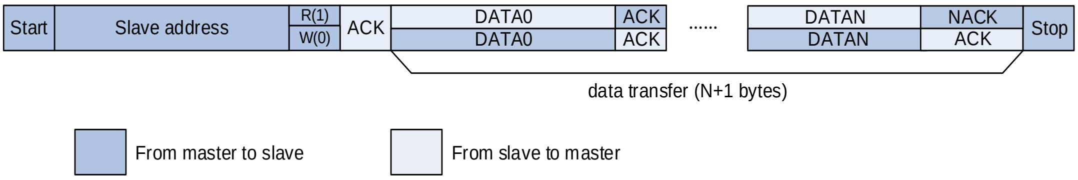


Figure 17-7. I2C communication flow with 10-bit address (Master Transmit)

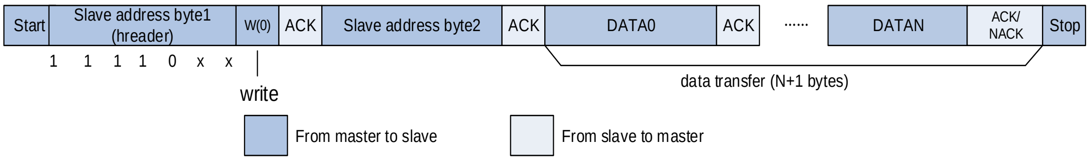
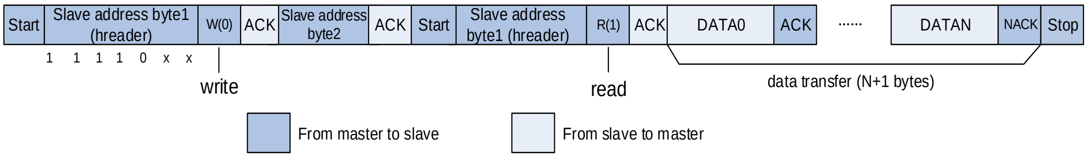


Figure 17-8. I2C communication flow with 10-bit address (Master Receive)



17.3.7. Programming model

An I2C device such as LCD driver may only be a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- Master Transmitter.
- Master Receiver.
- Slave Transmitter.
- Slave Receiver.

I2C block supports all of the four I2C modes. After system reset, it works in slave mode. If it's programmed by software and finished sending a START condition on I2C bus, it changes into master mode. The I2C changes back to slave mode after it's programmed by software and finished sending a STOP condition on I2C bus.

Programming model in slave transmitting mode

As is shown in the figure below, the following software procedure should be followed if users wish to make transaction in slave transmitter mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.

2. After receiving a START condition followed by a matched address, either in 7-bit format or in 10-bit format, the I2C hardware sets the ADDSEND bit in I2C_STAT0 register, which should be monitored by software either by polling or interrupt. After that software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. If 10-bit addressing format is selected, the I2C master should then send a repeated START(Sr) condition followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START(Sr) condition and the following header. Software needs to clear the ADDSEND bit again by reading I2C_STAT0 and then I2C_STAT1.

3. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Once TBE is set, Software should write the first byte of data to I2C_DATA register, TBE is not cleared in this case because the write byte in I2C_DATA is moved to the internal shift register immediately. I2C begins to transmit data to I2C bus as soon as the shift register is not empty.

4. During the first byte's transmission, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

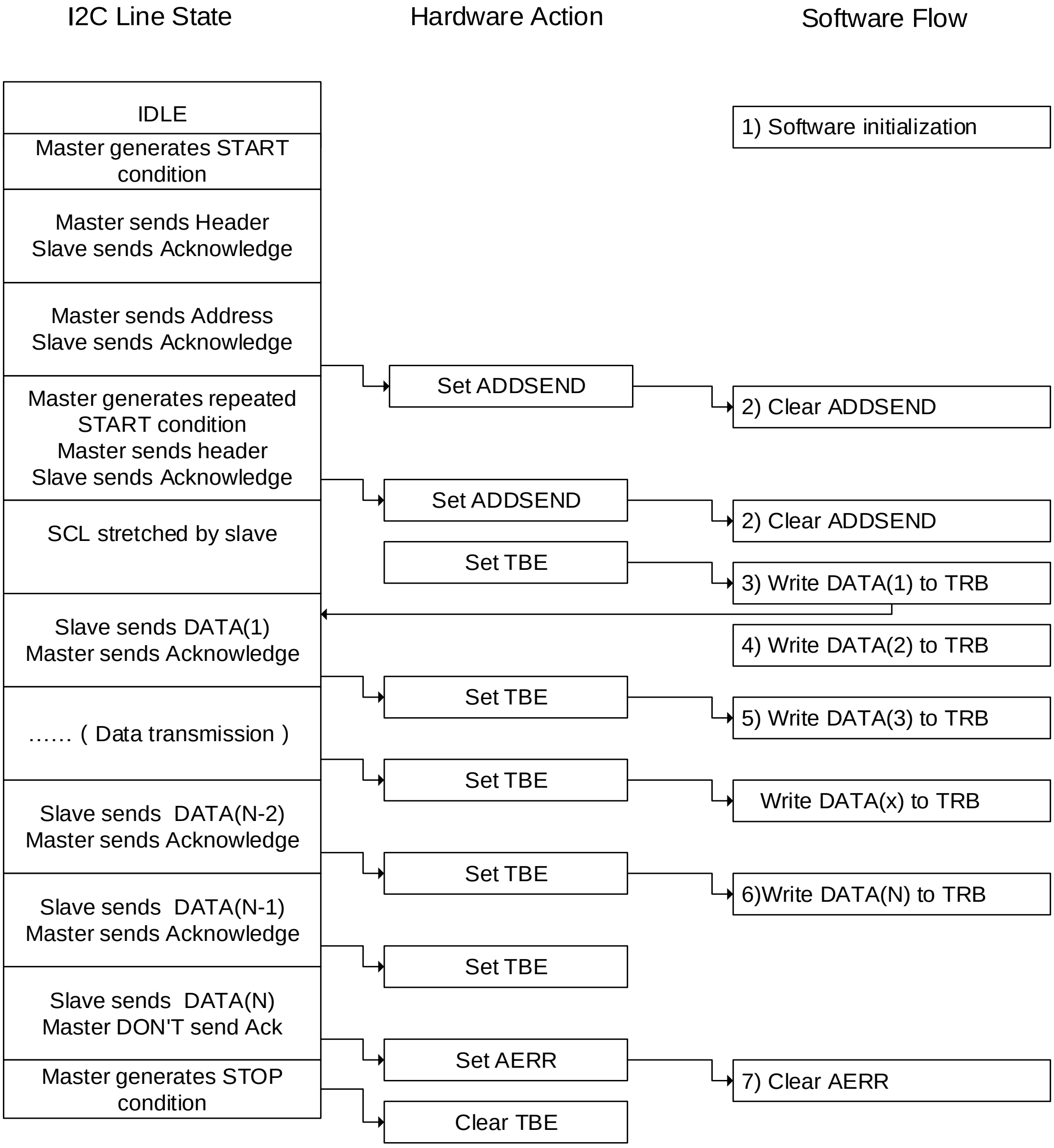
5. Any time TBE is set, software can write a byte to I2C_DATA as long as there are still data to be transmitted.

6. During the second last byte's transmission, software write the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be set after the byte's transmission and not cleared until a STOP condition.

7. I2C master doesn't acknowledge to the last byte according to the I2C protocol, so after sending the last byte, I2C slave will wait for the STOP condition on I2C bus and sets AERR (Acknowledge Error) bit to notify software that transmission completes. Software

clears AERR bit by writing 0 to it.

Figure 17-9. Programming model for slave transmitting



Programming model in slave receiving mode

As is shown in the figure below, the following software procedure should be followed if users wish to make reception in slave receiver mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.

2. After receiving a START condition followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register, which should be monitored by software either by polling or interrupt. After that software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. The I2C begins to receive data to I2C bus as

soon as ADDSEND bit is cleared.

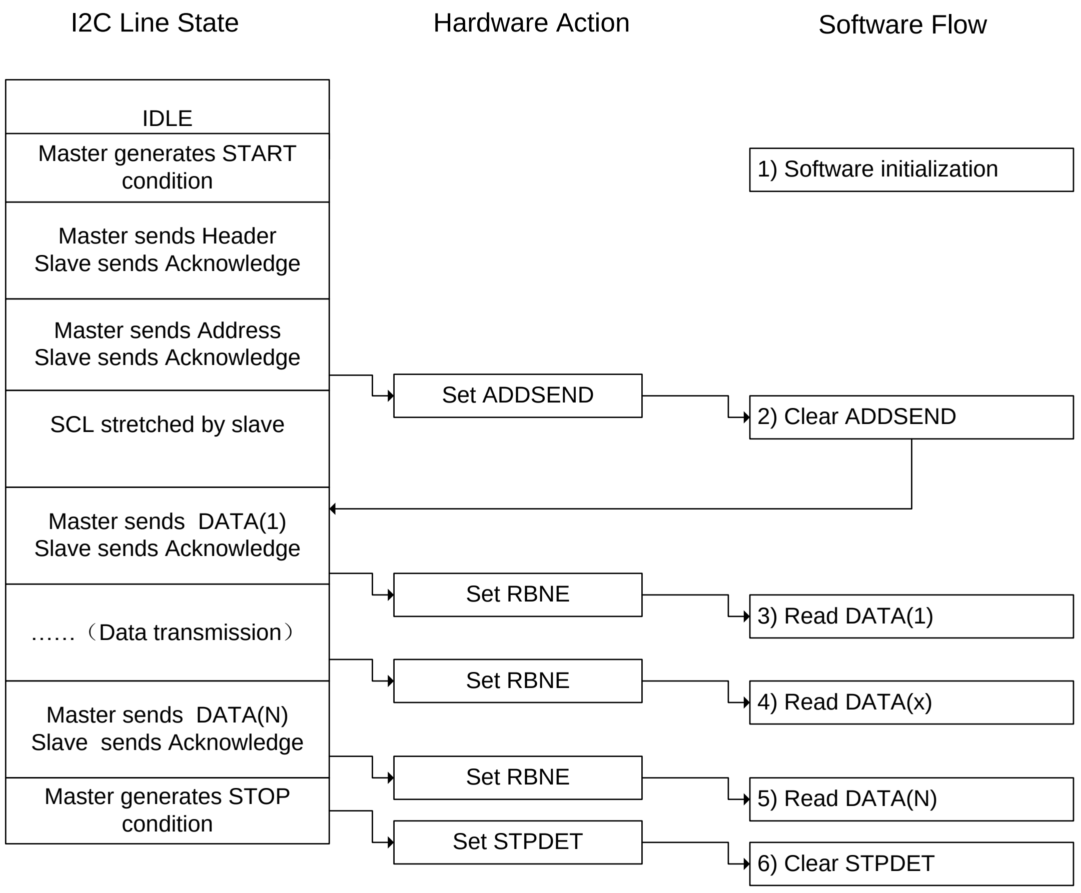
3. As soon as the first byte is received, RBNE is set by hardware. Software can now read the first byte from I2C_DATA and RBNE is cleared as well.

4. Any time RBNE is set, software can read a byte from I2C_DATA.

5. After last byte is received, RBNE is set. Software reads the last byte.

6. STPDET bit is set when I2C detects a STOP condition on I2C bus and software reads I2C_STAT0 and then write I2C_CTL0 to clear the STPDET bit.

Figure 17-10. Programming model for slave receiving


I2C Line State
Hardware Action
Software Flow
IDLE
Master generates START condition
Master sends Header Slave sends Acknowledge
Master sends Address Slave sends Acknowledge
SCL stretched by slave
Master sends DATA(1) Slave sends Acknowledge
……（Data transmission）
Master sends DATA(N) Slave sends Acknowledge
Master generates STOP condition
Set ADDSEND
Set RBNE
Set RBNE
Set RBNE
Set STPDET
1) Software initialization
2) Clear ADDSEND
3) Read DATA(1)
4) Read DATA(x)
5) Read DATA(N)
6) Clear STPDET

Programming model in master transmitting mode

As it shows in figure below, the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.

3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10S END bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.

4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1.

5. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Software now write the first byte data to I2C_DATA register, but the TBE is not cleared because the write byte in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as shift register is not empty.

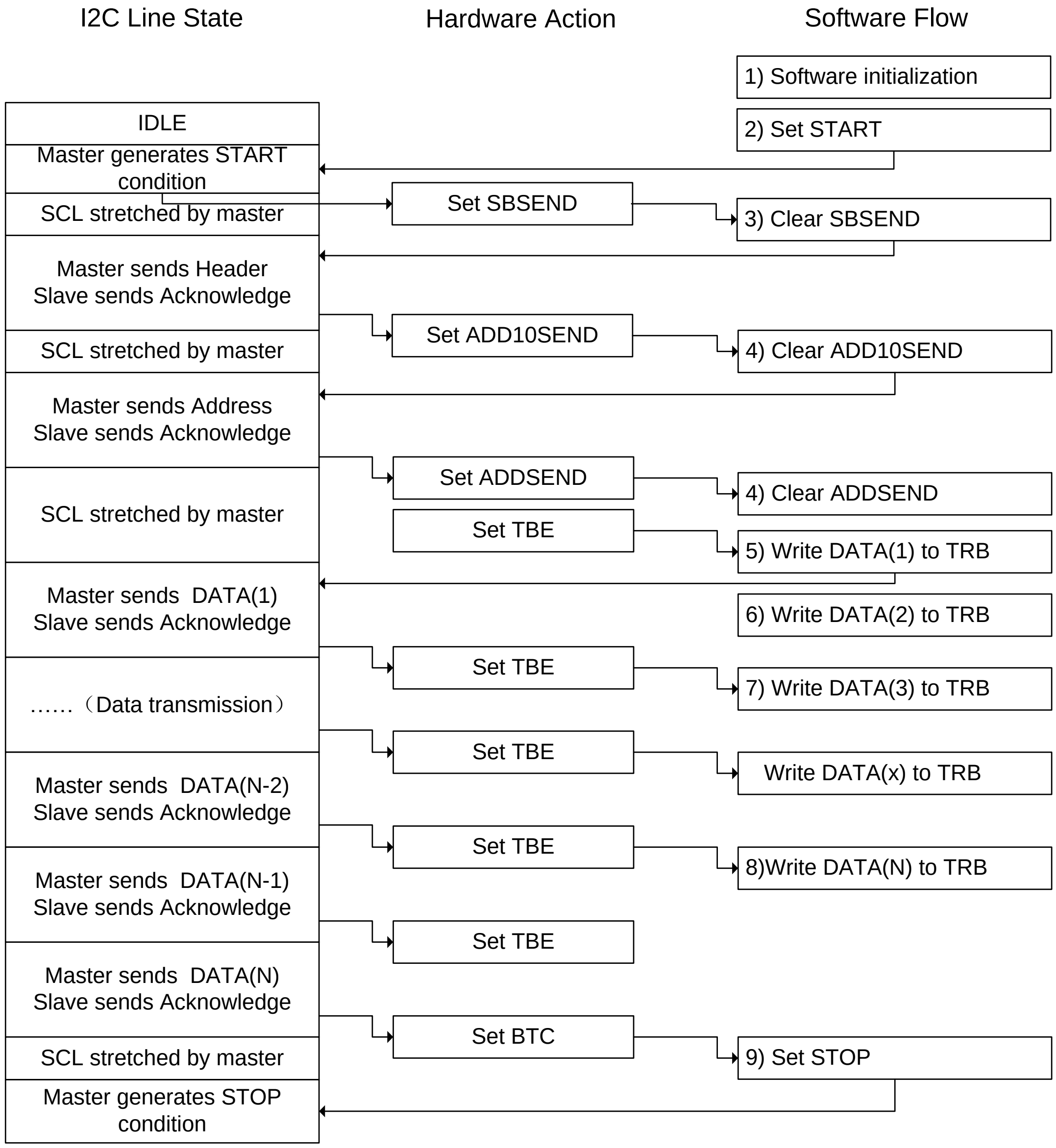
6. During the first byte's transmission, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

7. Any time TBE is set, software can write a byte to I2C_DATA as long as there are still data to be transmitted.

8. During the second last byte's transmission, software write the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the byte's transmission and not cleared until a STOP condition.

9. After sending the last byte, I2C master sets BTC bit because both shift register and I2C_DATA are empty. Software should program a STOP request now, and the I2C clears both TBE and BTC flags after sending a STOP condition.

Figure 17-11. Programming model for master transmitting



Programming model in master receiving mode

In master receiving mode, a master is responsible for generating NACK for the last byte reception and then sending STOP condition on I2C bus. So, special attention should be paid to ensure the correct ending of data reception. Two solutions for master receiving are provided here for your application: Solution A and B. Solution A requires the software's quick response to I2C events, while Solution B doesn't.

Solution A

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address

on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.

3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.

4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START condition on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header to I2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.

5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.

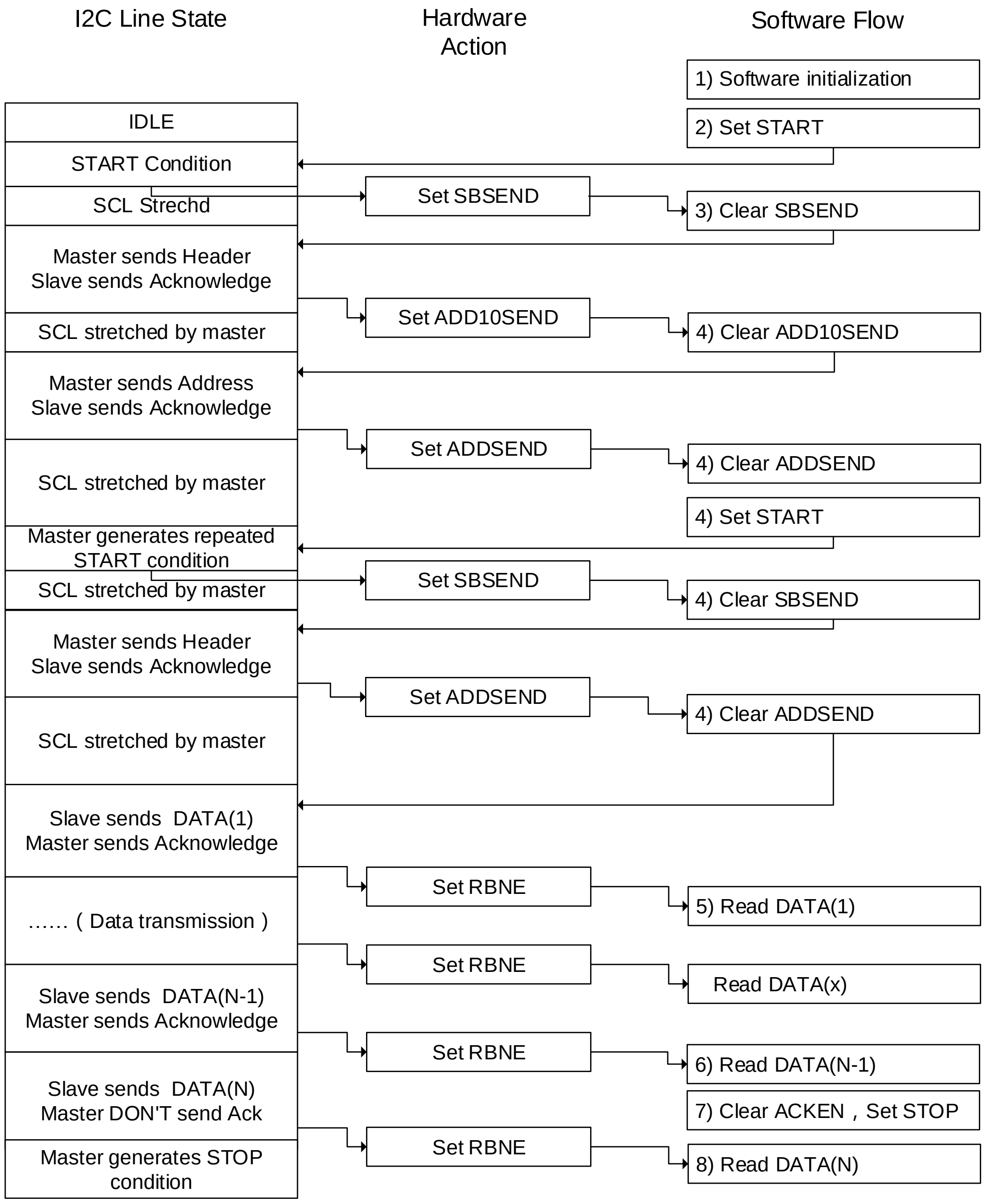
6. Any time RBNE is set, software can read a byte from I2C_DATA.

7. After the second last byte is received, the software should clear ACKEN bit and set STOP bit. These actions should complete before the end of the last byte’s receiving to ensure that NACK is sent for the last byte.

8. After last byte is received, RBNE is set. Software reads the last byte. I2C doesn’t send ACK to the last byte and generate a STOP condition after the transmission of the last byte.

Above steps require byte number N>1. If N=1, Step 7 should be performed after Step 4 and completed before the end of the single byte’s receiving.

Figure 17-12. Programming model for master receiving using Solution A



Solution B

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.
2. Software set START bit requesting I2C to generate a START condition to I2C bus.
3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address toI2C_DATA.

I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address toI2C_DATA.

4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set STARTbit again to generate a repeated START condition on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header toI2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C_DATA until the master receives N-3 bytes.

As shown in ***Figure 17-13. Programming model for master receiving using solution B***, the N-2 byte is not read out by software, so after the N-1 byte is received, both BTC and RBNE are asserted. The bus is stretched by master to prevent the reception of the last byte. Then software should clear ACKEN bit.

7. Software reads out N-2 byte, clearing BTC. After this the N-1 byte is moved from shift register to I2C_DATA and bus is released and begins to receive the last byte.
8. After last byte is received, both BTC and RBNE is set again. Software sets STOP bit and master sends out a STOP condition on bus.
9. Software reads the N-1 byte, clearing BTC. After this the last byte is moved from shift register to I2C_DATA.
10. Software reads the last byte, clearing RBNE.

Above steps require that byte number N>2. N=1 or N=2 are similar:

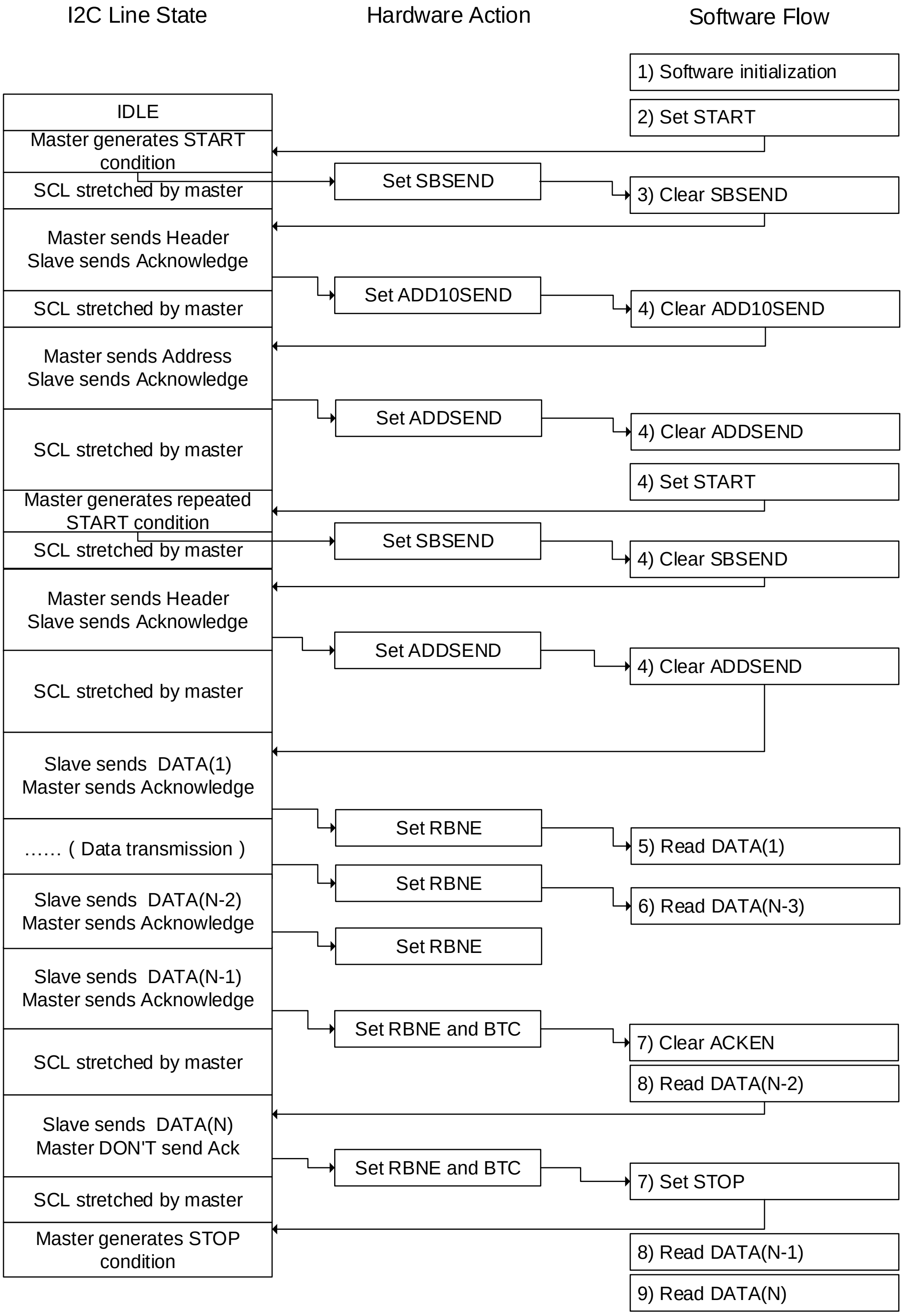
N=1

In Step4, software should reset ACK bit before clearing ADDSEND bit and set STOP bit after clearing ADDSEND bit. Step 5 is the last step when N=1.

N=2

In Step 2, software should set POAP bit before set START bit. In Step 4, software should reset ACKEN bit before clearing ADDSEND bit. In Step 5, software should wait until BTC is set and then set STOP bit and reads I2C_DATA twice.

Figure 17-13. Programming model for master receiving using solution B



17.3.8. SCL line stretching

The SCL line stretching function is designed to avoid overflow error in reception and underflow error in transmission. As is shown in Programming Model, when the TBE and BTC bit of a transmitter is set, the transmitter stretches the SCL line low until the transfer buffer register is filled with the next transmit data. When the RBNE and BTC bit of a receiver is set, the receiver stretches the SCL line low until the data in the transfer buffer is read out.

When works in slave mode, the SCL line stretching function can be disabled by setting the DISSTRC bit in the I2C_CTL0 register. If this bit is set, the software is required to be quick enough to serve the TBE, RBNE and BTC status, otherwise, overflow or underflow situation might occur.

17.3.9. Use DMA for data transfer

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU's high overload. The DMA controller can be used to process TBE and RBNE flag: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically.

The DMA request is enabled by the DMAON bit in the I2C_CTL1 register. This bit should be set after clearing the ADDSEND status. If the SCL line stretching function is disabled for a slave device, the DMAON bit should be set before the ADDSEND event.

Refer to the specification of the DMA controller for the configuration method of a DMA stream. The DMA controller must be configured and enabled before I2C transfer. When the configured number of byte has been transferred, the DMA controller generates End of Transfer (EOT) interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C_CTL1 register should be set. The I2C master will not send nack after the last byte. The software can set the STOP bit to generate a stop condition in the ISR of the DMA EOT interrupt.

When a master receives only one byte, the ACKEN bit must be cleared before clearing the ADDSEND status. Software can set the STOP bit to generate a stop condition after clearing the ADDSEND status, or in the ISR of the DMA EOT interrupt.

17.3.10. Packet error checking

There is a CRC-8 calculator in I2C block to perform Packet Error Checking for I2C data. The polynomial of the CRC is x8 + x2 + x + 1 which is compatible with the SMBus protocol. If enabled by setting PECEN bit, the PEC will calculate all the data transmitted through I2C including address. I2C is able to send out the PEC value after the last data byte or check the received PEC value with its calculated PEC using the PECTRANS bit. In DMA mode, the I2C will send or check PEC value automatically if PECEN bit is set.

17.3.11. SMBus support

The System Management Bus (abbreviated to SMBus or SMB) is a single-ended simple two-wire bus for the purpose of lightweight communication. Most commonly it is found in computer motherboards for communication with power source for ON/OFF instructions.It is derived from I2C for communication with low-bandwidth devices on a motherboard, especially power related chips such as a laptop's rechargeable battery subsystem (see Smart Battery Data).

SMBus protocol

Each message transaction on SMBus follows the format of one of the defined SMBus protocols. The SMBus protocols are a subset of the data transfer formats defined in the I2C specifications. I2C devices that can be accessed through one of the SMBus protocols are compatible with the SMBus specifications. I2C devices that do not adhere to these protocols cannot be accessed by standard methods as defined in the SMBus and Advanced Configuration and Power Management Interface (abbreviated to ACPI) specifications.

Address resolution protocol

The SMBus uses I2C hardware and I2C hardware addressing, but adds second-level software for building special systems. Additionally, its specifications include an Address Resolution Protocol that can make dynamic address allocations. Dynamic reconfiguration of the hardware and software allow bus devices to be ‘hot-plugged’ and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In both those protocols there is a very useful distinction made between a System Host and all the other devices in the system that can have the names and functions of masters or slaves.

Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I2C can be a ‘DC’ bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This will notify to the master that the slave is busy but does not want to lose the communication. The slave device will allow continuation after its task is completed. There is no limit in the I2C bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 35ms. SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to clear this mode. Slave devices are not allowed to hold the clock low too long.

Packet error checking

SMBus 2.0 and 1.1 allow Packet Error Checking (PEC). In that mode, a PEC (packet error code) byte is appended at the end of each transaction. The byte is calculated as CRC-8 checksum, calculated over the entire message including the address and read/write bit. The polynomial used is x8+x2+x+1 (the CRC-8-ATM HEC algorithm, initialized to zero).

SMBus alert

The SMBus has an extra optional shared interrupt signal called SMBALERT# which can be used by slaves to tell the host to ask its slaves about events of interest. SMBus also defines a less common "Host Notify Protocol", providing similar notifications but passing more data and building on the I2C multi-master mode.

SMBus programming flow

The programming flow for SMBus is similar to normal I2C. In order to use SMBus mode, the application should configure several SMBus specific registers, response to some SMBus specific flags and implement the upper protocols described in SMBus specification.

1. Before communication, SMBEN bit in I2C_CTL0 should be set and SMBSEL and ARPEN bits should be configured to desired value.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should response to HSTSMB flag in SMBus Host Mode (SMBTYPE =1) or DEFSMB flag in SMBus Device Mode, and implement the function of ARP protocol.
3. In order to support SMBus Alert Mode, the software should response to SMBALT flag and implement the related function.

17.3.12. Status, errors and interrupts

There are several status and error flags in I2C, and interrupt may be asserted from these flags by setting some register bits (refer to I2C register for detail).

Table17-2. Event status flags

Event Flag Name	Description
SBSEND	START condition sent (master)
ADDSEND	Address sent or received
ADD10SEND	Header of 10-bit address sent
STPDET	STOP condition detected
BTC	Byte transmission completed
TBE	I2C_DATA is empty when transmitting
RBNE	I2C_DATA is not empty when receiving

Table17-3. I2C error flags

I2C Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Over-run or under-run when SCL stretch is disabled.
AERR	No acknowledge received
PECERR	CRC value doesn’t match

I2C Error Name	Description
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

17.4. Register definition

17.4.1. Control register 0 (I2C_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRESET	Reserved	SALT	PECTRANS	POAP	ACKEN	STOP	START	SS	GCEN	PECEN	ARPEN	SMBSEL	Reserved	SMBEN	I2CEN
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits	Fields	Descriptions
15	SRESET	Software reset I2C, software should wait until the I2C lines are released to reset the I2C 0: I2C is not under reset 1: I2C is under reset
14	Reserved	Must be kept the reset value
13	SALT	SMBus Alert. Issue alert through SMBA pin. Software can set and clear this bit and hardware can clear this bit. 0: Don’t issue alert through SMBA pin 1: Issue alert through SMBA pin
12	PECTRANS	PEC Transfer Software set and clear this bit while hardware clears this bit when PEC is transferred or START/STOP condition detectedor I2CEN=0 0: Don’t transfer PEC value 1: Transfer PEC
11	POAP	Position of ACK and PEC when receiving This bit is set and cleared by software and cleared by hardware when I2CEN=0 0: ACKEN bit specifies whether to send ACK or NACK for the current byte that is being received. PECTRANS bit indicates that the current receiving byte is a PEC byte 1: ACKEN bit specifies whether to send ACK or NACK for the next byte that is to be received, PECTRANS bit indicates the next byte that is to be received is a PEC byte
10	ACKEN	Whether or not to send an ACK This bit is set and cleared by software and cleared by hardware when I2CEN=0

		0: ACK will not be sent 1: ACK will be sent
9	STOP	Generate a STOP condition on I2C bus This bit is set and cleared by software and set by hardware when SMBUs timeout and cleared by hardware when STOP condition detected. 0: STOP will not be sent 1: STOP will be sent
8	START	Generate a START condition on I2C bus This bit is set and cleared by software and and cleared by hardware when START condition detected or I2CEN=0 0: START will not be sent 1: START will be sent
7	SS	Whether to stretch SCL low when data is not ready in slave mode. This bit is set and cleared by software. 0: SCL Stretching is enabled 1: SCL Stretching is disabled
6	GCEN	Whether or not to response to a General Call (0x00) 0: Slave won’t response to a General Call 1: Slave will response to a General Call
5	PECEN	PEC Calculation Switch 0: PEC Calculation off 1: PEC Calculation on
4	ARPEN	ARP protocol in SMBus switch 0: ARP is disabled 1: ARP is enabled
3	SMBSEL	SMBusType Selection 0: Device 1: Host
2	Reserved	Must keep the reset value
1	SMBEN	SMBus/I2C mode switch 0: I2C mode 1: SMBus mode
0	I2CEN	I2C peripheral enable 0: I2C is disabled 1: I2C is enabled

17.4.2. Control register 1 (I2C_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMALST	DMAON	BUFIE	EVIE	ERRIE	Reserved		I2CCLK[5:0]					
			rw	rw	rw	rw	rw			rw					

Bits	Fields	Descriptions
15:13	Reserved	Must be kept the reset value
12	DMALST	Flag indicating DMA last transfer 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer
11	DMAON	DMA mode switch 0: DMA mode disabled 1: DMA mode enabled
10	BUFIE	Buffer interrupt enable 0: No interrupt asserted when TBE = 1 or RBNE = 1 1: Interrupt asserted when TBE = 1 or RBNE = 1 if EVIE=1
9	EVIE	Event interrupt enable 0: Event interrupt disabled 1: Event interrupt enabled, means that interrupt will be generated when SBSEND, ADDSEND, ADD10SEND, STPDET or BTC flag asserted or TBE=1 or RBNE=1 if BUFIE=1.
8	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled, means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag asserted.
7:6	Reserved	Must be kept the reset value
5:0	I2CCLK[5:0]	I2C Peripheral clock frequency I2CCLK[5:0]should be the frequency of input APB1 clock in MHz which is at least 2. 000000 - 000001: Not allowed 000010 - 110110: 2 MHz~54MHz 110111 - 111111: Not allowed due to the limitation of APB1 clock **Note:** In I2C standard mode, the frequencies of APB1 must be equal or greater than

2MHz. In I2C fast mode, the frequencies of APB1 must be equal or greater than 8MHz.

17.4.3. Slave address register 0 (I2C_SADDR0)

Address offset: 0x08
Reset value: 0x0000

This register can be accessed by word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDFOR MAT	Reserved					ADDRESS[9:8]		ADDRESS[7:1]							ADDRES S0
rw						rw		rw							rw

Bits	Fields	Descriptions
15	ADDFORMAT	Address mode for the I2C slave 0: 7-bit Address 1: 10-bit Address
14:10	Reserved	Must be kept the reset value
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address
7:1	ADDRESS[7:1]	7-bit address or bits 7:1 of a 10-bit address
0	ADDRESS0	Bit 0 of a 10-bit address

17.4.4. Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C
Reset value: 0x0000

This register can be accessed by word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADDRESS2[7:1]							DUADEN
								rw							rw

Bits	Fields	Descriptions
15:8	Reserved	Must be kept the reset value
7:1	ADDRESS2[7:1]	Second I2C address for the slave in Dual-Address mode
0	DUADEN	Dual-Address mode switch 0: Dual-Address mode disabled

1: Dual-Address mode enabled

17.4.5. Transfer buffer register (I2C_DATA)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by word (32-bit)

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

Bits	Fields	Descriptions
15:8	Reserved	Must be kept the reset value
7:0	TRB[7:0]	Transmission or reception data buffer

17.4.6. Transfer status register 0 (I2C_STAT0)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed byhalf-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALT	SMBTO	Reserved	PECERR	OUERR	AERR	LOSTARB	BERR	TBE	RBNE	Reserved	STPDET	ADD10SEND	BTC	ADDSEND	SBSEND
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bits	Fields	Descriptions
15	SMBALT	SMBus Alert status This bit is set by hardware and cleared by writing 0. 0: SMBA pin not pulled down (device mode) or no Alert detected (host mode) 1: SMBA pin pulled down (device mode) or Alert detected (host mode)
14	SMBTO	Timeout signal in SMBus mode This bit is set by hardware and cleared by writing 0. 0: No timeout error 1: Timeout event occurs (SCL is low for 25 ms)
13	Reserved	Must keep the reset value
12	PECERR	PEC error when receiving data This bit is set by hardware and cleared by writing 0.

		0: Received PEC and calculated PEC match 1: Received PEC and calculated PEC don’t match, I2C will send NACK careless of ACKEN bit.
11	OUERR	Over-run or under-run situation occurs in slave mode, when SCL stretching is disabled. In slave receiving mode, if the last byte in I2C_DATA is not read out while the following byte is already received, over-run occurs. In slave transmitting mode, if the current byte is already sent out, while the I2C_DATA is still empty, under-run occurs. This bit is set by hardware and cleared by writing 0. 0: No over-run or under-run occurs 1: Over-run or under-run occurs
10	AERR	Acknowledge Error This bit is set by hardware and cleared by writing 0. 0: No Acknowledge Error 1: Acknowledge Error
9	LOSTARB	Arbitration Lost in master mode This bit is set by hardware and cleared by writing 0. 0: No Arbitration Lost 1: Arbitration Lost occurs and the I2C block changes back to slave mode.
8	BERR	A bus error occurs indication a unexpected START or STOP condition on I2C bus This bit is set by hardware and cleared by writing 0. 0: No bus error 1: A bus error detected
7	TBE	I2C_DATA is Empty during transmitting This bit is set by hardware after it moves a byte from I2C_DATA to shift register and cleared by writing a byte to I2C_DATA. If both the shift register and I2C_DATA are empty, writing I2C_DATA won’t clear TBE (refer to Programming Model for detail). 0: I2C_DATA is not empty 1: I2C_DATA is empty, software can write
6	RBNE	I2C_DATA is not Empty during receiving This bit is set by hardware after it moves a byte from shift register to I2C_DATA and cleared by reading I2C_DATA. If both BTC and RBNE are asserted, reading I2C_DATA won’t clear RBNE because the shift register’s byte is moved to I2C_DATA immediately. 0: I2C_DATA is empty 1: I2C_DATA is not empty, software can read
5	Reserved	Must be kept the reset value
4	STPDET	STOP condition detected in slave mode

		This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0 0: STOP condition not detected in slave mode 1: STOP condition detected in slave mode
3	ADD10SEND	Header of 10-bit address is sent in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA. 0: No header of 10-bit address sent in master mode 1: Header of 10-bit address is sent in master mode
2	BTC	Byte transmission completed. If a byte is already received in shift register but I2C_DATA is still full in receiving mode or a byte is already sent out from shift register but I2C_DATA is still empty in transmitting mode, the BTC flag is asserted. This bit is set by hardware. This bit can be cleared by 3 ways as follow: Reading I2C_STAT0 followed by reading or writing Hardware clearing: sending the STOP condition or START condition Bit 0 (I2CEN bit) of the I2C_CTL0 is reset. 0: BTC not asserted 1: BTC asserted
1	ADDSEND	Address is sent in master mode or received and matches in slave mode. This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1. 0: No address sent or received 1: Address sent out in master mode or a matched address is received in salve mode
0	SBSEND	START condition sent out in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA 0: No START condition sent 1: START condition sent

17.4.7. Transfer status register 1 (I2C_STAT1)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PECV[7:0]								DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TR	I2CBSY	MASTER

r r r r r r r r

Bits	Fields	Descriptions
15:8	PECV[7:0]	Packet Error Checking Value that calculated by hardware when PEC is enabled.
7	DUMODF	Dual Flag in slave mode indicating which address is matched in Dual-Address mode This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 0: OAR1 address matches 1: OAR2 address matches
6	HSTSMB	SMBus Host Header detected in slave mode This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 0: No SMBus Host Header detected 1: SMBus Host Header detected
5	DEFSMB	Default address of SMBusDevice This bit is cleared by hardware after a STOP or a START condition or I2CEN=0. 0: Thedefault address has not beenreceived 1: Thedefault address has been received for SMBus Device
4	RXGC	General call address (00h) received. This bit is cleared by hardware after a STOP or a START condition or I2CEN=0. 0: No general call address (00h) received 1: General call address (00h) received
3	Reserved	Must be kept the reset value
2	TR	Whether the I2C is a transmitter or a receiver This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB=1. 0: Receiver 1: Transmitter
1	I2CBSY	Busy flag This bit is cleared by hardware after a STOP condition 0: No I2C communication. 1: I2C communication active.
0	MASTER	A flag indicating whether I2C block is in master or slave mode. This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB=1. 0: Slave mode 1: Master mode

17.4.8. Clock configure register (I2C_CKCFG)

Address offset: 0x1C
Reset value: 0x0000

This register can be accessed by word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAST	DTCY	Reserved		CLKC[11:0]											
rw	rw			rw											

Bits	Fields	Descriptions
15	FAST	I2C speed selection in master mode 0: Standard speed 1: Fast speed
14	DTCY	Duty cycle in fast mode 0: $T_{low}/T_{high} = 2$ 1: $T_{low}/T_{high} = 16/9$
13:12	Reserved	Must be kept the reset value
11:0	CLKC[11:0]	I2C Clock control in master mode In standard speed mode: $T_{high} = T_{low} = CLKC * T_{PCLK1}$ In fast speed mode, if DTCY=0: $T_{high} = CLKC * T_{PCLK1}$, $T_{low} = 2 * CLKC * T_{PCLK1}$ In fast speed mode, if DTCY=1: $T_{high} = 9 * CLKC * T_{PCLK1}$, $T_{low} = 16 * CLKC * T_{PCLK1}$

17.4.9. Rise time register (I2C_RT)

Address offset: 0x20
Reset value: 0x0002

This register can be accessed by word (32-bit)

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

Bits	Fields	Descriptions
15:6	Reserved	Must be kept the reset value
5:0	RISETIME[5:0]	Maximum rise time in master mode The RISETIME value should be the maximum SCL rise time incremented by 1.

18. Serial peripheral interface/Inter-IC sound (SPI/I2S)

18.1. Overview

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol.

The Serial Peripheral Interface (SPI) provides a SPI protocol of data transmission and reception function in master or slave mode. Both full-duplex and simplex communication modes are supported, with hardware CRC calculation and checking.

The inter-IC sound (I2S) supports four audio standards: I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. I2S works at either master or slave mode for transmission and reception.

18.2. Characteristics

18.2.1. SPI characteristics

- Master or slave operation with full-duplex or simplex mode.
- Separate transmit and receive buffer, 16 bits wide.
- Data frame size can be 8 or 16 bits.
- Bit order can be LSB first or MSB first.
- Software and hardware NSS management.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.

18.2.2. I2S characteristics

- Master or slave operation with transmission or reception mode.
- Four I2S standards supported: Phillips, MSB justified, LSB justified and PCM standard.
- Data length can be 16 bits, 24 bits or 32 bits.
- Channel length can be 16 bits or 32 bits.
- Transmission and reception using a 16 bits wide buffer.
- Audio sample frequency can be 8 kHz to 192 kHz using I2S clock divider.
- Programmable idle state clock polarity.
- Master clock (MCK) can be output.
- Transmission and reception using DMA.

18.3. SPI block diagram

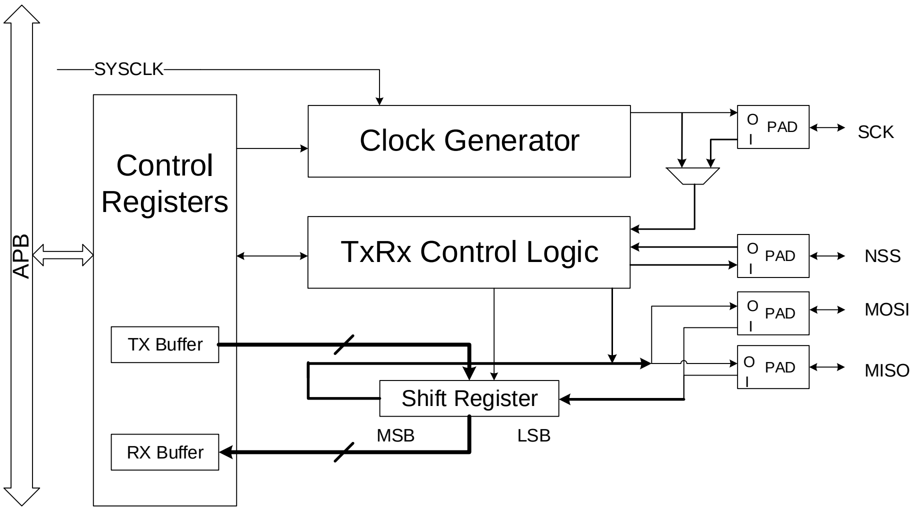


Figure 18-1. Block diagram of SPI

18.4. SPI signal description

18.4.1. Normal configuration

Table 18-1. SPI signal description

Pin Name	Direction	Description
SCK	I / O	Master: SPI Clock Output Slave: SPI Clock Input
MISO	I / O	Master: Data reception line Slave: Data transmission line Master with Bidirectional mode: Not used Slave with Bidirectional mode: Data transmission and reception Line.
MOSI	I / O	Master: Data transmission line Slave: Data reception line Master with Bidirectional mode: Data transmission and reception Line. Slave with Bidirectional mode: Not used
NSS	I / O	Software NSS Mode: Not Used Master in Hardware NSS Mode: NSS output (NSSDRV=1) for single master or (NSSDRV=0) for multi-master application. Slave in Hardware NSS Mode: NSS input, as a chip select signal for slave.

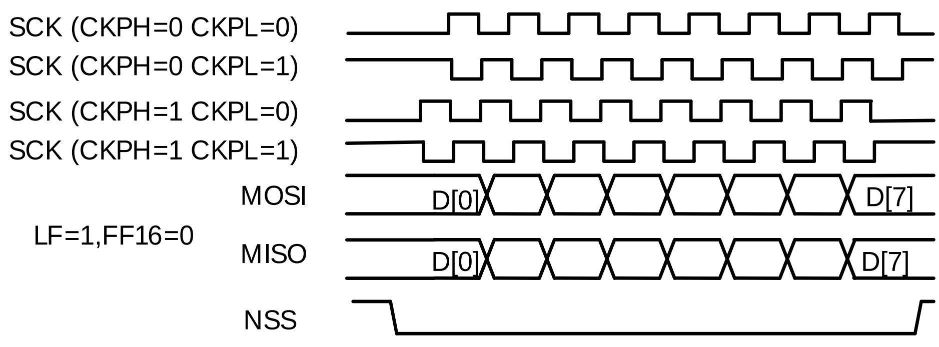
Note: The pin as input must be configured as in floating mode.

18.5. SPI function overview

18.5.1. SPI clock timing and data format

CKPL and CKPH bits in SPI_CTL0 register decide the timing of SPI clock and data signal. The CKPL bit decides the SCK level when idle and CKPH bit decides either first or second clock edge is a valid sampling edge.

Figure 18-2. SPI timing diagram in normal mode



In normal mode, the length of data is configured by the FF16 bit in the SPI_CTL0 register. Data length is 16 bits if FF16=1, otherwise is 8 bits.

Data order is configured by LF bit in SPI_CTL0 register, and SPI will first send the LSB if LF=1, or the MSB if LF=0.

18.5.2. NSS function

Slave Mode

When slave mode is configured (MSTMOD=0), SPI gets NSS level from NSS pin in hardware NSS mode (SWNSSEN = 0) or from SWNSS bit in software NSS mode (SWNSSEN = 1) and transmits/receives data only when NSS level is low. In software NSS mode, NSS pin is not used.

Master mode

In master mode (MSTMOD=1) if the application uses multi-master connection, NSS can be configured to hardware input mode (SWNSSEN=0, NSSDRV=0) or software mode (SWNSSEN=1). Then, once the NSS pin (in hardware NSS mode) or the SWNSS bit (in software NSS mode) goes low, the SPI automatically enters to slave mode and triggers a master fault flag CONFERR.

If the application wants to use NSS line to control the SPI slave, NSS should be configured to hardware output mode (SWNSSEN=0, NSSDRV=1). NSS stays high after SPI is enabled and goes low when transmission or reception process begins.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

18.5.3. SPI operation modes

Table 18-2. SPI operation modes

Mode	Description	Register Configuration	Data Pin Usage

MFD	Master Full-Duplex	MSTMOD = 1
		RO = 0
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Transmission
			MISO: Reception

MTU	Master Transmission with unidirectional connection	MSTMOD = 1
		RO = 0
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Transmission
			MISO: Not used

MRU	Master Reception with unidirectional connection	MSTMOD = 1
		RO = 1
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Not used
			MISO: Reception

MTB	Master Transmission with bidirectional connection	MSTMOD = 1
		RO = 0
		BDEN = 1
		BDOEN = 1
			MOSI: Transmission
			MISO: Not used

MRB	Master Reception with bidirectional connection	MSTMOD = 1
		RO = 0
		BDEN = 1
		BDOEN = 0
			MOSI: Reception
			MISO: Not used

SFD	Slave Full-Duplex	MSTMOD = 0
		RO = 0
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Reception
			MISO: Transmission

STU	Slave Transmission with unidirectional connection	MSTMOD = 0
		RO = 0
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Not used
			MISO: Transmission

SRU	Slave Reception with unidirectional connection	MSTMOD = 0
		RO = 1
		BDEN = 0
		BDOEN: Don’t care
			MOSI: Reception
			MISO: Not used

STB	Slave Transmission with	MSTMOD = 0	MOSI: Not used

Mode	Description	Register Configuration	Data Pin Usage

	bidirectional connection	RO = 0
		BDEN = 1
		BDOEN = 1
			MISO: Transmission

SRB	Slave Reception with bidirectional connection	MSTMOD = 0
		RO = 0
		BDEN = 1
		BDOEN = 0
			MOSI: Not used
			MISO: Reception

Figure 18-3. A typical Full-duplex connection

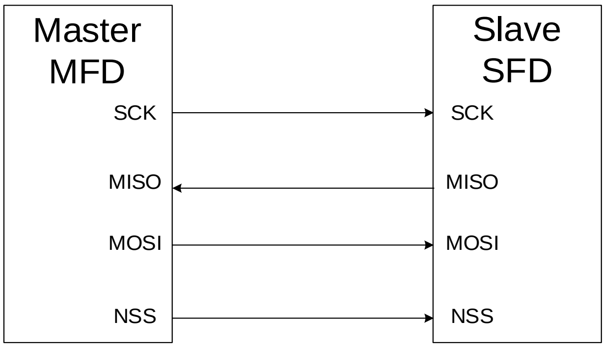


Figure 18-4. A typical simplex connection (Master: Receive, Slave: Transmit)

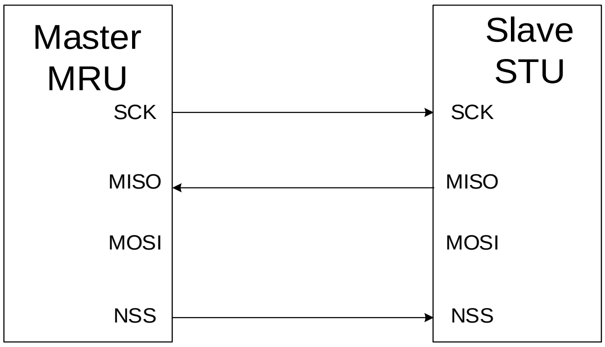


Figure 18-5. A typical simplex connection (Master: Transmit only, Slave: Receive)

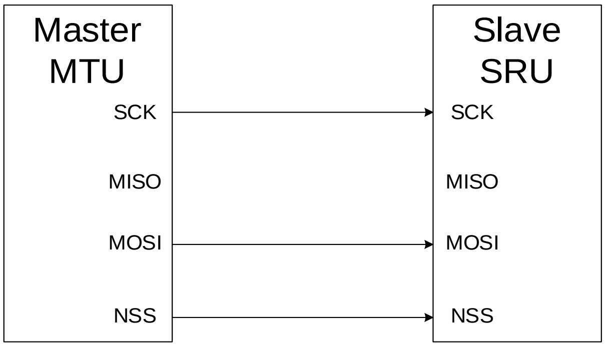
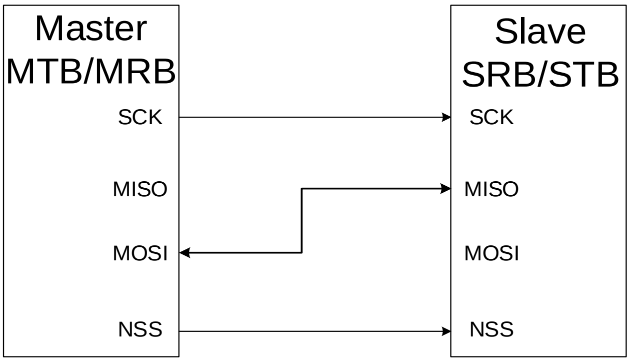


Figure 18-6. A typical bidirectional connection



SPI initialization sequence

Before transmiting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode is used, program the PSC [2:0] bits in SPI_CTL0 register to generate SCK with desired baud rate, otherwise, ignore this step.
2. Program data format (FF16 bit in the SPI_CTL0 register).
3. Program the clock timing register (CKPL and CKPH bits in the SPI_CTL0 register).
4. Program the frame format (LF bit in the SPI_CTL0 register).
5. Program the NSS mode (SWNSSEN and NSSDRV bits in the SPI_CTL0 register) according to the application's demand as described above in ***NSS function*** section.
6. Configure MSTMOD, RO, BDEN and BDOEN depending on the operation modes described above.
7. Enable the SPI (set the SPIEN bit).

Note: Before reconfiguring SPI clock, disable SPI. If done, then enable SPI to make reconfiguration take effect.

SPI basic transmission and reception sequence

Transmission sequence

After the initialization sequence, the SPI is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the transmit buffer. In slave mode the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is low, so application should ensure that data is already written into transmit buffer before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the data buffer to the shift register and then begins to transmit the loaded data frame, TBE (transmit buffer empty) flag is set after the first bit of this frame is transmited. After TBE flag is set, which means the transmit buffer is empty, the application should write SPI_DATA register again if it has more data to transmit.

In master mode, software should write the next data into SPI_DATA register before the transmission of current data frame is completed if it desires to generate continuous transmission.

Reception sequence

The incoming data will be moved from shift register to the receive buffer after the last valid sample clock and also, RBNE (receive buffer not empty) will be set. The application should read SPI_DATA register to get the received data and this will clear the RBNE flag automatically. In MRU and MRB modes, hardware continuously sends clock signal to receive the next data frame, while in full-duplex master mode (MFD), hardware only receives the next data frame when the transmit buffer is not empty.

Note: In SPI slave mode, if the number of input clock circle is not integral multiple of 8 or 16, which depends on bit width by configuration, and NSS is closed, SPI does not clear the count, if NSS is enabled, it transmits or receives new data after waiting for the corresponding number of clock circle. It can be solved by SPI disabled then SPI enabled.

SPI operation sequence in different modes

In full-duplex mode, either MFD or SFD, application should monitor the RBNE and TBE flags and follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to full-duplex mode, except that application should ignore the RBNE and OVRE flags and only perform transmission sequence described above.

In master reception mode (MRU or MRB), the behavior is different from full-duplex mode or transmission mode. In MRU or MRB mode, the SPI continuously generates SCK just after SPI is enabled, until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer in time after the RBNE flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to full-duplex mode, except that application should ignore the TBE flag and only perform reception sequence described above.

SPI disabling sequence

Different sequences are used to disable the SPI in different operation modes:

MFD SFD

Wait for the last RBNE flag and then receive the last data. Confirm that TBE=1 and TRANS=0. At last, disable the SPI by clearing SPIEN bit.

MTU MTB STU STB

Write the last data into SPI_DATA and wait until the TBE flag is set and then wait until the TRANS flag is cleared. Disable the SPI by clearing SPIEN bit.

MRU MRB

After getting the second last RBNE flag, read out this data and delay for a SCK clock time and then, disable the SPI by clearing SPIEN bit. Wait until the last RBNE flag is set and read out the last data.

SRU SRB

Application can disable the SPI when it doesn’t want to receive data, and then wait until the TRANS=0 to ensure the on-going transfer completes.

18.5.4. DMA function

The DMA function frees the application from data writing and reading process during transfer, to improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI_CTL1 register. To use DMA function, application should first correctly configure DMA modules, then configure SPI module according to the initialization sequence, at last enable SPI.

After being enabled, If DMATEN is set, SPI will generate a DMA request each time TBE=1, then DMA will acknowledge to this request and write data into the SPI_DATA register automatically. If DMAREN is set, SPI will generate a DMA request each time RBNE=1, then DMA will acknowledge to this request and read data from the SPI_DATA register automatically.

18.5.5. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial in SPI_CRCPOLY register.

Application can switch on the CRC function by setting CRCEN bit in SPI_CTL0 register. The CRC calculators continuously calculate CRC for each bit transmitted and received on lines, and the calculated CRC values can be read from SPI_TCRC and SPI_RCRC register.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI_CTL0 register after the last data is written to the transmit buffer. In full-duplex mode (MFD or SFD) the SPI treats the incoming data as a CRC value when it transmits a CRC and will check the received CRC value. In reception mode (MRB, MRU, SRU and SRB), the application should set the CRCNT bit after the second-last data frame is received. When CRC checking fails, the CRCERR flag will be set.

If DMA function is enabled, application doesn’t need to operate CRCNT bit and hardware will automatically process the CRC transmitting and checking.

18.6. SPI interrupts

18.6.1. Status flags

- Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI_DATA register.

- Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI_DATA register.

- SPI Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn’t generate any interrupt.

Note: TRANS is set after the first bit is transmitted. So TBE or RBNE must be judged as the communication finished, instead of TRANS.

18.6.2. Error conditions

- Configuration Fault Error (CONFERR)

CONFERR is an error flag in master mode. In NSS hardware mode and the NSSDRV is not enabled, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the SWNSS bit is 0. When the CONFERR is set, the SPIEN bit and the MSTMOD bit are cleared by hardware, the SPI is disabled and the device is forced into slave mode.

The SPIEN and MSTMOD bit are write protection until the CONFERR is cleared. The CONFERR bit of the slave cannot be set. In a multi-master configuration, the device can be in slave mode with CONFERR bit set, which means there might have been a multi-master conflict for system control.

- Rx Overrun Error (RXORERR)

The RXORERR bit is set if a data is received when the RBNE is set. That means, the last data has not been read out and the newly incoming data is received. The receive buffer contents won’t be covered with the newly incoming data, so the newly incoming data is lost.

- CRC Error (CRCERR)

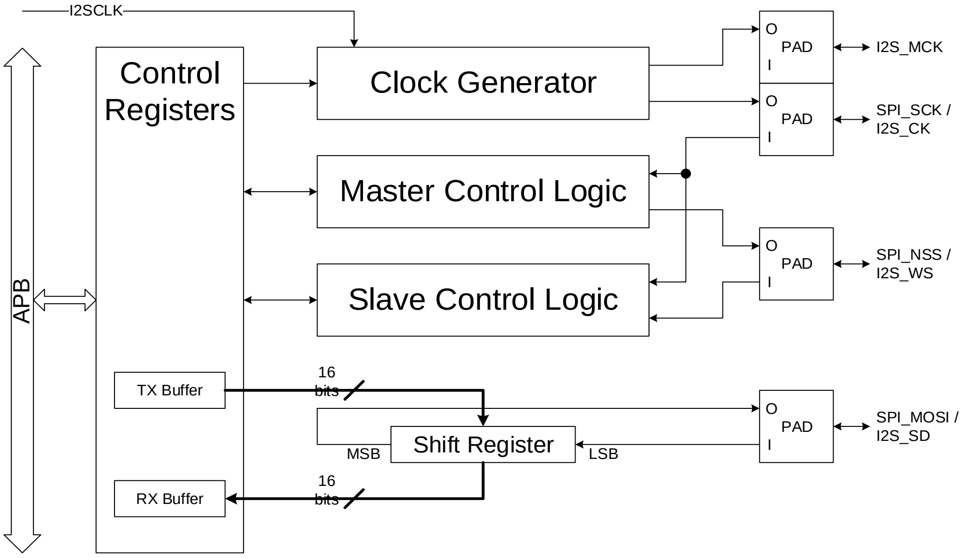
When the CRCEN bit is set, the CRC calculation result of the received data in the SPI_RCRC register is compared with the received CRC value after the last data, the CRCERR is set when they are different.

Table 18-3. SPI interrupt requests

Flag	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register.	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
CONFERR	Configuration Fault Error	Read or write SPI_STAT register, then write SPI_CTL0 register.	ERRIE
RXORERR	Rx Overrun Error	Read SPI_DATA register, then read SPI_STAT register.	
CRCERR	CRC error	Write 0 to CRCERR bit	

18.7. I2S block diagram

Figure 18-7. Block diagram of I2S



There are five sub modules to support I2S function, including control registers, clock generator, master control logic, slave control logic and shift register. All the user configuration registers are implemented in the control registers module, including the TX buffer and RX buffer. The clock generator is used to produce I2S communication clock in master mode. The master control logic is implemented to generate the I2S_WS signal and control the communication in master mode. The slave control logic is implemented to control the communication in slave mode according to the received I2SCK and I2S_WS. The shift register handles the serial data transmission and reception on I2S_SD.

18.8. I2S signal description

There are four pins on the I2S interface, including I2S_CK, I2S_WS, I2S_SD and I2S_MCK. I2S_CK is the serial clock signal, which shares the same pin with SPI_SCK. I2S_WS is the frame control signal, which shares the same pin with SPI_NSS. I2S_SD is the serial data signal, which shares the same pin with SPI_MOSI. I2S_MCK is the master clock signal. It produces a frequency rate equal to 256 x Fs, and Fs is the audio sampling frequency.

18.9. I2S function overview

18.9.1. I2S audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI_I2SCTL register. Four audio standards are supported, including I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. All standards except PCM handle audio data time-multiplexed on two channels (the left channel and the right channel). For these standards, the I2S_WS signal indicates the channel side. For PCM standard, the I2S_WS signal indicates frame synchronization information.

The data length and the channel length are configured by the DTLEN bits and CHLEN bit in the SPI_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 24-bit data packed in 32-bit frame, and 32-bit data packed in 32-bit frame. The data buffer for transmission and reception is 16-bit wide. In the case that the data length is 24 bits or 32 bits, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In the case that the data length is 16 bits, only one write or read operation to or from the SPI_DATA register is needed to complete a frame. When using 16-bit data packed in 32-bit frame, 16-bit 0 is inserted by hardware automatically to extend the data to 32-bit format.

For all standards and packet types, the most significant bit (MSB) is always sent first. For all standards based on two channels time-multiplexed, the channel left is always sent first followed by the channel right.

I2S Phillips standard

For I2S Phillips standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The timing diagrams for each configuration are shown below.

Figure 18-8. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

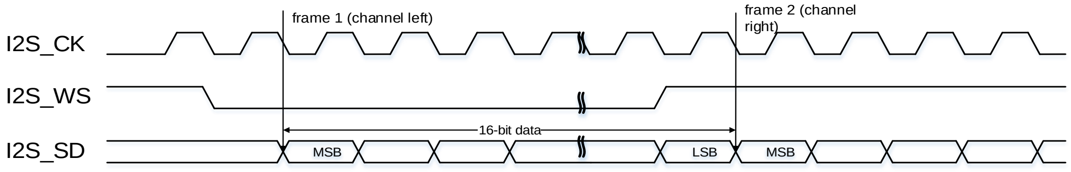
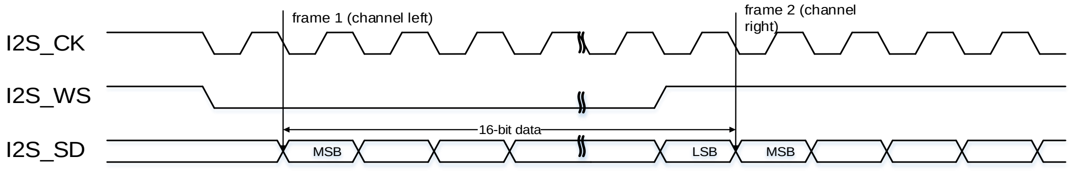


Figure 18-9. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)



When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete a frame.

Figure 18-10. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

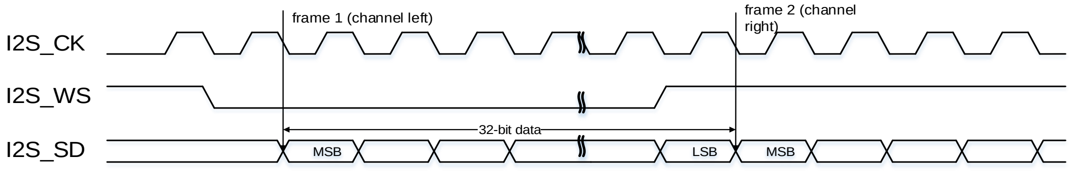
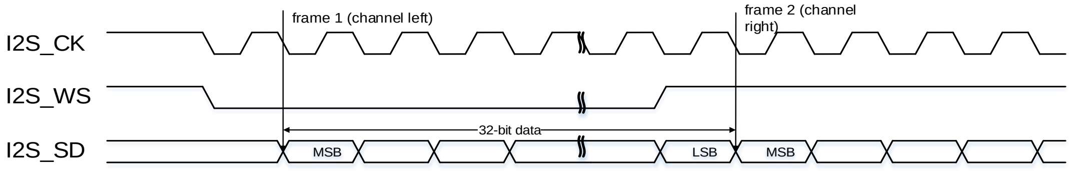


Figure 18-11. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)



When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits, and the second one should be the lower 16 bits. In reception mode, if a 32-bit data is received, the first data read from the SPI_DATA register should be higher 16 bits, and the second one should be the lower 16 bits.

Figure 18-12. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

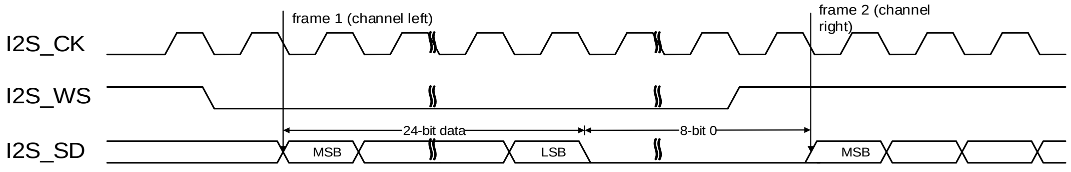
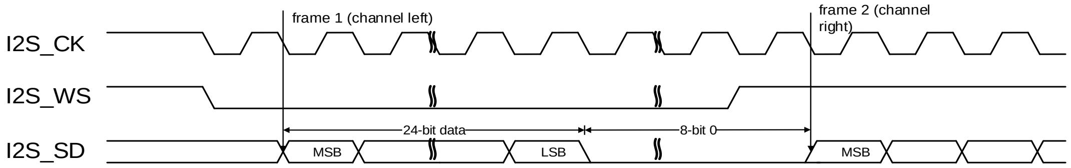


Figure 18-13. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits: D[23:8], and the second one should be a 16-bit data. The higher 8 bits of this 16-bit data should be D[7:0] and the lower 8 bits can be any value. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI_DATA register is D[23:8], and the second one is a 16-bit data. The higher 8 bits of this 16-bit data are D[7:0] and the lower 8 bits are zeros.

Figure 18-14. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

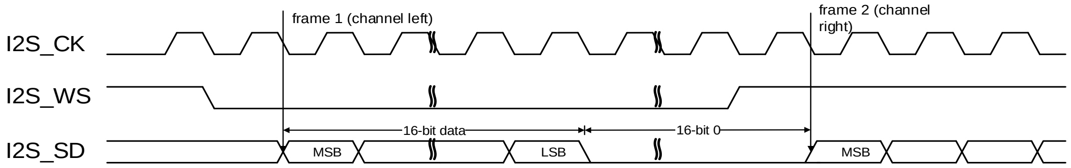
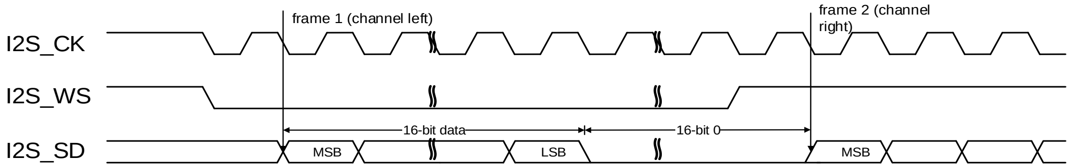


Figure 18-15. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

MSB justified standard

For MSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The SPI_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration are shown below.

Figure 18-16. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

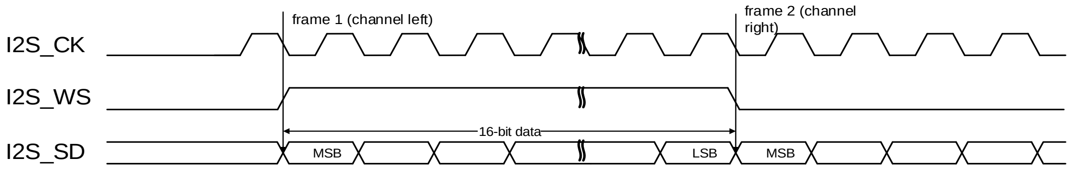


Figure 18-17. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

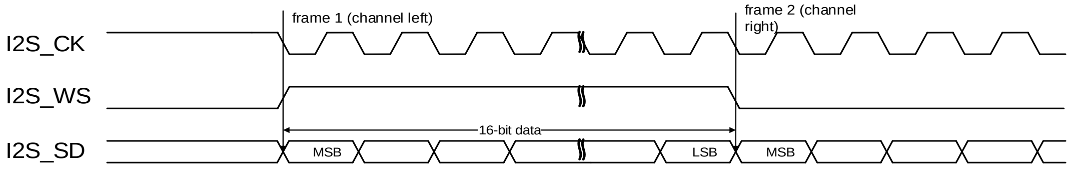


Figure 18-18. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

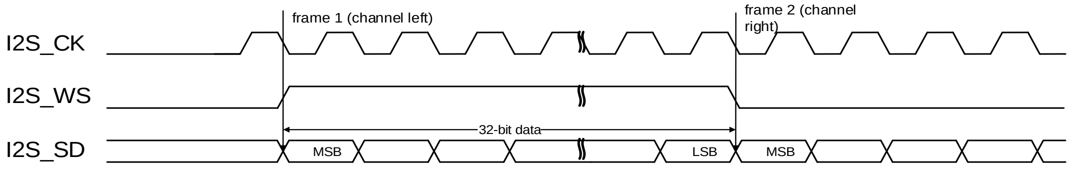


Figure 18-19. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

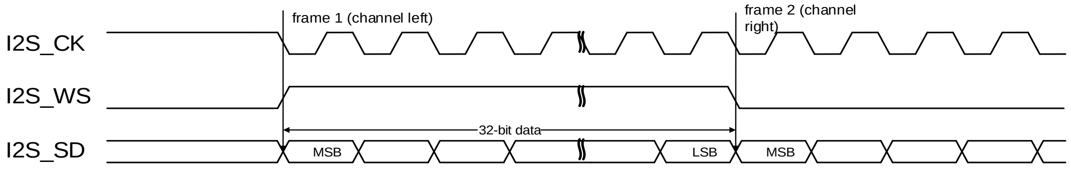


Figure 18-20. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

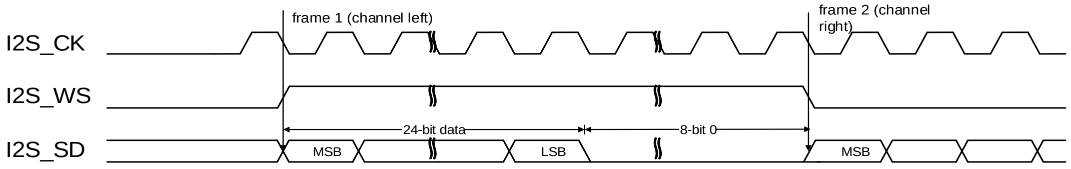


Figure 18-21.MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

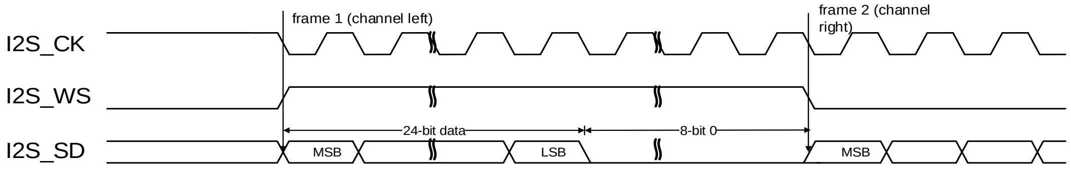


Figure 18-22. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

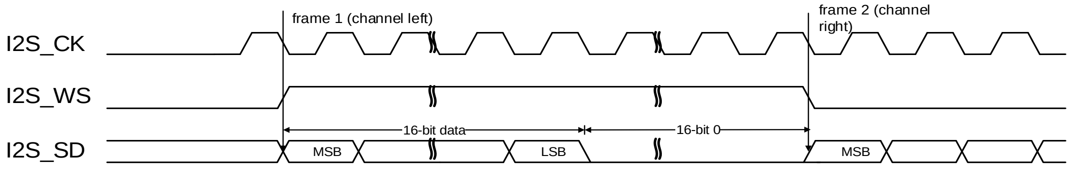
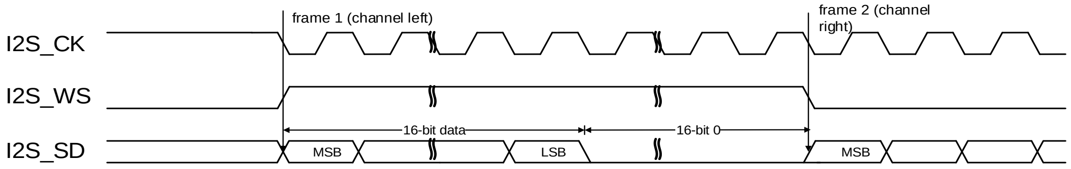


Figure 18-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



LSB justified standard

For LSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. In the case that the channel length is equal to the data length, LSB justified standard and MSB justified standard are exactly the same. In the case that the channel length is greater than the data length, the valid data is aligned to LSB for LSB justified standard while the valid data is aligned to MSB for MSB justified standard. The timing diagrams for the cases that the channel length is greater than the data length are shown below.

Figure 18-24. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

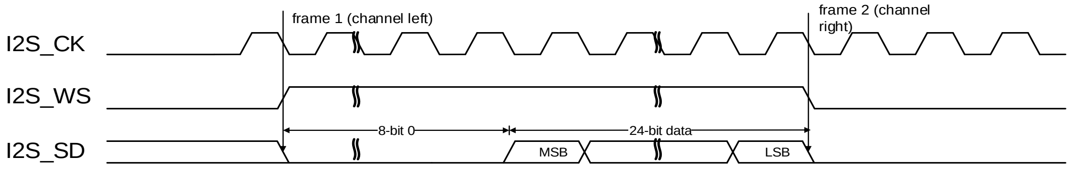
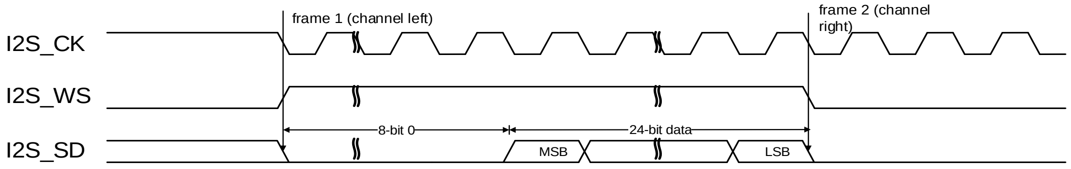


Figure 18-25. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D [23:0] is going to be sent, the first data written to the SPI_DATA register should be a 16-bit data. The higher 8 bits of the 16-bit data can be any value and the lower 8 bits should be D [23:16]. The second data written to the SPI_DATA register should be D [15:0]. In reception mode, if a 24-bit data D [23:0] is received, the first data read from the SPI_DATA register is a 16-bit data. The high 8 bits of this 16-bit data are zeros and the lower 8 bits are D [23:16]. The second data read from the SPI_DATA register is D [15:0].

Figure 18-26. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

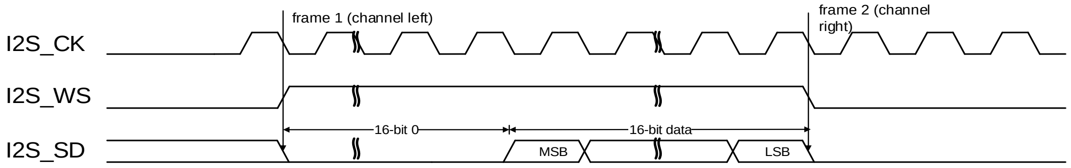
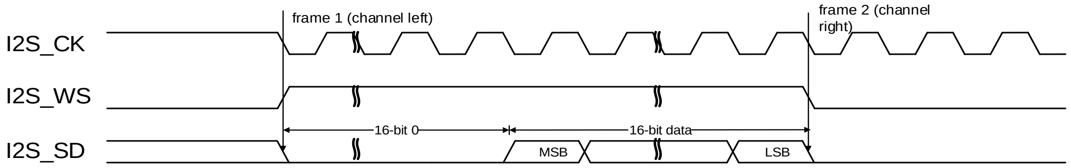


Figure 18-27. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

PCM standard

For PCM standard, I2S_WS and I2S_SD are updated on the rising edge of I2S_CK, and the I2S_WS signal indicates frame synchronization information. Both the short frame synchronization mode and the long frame synchronization mode are available and configurable using the PCMSMOD bit in the SPI_I2SCTL register. The SPI_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

Figure 18-28. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

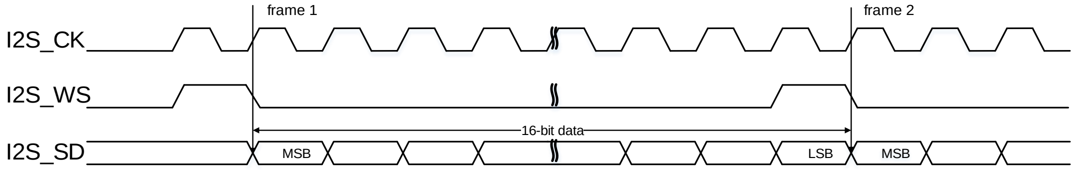


Figure 18-29. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

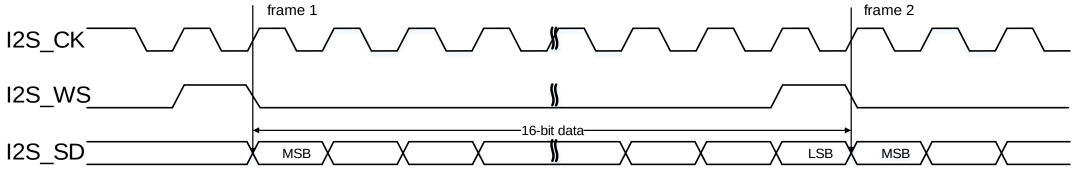


Figure 18-30. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

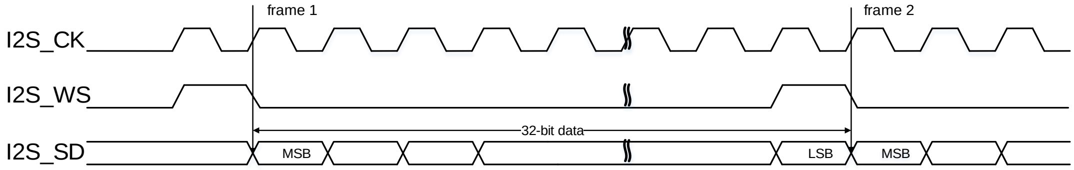


Figure 18-31. PCM standard short frame synchronization mode timing diagram

(DTLEN=10, CHLEN=1, CKPL=1)

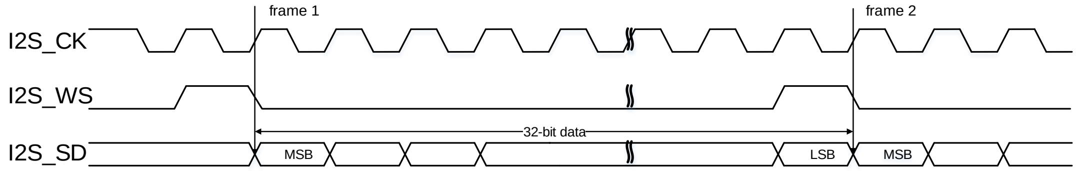


Figure 18-32. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

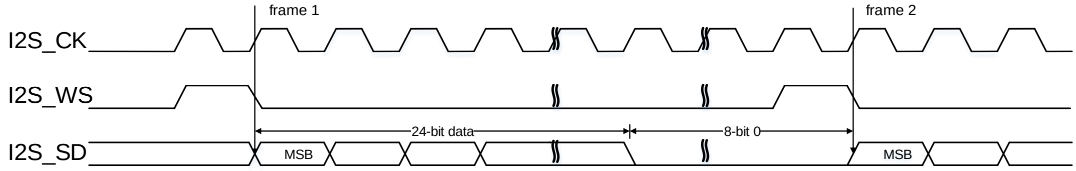


Figure18-33. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

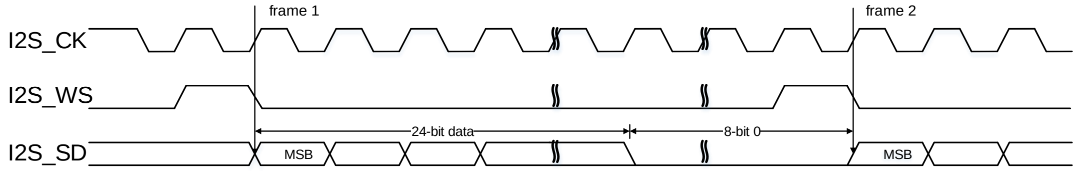


Figure 18-34. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

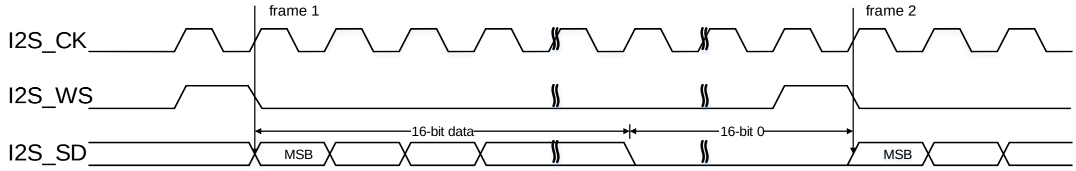
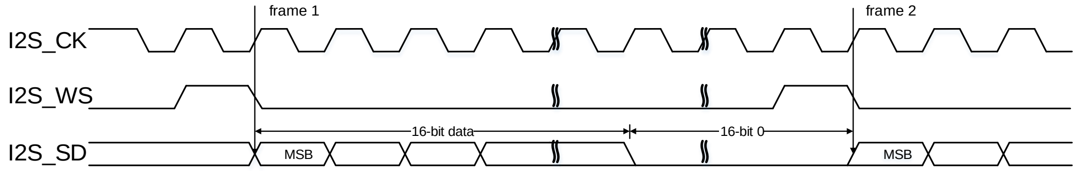


Figure 18-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



The timing diagrams for each configuration of the long frame synchronization mode are shown below.

Figure 18-36. PCM standard long frame synchronization mode timing diagram

(DTLEN=00, CHLEN=0, CKPL=0)

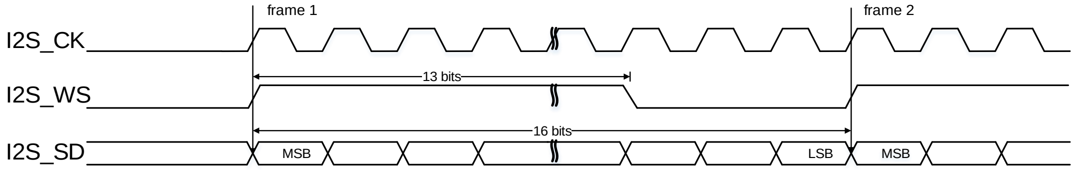


Figure18-37. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

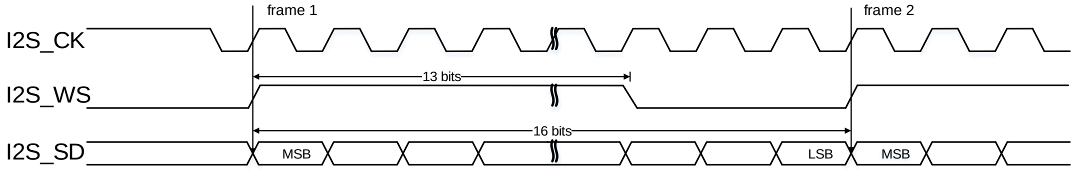


Figure 18-38. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

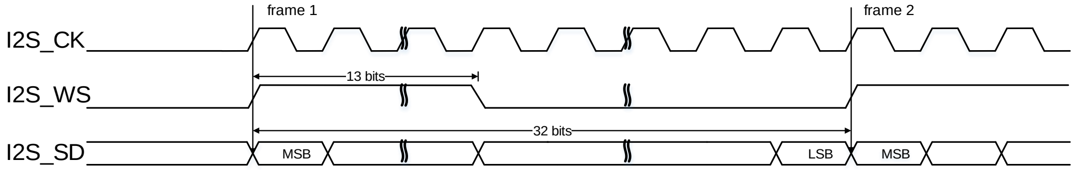


Figure 18-39. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

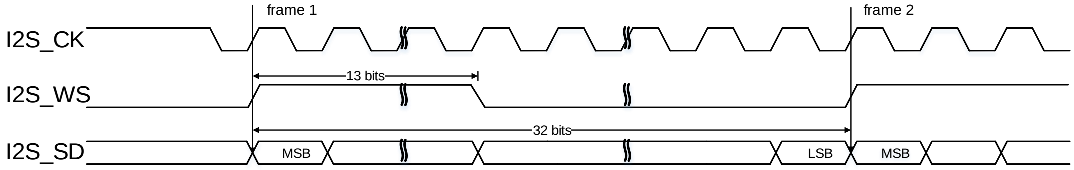


Figure 18-40. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

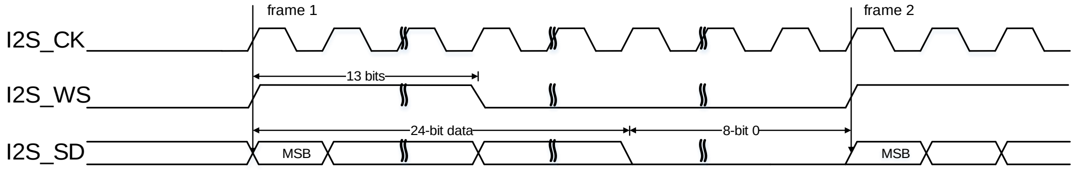


Figure 18-41. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

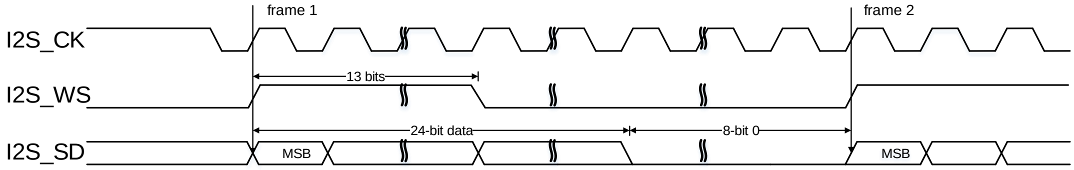


Figure 18-42. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

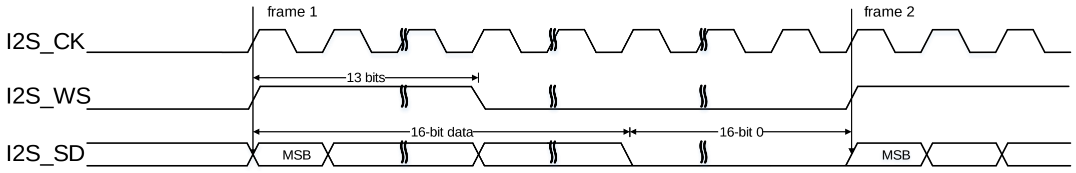
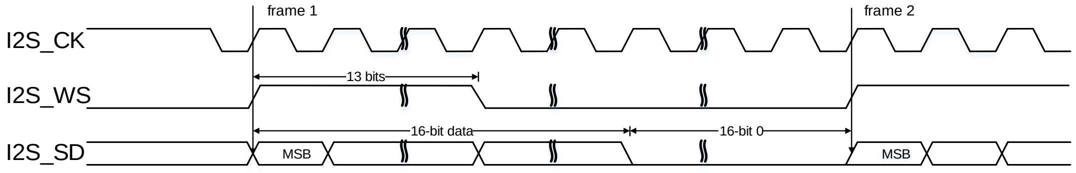
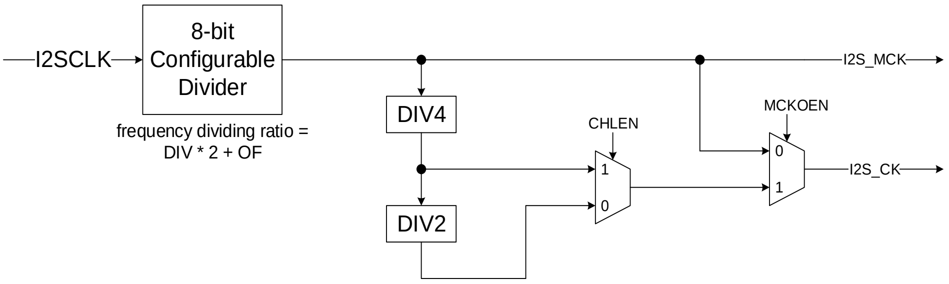


Figure 18-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



18.9.2. I2S clock

Figure 18-44. Block diagram of I2S clock generator


I2SCLK
8-bit Configurable Divider
frequency dividing ratio = DIV * 2 + OF
DIV4
DIV2
CHLEN
1
0
MCKOEN
0
1
I2S_MCK
I2S_CK

The block diagram of I2S clock generator is shown as ***Figure 18-44. Block diagram of I2S clock generator***. The I2S interface clocks are configured by the DIV bits, the OF bit, the MCKOEN bit in the SPI_I2SPSC register and the CHLEN bit in the SPI_I2SCTL register. The source of I2S clock can be either PLL2(CK_PLL2*2) or CK_SYS in order to get the maximum accuracy. The I2S bitrate can be calculated by the formulas shown in ***Table 18-4. I2S bitrate calculation formulas.***

Table 18-4. I2S bitrate calculation formulas

MCKOEN	CHLEN	Formula
0	0	I2SCLK / (DIV * 2 + OF)
0	1	I2SCLK / (DIV * 2 + OF)
1	0	I2SCLK / (8 * (DIV * 2 + OF))
1	1	I2SCLK / (4 * (DIV * 2 + OF))

The relationship between audio sampling frequency (Fs) and I2S bitrate is defined by the following formula:

Fs = I2S bitrate / (number of bits per channel * number of channels)

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in ***Table 18-5. Audio sampling frequency calculation formulas.***

Table 18-5. Audio sampling frequency calculation formulas

MCKOEN	CHLEN	Formula
0	0	I2SCLK / (32 * (DIV * 2 + OF))
0	1	I2SCLK / (64 * (DIV * 2 + OF))
1	0	I2SCLK / (256 * (DIV * 2 + OF))
1	1	I2SCLK / (256 * (DIV * 2 + OF))

18.9.3. Operation

Operation modes

The operation mode is selected by the I2SOPMOD bits in the SPI_I2SCTL register. There are four available operation modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode. The direction of I2S interface signals for each operation mode is shown in the ***Table 18-6. Direction of I2S interface signals for each operation mode.***

Table 18-6. Direction of I2S interface signals for each operation mode

Operation mode	I2S_MCK	I2S_CK	I2S_WS	I2S_SD
Master transmission	output or NU(1)	output	output	output
Master reception	output or NU(1)	output	output	input
Slave transmission	input or NU(1)	input	input	output
Slave reception	input or NU(1)	input	input	input

1. NU means the pin is not used by I2S and can be used by other functions.

I2S initialization sequence

I2S initialization sequence contains five steps shown below. In order to initialize I2S working in master mode, all the five steps should be done. In order to initialize I2S working in slave mode, only step 2, step 3, step 4 and step 5 should be done.

- Step 1: Configure the DIV [7:0] bits, the OF bit, and the MCKOEN bit in the SPI_I2SPSC register, in order to define the I2S bitrate and whether I2S_MCK needs to be provided or not.

- Step 2: Configure the CKPL in the SPI_I2SCTL register, in order to define the idle state clock polarity.

- Step 3: Configure the I2SSEL bit, the I2SSTD [1:0] bits, the PCMSMOD bit, the I2SOPMOD [1:0] bits, the DTLEN [1:0] bits, and the CHLEN bit in the SPI_I2SCTL register, in order to define the I2S feature.
- Step 4: Configure the TBEIE bit, the RBNEIE bit, the ERRIE bit, the DMATEN bit, and the DMAREN bit in the SPI_CTL1 register, in order to select the potential interrupt sources and the DMA capabilities. This step is optional.
- Step 5: Set the I2SEN bit in the SPI_I2SCTL register to enable I2S.

I2S master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmit buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI_CTL1 register is set. At the beginning, the transmit buffer is empty (TBE is high) and no transmission sequence is processing in the shift register. When a half word is written to the SPI_DATA register (TBE goes low), the data is transferred from the transmit buffer to the shift register (TBE goes high) immediately. At the moment, the transmission sequence begins.

The data is parallel loaded into the 16-bit shift register, and shifted out serially to the I2S_SD pin, MSB first. The next data should be written to the SPI_DATA register, when the TBE flag is high. After a write operation to the SPI_DATA register, the TBE flag goes low. When the current transmission finishes, the data in the transmit buffer is loaded into the shift register, and the TBE flag goes back high. Software should write the next audio data into SPI_DATA register before the current data finishes, otherwise, the audio data transmission is not continuous.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the data to transfer belongs. The I2SCH flag is refreshed at the moment when the TBE flag goes high. At the beginning, the I2SCH flag is low, indicating the left channel data should be written to the SPI_DATA register.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

I2S master reception sequence

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the receive buffer is not empty, and an interrupt will be generated if the RBNEIE bit in the SPI_CTL1 register is set. The reception sequence begins immediately when the I2SEN bit in the SPI_I2SCTL register is set. At the beginning, the receive buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the receive buffer (RBNE goes high). The data should be read from the SPI_DATA register, when the RBNE flag is high. After a read operation to the SPI_DATA register, the

RBNE flag goes low. It is mandatory to read the SPI_DATA register before the end of the next reception. Otherwise, reception overrun error occurs. The RXORERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is necessary to switch off and then switch on I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the received data belongs. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

Different sequences are used to disable the I2S in different standards, data length and channel length. The sequences for each case are described below.

- 16-bit data packed in 32-bit frame in the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD = 10)

1. Wait for the second last RBNE
2. Then wait 17 I2S CK clock (clock on I2S_CK pin) cycles
3. Clear the I2SEN bit

- 16-bit data packed in 32-bit frame in the audio standards except the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD is not equal to 10)

1. Wait for the last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

- For all other cases

1. Wait for the second last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The transmission sequence begins when the external master sends the clock and when the I2S_WS signal requests the transfer of data. The data has to be written to the SPI_DATA register before the master initiates the communication. Software should write the next audio data into SPI_DATA register before the current data finishe. Otherwise, transmission underrun error occurs. The TXURERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is mandatory to switch off and switch on I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

I2S slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The reception sequence begins when the external master sends the clock and when the I2S_WS signal indicates a start of the data transfer. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

18.9.4. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

18.10. I2S interrupts

18.10.1. Status flags

There are four status flags implemented in the SPI_STAT register, including TBE, RBNE, TRANS and I2SCH. The user can use them to fully monitor the state of the I2S bus.

- Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI_DATA register.

- Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI_DATA register.

- I2S Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn’t generate any interrupt.

- I2S channel side flag (I2SCH)

This flag indicates the channel side information of the current transfer and has no meaning in

PCM mode. It is updated when TBE rises in transmission mode or RBNE rises in reception mode. This flag doesn’t generate any interrupt.

18.10.2. Error conditions

There are two error conditions:

- Transmission Underrun Error Flag (TXURERR)

This condition occurs when the transmit buffer is empty when the valid SCK signal starts in slave transmission mode.

- Reception Overrun Error Flag (RXORERR)

This condition occurs when the receive buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in receive buffer is not updated and the newly incoming data is lost.

I2S interrupt events and corresponding enabled bits are summed up in the ***Table 18-7. I2S interrupt.***

Table 18-7. I2S interrupt

Flag Name	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	

18.11. Register definition

18.11.1. Control register 0 (SPI_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register has to be accessed by word (32-bit)

This register has no meaning in I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDEN	BDOEN	CRCEN	CRCNT	FF16	RO	SWNSS EN	SWNSS	LF	SPIEN	PSC [2:0]			MSTMOD	CKPL	CKPH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	BDEN	Bidirectional enable 0: 2 line unidirectional transmit mode 1: 1 line bidirectional transmit mode. The information transfers between the MOSI pin in master and the MISO pin in slave.
14	BDOEN	Bidirectional Transmit Output Enable When BDEN is set, this bit determines the direction of transfer. 0: Work in receive-only mode 1: Work in transmit-only mode
13	CRCEN	CRC Calculation Enable 0: CRC calculation is disabled 1: CRC calculation is enabled.
12	CRCNT	CRC Next Transfer 0: Next transfer is Data 1: Next transfer is CRC value (TCR) When the transfer is managed by DMA, CRC value is transferred by hardware. This bit should be cleared. In full-duplex or transmit-only mode, set this bit after the last data is written to SPI_DATA register. In receive only mode, set this bit after the second last data is received.
11	FF16	Data frame format

		0: 8-bit data frame format 1: 16-bit data frame format
10	RO	Receive only When BDEN is cleared, this bit determines the direction of transfer. 0: Full-duplex 1: Receive-only
9	SWNSSEN	NSS Software Mode Selection 0: NSS hardware mode. The NSS level depends on NSS pin. 1: NSS software mode. The NSS level depends on SWNSS bit.
8	SWNSS	NSS Pin Selection In NSS Software Mode 0: NSS pin is pulled low 1: NSS pin is pulled high This bit has an effect only when the SWNSSEN bit is set.
7	LF	LSB First Mode 0: Transmit MSB first 1: Transmit LSB first
6	SPIEN	SPI Enable 0: SPI peripheral is disabled 1: SPI peripheral is enabled
5:3	PSC[2:0]	Master Clock Prescaler Selection 000: PCLK/2 100: PCLK/32 001: PCLK/4 101: PCLK/64 010: PCLK/8 110: PCLK/128 011: PCLK/16 111: PCLK/256 PCLK means PCLK2 when using SPI0 or PCLK1 when using SPI1 and SPI2.
2	MSTMOD	Master Mode Enable 0: Slave mode 1: Master mode
1	CKPL	Clock Polarity Selection 0: CLK pin is pulled low when SPI is idle 1: CLK pin is pulled high when SPI is idle
0	CKPH	Clock Phase Selection 0: Capture the first data at the first clock transition. 1: Capture the first data at the second clock transition

18.11.2. Control register 1 (SPI_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TBEIE	RBNEIE	ERRIE	Reserved		NSSDRV	DMATEN	DMAREN
								rw	rw	rw			rw	rw	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	TBEIE	Transmit Buffer Empty Interrupt Enable 0: TBE interrupt is disabled. 1: TBE interrupt is enabled. An interrupt is generated when the TBE bit is set
6	RBNEIE	Receive Buffer Not Empty Interrupt Enable 0: RBNE interrupt is disabled. 1: RBNE interrupt is enabled. An interrupt is generated when the RBNE bit is set
5	ERRIE	Errors Interrupt Enable. 0: Error interrupt is disabled. 1: Error interrupt is enabled. An interrupt is generated when the CRCERR bit or the CONFERR bit or the RXORERR bit or the TXURERR bit is set.
4:3	Reserved	Must be kept at reset value
2	NSSDRV	Drive NSS Output 0: NSS output is disabled. 1: NSS output is enabled. If the NSS pin is configured as output, the NSS pin is pulled low in master mode when SPI is enabled. If the NSS pin is configured as input, the NSS pin should be pulled high in master mode, and this bit has no effect.
1	DMATEN	Transmit Buffer DMA Enable 0: Transmit buffer DMA is disabled 1: Transmit buffer DMA is enabled, when the TBE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.
0	DMAREN	Receive Buffer DMA Enable 0: Receive buffer DMA is disabled 1: Receive buffer DMA is enabled, when the RBNE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.

18.11.3. Status register (SPI_STAT)

Address offset: 0x08

Reset value: 0x0002

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRANS	RXORERR	CONFERR	CRCERR	TXURERR	I2SCH	TBE	RBNE
								r	r	r	rc_w0	r	r	r	r

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	TRANS	Transmitting On-going Bit 0: SPI or I2S is idle. 1: SPI or I2S is currently transmitting and/or receiving a frame This bit is set and cleared by hardware.
6	RXORERR	Reception Overrun Error Bit 0: No reception overrun error occurs. 1: Reception overrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_DATA register followed by a read access to the SPI_STAT register.
5	CONFERR	SPI Configuration error Bit 0: No configuration fault occurs 1: Configuration fault occurred. (In master mode, the NSS pin is pulled low in NSS hardware mode or SWNSS bit is low in NSS software mode.) This bit is set by hardware and cleared by a read or write operation on the SPI_STAT register followed by a write access to the SPI_CTL0 register. This bit is not used in I2S mode.
4	CRCERR	SPI CRC Error Bit 0: The SPI_RCRC value is equal to the received CRC data at last. 1: The SPI_RCRC value is not equal to the received CRC data at last. This bit is set by hardware and is able to be cleared by writing 0. This bit is not used in I2S mode.
3	TXURERR	Transmission underrun error Bit 0: No transmission underrun error occurs. 1: Transmission underrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_STAT

		register. This bit is not used in SPI mode.
2	I2SCH	I2S channel side 0: The next data needs to be transmitted or the data just received is channel left. 1: The next data needs to be transmitted or the data just received is channel right. This bit is set and cleared by hardware. This bit is not used in SPI mode, and has no meaning in the I2S PCM mode.
1	TBE	Transmit Buffer Empty 0: Transmit buffer is not empty 1: Transmit buffer is empty
0	RBNE	Receive Buffer Not Empty 0: Receive buffer is empty 1: Receive buffer is not empty

18.11.4. Data register (SPI_DATA)

Address offset: 0x0C

Reset value: 0x0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	SPI_DATA[15:0]	Data transfer register. The hardware has two buffers, including transmit buffer and receive buffer. Write data to SPI_DATA will save the data to transmit buffer and read data from SPI_DATA will get the data from receive buffer. When the data frame format is set to 8-bit data, the SPI_DATA [15:8] is forced to 0 and the SPI_DATA [7:0] is used for transmission and reception, transmit buffer and receive buffer are 8-bits. If the Data frame format is set to 16-bit data, the SPI_DATA [15:0] is used for transmission and reception, transmit buffer and receive buffer are 16-bit.

18.11.5. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x10

Reset value: 0x0007

This register has to be accessed by word(32-bit),

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CPR[15:0]	CRC polynomial register This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h.

18.11.6. RX CRC register (SPI_RCRC)

Address offset: 0x14

Reset value: 0x0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RCR[15:0]	RX CRC register When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCRC register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in RCR [7:0], when the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in RCR[15:0]. The hardware computes the CRC value after each received bit, when the TRANS is set, a read to this register could return an intermediate value. This register is reset when the CRCEN bit or the SPIEN bit in SPI_CTL0 register is cleared.

18.11.7. TX CRC register (SPI_TCRC)

Address offset: 0x18

Reset value: 0x0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	TCR[15:0]	TX CRC register When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in TCRC register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in TCR [7:0], when the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in TCR [15:0]. The hardware computes the CRC value after each transmitted bit, when the TRANS is set, a read to this register could return an intermediate value. The different frame format (LF bit of the SPI_CTL0) will get different CRC value. This register is reset when the CRCEN bit or the SPIEN bit in SPI_CTL0 register is cleared.

18.11.8. I2S control register (SPI_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I2SSEL	I2SEN	I2SOPMOD[1:0]		PCMSMOD	Reserved	I2SSTD[1:0]		CKPL	DTLEN[1:0]		CHLEN
				rw	rw	rw		rw		rw		rw	rw		rw

Bits	Fields	Descriptions

31:12	Reserved	Must be kept at reset value
11	I2SSEL	I2S mode selection 0: SPI mode 1: I2S mode This bit should be configured when SPI mode or I2S mode is disabled.
10	I2SEN	I2S enable 0: I2S is disabled 1: I2S is enabled This bit is not used in SPI mode.
9:8	I2SOPMOD[1:0]	I2S operation mode 00: Slave transmission mode 01: Slave reception mode 10: Master transmission mode 11: Master reception mode This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7	PCMSMOD	PCM frame synchronization mode 0: Short frame synchronization 1: long frame synchronization This bit has a meaning only when PCM standard is used. This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
6	Reserved	Must be kept at reset value
5:4	I2SSTD[1:0]	I2S standard selection 00: I2S Phillips standard 01: MSB justified standard 10: LSB justified standard 11: PCM standard These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.
3	CKPL	Idle state clock polarity 0: The idle state of I2S_CK is low level 1: The idle state of I2S_CK is high level This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
2:1	DTLEN[1:0]	Data length 00: 16 bits 01: 24 bits 10: 32 bits

		11: Reserved These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.
0	CHLEN	Channel length 0: 16 bits 1: 32 bits The channel length must be equal to or greater than the data length. This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.

18.11.9. I2S clock prescaler register (SPI_I2SPSC)

Address offset: 0x20

Reset value: 0x0002

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCKOEN	OF	DIV[7:0]							
						rw	rw	rw							

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	MCKOEN	I2S_MCK output enable 0: I2S_MCK output is disabled 1: I2S_MCK output is enabled This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
8	OF	Odd factor for the prescaler 0: Real divider value is DIV * 2 1: Real divider value is DIV * 2 + 1 This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7:0	DIV[7:0]	Dividing factor for the prescaler Real divider value is DIV * 2 + OF. DIV must not be 0. These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.

19. Secure digital input/output interface (SDIO)

This section applies to GD32F103xx high-density (HD) devices and extra-density (XD) devices only.

19.1. Overview

The secure digital input/output interface (SDIO) defines the SD, SD I/O, MMC and CE-ATA card host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards, Multimedia Card (MMC) and CE-ATA devices.

The supported SD memory card and SD I/O card system specifications are defined in the SD card Association website at ***www.sdcard.org***.

The supported Multimedia Card system specifications are defined through the Multimedia Card Association website at ***www.jedec.org***, published by the JEDEC SOLID STATE TECHNOLOGY ASSOCIATION.

The supported CE-ATA system specifications are defined through the CE-ATA workgroup website at ***www.ce-ata.org***.

19.2. Characteristics

The SDIO features include the following:

- **MMC:** Full support for Multimedia Card System Specification Version 4.2(and previous versions) Card and three different data bus modes: 1-bit (default), 4-bit and 8-bit.
- **SD Card:** Full support for *SD Memory Card Specifications Version 2.0.*
- **SD I/O:** Full support for *SD I/O Card Specification Version 2.0* card and two different data bus modes: 1-bit (default) and 4-bit.
- **CE-ATA:** Full compliance with *CE-ATA digital protocol Version 1.1.*
- 48MHz data transfer frequency and 8-bit data transfer mode.
- Interrupt and DMA request to processor.
- Completion Signal enables and disable feature (CE-ATA).

Note: SDIO supports only one SD, SD I/O, MMC4.2 card or CE-ATA device at any one time and a stack of MMC 4.1 or previous.

19.3. SDIO bus topology

After a power-on reset, the host must initialize the card by a special message-based bus protocol.

Each message is represented by one of the following tokens:

Command: a command is a token which starts an operation. A command is sent from the host to a card. A command is transferred serially on the CMD line.

Response: a response is a token which is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

Data: data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. The number of data lines used for the data transfer can be 1(DAT0), 4(DAT0-DAT3) or 8(DAT0-DAT7).

The structure of commands, responses and data blocks is described in ***Card functional description***. One data transfer is a bus operation.

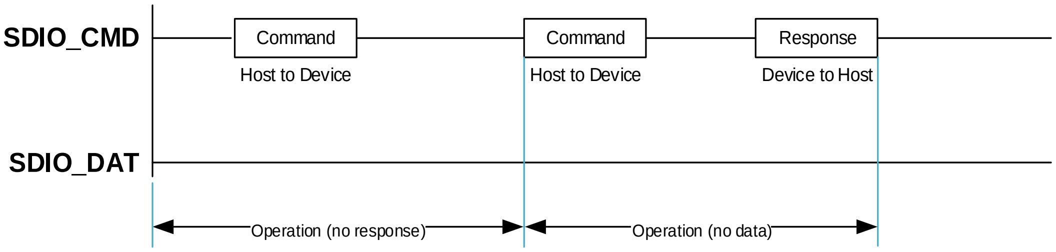
There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the DAT0-DAT7 and CMD lines are transferred synchronous to the host clock.

Two types of data transfer commands are defined:

- Stream commands: These commands initiate a continuous data stream; they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum (only MMC supports).
- Block-oriented commands: These commands send a data block successfully by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

The basic transaction on the bus is the command/response transaction (refer to ***Figure 19-1. SDIO "no response" and "no data" operations***). This type of bus transaction transfers their information directly within the command or response structure. In addition, some operations have a data token. Data transfers to/from the Card/Device are done in blocks.

Figure 19-1. SDIO "no response" and "no data" operations



Note that the Multiple Block operation mode is faster than Single Block operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the host to use single or multiple data lines. ***Figure 19-2. SDIO multiple blocks read operation*** is the multiple blocks read operation and ***Figure 19-3. SDIO multiple blocks write operation*** is the multiple block write operation. The block write operation uses a simple busy signal of the write operation duration on the data (DAT0) line. CE-ATA device has an optional busy before it is ready to receive the data.

Figure 19-2. SDIO multiple blocks read operation

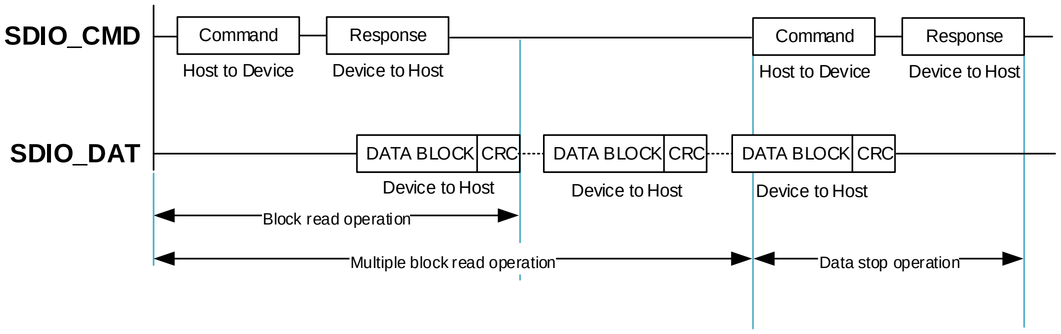
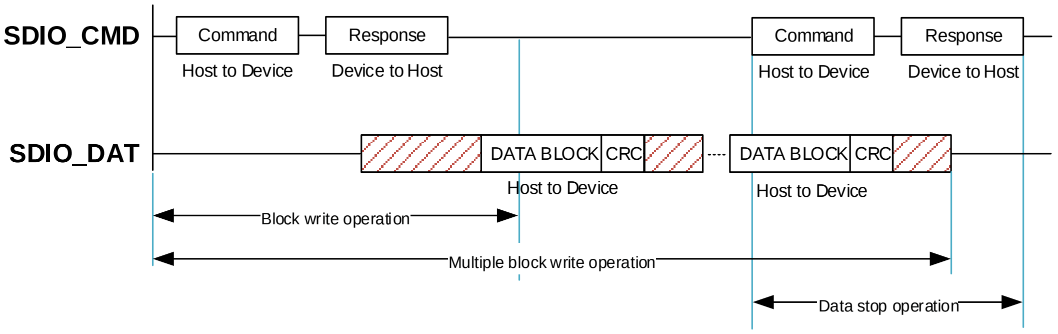


Figure 19-3. SDIO multiple blocks write operation



Data transfers to/from SD memory cards, SD I/O cards (both IO only card and combo card) and CE-ATA device are done in data blocks. Data transfers to/from MMC are done in data blocks or streams. ***Figure 19-4. SDIO sequential read operation*** and ***Figure 19-5. SDIO sequential write operation*** are the stream read and write operation.

Figure 19-4. SDIO sequential read operation

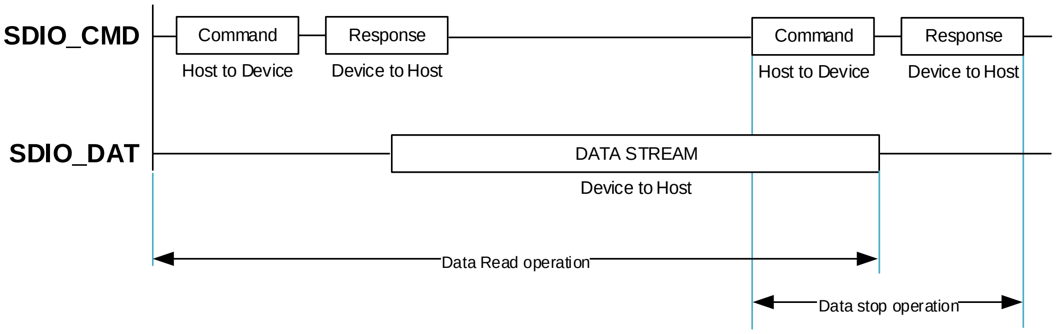
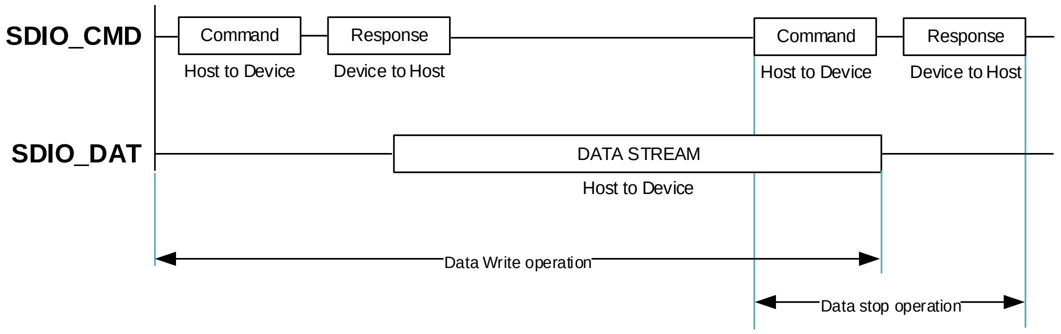


Figure 19-5. SDIO sequential write operation

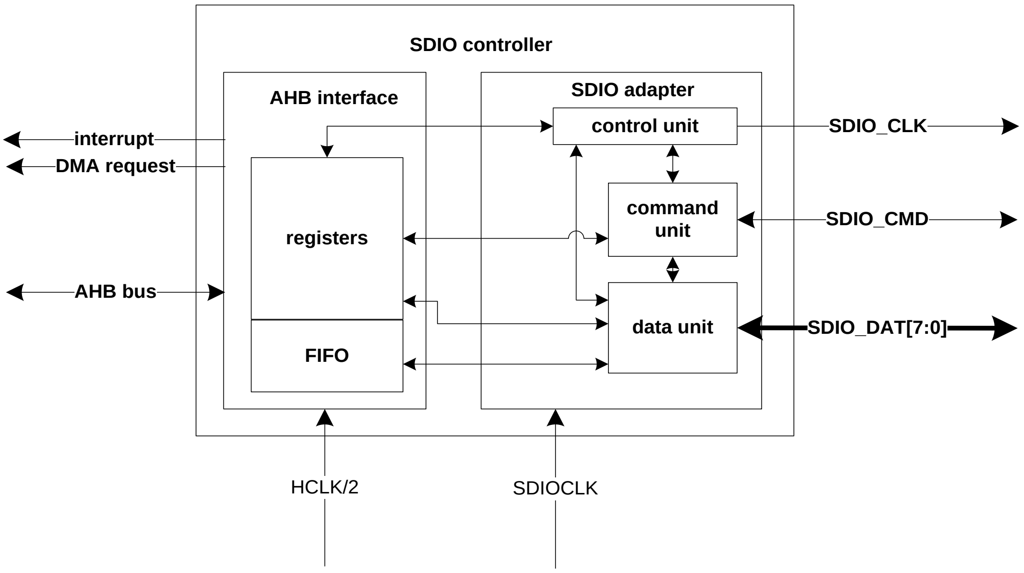


19.4. SDIO functional description

The following figure shows the SDIO structure. There have two main parts:

- The SDIO adapter block consists of control unit which manage clock, command unit which manage command transfer, data unit which manage data transfer.
- The AHB interface block contains access registers by AHB bus, contains FIFO unit which is data FIFO used for data transfer, and generates interrupt and DMA request signals.

Figure 19-6. SDIO block diagram



19.4.1. SDIO adapter

The SDIO adapter contains control unit, command unit and data unit, and generates signals to cards. The signals is descript bellow:

SDIO_CLK: The SDIO_CLK is the clock provided to the card. Each cycle of this signal directs a one bit transfer on the command line (SDIO_CMD) and on all the data lines (SDIO_DAT).

The SDIO_CLK frequency can vary between 0 MHz and 20 MHz for a Multimedia Card V3.31, between 0 and 48 MHz for a Multimedia Card V4.2, or between 0 and 25 MHz for an SD/SD I/O card.

The SDIO uses two clock signals: SDIO adapter clock (SDIOCLK = HCLK) and AHB bus clock (HCLK/2)

SDIO_CMD: This signal is a bidirectional command channel used for card initialization and transfer of commands. Commands are sent from the SDIO controller to the card and responses are sent from the card to the host. The CMD signal has two operation modes: open-drain for initialization (only for MMC3.31 or previous), and push-pull for command transfer (SD/SD I/O card MMC4.2 use push-pull drivers also for initialization).

SDIO_DAT[7:0]: These are bidirectional data channels. The DAT signals operate in push-pull mode. Only the card or the host is driving these signals at a time. By default, after power up or reset, only DAT0 is used for data transfer. A wider data bus can be configured for data transfer, using either DAT0-DAT3 or DAT0-DAT7 (just for MMC4.2), by the SDIO controller. The SDIO includes internal pull-ups for data lines DAT1-DAT7. Right after entering to the 4-bit mode the card disconnects the internal pull-ups of lines DAT1 and DAT2 (DAT3 internal pull-up is left connected due to the SPI mode CS usage). Correspondingly right after entering to the 8-bit mode the card disconnects the internal pull-ups of lines DAT1, DAT2 and DAT4-DAT7.

Table 19-1. SDIO I/O definitions

Pin function	Direction	Description
SDIO_CLK	O	SD/SD I/O /MMC clock
SDIO_CMD	I/O	Command input/output
SDIO_DAT[7:0]	I/O	Data input/output for data lines DAT[7:0]

The SDIO adapter is an interface to SD, SD I/O, MMC and CE-ATA. It consists of three subunits:

Control unit

The control unit contains the power management functions and the clock management functions for the memory card clock. The power management is controlled by SDIO_PWRCTL register which implements power off or power on. The power saving mode configured by setting CLKPWRSAV bit in SDIO_CLKCTL register, which implements close the SDIO_CLK when the bus is idle. The clock management generates SDIO_CLK to card. The SDIO_CLK is generated by a divider of SDIOCLK when CLKBYP bit in SDIO_CLKCTL register is 0, or directly SDIOCLK when CLKBYP bit in SDIO_CLKCTL register is 1.

The hardware clock control is enabled by setting HWCLKEN in SDIO_CLKCTL register. This functionality is used to avoid FIFO underrun and overrun errors by hardware control the SDIO_CLK on/off depending on the system bus is very busy or not. When the FIFO cannot receive or transmit data, the host will stop the SDIO_CLK and freeze SDIO state machines to avoid the corresponded error. Only state machines are frozen, the AHB interface is still alive. So, the FIFO can access by AHB bus.

Command unit

The command unit implements command transfer to the card. The data transfer flow is controlled by Command State Machine (CSM). After a write operation to SDIO_CMDCTL register and CSMEN in SDIO_CMDCTL register is 1, the command transfer starts. It firstly sends a command to the card. The command contains 48 bits send by SDIO_CMD signal which sends 1 bits to card at one SDIO_CLK. The 48 bits command contains 1 bit Start bit, 1 bit Transmission bit, 6 bits command index defined by CMDIDX bits in SDIO_CMDCTL register, 32 bits argument defined in SDIO_CMDAGMT register, 7 bits CRC, and 1 bit end bit. Then receive response from the card if CMDRESP in SDIO_CMDCTL register is not 0b00/0b10. There are short response which have 48 bits or long response which have 136 bits. The response stores in SDIO_RESP0 - SDIO_RESP3 registers. The command unit also generates the command status flags defined in SDIO_STAT register.

Command state machine

CS_Idle	After reset, ready to send command.		
	1.CSM enabled and WAITDEND enabled	→	CS_Pend
	2.CSM enabled and WAITDEND disabled	→	CS_Send
	3.CSM disabled	→	CS_Idle
	Note: The state machine remains in the Idle state for at least eight SDIO_CLK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the response.		

CS_Pend	Waits for the end of data transfer.		
	1.The data transfer complete	→	CS_Send
	2.CSM disabled	→	CS_Idle

CS_Send	Sending the command.		
	1.The command transmitted has response	→	CS_Wait
	2.The command transmitted doesn’t have response	→	CS_Idle
	3.CSM disabled	→	CS_Idle

CS_Wait	Wait for the start bit of the response.		
	1.Receive the response(detected the start bit)	→	CS_Receive
	2.Timeout is reached without receiving the response	→	CS_Idle
	3.CSM disabled	→	CS_Idle
	Note: The command timeout has a fixed value of 64 SDIO_CLK clock periods.		

CS_Receive	Receive the response and check the CRC.	
1.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command Completion signal enabled	→	CS_Waitcompl
2.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command Completion signal disabled	→	CS_Pend
3.CSM disabled	→	CS_Idle
4.Response received	→	CS_Idle
5.Command CRC failed	→	CS_Idle

CS_Waitcompl	Wait for the Command Completion signal.	
1.CE-ATA Command Completion signal received	→	CS_Idle
2.CSM disabled	→	CS_Idle
3.Command CRC failed	→	CS_Idle

Data unit

The data unit performs data transfers to and from cards. The data transfer uses SDIO_DAT[7:0] signals when 8-bits data width (BUSMODE bits in SDIO_CLKCTL register is 0b10), use SDIO_DAT[3:0] signals when 4-bits data width (BUSMODE bits in SDIO_CLKCTL register is 0b01), or SDIO_DAT[0] signal when 1-bit data width (BUSMODE bits in SDIO_CLKCTL register is 0b00). The data transfer flow is controlled by Date State Machine (DSM). After a write operation to SDIO_DATACTL register and DATAEN in SDIO_DATACTL register is 1, the data transfer starts. It sends data to card when DATADIR in SDIO_DATACTL register is 0, or receive data from card when DATADIR in SDIO_DATACTL register is 1. The data unit also generates the data status flags defined in SDIO_STAT register.

Data state machine

DS_Idle	The data unit is inactive, waiting for send and receive.	
1.DSM enabled and data transfer direction is from host to card	→	DS_WaitS
2.DSM enabled and data transfer direction is from card to host	→	DS_WaitR
3.DSM enabled and Read Wait Started and SD I/O mode enabled	→	DS_Readwait

DS_WaitS	Wait until the data FIFO empty flag is deasserted or data transfer ended.	
1.Data transfer ended	→	DS_Idle
2.DSM disabled	→	DS_Idle

3.Data FIFO empty flag is deasserted	→	DS_Send

DS_Send	Transmit data to the card.	
1.Data block transmitted	→	DS_Busy
2.DSM disabled	→	DS_Idle
3.Data FIFO underrun error occurs	→	DS_Idle
4. Internal CRC error	→	DS_Idle

DS_Busy	Waits for the CRC status flag.	
1.Receive a positive CRC status	→	DS_WaitS
2.Receive a negative CRC status	→	DS_Idle
3.DSM disabled	→	DS_Idle
4.Timeout occurs	→	DS_Idle
Note: The command timeout programmed in the data timer register (SDIO_DATATO).		

DS_WaitR	Wait for the start bit of the receive data.	
1.Data receive ended	→	DS_Idle
2.DSM disabled	→	DS_Idle
3.Data timeout reached	→	DS_Idle
4.Receives a start bit before timeout	→	DS_Receive
Note: The command timeout programmed in the data timer register (SDIO_DATATO).		

DS_Receive	Receive data from the card and write it to the data FIFO.	
1.Data block received	→	DS_WaitR
2.Data transfer ended	→	DS_WaitR
3.Data FIFO overrun error occurs	→	DS_Idle
4.Data received and Read Wait Started and SD I/O mode enabled	→	DS_Readwait
5.DSM disabled or CRC fails	→	DS_Idle

DS_Readwait	Wait for the read wait stop command.	
1.ReadWait stop enabled	→	DS_WaitR
2.DSM disabled	→	DS_Idle

19.4.2. AHB interface

The AHB interface implements access to SDIO registers, data FIFO and generates interrupt and DMA request. It includes a data FIFO unit, registers unit, and the interrupt / DMA logic.

The interrupt logic generates interrupt when at least one of the selected status flags is high.

An interrupt enable register is provided to allow the logic to generate a corresponding interrupt.

The DMA interface provides a method for fast data transfers between the SDIO data FIFO and memory. The following example describes how to implement this method:

1. Complete the card identification process

2. Increase the SDIO_CLK frequency

3. Send CMD7 to select the card and configure the bus width

4. Configure the DMA1 as follows:

Enable DMA1 controller and clear any pending interrupts. Configure the DMA1_Channel3 source address register with the memory base address and DMA1_Channel3 destination address register with the SDIO_FIFO register address. Program DMA1_Channel3 control register (memory increment, not peripheral increment, peripheral and source width is word size, M2M disable).

5. Write block to card as follows:

Write the data size in bytes in the SDIO_DATALEN register. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register; the host sends data in blocks of size BLKSZ each. Program SDIO_CMDAGMT register with the data address, where data should be written. Program the SDIO command control register (SDIO_CMDCTL): CMDIDX with 24, CMDRESP with 1 (SDIO card host waits for a short response); CSMEN with ‘1’ (enable to send a command). Other fields are their reset value.

When the CMDRECV flag is set, program the SDIO data control register (SDIO_DATACTL): DATAEN with 1 (enable to send data); DATADIR with 0 (from controller to card); TRANSMOD with 0 (block data transfer); DMAEN with 1 (DMA enabled); BLKSZ with 0x9 (512 bytes). Other bits don’t care.

Wait for DTBLKEND flag is set. Check that no channels are still enabled by polling the DMA Interrupt Flag register.

It consists the following subunits:

Register unit

The register unit which contains all system registers generates the signals to control the communication between the controller and card.

Data FIFO

The data FIFO unit has a data buffer, uses as transmit and receive FIFO. The FIFO contains a 32-bit wide, 32-word deep data buffer. The transmit FIFO is used when write data to card and TXRUN in SDIO_STAT register is 1. The data to be transferred is written to transmit FIFO by AHB bus, the data unit in SDIO adapter read data from transmit FIFO, and then send the data to card. The receive FIFO is used when read data from card and RXRUN in SDIO_STAT register is 1. The data to be transferred is read from the card and then write to receive FIFO. The data in receive FIFO is read to AHB bus when needed. This unit also generates FIFO flags in SDIO_STAT registers.

19.5. Card functional description

19.5.1. Card registers

Within the card interface registers are defined: OCR, CID, CSD, EXT_CSD, RCA, DSR and SCR. These can be accessed only by corresponding commands. The OCR, CID, CSD and SCR registers carry the card/content specific information, while the RCA and DSR registers are configuration registers storing actual configuration parameters. The EXT_CSD register carries both, card specific information and actual configuration parameters. For specific information, please refer to the relevant specifications.

OCR register: The 32-bit operation conditions register (OCR) stores the V_{DD} voltage profile of the card and the access mode indication (MMC). In addition, this register includes a status information bit. This status bit is set if the card power up procedure has been finished. The register is a little different between MMC and SD card. The host can use CMD1 (MMC), ACMD41 (SD memory), CMD5 (SD I/O) to get the content of this register.

CID register: The Card Identification (CID) register is 128 bits wide. It contains the card identification information used during the card identification phase. Every individual Read/Write (RW) card shall have a unique identification number. The host can use CMD2 and CMD10 to get the content of this register.

CSD register: The Card-Specific Data register provides information regarding access to the card contents. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, whether the DSR register can be used, etc. The programmable part of the register can be changed by CMD27. The host can use CMD9 to get the content of this register.

Extended CSD Register: Just MMC4.2 has this register. The Extended CSD register defines the card properties and selected modes. It is 512 bytes long. The most significant 320 bytes are the Properties segment, which defines the card capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the card is working in. These modes can be changed by the host by means of the SWITCH command. The host can use CMD8 (just MMC supports this command) to get the content of this register.

RCA register: The writable 16-bit relative card address register carries the card address that is published by the card during the card identification. This address is used for the addressed host-card communication after the card identification procedure. The host can use CMD3 to ask the card to publish a new relative address (RCA).

Note: The default value of the RCA register is 0x0001(MMC) or 0x0000(SD/SD I/O). The default value is reserved to set all cards into the Stand-by State with CMD7.

DSR register (Optional): The 16-bit driver stage register can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of cards). The CSD register carries the information about the DSR register usage. The default value of the DSR register is 0x404. The host can use CMD4 to get the content of this register.

SCR register: Just SD/SD I/O (if has memory port) have this register. In addition to the CSD register, there is another configuration register named SD CARD Configuration Register (SCR), which is only for SD card. SCR provides information on the SD Memory Card's special features that were configured into the given card. The size of SCR register is 64 bits. This register shall be set in the factory by the SD Memory Card manufacturer. The host can use ACMD51 to get the content of this register.

19.5.2. Commands

Commands types

There are four kinds of commands defined to control the Card:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr) response from all cards simultaneously
- Addressed (point-to-point) commands (ac) no data transfer on DAT
- Addressed (point-to-point) data transfer commands (adtc) data transfer on DAT

Command format

All commands have a fixed code length of 48 bits, as show in ***Figure 19-7. Command Token Format***, needing a transmission time of 1.92μs (25 MHz) 0.96μs (50 MHz) and 0.92us (52 MHz).

Figure 19-7. Command Token Format

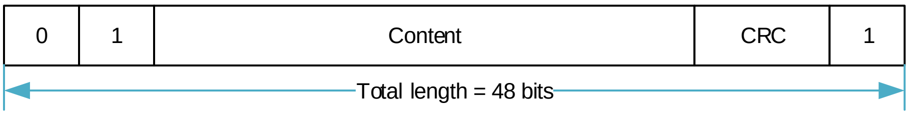


Table 19-2. Command format

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	‘0’	‘1’	x	x	x	‘1’
Description	start bit	transmission bit	command index	argument	CRC7	end bit

A command always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (host = 1). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g. an address), which is coded by 32 bits. A value denoted by 'x' in the table above indicates this variable is dependent on the command. All commands are protected by a CRC7. Every command code word is terminated by the end bit (always 1).

Command classes

The command set of the Card system is divided into several classes (See ***Table 19-3. Card command classes (CCCs)***). Each class supports a set of card functionalities. ***Table 19-3. Card command classes (CCCs)*** determines the setting of CCC from the card supported commands.

For SD cards, Class 0, 2, 4, 5 and 8 are mandatory and shall be supported. Class 7 except CMD40 is mandatory for SDHC. The other classes are optional. The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

For MMC cards, Class 0 is mandatory and shall be supported. The other classes are either mandatory only for specific card types or optional. By using different classes, several configurations can be chosen (e.g. a block writable card or a stream readable card). The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

For CE-ATA device, the device shall support the MMC commands required to achieve the transfer state during device initialization. Other interface configuration settings, such as bus width, may require additional MMC commands also be supported. See the MMC reference. CE-ATA makes use of the following MMC commands: CMD0 - GO_IDLE_STATE, CMD12 - STOP_TRANSMISSION, CMD39 - FAST_IO, CMD60 - RW_MULTIPLE_REGISTER, CMD61 - RW_MULTIPLE_BLOCK. GO_IDLE_STATE (CMD0), STOP_TRANSMISSION (CMD12), and FAST_IO (CMD39) are as defined in the MMC reference. RW_MULTIPLE_REGISTER (CMD60) and RW_MULTIPLE_BLOCK (CMD61) are MMC commands defined by CE-ATA.

Table 19-3. Card command classes (CCCs)

	Card command class(CCC)	0	1	2	3	4	5	6	7	8	9	10	11
Supported command	**Class description**	**basic**	**Stream read**	**Block read**	**Stream write**	**Block write**	**erase**	**write protection**	**Lock card**	**application specific**	**I/O mode**	**switch**	**reserved**

CMD0	M	+											
CMD1	M	+											
CMD2	M	+											
CMD3	M	+											
CMD4	M	+											
CMD5	O										+		
CMD6	M											+	
CMD7	M	+											
CMD8	M	+											
CMD9	M	+											
CMD10	M	+											
CMD11	M		+										
CMD12	M	+											
CMD13	M	+											
CMD14	M	+											
CMD15	M	+											
CMD16	M			+		+			+				
CMD17	M			+									
CMD18	M			+									
CMD19	M	+											
CMD20	M				+								
CMD23	M			+		+							
CMD24	M					+							
CMD25	M					+							
CMD26	M					+							
CMD27	M					+							
CMD28	M							+					
CMD29	M							+					
CMD30	M							+					
CMD32	M						+						
CMD33	M						+						
CMD34	O											+	
CMD35	O											+	
CMD36	O											+	
CMD37	O											+	
CMD38	M						+						
CMD39											+		
CMD40											+		
CMD42									+				
CMD50	O											+	
CMD52	O										+		
CMD53	O										+		

CMD55	M									+			
CMD56	M									+			
CMD57	O											+	
CMD60	M									+			
CMD61	M									+			
ACMD6	M									+			
ACMD13	M									+			
ACMD22	M									+			
ACMD23	M									+			
ACMD41	M									+			
ACMD42	M									+			
ACMD51	M									+			

Note: 1.CMD1, CMD11, CMD14, CMD19, CMD20, CMD23, CMD26, CMD39 and CMD40 are only available for MMC.CMD5, CMD32-34, CMD50, CMD52, CMD53, CMD57 and ACMDx are only available for SD card. CMD60, CMD61 are only available for CE-ATA device.

2. All the ACMDs shall be preceded with APP_CMD command (CMD55).

3. CMD8 has different meaning for MMC and SD memory.

Detailed command description

The following tables describe in detail all bus commands. The responses R1-R7 are defined in ***Responses***. The registers CID, CSD and DSR are described in ***Card registers***. The card shall ignore stuff bits and reserved bits in an argument.

Table 19-4. Basic commands (class 0)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all cards to idle state
CMD1	bc	[31:0] OCR without busy	R3	SEND_OP_COND	Asks the card, in idle state, to send its Operating Conditions Register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks any card to send the CID numbers on the CMD line (any card that is connected to the host will respond)
CMD3	bcr	[31:0] stuff bits	R6	SEND_RELATIVE_ADDR	Ask the card to publish a new relative address (RCA)
CMD4	bc	[31:16] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards

CMD5	bcr	[31:25]reserved bits [24]S18R [23:0] I/O OCR	R4	IO_SEND_OP_COND	Only for I/O cards. It is similar to the operation of ACMD41 for SD memory cards, used to inquire about the voltage range needed by the I/O card.
CMD6	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Only for MMC. Switches the mode of operation of the selected card or modifies the EXT_CSD registers.
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnects states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects the card.
CMD8	bcr	[31:12]reserved bits [11:8]supply voltage(VHS) [7:0]check pattern	R7	SEND_IF_COND	Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to '0'.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	For MMC only. The card sends its EXT_CSD register as a block of data.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on CMD the line.
CMD12	ac	[31:0] stuff bits	R1b	STOP TRANSMISSION	Forces the card to stop transmission
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	adtc	[31:0] stuff bits	R1	BUSTEST_R	A host reads the reversed bus testing data pattern from a card.

CMD15	ac	[31:16] RCA [15:0] reserved bits	-	GO_INACTIVE_STATE	Sends an addressed card into the Inactive State. This command is used when the host explicitly wants to deactivate a card.
CMD19	adtc	[31:0] stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a card.

Table 19-5. Block-Oriented read commands (class 2)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	In the case of a Standard Capacity SD and MMC, this command sets the block length (in bytes) for all following block commands (read, write, lock). Default is 512 Bytes. Set length is valid for memory access commands only if partial block read operation are allowed in CSD. In the case of a High Capacity SD Memory Card, block length set by CMD16 command does not affect the memory read and write commands. Always 512 Bytes fixed block length is used. In both cases, if block length is set larger than 512Bytes, the card sets the BLOCK_LEN_ERROR bit.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	In the case of a Standard Capacity SD and MMC, this command reads a block of the size selected by the SET_BLOCKLEN command. In the case of a High Capacity Card, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_TRANSMISSION command. Block length is specified the same as READ_SINGLE_BLOCK command.
Note: The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register					

Table 19-6. Stream read commands (class 1) and stream write commands (class 3)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.
Note: The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register					

Table 19-7. Block-Oriented write commands (class 4)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	See description in ***Table 19-5. Block-Oriented read commands (class 2)***.
CMD23	ac	[31:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. If the argument is all 0s, the subsequent read/write operation will be open-ended.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	In the case of a Standard Capacity SD, this command writes a block of the size selected by the SET_BLOCKLEN command. In the case of a SDHC, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE _BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows. Block length is specified the same as WRITE_BLOCK command.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

Note: 1.The data transferred shall not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD. In the case that write partial blocks is not supported, then the block length=default block length (given in CSD).

2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card.

Table 19-8. Erase commands (class 5)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD32	ac	[31:0] data address	R1	ERASE_WR_ BLK_START	Sets the address of the first write block to be erased.(SD)
CMD33	ac	[31:0] data address	R1	ERASE_WR_ BLK_END	Sets the address of the last write block of the continuous range to be erased.(SD)
CMD35	ac	[31:0]data address	R1	ERASE_GRO UP_ START	Sets the address of the first erase group within a range to be selected for erase.(MMC)
CMD36	ac	[31:0]data address	R1	ERASE_GRO UP_END	Sets the address of the last erase group within a continuous range to be selected for erase.(MMC)
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erases all previously selected write blocks.

Note: 1.CMD34 and CMD37 are reserved in order to maintain backwards compatibility with older versions of the MMC.

2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card.

Table 19-9. Block oriented write protection commands (class 6)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_ PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE). A High Capacity SD Memory Card does not support this command.
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_ PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE _PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.

Note: 1. High Capacity SD Memory Card does not support these three commands.

Table 19-10. Lock card (class 7)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCK_LEN	See description in *Table 19-5. Block-Oriented read commands (class 2)*.
CMD42	adtc	[31:0] Reserved bits (Set all 0)	R1	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command. Reserved bits in the argument and in Lock Card Data Structure shall be set to 0.

Table 19-11. Application-specific commands (class 8)

Cmd index	type	argument	Response format	Abbreviation	Description
ACMD41	bcr	[31]reserved bit [30]HCS [29:24]reserved bits [23:0]V_{DD} Voltage Window(OCR[23:0])	R3	SD_SEND_OP_COND	Sends host capacity support information (HCS) and asks the accessed card to send its operating condition register(OCR) content in the response. HCS is effective when card receives SEND_IF_COND command. CCS bit is assigned to OCR[30].
ACMD42	ac	[31:1] stuff bits [0]set_cd	R1	SET_CLR_CARD_DETECT	Connect[1]/Disconnect[0] the 50K pull-up resistor on CD/DAT3 (pin 1) of the card.
ACMD51	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits. [0] RD/WR	R1	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose/application specific command. The host sets RD/WR=1 for reading data from the card and sets to 0 for writing data to the card.
CMD60	adtc	[31] WR [23:18] Address [7:2] Byte	R1(read)/ R1b(write)	RW_MULTIPLE_REGISTER	Read or write register in address range.

		Count Other bits are reserved bits.			
CMD61	adtc	[31] WR [15:0] Data Unit Count Other bits are reserved bits	R1(read)/ R1b(write)	RW_MULTIPLE_BLOCK	Read or write data block in address range.
Note: 1.ACMDx is Application-specific Commands for SD memory. 2. CMD60, CMD61 are Application-specific Commands for CE-ATA device.					

Table 19-12. I/O mode commands (class 9)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD39	ac	[31:16] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8 bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register if the write flag is cleared to 0. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode
CMD52	adtc	[31] R/W Flag [30:28] Function Number [27] RAW Flag [26] Stuff Bits [25:9] Register Address [8] Stuff Bits [7:0] Write Data/Stuff Bits	R5	IO_RW_DIRECT	The IO_RW_DIRECT is the simplest means to access a single register within the total 128K of register space in any I/O function, including the common I/O area (CIA). This command reads or writes 1 byte using only 1 command/response pair. A common use is to initialize registers or monitor status values for I/O functions. This command is the fastest means to read or write single I/O registers, as it requires only a single command/response pair.

CMD53	adtc	[31] R/W Flag [30:28] Function Number [27] Block Mode [26] OP code [25:9] Register Address [8:0] Byte/Block Count		IO_RW_EXTENDED	This command allows the reading or writing of a large number of I/O registers with a single command.
Note: 1.CMD39, CMD40 are only for MMC. 2. CMD52, CMD53 are only for SD I/O card.					

Table 19-13. Switch function commands (class 10)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD6	adtc	[31] Mode 0:Check function 1:Switch function [30:24] reserved [23:20] reserved for function group 6 (0h or Fh) [19:16] reserved for function group 5 (0h or Fh) [15:12] reserved for function group 4 (0h or Fh) [11:8] reserved for function group 3 (0h or Fh) [7:4] function group 2 for command system [3:0] function group 1 for access mode	R1	SWITCH_FUNC	Only for SD memory and SD I/O. Checks switchable function (mode 0) and switch card function (mode 1).

19.5.3. Responses

All responses are sent on the CMD line. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type.

Responses types

There are 7 types of responses show as follows.

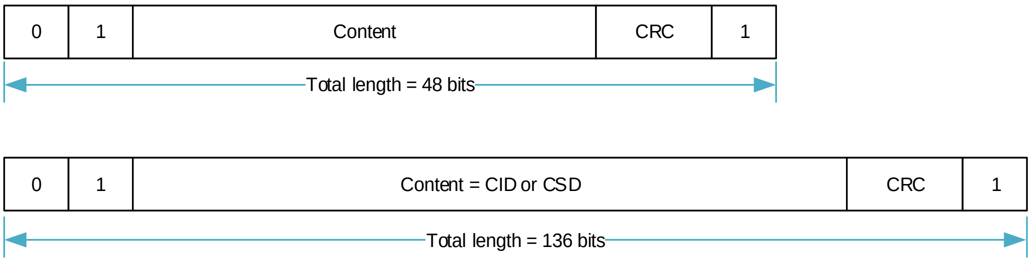
- **R1 / R1b :** normal response command.
- **R2 :** CID, CSD register.
- **R3 :** OCR register.
- **R4 :** Fast I/O.
- **R5 :** Interrupt request.
- **R6 :** Published RCA response.
- **R7 :** Card interface condition.

The SD Memory Card support five types of them, R1 / R1b, R2, R3, R6, R7. And the SD I/O Card and MMC supports additional response types named R4 and R5, but they are not exactly the same for SD I/O Card and MMC.

Responses format

Responses have two formats, as show in ***Figure 19-8. Response Token Format***, all responses are sent on the CMD line. The code length depends on the response type. Except R2 is 136 bits length, others are all 48 bits length.

Figure 19-8. Response Token Format



A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value ‘x’ in the tables below indicates a variable entry. All responses except for the type R3 are protected by a CRC. Every command code word is terminated by the end bit (always 1).

R1 (normal response command)

Code length is 48 bits. The bits 45:40 indicate the index of the command to be responded to, this value being interpreted as a binary coded number (between 0 and 63). The status of the card is coded in 32 bits. Note that if a data transfer to the card is involved, then a busy signal may appear on the data line after the transmission of each block of data. The host shall check for busy after data block transmission. The card status is described in ***Two status fields of the card***.

Table 19-14. Response R1

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	‘0’	‘0’	x	x	x	‘1’
description	start bit	transmission bit	command	card status	CRC7	end bit

			index			

R1b

R1b is identical to R1 with an optional busy signal transmitted on the data line DAT0. The card may become busy after receiving these commands based on its state prior to the command reception. The Host shall check for busy at the response.

R2 (CID, CSD register)

Code length is 136 bits. The contents of the CID register are sent as a response to the commands CMD2 and CMD10. The contents of the CSD register are sent as a response to CMD9. Only the bits [127..1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response.

Table 19-15. Response R2

Bit position	135	134	[133:128]	[127:1]	0
Width	1	1	6	127	1
Value	‘0’	‘0’	‘111111’	x	‘1’
description	start bit	transmission bit	reserved	CID or CSD register and internal CRC7	end bit

R3 (OCR register)

Code length is 48 bits. The contents of the OCR register are sent as a response to ACMD41 (SD memory), CMD1 (MMC). The response of different cards may have a little different.

Table 19-16. Response R3

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	‘0’	‘0’	‘111111’	x	‘1111111’	‘1’
description	start bit	transmission bit	reserved	OCR register	reserved	end bit

R4 (Fast I/O)

For MMC only. Code length 48 is bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its contents. The status bit in the argument is set if the operation was successful.

Table 19-17. Response R4 for MMC

Bit position	47	46	[45:40]	[39:8] Argument field				[7:1]	0
Width	1	1	6	16	1	7	8	7	1
Value	‘0’	‘0’	‘100111’	x	x	x	x	x	‘1’
description	start	transmission	CMD39	RCA	status	register	read	CRC7	end

	bit	bit		[31:16]	[15]	address [14:8]	register contents [7:0]		bit

R4b

For SD I/O only. Code length is 48 bits. The SDIO card receive the CMD5 will respond with a unique SD I/O response R4.

Table 19-18. Response R4 for SD I/O

Bit position	47	46	[45:40]	39	[38:36]	35	[34:32]	31	[30:8]	[7:1]	0
Width	1	1	6	1	3	1	3	1	23	7	1
Value	‘0’	‘0’	‘111111’	x	x	x	‘000’	x	x	‘1111111’	1
description	start bit	transmission bit	Reserved	C	Number of I/O functions	Memory Present	Stuff Bits	S18A	I/O OCR	Reserved	end bit

R5 (Interrupt request)

For MMC only. Code length is 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

Table 19-19. Response R5 for MMC

Bit position	47	46	[45:40]	[39:8] Argument field		[7:1]	0
Width	1	1	6	16	16	7	1
Value	‘0’	‘0’	‘101000’	x	x	x	‘1’
description	start bit	transmission bit	CMD40	RCA [31:16] of winning card or of the host	[15:0] Not defined. May be used for IRQ data	CRC7	end bit

R5b

For SD I/O only. The SDIO card's response to CMD52 and CMD53 is R5. If the communication between the card and host is in the 1-bit or 4-bit SD mode, the response shall be in a 48-bit response (R5).

Table 19-20. Response R5 for SD I/O

Bit position	47	46	[45:40]	[39:24]	[23:16]	[15:8]	[7:1]	0
Width	1	1	6	16	8	8	7	1
Value	‘0’	‘0’	‘11020X’	‘0’	x	x	x	‘1’
description	start bit	transmission bit	CMD52/53	Stuff Bits	Response Flags	Read or Write Data	CRC7	end bit

R6 (Published RCA response)

Code length is 48 bit. The bits [45:40] indicate the index of the command to be responded to (CMD3). The 16 MSB bits of the argument field are used for the Published RCA number.

Table 19-21. Response R6

Bit position	47	46	[45:40]	[39:8] Argument field		[7:1]	0
Width	1	1	6	16	16	7	1
Value	‘0’	‘0’	‘000011’	x	x	x	‘1’
description	start bit	transmission bit	CMD3	New published RCA of the card	card status bits:23,22,19,12:0	CRC7	end bit

R7 (Card interface condition)

For SD memory only. Code length is 48 bits. The card support voltage information is sent by the response of CMD8. Bits 19-16 indicate the voltage range that the card supports. The card that accepted the supplied voltage returns R7 response. In the response, the card echoes back both the voltage range and check pattern set in the argument.

Table 19-22. Response R7

Bit position	47	46	[45:40]	[39:20]	[19:16]	[15:8]	[7:1]	0
Width	1	1	6	20	4	8	7	1
Value	‘0’	‘0’	‘001000’	‘00000h’	x	x	x	‘1’
description	start bit	transmission bit	CMD8	Reserved bits	Voltage accepted	echo-back of check pattern	CRC7	end bit

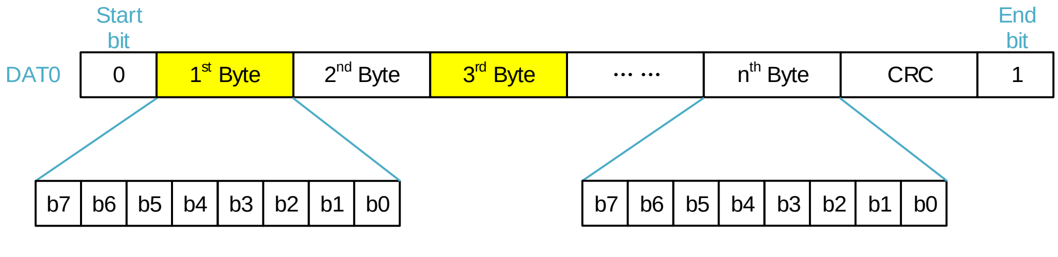
19.5.4. Data packets format

There are 3 data bus mode, 1-bit, 4-bit and 8-bit width. 1-bit mode is mandatory, 4-bit and 8-bit mode is optional. Although using 1-bit mode, DAT3 also need to notify card current working mode is SDIO or SPI, when card reset and initialize.

1-bit data packet format

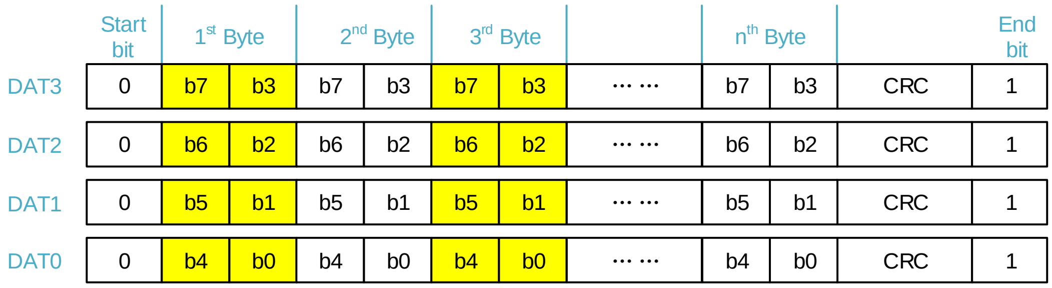
After card reset and initialize, only DAT0 pin is used to transfer data. And other pin can be used freely. ***Figure 19-9. 1-bit data bus width***, ***Figure 19-10. 4-bit data bus width*** and ***Figure 19-11. 8-bit data bus width*** show the data packet format when data bus wide is 1-bit, 4-bit and 8-bit.

Figure 19-9. 1-bit data bus width



4-bit data packet format

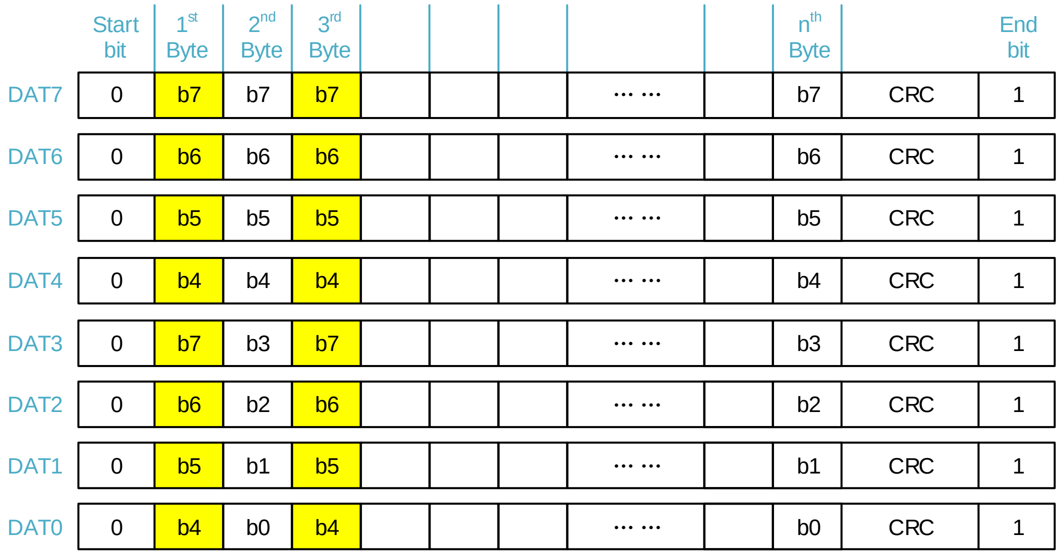
Figure 19-10. 4-bit data bus width



	Start bit	1st Byte		2nd Byte		3rd Byte			nth Byte			End bit
DAT3	0	b7	b3	b7	b3	b7	b3	……	b7	b3	CRC	1
DAT2	0	b6	b2	b6	b2	b6	b2	……	b6	b2	CRC	1
DAT1	0	b5	b1	b5	b1	b5	b1	……	b5	b1	CRC	1
DAT0	0	b4	b0	b4	b0	b4	b0	……	b4	b0	CRC	1

8-bit data packet format

Figure 19-11. 8-bit data bus width



	Start bit	1st Byte	2nd Byte	3rd Byte						nth Byte		End bit
DAT7	0	b7	b7	b7				……		b7	CRC	1
DAT6	0	b6	b6	b6				……		b6	CRC	1
DAT5	0	b5	b5	b5				……		b5	CRC	1
DAT4	0	b4	b4	b4				……		b4	CRC	1
DAT3	0	b7	b3	b7				……		b3	CRC	1
DAT2	0	b6	b2	b6				……		b2	CRC	1
DAT1	0	b5	b1	b5				……		b1	CRC	1
DAT0	0	b4	b0	b4				……		b0	CRC	1

19.5.5. Two status fields of the card

The SD Memory supports two status fields and others just support the first one:

Card Status: Error and state information of a executed command, indicated in the response

SD Status: Extended status field of 512 bits that supports special features of the SD Memory Card and future Application-Specific features.

Card status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

The type and clear condition fields in the table are abbreviated as follows:

Type

•E: Error bit. Send an error condition to the host. These bits are cleared as soon as the response (reporting the error) is sent out.

•S: Status bit. These bits serve as information fields only, and do not alter the execution of the command being responded to. These bits are persistent, they are set and cleared in accordance with the card status.

•R: Exceptions are detected by the card during the command interpretation and validation phase (Response Mode).

•X: Exceptions are detected by the card during command execution phase (Execution Mode).

Clear condition

•A: According to current state of the card.

•B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).

•C: Cleared by read

Table 19-23. Card status

Bits	Identifier	Type	Value	Description	Clear Condition
31	OUT_OF_RANGE	ERX	’0’= no error ’1’= error	The command’s argument was out of the allowed range for this card.	C
30	ADDRESS_ERROR	ERX	’0’= no error ’1’= error	A misaligned address which did not match the block length was used in the command.	C
29	BLOCK_LEN_ERROR	ERX	’0’= no error ’1’= error	The transferred block length is not allowed for this card, or the number of transferred bytes does not match the block length.	C
28	ERASE_SEQ_ERROR	ER	’0’= no error ’1’= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	ERX	’0’= no error ’1’= error	An invalid selection of write-blocks for erase occurred.	C
26	WP_VIOLATION	ERX	’0’= not protected ’1’= protected	Set when the host attempts to write to a protected block or to the temporary or permanent write protected card.	C

25	CARD_IS_LOCKED	SX	‘0’ = card unlocked ‘1’ = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAILED	ERX	’0’= no error ’1’= error	Set when a sequence or password error has been detected in lock/unlock card command.	C
23	COM_CRC_ERROR	ER	’0’= no error ’1’= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	ER	’0’= no error ’1’= error	Command not legal for the card state.	B
21	CARD_ECC_FAILED	ERX	’0’= success ’1’= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	ERX	’0’= no error ’1’= error	Internal card controller error.	C
19	ERROR	ERX	’0’= no error ’1’= error	A general or an unknown error occurred during the operation.	C
18	UNDERRUN	ERX	’0’= no error ’1’= error	Only for MMC. The card could not sustain data transfer in stream read mode.	C
17	OVERRUN	ERX	’0’= no error ’1’= error	Only for MMC. The card could not sustain data programming in stream write mode.	C
16	CID/ CSD_OVERWRITE	ERX	’0’= no error ’1’= error	Can be either one of the following errors: - The read only section of the CSD does not match the card content. - An attempt to reverse the copy (set as original) or permanent WP(unprotected) bits was made.	C
15	WP_ERASE_SKIP	ERX	’0’= not protected ’1’= protected	Set when only partial address space was erased due to existing write protected blocks or the temporary or permanent write protected card was erased.	C
14	CARD_ECC_DISABLED	SX	’0’= enabled ’1’= disabled	The command has been executed without using the	A

				internal ECC.	
13	ERASE_RESET	SR	’0’= cleared ’1’= set	An erase sequence was cleared before executing because an out of erase sequence command was received.	C
[12:9]	CURRENT_STATE	SX	0 = idle 1 = ready 2 = identification 3 = stand by 4 = transfer 5 = send data 6 = receive data 7 = programming 8 = disconnect 9-14 = reserved 15 = reserved for I/O mode	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as a binary coded number between 0 and 15.	B
8	READY_FOR_DATA	SX	’0’= not ready ’1’= ready	Corresponds to buffer empty signaling on the bus.	A
7	SWITCH_ERROR	EX	’0’= no error ’1’= switch error	If set, the card don’t switch to the expected mode as requested by the SWITCH command.	B
6	Reserved				
5	APP_CMD	SR	’0’= enabled ’1’= disabled	The card will expect ACMD, or an indication that the command has been interpreted as ACMD.	C
4	Reserved				
3	AKE_SEQ_ERROR	ER	’0’= no error ’1’= error	Only for SD memory. Error in the sequence of the authentication process.	C
2	Reserved for application specific commands.				
[1:0]	Reserved for manufacturer test mode.				

Note: 18, 17, 7 bits are only for MMC. 14, 3 bits are only for SD memory.

SD status register

The SD Status contains status bits that are related to the SD Memory Card proprietary features and may be used for future application-specific usage. The size of the SD Status is one data block of 512 bits. The content of this register is transmitted to the Host over the DAT bus along with a 16-bit CRC. The SD Status is sent to the host over the DAT bus as a response to ACMD13 (CMD55 followed with CMD13). ACMD13 can be sent to a card only in 'transfer state' (card is selected). The SD Status structure is described below.

The same abbreviation for 'type' and 'clear condition' were used as for the Card Status above.

Table 19-24. SD status

Bits	Identifier	Type	Value	Description	Clear Condition
[511:510]	DAT_BUS_WIDTH	SR	’00’= 1 (default) ‘01’= reserved ‘10’= 4 bit width ‘11’= reserved	Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	SR	’0’= Not in the mode ’1’= In Secured Mode	Card is in Secured Mode of operation (refer to the “SD Security Specification”).	A
[508:496]	reserved				
[495:480]	SD_CARD_TYPE	SR	The following cards are currently defined: ’0000’= Regular SD RD/WR Card. ’0001’= SD ROM Card '0002'= OTP	In the future, the 8 LSBs will be used to define different variations of an SD Memory Card (Each bit will define different SD Types). The 8 MSBs will be used to define SD Cards that do not comply with current SD Physical Layer Specification.	A
[479:448]	SIZE_OF_PROTECTED_AREA	SR	Size of protected area	(See below)	A
[447:440]	SPEED_CLASS	SR	Speed class of the card	(See below)	A
[439:432]	PERFORMANCE_MOVE	SR	Performance of move indicated by 1 [MB/s] step.	(See below)	A
[431:428]	AU_SIZE	SR	Size of AU	(See below)	A
[427:424]	reserved				
[423:408]	ERASE_SIZE	SR	Number of AUs to be erased at a time	(See below)	A
[407:4	ERASE_TIMEOUT	SR	Timeout value for	(See below)	A

02]			erasing areas specified by UNIT_OF_ERASE_AU		
[401:400]	ERASE_OFFSET	SR	Fixed offset value added to erase time.	(See below)	A
[399:312]	reserved				
[311:0]	reserved for manufacturer				

SIZE_OF_PROTECTED_AREA

Setting this field differs between SDSC and SDHC/SDXC.

In case of SDSC Card, the capacity of protected area is calculated as follows:

Protected Area = SIZE_OF_PROTECTED_AREA_* MULT * BLOCK_LEN.

SIZE_OF_PROTECTED_AREA is specified by the unit in MULT*BLOCK_LEN.

In case of SDHC and SDXC Cards, the capacity of protected area is calculated as follows:

Protected Area = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA is specified by the unit in byte.

SPEED_CLASS

This 8-bit field indicates the Speed Class.

00h: Class 0

01h: Class 2

02h: Class 4

03h: Class 6

04h: Class 10

05h–FFh: Reserved

PERFORMANCE_MOVE

This 8-bit field indicates Pm and the value can be set by 1 [MB/sec] step. If the card does not move useing RUs, Pm should be considered as infinity. Setting to FFh means infinity. The minimum value of Pm is defined in ***Table 19-25. Performance move field***.

Table 19-25. Performance move field

PERFORMANCE_MOVE	Value Definition
00h	Sequential Write

01h	1 [MB/sec]
02h	2 [MB/sec]
....	......
FEh	254 [MB/sec]
FFh	Infinity

AU_SIZE

This 4-bit field indicates AU Size and the value can be selected from 16 KB.

Table 19-26. AU_SIZE field

AU_SIZE	Value Definition
0h	Not Defined
1h	16 KB
2h	32 KB
3h	64 KB
4h	128 KB
5h	256 KB
6h	512 KB
7h	1 MB
8h	2 MB
9h	4 MB
Ah	8 MB
Bh	12 MB
Ch	16 MB
Dh	24 MB
Eh	32 MB
Fh	64 MB

The maximum AU size, depends on the card capacity, is defined in ***Table 19-26. AU_SIZE field***. The card can set any AU size specified in ***Table 19-27. Maximum AU size*** that is less than or equal to the maximum AU size. The card should set smaller AU size as possible.

Table 19-27. Maximum AU size

Card Capacity	up to 64MB	up to 256MB	up to 512MB	up to 32GB	up to 2TB
Maximum AU Size	512 KB	1 MB	2 MB	4 MB1	64MB

ERASE_SIZE

This 16-bit field indicates NERASE. When NERASE of AUs are erased, the timeout value is specified by ERASE_TIMEOUT (Refer to ERASE_TIMEOUT). The host should determine proper number of AUs to be erased in one operation so that the host can indicate progress of erase operation. If this field is set to 0, the erase timeout calculation is not supported.

Table 19-28. Erase size field

ERASE_SIZE	Value Definition
0000h	Erase Time-out Calculation is not supported.
0001h	1 AU
0002h	2 AU
0003h	3 AU
....	....
FFFFh	65535 AU

ERASE_TIMEOUT

This 6-bit field indicates the T_{ERASE} and the value indicates erase timeout from offset when multiple AUs are erased as specified by ERASE_SIZE. The range of ERASE_TIMEOUT can be defined as up to 63 seconds and the card manufacturer can choose any combination of ERASE_SIZE and ERASE_TIMEOUT depending on the implementation. Once ERASE_TIMEOUT is determined, it determines the ERASE_SIZE. The host can determine timeout for any number of AU erase by the equation below.

$$\text{Erase timeout of X AU} = \frac{T_{ERASE}}{N_{ERASE}} * X + T_{OFFSET} \tag{19-1}$$

Table 19-29. Erase timeout field

ERASE_TIMEOUT	Value Definition
00	Erase Time-out Calculation is not supported.
01	1 [sec]
02	2 [sec]
03	3 [sec]
....	....
63	63 [sec]

If ERASE_SIZE field is set to 0, this field shall be set to 0.

ERASE_OFFSET

This 2-bit field indicates the T_{OFFSET} and one of four values can be selected. This field is meaningless if ERASE_SIZE and ERASE_TIMEOUT fields are set to 0.

Table 19-30. Erase offset field

ERASE_OFFSET	Value Definition
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

19.6. Programming sequence

19.6.1. Card identification

The host will be in card identification mode after reset and while it is looking for new cards on the bus. While in card identification mode the host resets all the cards, validates operation voltage range, identifies cards and asks them to publish Relative Card Address (RCA). This operation is done to each card separately on its own CMD line. All data communication in the Card Identification Mode uses the command line (CMD) only.

During the card identification process, the card shall operate in the clock frequency of the identification clock rate F_{OD} (400 kHz).

Card reset

The command GO_IDLE_STATE (CMD0) is the software reset command and sets MMC and SD memory card into Idle State regardless of the current card state. The reset command (CMD0) is only used for memory or the memory portion of Combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write 1 to the RES bit in the CCCR. Cards in Inactive State are not affected by this command.

After power-on by the host, all cards are in Idle State, including the cards that have been in Inactive State before. After power-on or CMD0, all cards' CMD lines are in input mode, waiting for start bit of the next command. The cards are initialized with a default relative card address (RCA) and with a default driver strength with 400 KHz clock frequency.

Operating voltage range validation

At the start of communication between the host and the card, the host may not know the card supported voltage and the card may not know whether it supports the current supplied voltage. To verify the voltage, the following commands are defined in the related specification.

The SEND_OP_COND (CMD1 for MMC), SD_SEND_OP_COND (ACMD41 for SD memory), IO_SEND_OP_COND (CMD5 for SD I/O) command is designed to provide hosts with a mechanism to identify and reject cards which do not match the V_{DD} range desired by the host. This is accomplished by the host sending the required V_{DD} voltage window as the operand of this command. If the card cannot perform data transfer in the specified range it must discard itself from further bus operations and go into Inactive State. Otherwise, the card shall respond sending back its V_{DD} range.

If the card can operate on the supplied voltage, the response echoes back the supply voltage and the check pattern that were set in the command argument.

If the card cannot operate on the supplied voltage, it returns no response and stays in idle state. It is mandatory to issue CMD8 prior to ACMD41 to initialize SDHC Card. Receipt of

CMD8 makes the cards realize that the host supports the Physical Layer Version 2.00 and the card can enable new functions.

Card identification process

The card identification process differs in different cards. The card can be of the type MMC, CE-ATA, SD, or SD I/O. All types of SD I/O cards are supported, that is, SDIO_IO_ONLY, SDIO_MEM_ONLY, and SDIO COMBO cards. The identification process sequence includes the following steps:

1. Check if the card is connected.

2. Identify the card type; SD, MMC(CE-ATA), or SD I/O.

– Send CMD5 first. If a response is received, then the card is SD I/O

– If not, send ACMD41; if a response is received, then the card is SD.

– Otherwise, the card is an MMC or CE-ATA.

3. Initialization the card according to the card type.

Use a clock source with a frequency = F_{OD} (that is, 400 KHz) and use the following command sequence:

– SD card - Send CMD0, ACMD41, CMD2, CMD3.

– SDHC card - send CMD0, CMD8, ACMD41, CMD2, CMD3.

– SD I/O - Send CMD52, CMD0, CMD5, if the card doesn’t have memory port, send CMD3; otherwise send ACMD41, CMD11 (optional), CMD2, and CMD3.

– MMC/CE-ATA - Send CMD0, CMD1, CMD2, CMD3.

4. Identify the MMC/CE-ATA device.

– CPU should query the byte 504 (S_CMD_SET) of EXT_CSD register by sending CMD8. If bit 4 is set to 1, then the device supports ATA mode.

– If ATA mode is supported, the CPU should select the ATA mode by setting the ATA bit (bit 4) in the EXT_CSD register slice 191(CMD_SET) to activate the ATA command set. The CPU selects the command set using the SWITCH (CMD6) command.

– In the presence of a CE-ATA device, the FAST_IO (CMD39) and RW_MULTIPLE_REGISTER (CMD60) commands will succeed and the returned data will be the CE-ATA reset signature.

19.6.2. No data commands

To send any non-data command, the software needs to program the SDIO_CMDCTL register and the SDIO_CMDAGMT register with appropriate parameters. Using these two registers, the host forms the command and sends it to the command bus. The host reflects the errors in the command response through the error bits of the SDIO_STAT register.

When a response is received the host sets the CMDRECV (CRC check passed) or CCRCERR (CRC check error) bit in the SDIO_STAT register. A short response is copied in SDIO_RESP0, while a long response is copied to all four response registers. The SDIO_RESP3 bit 31 represents the MSB, and the SDIO_RESP0 bit 0 represents the LSB of a long response.

19.6.3. Single block or multiple block write

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card. The block consists of start bits (1 or 4 bits LOW), data block, CRC and end bits(1 or 4 bits HIGH). If the CRC fails, the card indicates the failure on the SDIO_DAT line and the transferred data are discarded and not written, and all further transmitted blocks are ignored.

If the host uses partial blocks whose accumulated length is not block aligned, block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. The card shall set the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer. The write operation will also be aborted if the host tries to write data on a write protected area. In this case, however, the card will set the WP_VIOLATION bit (in the status register).

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

Some cards may require long and unpredictable time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT0 line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

For SD card. Setting a number of write blocks to be pre-erased (ACMD23) will make a following Multiple Block Write operation faster compared to the same operation without preceding ACMD23. The host will use this command to define how many number of write blocks are going to be send in the next write operation.

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the SDIO_DATALEN register.

2. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register; the host sends data in blocks of size BLKSZ.

3. Program SDIO_CMDAGMT register with the data address to which data should be written.

4. Program the SDIO_CMDCTL register. For SD memory and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SD I/O cards, use CMD53 for both single-block and multiple-block transfers. For CE-ATA, first use CMD60 to write the ATA task file, then use CMD61 to write the data. After writing to the CMD register, host starts executing a command, when the command is sent to the bus, the CMDRECV flag is set.

5. Write data to SDIO_FIFO.

6. Software should look for data error interrupts. If required, software can terminate the data transfer by sending the STOP command (CMD12).

7. When a DTEND interrupt is received, the data transfer is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the host should send the STOP command.

19.6.4. Single block or multiple block read

Block read is block oriented data transfer. The basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN), it is always 512 bytes. If READ_BL_PARTIAL(in the CSD) is set, smaller blocks whose starting and ending address are entirely contained within 512 bytes boundary may be transmitted.

CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. CRC is appended to the end of each block, ensuring data transfer integrity.

Block Length set by CMD16 can be set up to 512 bytes regardless of READ_BL_LEN.

Blocks will be continuously transferred until a STOP_TRANSMISSION command (CMD12) is issued. The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

When the last block of user area is read using CMD18, the host should ignore OUT_OF_RANGE error that may occur even the sequence is correct.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first misaligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.

Steps involved in a single block or multiple block read are:

1. Write the data size in bytes in the SDIO_DATALEN register.

2. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register. The host expects data from the card in blocks of size BLKSZ each.

3. Program the SDIO_CMDAGMT register with the data address of the beginning of a data read.

4. Program the SDIO_CMDCTL. For SD and MMC cards, using CMD17 for a single-block read and CMD18 for a multiple-block read. For SD I/O cards, using CMD53 for both single-block and multiple-block transfers. For CE-ATA, first using CMD60 to write the ATA task file, then using CMD 61 to read the data. After writing to the CMD register, the host starts executing the command, when the command is sent to the bus, the CMDRECV flag is set.

5. Software should look for data error interrupts. If required, software can terminate the data transfer by sending a STOP command.

6. The software should read data from the FIFO and make space in the FIFO for receiving more data.

7. When a DTEND interrupt is received, the software should read the remaining data in the FIFO.

19.6.5. Stream write and stream read (MMC only)

Stream write

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. Since the amount of data to be transferred is not determined in advance, CRC cannot be used.

If the host provides an out of range address as an argument to CMD20, the card will reject the command, remain in Tran state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream write command works only on a 1 bit bus configuration (on DAT0). If CMD20 is issued in other bus configurations, it is regarded as an illegal command.

In order to sustain data transfer in stream mode of the card, the time it takes to receive the data (defined by the bus clock rate) must be less than the time it takes to program it into the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for the stream write operation is given by the following formula:

$$\text{max write frequence} = \min\left(\text{TRAN\_SPEED}, \frac{8*2^{\text{WRITE\_BL\_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W\_FACTOR}}\right) \qquad (19\text{-}2)$$

TRAN_SPEED: Max bus clock frequency.

WRITE_BL_LEN: Max write data block length.

NSAC: Data read access-time 2 in CLK cycles.

TAAC: Data read access-time 1.

R2W_FACTOR: Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the TXURE bit set and return to Transfer state

Stream read

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its data, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

If the host provides an out of range address as an argument to CMD11, the card will reject the command, remain in Transfer state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream read command works only on a 1 bit bus configuration (on DAT0). If CMD11 is issued in other bus configurations, it is regarded as an illegal command.

If the end of the memory range is reached while sending data, and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined. As the host sends CMD12 the card will respond with the ADDRESS_OUT_OF_RANGE bit set and return to Tran state.

In order to sustain data transfer in stream mode of the card, the time it takes to transmit the data (defined by the bus clock rate) must be less than the time it takes to read it out of the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for stream read operation is given by the following formula:

$$\text{max read frequence} = \min\left(\text{TRAN\_SPEED}, \frac{8*2^{\text{READ\_BL\_LEN}} - 100*\text{NSAC}}{\text{TAAC}*\text{R2W\_FACTOR}}\right) \tag{19-3}$$

TRAN_SPEED: Max bus clock frequency.

READ_BL_LEN: Max read data block length.

NSAC: Data read access-time 2 in CLK cycles.

TAAC: Data read access-time 1.

R2W_FACTOR: Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the RXORE bit set and return to Transfer state

19.6.6. Erase

The erasable unit of the MMC/SD memory is the “Erase Group”; Erase group is measured in write blocks which are the basic writable units of the card. The size of the Erase Group is a card specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35)/ERASE_WR_BLK_START(CMD32) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36)/ERASE_WR_BLK_END(CMD33) command and finally it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card will ignore all LSB’s below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command (CMD35, CMD36, and CMD38) is received out of the defined erase sequence, the card shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence.

If the host provides an out of range address as an argument to CMD35 or CMD36, the card will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence.

If an ‘non erase’ command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the card shall respond with the ERASE_RESET bit set, reset the erase sequence and execute the last command.

If the erase range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set.

As described above for block write, the card will indicate that an erase is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

19.6.7. Bus width selection

After the host has verified the functional pins on the bus it should change the bus width configuration.

For MMC, using the SWITCH command (CMD6).The bus width configuration is changed by writing to the BUS_WIDTH byte in the Modes Segment of the EXT_CSD register (using the SWITCH command to do so). After power-on or software reset, the contents of the BUS_WIDTH byte is 0x00. If the host tries to write an invalid value, the BUS_WIDTH byte is not changed and the SWITCH_ERROR bit is set. This register is write only.

For SD memory, using SET_BUS_WIDTH command (ACMD6) to change the bus width. The default bus width after power up or GO_IDLE_STATE command (CMD0) is 1 bit. SET_BUS_WIDTH (ACMD6) is only valid in a transfer state, which means that the bus width can be changed only after a card is selected by SELECT/DESELECT_CARD (CMD7).

19.6.8. Protection management

In order to allow the host to protect data against erase or write, three methods for the cards are supported in the card:

CSD register for card protection (optional)

The entire card may be write protected by setting the permanent or temporary write protect bits in the CSD. Some cards support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD. It is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. The SET_WRITE_PROT command sets the write protection of the addressed write protected group, and the CLR_WRITE_PROT command clears the write protection of the addressed write protected group.

The High Capacity SD Memory Card does not support Write Protection and does not respond to write protection commands (CMD28, CMD29 and CMD30).

Write protect switch on the card (SD memory and SD I/O card)

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open it means that the card is write protected. If the window is closed the card is not write protected.

Password card Lock/Unlock Operation

The Password Card Lock/Unlock protection is described in ***Card Lock/Unlock operation***.

19.6.9. Card Lock/Unlock operation

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size are kept in a 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0), ACMD41, CMD16 and lock card command class (class 7). Thus, the host is allowed to reset, initialize, select, query for status, but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0), the card will be locked automatically after power on.

Similar to the existing CSD register write commands, the lock/unlock command is available in "transfer state" only. This means that it does not include an address argument and the card shall be selected before using it.

The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). ***Table 19-31. Lock card data structure*** describes the structure of the command data block.

Table 19-31. Lock card data structure

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved(all set to 0)				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	Password data(PWD)							
……								
PWDS_LEN+1								

ERASE: 1 Defines Forced Erase Operation. In byte 0, bit 3 will be set to 1 (all other bits shall be 0). All other bytes of this command will be ignored by the card.

LOCK/UNLOCK: 1 = Locks the card. 0 = Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).

CLR_PWD: 1 = Clears PWD.

SET_PWD: 1 = Set new password to PWD.

PWDS_LEN: Defines the following password(s) length (in bytes). In case of a password change, this field includes the total password length of old and new passwords. The password length is up to 16 bytes. In case of a password change, the total length of the old password and the new password can be up to 32 bytes.

Password data: In case of setting a new password, it contains the new password. In case of a password change, it contains the old password followed by the new password.

Setting the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the new password. In the case that a password replacement is done, then the block size shall consider that both passwords (the old and the new one) are sent with the command.
- Send the Card Lock/Unlock command with the appropriate data block size on the data

line including the 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWDS_LEN) and the password itself. In the case that a password replacement is done, then the length value (PWDS_LEN) shall include both passwords (the old and the new one) and the password data field shall include the old password (currently used) followed by the new password. Note that the card shall handle the calculation of the new password length internally by subtracting the old password length from PWDS_LEN field.

- In the case that the sent old password is not correct (not equal in size and content), then the LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In the case that the sent old password is correct (equal in size and content), then the given new password and its size will be saved in the PWD and PWD_LEN registers, respectively.

Reset the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWDS_LEN), and the password itself. If the PWD and PWD_LEN content match the sent password and its size, then the content of the PWD register is cleared and PWD_LEN is set to 0. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Locking a card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWDS_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Unlocking the card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line

including the 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWDS_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

19.7. Specific operations

19.7.1. SD I/O specific operations

The SD I/O only card and SD I/O combo card support these specific operations:

Read Wait operation

Suspend/resume operation

Interrupts

The SD I/O supports these operations only if the SDIO_DATACTL[11] bit is set, except for read suspend that does not need specific hardware implementation.

SD I/O read wait operation

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation allows a host to signal a card that is executing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O card. To determine if a card supports the Read Wait protocol, the host shall test SRW capability bit in the Card Capability byte of the CCCR. The timing for Read Wait is based on the Interrupt Period. If a card does not support the Read Wait protocol, the only means a host has to stall (not abort) data in the middle of a read multiple command is to control the SDIO_CLK. The limitation of this method is that with the clock stopped, the host cannot issue any commands, and so cannot perform other operations during the delay time. Read Wait support is mandatory for the card to support suspend/resume. ***Figure 19-12. Read wait control by stopping SDIO_CLK*** and ***Figure 19-13. Read wait operation using SDIO_DAT[2]*** show the Read Wait mode about stop the SDIO_CLK and use SDIO_DAT[2].

Figure 19-12. Read wait control by stopping SDIO_CLK

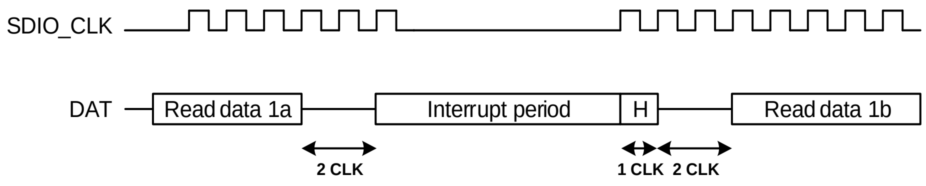
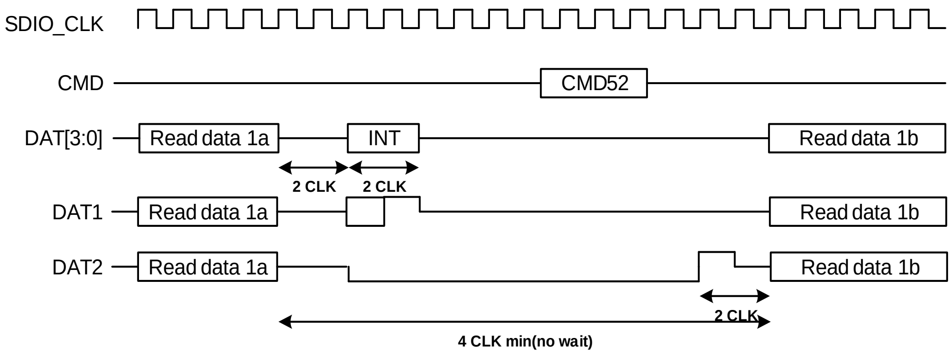


Figure 19-13. Read wait operation using SDIO_DAT[2]



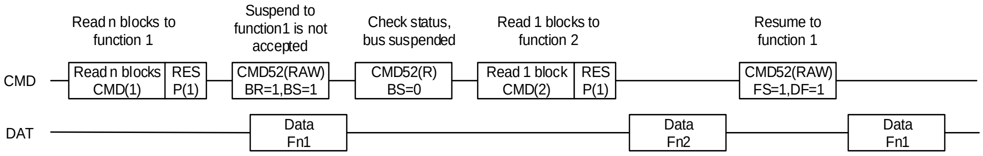
We can start the Read Wait interval before the data block is received: when the data unit is enabled (SDIO_DATACTL[0] bit set), the SD I/O specific operation is enabled (SDIO_DATACTL[11] bit set), Read Wait starts (SDIO_DATACTL[10] = 0 and SDIO_DATACTL[8] = 1) and data direction is from card to SD I/O (SDIO_DATACTL[1] = 1), the DSM directly moves from Idle to Read Wait. In Read Wait the DSM drives SDIO_DAT[2] to 0 after 2 SDIO_CLK clock cycles. In this state, when you set the RWSTOP bit (SDIO_DATACTL[9]), the DSM remains in Wait for two more SDIO_CLK clock cycles to drive SDIO_DAT[2] to 1 for one clock cycle. The DSM then starts waiting again until it receives data from the card. The DSM will not start a Read Wait interval while receiving a block even if Read Wait start is set: the Read Wait interval will start after the CRC is received. The RWSTOP bit has to be cleared to start a new Read Wait operation. During the Read Wait interval, the SDIO can detect SD I/O interrupts on SDIO_DAT[1].

SD I/O suspend/resume operation

Within a multi-function SD I/O or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SD I/O and combo cards can implement the optional concept of suspend/resume. If a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory. Once this higher-priority transfer is completed, the original transfer is re-started where it left off (resume).

Figure 19-14. Function2 read cycle inserted during function1 multiple read cycle shows a condition where the first suspend request is not immediately accepted. The host then checks the status of the request with a read and determines that the bus has now been released (BS=0). At this time, a read to function 2 is started. Once that single block read is completed, the resume is issued to function, causing the data transfer to resume (DF=1).

Figure 19-14. Function2 read cycle inserted during function1 multiple read cycle



When the host sends data to the card, the host can suspend the write operation. The SDIO_CMDCTL[11] bit is set and indicates to the CSM that the current command is a suspend command. The CSM analyzes the response and when the response is received from the card (suspend accepted), it acknowledges the DSM that goes Idle after receiving the CRC token of the current block.

To suspend a read operation, the DSM waits in the WaitR state, when the function to be suspended sends a complete packet just before stopping the data transaction. The application should continue reading receive FIFO until the FIFO is empty, and the DSM goes Idle state automatically.

Interrupts

In order to allow the SD I/O card to interrupt the host, an interrupt function is added to a pin on the SD interface. Pin number 8, which is used as SDIO_DAT[1] when operating in the 4-bit SD mode, is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SD I/O interrupt is "level sensitive", that is, the interrupt line shall be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via function unique I/O operation.

When setting the SDIO_DATACTL[11] bit SD I/O interrupts can detect on the SDIO_DAT[1] line.

Figure 19-15. Read Interrupt cycle timing shows the timing of the interrupt period for single data transaction read cycles.

Figure 19-15. Read Interrupt cycle timing

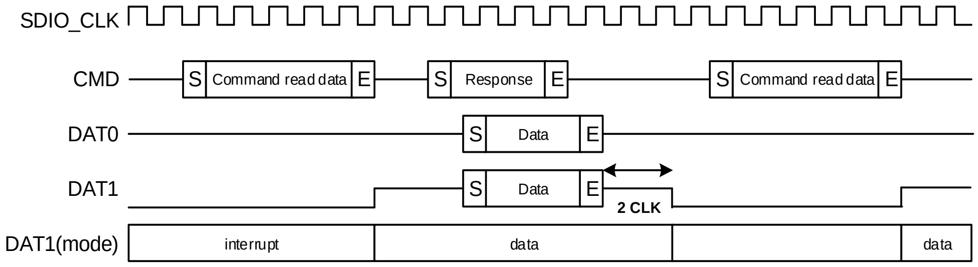
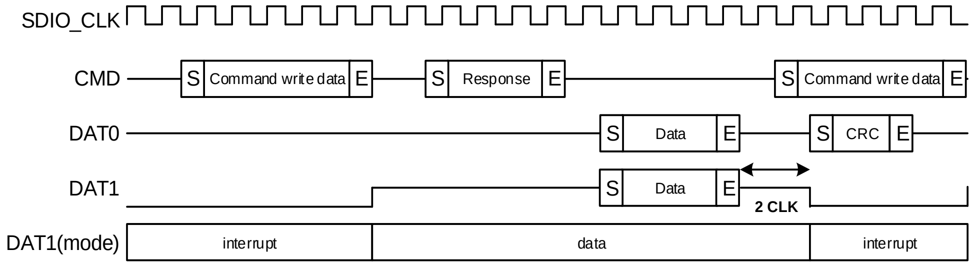


Figure 19-16. Write interrupt cycle timing



When transferring multiple blocks of data in the 4-bit SD mode, a special definition of the interrupt period is required. In order to allow the highest speed of communication, the interrupt period is limited to a 2-clock interrupt period. Card that wants to send an interrupt signal to the host shall assert DAT1 low for the first clock and high for the second clock. The card shall then release DAT1 into the hi-Z State. ***Figure 19-17. Multiple block 4-Bit read interrupt cycle timing*** shows the operation for an interrupt during a 4-bit multi-block read and ***Figure 19-18. Multiple block 4-Bit write interrupt cycle timing*** shows the operation for an interrupt during a 4-bit multi-block write

Figure 19-17. Multiple block 4-Bit read interrupt cycle timing

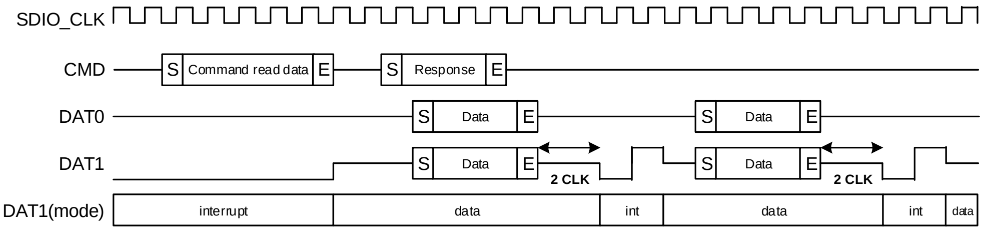
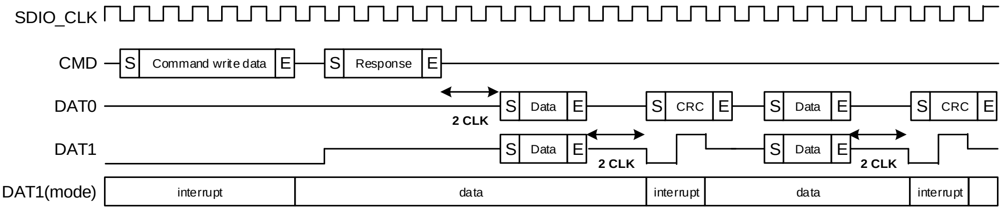


Figure 19-18. Multiple block 4-Bit write interrupt cycle timing



19.7.2. CE-ATA specific operations

The CE-ATA device supports these specific operations:

Receive command completion signal

Send command completion disable signal

The SDIO supports these operations only when SDIO_CMDCTL[14] is set.

Command completion signal

CE-ATA defines a command completion signal that the device uses to notify the host upon normal ATA command completion or when ATA command termination has occurred due to an error condition the device has encountered.

If the ‘enable CMD completion’ bit SDIO_CMDCTL[12] is set and the ‘not interrupt Enable’ bit SDIO_CMDCTL[13] is reset, the CSM waits for the command completion signal in the Waitcompl state.

When start bit is received on the CMD line, the CSM enters the Idle state. No new command can be sent for 7 bit cycles. Then, for the last 5 cycles (out of the 7) the CMD line is driven to‘1’ in push-pull mode.

After the host detects a command completion signal from the device, it should issue a

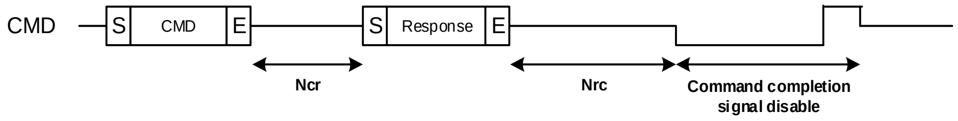
FAST_IO (CMD39) command to read the ATA Status register to determine the ending status for the ATA command.

Command completion disable signal

The host may cancel the ability for the device to return a command completion signal by issuing the command completion signal disable. The host shall only issue the command completion signal disable when it has received an R1b response for an outstanding RW_MULTIPLE_BLOCK (CMD61) command.

Command completion signal disable is sent 8 bit cycles after the reception of a short response if the 'enable CMD completion' bit, SDIO_CMDCTL[12] is not set and the 'not interrupt Enable' bit SDIO_CMDCTL[13] is reset.

Figure 19-19. The operation for command completion disable signal



19.8. Register definition

19.8.1. Power control register (SDIO_PWRCTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1:0	PWRCTL[1:0]	SDIO power control bits. These bits control the SDIO state, card input or output. 00: SDIO power off: SDIO cmd/data state machine reset to IDLE, clock to card stopped, no cmd/data output to card 01: Reserved 10: Reserved 11: SDIO Power on

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

19.8.2. Clock control register (SDIO_CLKCTL)

Address offset: 0x04

Reset value: 0x0000 0000

This register controls the output clock SDIO_CLK.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HWCLKEN	CLKEDGE	BUSMODE[1:0]		CLKBYP	CLKPWRSAV	CLKEN	DIV[7:0]							
	rw	rw	rw		rw	rw	rw	rw							

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	HWCLKEN	Hardware Clock Control enable bit If this bit is set, hardware controls the SDIO_CLK on/off depending on the system bus is very busy or not. There is no underrun/overrun error when this bit is set, because hardware can close the SDIO_CLK when almost underrun/overrun. 0: HW Clock control is disabled 1: HW Clock control is enabled
13	CLKEDGE	SDIO_CLK clock edge selection bit 0: Select the rising edge of the SDIOCLK to generate SDIO_CLK 1: Select the falling edge of the SDIOCLK to generate SDIO_CLK
12:11	BUSMODE[1:0]	SDIO card bus mode control bit 00: 1-bit SDIO card bus mode selected 01: 4-bit SDIO card bus mode selected 10: 8-bit SDIO card bus mode selected
10	CLKBYP	Clock bypass enable bit This bit defines the SDIO_CLK is directly SDIOCLK or not. 0: NO bypass, the SDIO_CLK refers to DIV bits in SDIO_CLKCTL register. 1: Clock bypass, the SDIO_CLK is directly from SDIOCLK (SDIOCLK/1).
9	CLKPWRSAV	SDIO_CLK clock dynamic switch on/off for power saving. This bit controls SDIO_CLK clock dynamic switch on/off when the bus is idle for power saving 0: SDIO_CLK clock is always on 1: SDIO_CLK closed when bus idle
8	CLKEN	SDIO_CLK clock output enable bit 0: SDIO_CLK is disabled 1: SDIO_CLK is enabled
7:0	DIV[7:0]	Clock division This field and DIV[8] bit defines the division factor to generator SDIO_CLK clock to card. The SDIO_CLK is divider from SDIOCLK if CLKBYP bit is 0, and the SDIO_CLK frequency = SDIOCLK / (DIV[8:0] + 2).

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

19.8.3. Command argument register (SDIO_CMDAGMT)

Address offset: 0x08

Reset value: 0x0000 0000

This register defines 32 bit command argument, which will be used as part of the command (bit 39 to bit 8).

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">CMDAGMT[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CMDAGMT[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:0	CMDAGMT[31:0]	SDIO card command argument This field defines the SDIO card command argument which sent to card. This field is the bits [39:8] of command message. If the command message contains an argument, this field must update before writing SDIO_CMDCTL register when sending a command.

19.8.4. Command control register (SDIO_CMDCTL)

Address offset: 0x0C

Reset value: 0x0000 0000

The SDIO_CMDCTL register contains the command index and other command control bits to control command state machine.

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Reserved</td><td>ATAEN</td><td>NINTEN</td><td>ENCMDC</td><td>SUSPEND</td><td>CSMEN</td><td>WAITDEND</td><td>INTWAIT</td><td colspan="2">CMDRESP[1:0]</td><td colspan="6">CMDIDX[5:0]</td></tr>
<tr><td></td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td colspan="2">rw</td><td colspan="6">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	ATAEN	CE-ATA command enable(CE-ATA only) If this bit is set, the host enters the CE-ATA mode, and the CSM transfers CMD61. 0: CE-ATA disable 1: CE-ATA enable
13	NINTEN	No CE-ATA Interrupt (CE-ATA only) This bit defines if there is CE-ATA interrupt or not. This bit is only used when CE-ATA card. 0: CE-ATA interrupt enable

		1: CE_ATA interrupt disable
12	ENCMDC	CMD completion signal enabled (CE-ATA only) This bit defines if there is command completion signal or not in CE-ATA card. 0: no completion signal 1: have completion signal
11	SUSPEND	SD I/O suspend command(SD I/O only) This bit defines whether the CSM to send a suspend command or not. This bit is only used for SDIO card. 0: no effect 1: suspend command
10	CSMEN	Command state machine (CSM) enable bit 0: Command state machine disable (stay on CS_Idle) 1: Command state machine enable
9	WAITDEND	Waits for ends of data transfer. If this bit is set, the command state machine starts to send a command must wait the end of data transfer. 0: no effect 1: Wait the end of data transfer
8	INTWAIT	Interrupt wait instead of timeout This bit defines the command state machine to wait card interrupt at CS_Wait state in command state machine. If this bit is set, no command wait timeout generated. 0: Not wait interrupt. 1: Wait interrupt.
7:6	CMDRESP[1:0]	Command response type bits These bits define the response type after sending a command message. 00: No response 01: Short response 10: No response 11: Long response
5:0	CMDIDX[5:0]	Command index This field defines the command index to be sent to SDIO card.

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

19.8.5. Command index response register (SDIO_RSPCMDIDX)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RSPCMDIDX[5:0]					
										r					

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5:0	RSPCMDIDX[5:0]	Last response command index Read-only bits field. This field contains the command index of the last command response received. If the response doesn’t have the command index (long response and short response of R3), the content of this register is undefined.

19.8.6. Response register (SDIO_RESPx x=0..3)

Address offset: 0x14+(4*x), x=0..3

Reset value: 0x0000 0000

These register contains the content of the last card response received.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESPx[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	RESPx[31:0]	Card state. The content of the response, see ***Table 19-32. SDIO_RESPx register at different response type***.

The short response is 32 bits, the long response is 127 bits (bit 128 is the end bit 0).

Table 19-32. SDIO_RESPx register at different response type

Register	Short response	Long response
SDIO_RESP0	Card response[31:0]	Card response[127:96]
SDIO_RESP1	reserved	Card response [95:64]
SDIO_RESP2	reserved	Card response [63:32]
SDIO_RESP3	reserved	Card response [31:1],plus bit 0

19.8.7. Data timeout register (SDIO_DATATO)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">DATATO[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DATATO[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:0	DATATO[31:0]	Data timeout period These bits define the data timeout period count by SDIO_CLK. When the DSM enter the state WaitR or BUSY, the internal counter which loads from this register starts decrement. The DSM timeout and enter the state Idle and set the DTTMOUT flag when the counter decreases to 0.

Note: The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

19.8.8. Data length register (SDIO_DATALEN)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">DATALEN[24:16]</td></tr>
<tr><td colspan="7"></td><td colspan="9">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DATALEN[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value
24:0	DATALEN[24:0]	Data transfer length This register defined the number of bytes to be transferred. When the data transfer starts, the data counter loads this register and starts decrement.

Note: If block data transfer selected, the content of this register must be a multiple of the block size (refer to SDIO_DATACTL). The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

19.8.9. Data control register (SDIO_DATACTL)

Address offset: 0x2C

Reset value: 0x0000 0000

This register controls the DSM.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IOEN	RWTYPE	RWSTOP	RWEN	BLKSZ[3:0]				DMAEN	TRANS MOD	DATADIR	DATAEN
				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	IOEN	SD I/O specific function enable(SD I/O only) 0: Not SD I/O specific function 1: SD I/O specific function
10	RWTYPE	Read wait type(SD I/O only) 0: Read Wait control using SDIO_DAT[2] 1: Read Wait control by stopping SDIO_CLK
9	RWSTOP	Read wait stop(SD I/O only) 0: No effect 1: Stop the read wait process if RWEN bit is set
8	RWEN	Read wait mode enabled(SD I/O only) 0: Read wait mode is disabled 1: Read wait mode is enabled
7:4	BLKSZ[3:0]	Data block size These bits defined the block size when data transfer is block transfer. 0000: block size = 2^0 = 1 byte 0001: block size = 2^1 = 2 bytes 0010: block size = 2^2 = 4 bytes 0011: block size = 2^3 = 8 bytes 0100: block size = 2^4 = 16 bytes 0101: block size = 2^5 = 32 bytes 0110: block size = 2^6 = 64 bytes 0111: block size = 2^7 = 128 bytes 1000: block size = 2^8 = 256 bytes

		1001: block size = 2^9 = 512 bytes 1010: block size = 2^{10} = 1024 bytes 1011: block size = 2^{11} = 2048 bytes 1100: block size = 2^{12} = 4096 bytes 1101: block size = 2^{13} = 8192 bytes 1110: block size = 2^{14} = 16384 bytes 1111: reserved
3	DMAEN	DMA enable bit 0: DMA is disabled. 1: DMA is enabled.
2	TRANSMOD	Data transfer mode 0: Block transfer 1: Stream transfer or SDIO multibyte transfer
1	DATADIR	Data transfer direction 0: Write data to card. 1: Read data from card.
0	DATAEN	Data transfer enable bit Write 1 to this bit to start data transfer regardless this bit is 0 or 1. The DSM moves to Readwait state if RWEN is set or to the WaitS, WaitR state depend on DATADIR bit. Start a new data transfer, it not need to clear this bit to 0.

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

19.8.10. Data counter register (SDIO_DATACNT)

Address offset: 0x30

Reset value: 0x0000 0000

This register is read only. When the DSM from Idle to WaitR or WaitS, it loads value from data length register (SDIO_DATALEN). It decrements with the data transferred, when it reaches 0, the flag DTEND is set.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							DATACNT[24:16]								
							r								

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

Bits	Fields	Descriptions

31:25	Reserved	Must be kept at reset value
24:0	DATACNT[24:0]	Data count value Read-only bits field. When these bits are read, the number of remaining data bytes to be transferred is returned.

19.8.11. Status register (SDIO_STAT)

Address offset: 0x34
Reset value: 0x0000 0000

This register is read only. The following descripts the types of flag:
The flags of bit [23:22, 10:0] can only be cleared by writing 1 to the corresponding bit in interrupt clear register (SDIO_INTC).
The flags of bit [21:11] are changing depend on the hardware logic.
This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								ATAEND	SDIOINT	RXDTVAL	TXDTVAL	RFE	TFE	RFF	TFF
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFH	TFH	RXRUN	TXRUN	CMDRUN	DTBLKEND	STBITE	DTEND	CMDSEND	CMDRECV	RXORE	TXURE	DTTMOUT	CMDTMOUT	DTCRCERR	CCRCERR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	ATAEND	CE-ATA command completion signal received (only for CMD61)
22	SDIOINT	SD I/O interrupt received
21	RXDTVAL	Data is valid in receive FIFO
20	TXDTVAL	Data is valid in transmit FIFO
19	RFE	Receive FIFO is empty
18	TFE	Transmit FIFO is empty When HW Flow control is enabled, TFE signals becomes activated when the FIFO contains 2 words.
17	RFF	Receive FIFO is full When HW Flow control is enabled, RFF signals becomes activated 2 words before the FIFO is full.
16	TFF	Transmit FIFO is full

15	RFH	Receive FIFO is half full: at least 8 words can be read in the FIFO
14	TFH	Transmit FIFO is half empty: at least 8 words can be written into the FIFO
13	RXRUN	Data reception in progress
12	TXRUN	Data transmission in progress
11	CMDRUN	Command transmission in progress
10	DTBLKEND	Data block sent/received (CRC check passed)
9	STBITE	Start bit error in the bus.
8	DTEND	Data end (data counter, SDIO_DATACNT, is zero)
7	CMDSEND	Command sent (no response required)
6	CMDRECV	Command response received (CRC check passed)
5	RXORE	Received FIFO overrun error occurs
4	TXURE	Transmit FIFO underrun error occurs
3	DTTMOUT	Data timeout The data timeout period depends on the SDIO_DATATO register.
2	CMDTMOUT	Command response timeout The command timeout period has a fixed value of 64 SDIO_CLK clock periods.
1	DTCRCERR	Data block sent/received (CRC check failed)
0	CCRCERR	Command response received (CRC check failed)

19.8.12. Interrupt clear register (SDIO_INTC)

Address offset: 0x38

Reset value: 0x0000 0000

This register is write only. Writing 1 to the bit can clear the corresponding bit in the SDIO_STAT register.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								ATAENDC	SDIOINTC	Reserved					
								w	w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTBLKENDC	STBITEC	DTENDC	CMDSENDC	CMDRECVC	RXOREC	TXUREC	DTTMOUTC	CMDTMOUTC	DTCRCERRC	CCRCERRC
					w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	ATAENDC	ATAEND flag clear bit Write 1 to this bit to clear the flag.
22	SDIOINTC	SDIOINT flag clear bit Write 1 to this bit to clear the flag.
21:11	Reserved	Must be kept at reset value
10	DTBLKENDC	DTBLKEND flag clear bit Write 1 to this bit to clear the flag.
9	STBITEC	STBITE flag clear bit Write 1 to this bit to clear the flag.
8	DTENDC	DTEND flag clear bit Write 1 to this bit to clear the flag.
7	CMDSENDC	CMDSEND flag clear bit Write 1 to this bit to clear the flag.
6	CMDRECVC	CMDRECV flag clear bit Write 1 to this bit to clear the flag.
5	RXOREC	RXORE flag clear bit Write 1 to this bit to clear the flag.
4	TXUREC	TXURE flag clear bit Write 1 to this bit to clear the flag.
3	DTTMOUTC	DTTMOUT flag clear bit Write 1 to this bit to clear the flag.
2	CMDTMOUTC	CMDTMOUT flag clear bit Write 1 to this bit to clear the flag.
1	DTCRCERRC	DTCRCERR flag clear bit Write 1 to this bit to clear the flag.
0	CCRCERRC	CCRCERR flag clear bit Write 1 to this bit to clear the flag.

19.8.13. Interrupt enable register (SDIO_INTEN)

Address offset: 0x3C
Reset value: 0x0000 0000

This register enables the corresponding interrupt in the SDIO_STAT register.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								ATAENDIE	SDIOINTIE	RXDTVALIE	TXDTVALIE	RFEIE	TFEIE	RFFIE	TFFIE
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFHIE	TFHIE	RXRUNIE	TXRUNIE	CMDRUNIE	DTBLKENDIE	STBITEIE	DTENDIE	CMDSENDIE	CMDRECVIE	RXOREIE	TXUREIE	DTTMOUTIE	CMDTMOUTIE	DTCRCERRIE	CCRCERRIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	ATAENDIE	CE-ATA command completion signal received interrupt enable Write 1 to this bit to enable the interrupt.
22	SDIOINTIE	SD I/O interrupt received interrupt enable Write 1 to this bit to enable the interrupt.
21	RXDTVALIE	Data valid in receive FIFO interrupt enable Write 1 to this bit to enable the interrupt.
20	TXDTVALIE	Data valid in transmit FIFO interrupt enable Write 1 to this bit to enable the interrupt.
19	RFEIE	Receive FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
18	TFEIE	Transmit FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
17	RFFIE	Receive FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
16	TFFIE	Transmit FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
15	RFHIE	Receive FIFO half full interrupt enable Write 1 to this bit to enable the interrupt.
14	TFHIE	Transmit FIFO half empty interrupt enable Write 1 to this bit to enable the interrupt.
13	RXRUNIE	Data reception interrupt enable Write 1 to this bit to enable the interrupt.
12	TXRUNIE	Data transmission interrupt enable Write 1 to this bit to enable the interrupt.
11	CMDRUNIE	Command transmission interrupt enable

		Write 1 to this bit to enable the interrupt.
10	DTBLKENDIE	Data block end interrupt enable Write 1 to this bit to enable the interrupt.
9	STBITEIE	Start bit error interrupt enable Write 1 to this bit to enable the interrupt.
8	DTENDIE	Data end interrupt enable Write 1 to this bit to enable the interrupt.
7	CMDSENDIE	Command sent interrupt enable Write 1 to this bit to enable the interrupt.
6	CMDRECVIE	Command response received interrupt enable Write 1 to this bit to enable the interrupt.
5	RXOREIE	Received FIFO overrun error interrupt enable Write 1 to this bit to enable the interrupt.
4	TXUREIE	Transmit FIFO underrun error interrupt enable Write 1 to this bit to enable the interrupt.
3	DTTMOUTIE	Data timeout interrupt enable Write 1 to this bit to enable the interrupt.
2	CMDTMOUTIE	Command response timeout interrupt enable Write 1 to this bit to enable the interrupt.
1	DTCRCERRIE	Data CRC fail interrupt enable Write 1 to this bit to enable the interrupt.
0	CCRCERRIE	Command response CRC fail interrupt enable Write 1 to this bit to enable the interrupt.

19.8.14. FIFO counter register (SDIO_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								FIFOCNT[23:16]							
								r							

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

Bits	Fields	Descriptions

31:24	Reserved	Must be kept at reset value
23:0	FIFOCNT[23:0]	FIFO counter. These bits define the remaining number words to be written or read from the FIFO. It loads the data length register (SDIO_DATALEN[24:2] if SDIO_DATALEN is word-aligned or SDIO_DATALEN[24:2]+1 if SDIO_DATALEN is not word-aligned) when DATAEN is set, and start count decrement when a word write to or read from the FIFO.

19.8.15. FIFO data register (SDIO_FIFO)

Address offset: 0x80

Reset value: 0x0000 0000

This register occupies 32 entries of 32-bit words, the address offset is from 0x80 to 0xFC.

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODT[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	FIFODT[31:0]	Receive FIFO data or transmit FIFO data These bits are the data of receive FIFO or transmit FIFO. Write to or read from this register is write data to FIFO or read data from FIFO.

20. External memory controller (EXMC)

20.1. Overview

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory, it automatically converts AMBA memory access protocol into a specific memory access protocol defined in the configuration register, such as SRAM, ROM, NOR Flash, NAND Flash and PC Card. Users could also tweak with the timing parameters in the configuration registers to boost up memory access efficiency. EXMC access space is divided into multiple banks; each bank is assigned to access a specific memory type with flexible parameter configuration as defined in the control registers.

20.2. Characteristics

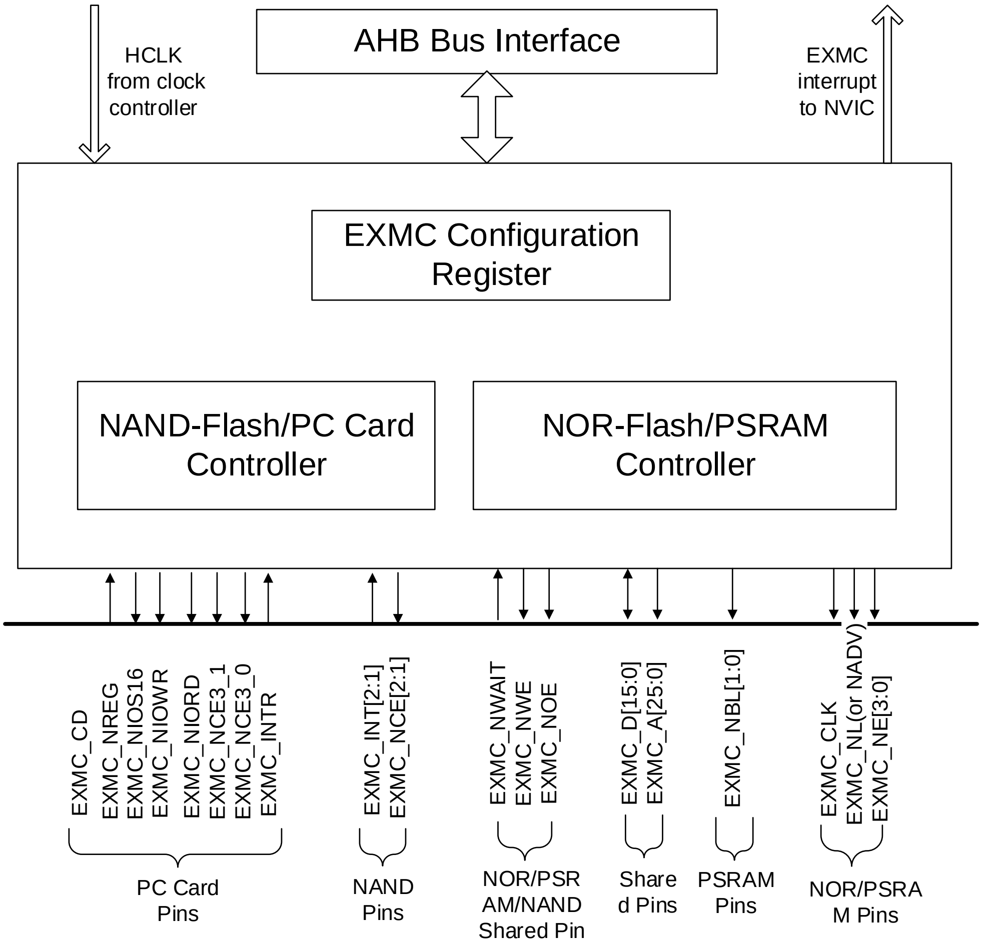
- Supported external memory:
  - SRAM
  - PSRAM
  - ROM
  - NOR Flash
  - 8-bit or 16-bit NAND Flash
  - 16-bit PC Card
- Protocol translation between the AMBA and the multitude of external memory protocol
- Offering a variety of programmable timing parameters to meet user’s specific needs
- Each bank has its own chip-select signal which can be configured independently
- Independent read/write timing configuration to a sub-set memory type
- Embedded ECC hardware for NAND Flash access
- 8 or 16 bits bus width
- Address and data bus multiplexing mechanism for NOR Flash and PSRAM
- Write enable and byte select are provided as needed
- Automatic AMBA transaction split when internal and external bus width is not compatible

20.3. Function overview

20.3.1. Block diagram

EXMC is the combination of five modules: The AHB bus interface, EXMC configuration registers, NOR/PSRAM controller, NAND/PC Card controller and external device interface. AHB clock (HCLK) is the reference clock.

Figure 20-1. The EXMC block diagram



20.3.2. Basic regulation of EXMC access

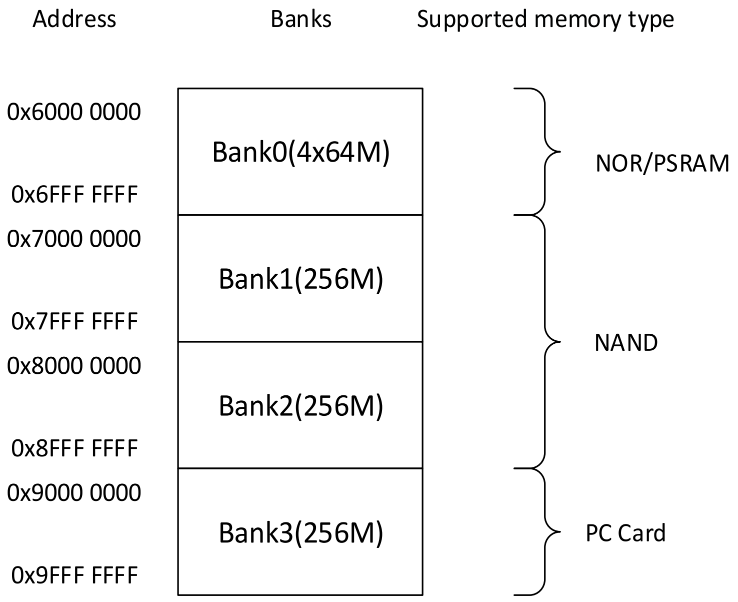
EXMC is the conversion interface between AHB bus and external device protocol. 32-bit of AHB read/write accesses can be split into several consecutive 8-bit or 16-bit read/write operations respectively. In the process of data transfer, AHB access data width and memory data width may not be the same. In order to ensure consistency of data transmission, EXMC's read/write accesses follow the following basic regulation.

- When the width of AHB bus equals to the memory bus width. No conversion is applied.

- When the width of AHB bus is greater than memory bus width. The AHB accesses are automatically split into several continuous memory accesses.
- When the width of AHB bus is smaller than memory bus width. If the external memory devices have the byte selection function, such as SRAM, ROM. PSRAM, the application can access the corresponding byte through their byte lane EXMC_NBL[1:0]. Otherwise, write operation is prohibited, but read operation is allowed unconditionally.

20.3.3. External device address mapping

Figure 20-2. EXMC memory banks



EXMC access space is divided into multiple banks. Each bank is 256 Mbytes. The first bank (Bank0) is further divided into four Regions, and each Region is 64 Mbytes. Bank1 and bank2 is each divided into two spaces, the attribute memory space and the common memory space. Bank3 is divided into three spaces, which are the attribute memory space, the common memory space and the I/O memory space.

Each bank or region has a separate chip-select control signal, which can be configured independently.

Bank0 is used for NOR and PSRAM device access.

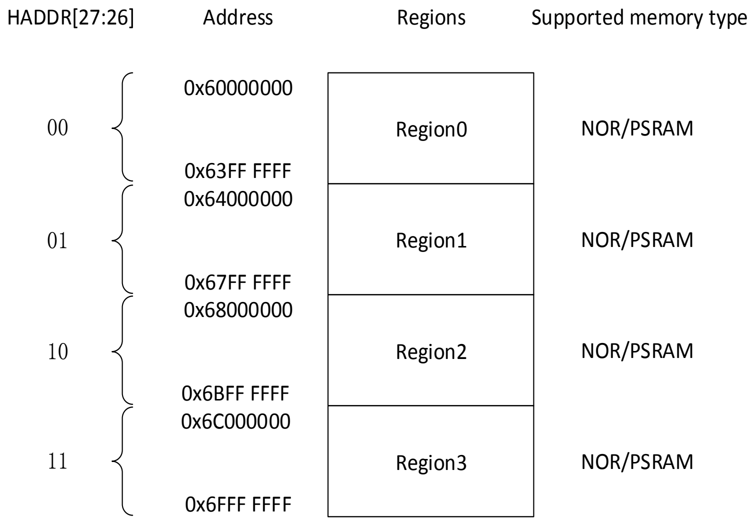
Bank1 and bank2 are used to access NAND Flash exclusively.

Bank3 is used for PC Card access.

NOR/PSRAM address mapping

Figure 20-3. Four regions of bank0 address mapping reflects the address mapping of the four regions of bank0. Internal AHB address lines HADDR [27:26] bits are used to select the four regions.

Figure 20-3. Four regions of bank0 address mapping



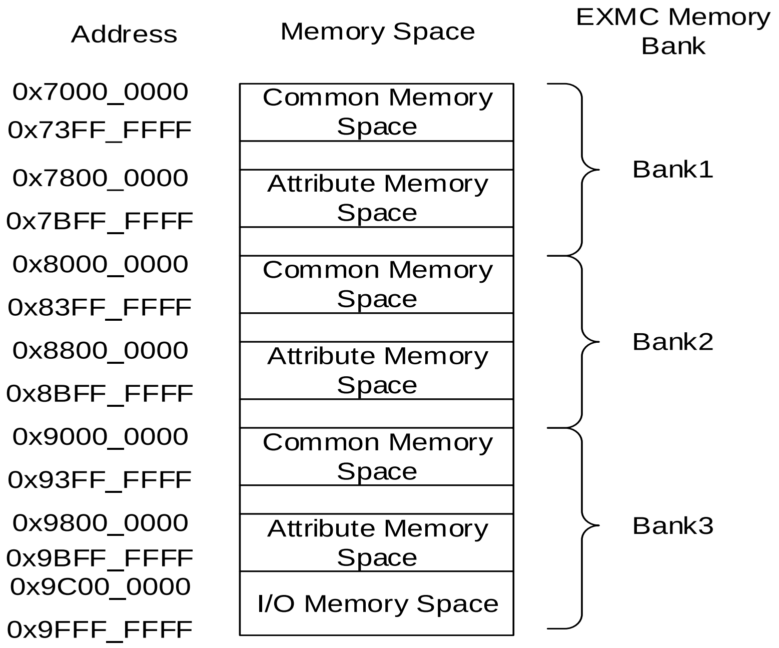
HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

- When data bus width of the external memory is 8-bits. In this case the memory address is byte aligned. HADDR[25:0] is connected to EXMC_A[25:0] and then the EXMC_A[25:0] is connected to the external memory address lines.
- When data bus width of the external memory is 16-bits. In this case the memory address is half-word aligned. HADDR byte address must be converted into half-word aligned by connecting HADDR[25:1] with EXMC_A[24:0]. The EXMC_A[24:0] is connected to the external memory address lines.

NAND/PC Card address mapping

Bank1 and bank2 are designed to access NAND Flash, and bank3 is designed to access PC Card. Each bank is further divided into several memory spaces as shown in Figure 20-4. NAND/PC Card address mapping.

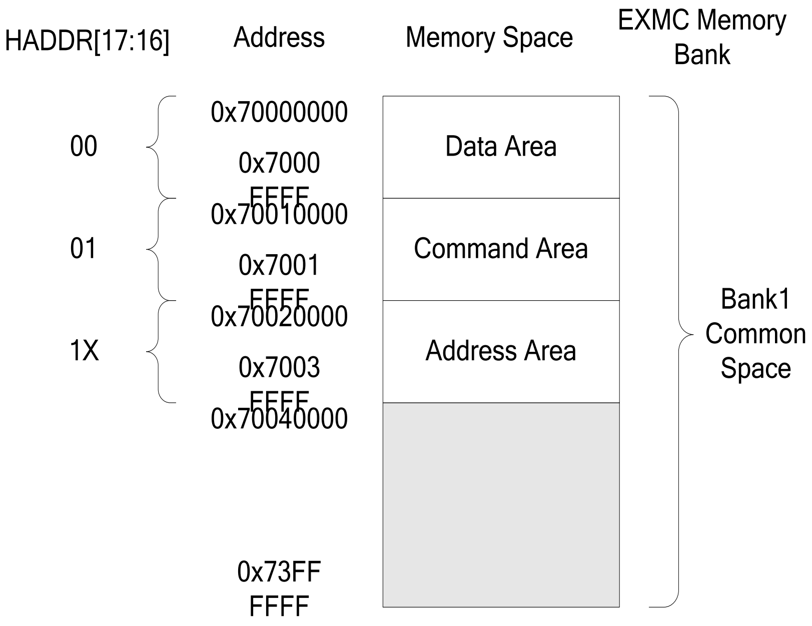
Figure 20-4. NAND/PC Card address mapping



NAND address mapping

For NAND Flash, the common space and the attribute space are further-divided into three areas individually, the data area, the command area and the address area as shown in ***Figure 20-5. Diagram of bank1 common space.***

Figure 20-5. Diagram of bank1 common space



HADDR [17:16] bits are used to select one of the three areas.

- When HADDR [17:16] = 00, the data area is selected.
- When HADDR [17:16] = 01, the command area is selected.
- When HADDR [17:16] = 1X, the address area is selected.

Application software uses these three areas to access NAND Flash, their definitions are as follows.

- Address area: This area is where the NAND Flash access address should be issued by software, the EXMC will pull the address latch enable (ALE) signal automatically in address transfer phase. ALE is mapped to EXMC_A [17].
- Command area: This area is where the NAND Flash access command should be issued by the software, the EXMC will pull the command latch enable (CLE) signal automatically in command transfer phase. CLE is mapped to EXMC_A [16].
- Data area: This area is where the NAND Flash read/write data should be accessed. When the EXMC is in data transfer mode, software should write the data to be transferred to the NAND Flash in this area. When the EXMC is in data reception mode, software should read the data from the NAND Flash by reading this area. Data access address is incremented automatically in consecutive mode, users need not to be concerned with access address area.

20.3.4. NOR/PSRAM controller

NOR/PSRAM memory controller controls bank0, which is designed to support NOR Flash, PSRAM, SRAM, ROM and honeycomb RAM external memory. EXMC has 4 independent chip-select signals for each of the 4 sub-banks within bank0, named NE[x] (x = 0, 1, 2, 3). Other signals for NOR/PSRAM access are shared. Each sub-bank has its own set of configuration register.

Note:

In asynchronous mode, all output signals of controller will change on the rise edge of internal AHB bus clock (HCLK).

In synchronous mode, all output data of controller will change on the fall edge of extern memory device clock (EXMC_CLK).

NOR/PSRAM memory device interface description

Table 20-1. NOR Flash interface signals description

EXMC Pin	Direction	Mode	Functional description
EXMC_CLK	Output	Sync	Clock signal for sync
Non-muxed EXMC_A[25:0]	Output	Async/Sync	Address bus signal
Muxed EXMC_A[25:16]			

EXMC Pin	Direction	Mode	Functional description
EXMC_D[15:0]	Input/output	Async/Sync (muxed)	Address/Data bus
	Input/output	Async/Sync (non-muxed)	Data bus
EXMC_NE[x]	Output	Async/Sync	Chip selection, x=0/1/2/3
EXMC_NOE	Output	Async/Sync	Read enable
EXMC_NWE	Output	Async/Sync	Write enable
EXMC_NWAIT	Input	Async/Sync	Wait input signal
EXMC_NL(NADV)	Output	Async/Sync	Address valid

Table 20-2. PSRAM non-muxed signal description

EXMC Pin	Direction	Mode	Functional description
EXMC_CLK	Output	Sync	Clock signal for sync
EXMC_A[25:0]	Output	Async/Sync	Address Bus
EXMC_D[15:0]	Input/output	Async/Sync	Data Bus
EXMC_NE[x]	Output	Async/Sync	Chip selection, x=0/1/2/3
EXMC_NOE	Output	Async/Sync	Read enable
EXMC_NWE	Output	Async/Sync	Write enable
EXMC_NWAIT	Input	Async/Sync	Wait input signal
EXMC_NL(NADV)	Output	Async/Sync	Latch enable (address valid enable, NADV)
EXMC_NBL[1]	Output	Async/Sync	Upper byte enable
EXMC_NBL[0]	Output	Async/Sync	Lower byte enable

Supported memory access mode

Table below shows an example of the supported devices type, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM.

Table 20-3. EXMC bank 0 supports all transactions

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
NOR Flash	Async	R	8	16	
	Async	R	16	16	
	Async	W	16	16	
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	32	16	Split into 2 EXMC accesses

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
	Sync	R	16	16	
	Sync	R	32	16	
PSRAM	Async	R	8	16	
	Async	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Async	R	16	16	
	Async	W	16	16	
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	32	16	Split into 2 EXMC accesses
	Sync	R	16	16	
	Sync	R	32	16	
	Sync	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Sync	W	16	16	
	Sync	W	32	16	Split into 2 EXMC accesses
SRAM and ROM	Async	R	8	8	
	Async	R	8	16	
	Async	R	16	8	Split into 2 EXMC accesses
	Async	R	16	16	
	Async	R	32	8	Split into 4 EXMC accesses
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	8	8	
	Async	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Async	W	16	8	
	Async	W	16	16	
	Async	W	32	8	
	Async	W	32	16	

NOR Flash/PSRAM controller timing

EXMC provides various programmable timing parameters and timing models for SRAM, ROM, PSRAM, NOR Flash and other external static memory.

Table 20-4. NOR / PSRAM controller timing parameters

Parameter	Function	Access mode	Unit	Min	Max
CKDIV	Sync Clock divide ratio	Sync	HCLK	2	16
DLAT	Data latency	Sync	EXMC_CLK	2	17
BUSLAT	Bus latency	Async/Sync read	HCLK	1	16
DSET	Data setup time	Async	HCLK	2	256
AHLD	Address hold time	Async(muxed)	HCLK	2	16
ASET	Address setup time	Async	HCLK	1	16

Table 20-5. EXMC_timing models

Timing model		Extend mode	Mode description	Write timing parameter	Read timing parameter
Async	Mode 1	0	SRAM/PSRAM/CRAM	DSET ASET	DSET ASET
	Mode 2	0	NOR Flash	DSET ASET	DSET ASET
	Mode A	1	SRAM/PSRAM/CRAM with EXMC_NOE toggling on data phase	WDSET WASET	DSET ASET
	Mode B	1	NOR Flash	WDSET WASET	DSET ASET
	Mode C	1	NOR Flash with EXMC_NOE toggling on data phase	WDSET WASET	DSET ASET
	Mode D	1	With address hold capability	WDSET WAHLD WASET	DSET AHLD ASET
	Mode AM	0	NOR Flash address/data mux	DSET AHLD ASET BUSLAT	DSET AHLD ASET BUSLAT
Sync	Mode E	0	NOR/PSRAM/CRAM synchronous read PSRAM/CRAM synchronous write	DLAT CKDIV	DLAT CKDIV
	Mode SM	0	NOR Flash address/data mux	DLAT CKDIV	DLAT CKDIV

As shown in ***Table 20-5. EXMC_timing models***, EXMC NOR Flash / PSRAM controller provides a variety of timing model, users can modify those parameters listed in ***Table 20-4. NOR / PSRAM controller timing parameters*** to satisfy different external memory type and user's requirements. When extended mode is enabled via the EXMODEN bit in EXMC_SNCTLx register, different timing patterns for read and write access could be generated independently according to EXMC_SNTCFGx and EXMC_SNWTCFGx register's configuration.

Asynchronous access timing diagram

Mode 1 - SRAM/CRAM

Figure 20-6. Mode 1 read access

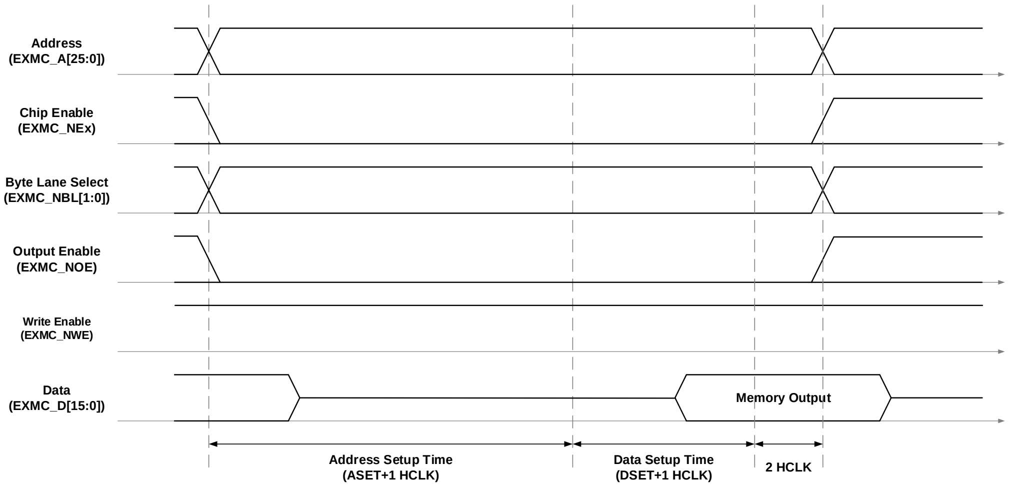


Figure 20-7. Mode 1 write access

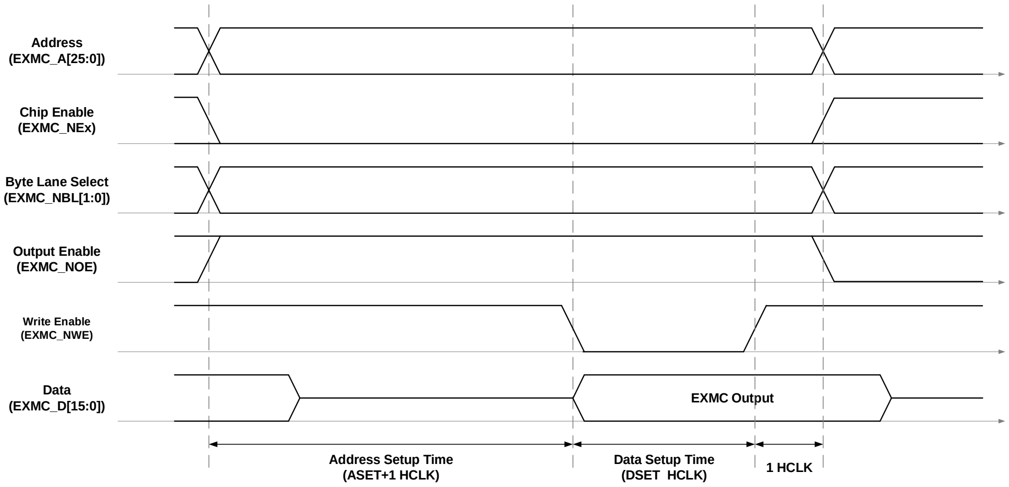


Table 20-6. Mode 1 related registers configuration

EXMC_SNCTLx		
Bit Position	Bit Name	Reference Setting Value
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWAIT	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WREN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1

8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	No effect
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory, except 2(Nor Flash)
1	NRMUX	0x0
0	NRBKEN	0x1
EXMC_SNTCFGx		
31-30	Reserved	0x0000
29-28	ASYNCMOD	No effect
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+1 HCLK for write, DSET+3 HCLK for read)
7-4	AHLD	No effect
3-0	ASET	Depends on memory and user

Mode A - SRAM/PSRAM(CRAM) OE toggling

Figure 20-8. Mode A read access

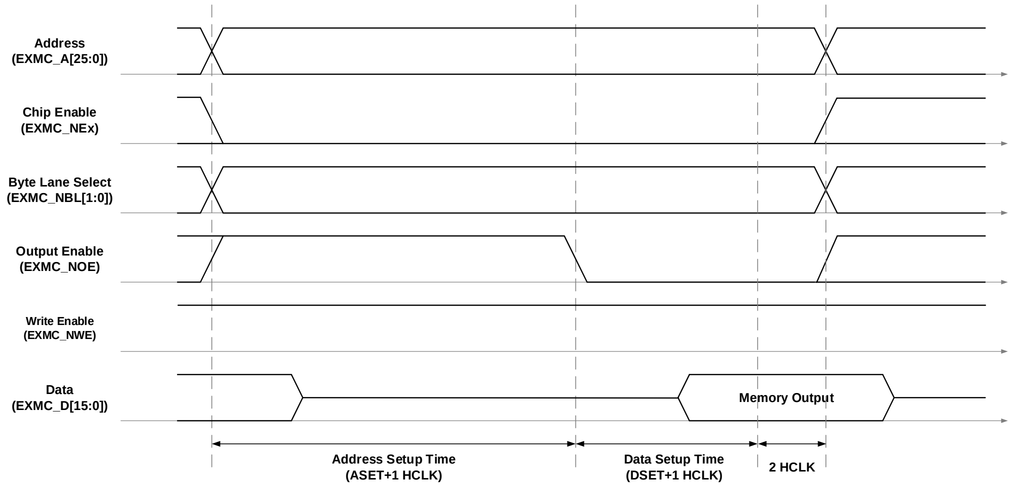
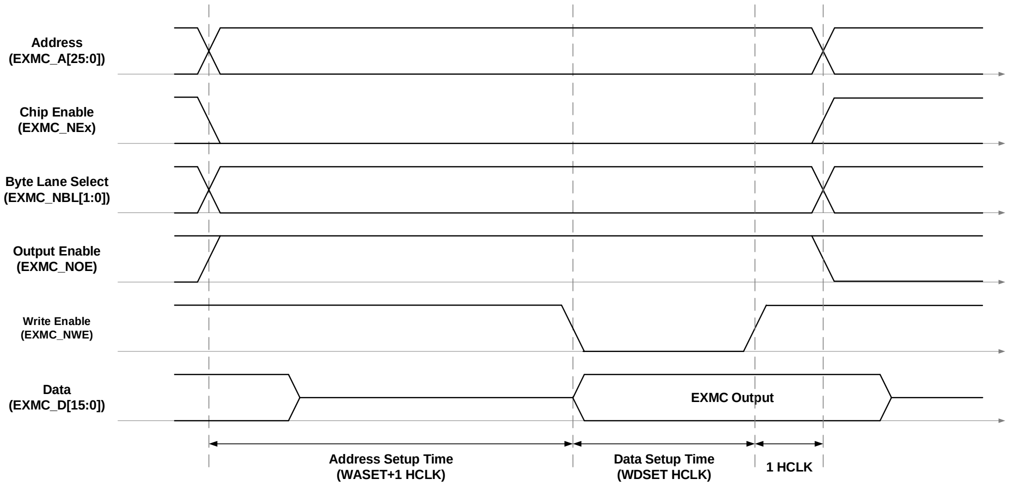


Figure 20-9. Mode A write access



The different between mode A and mode 1 write timing is that read/write timing is specified by the same set of timing configuration, while mode A write timing configuration is independent of its read configuration.

Table 20-7. Mode A related registers configuration

EXMC_SNCTLx		
Bit Position	**Bit Name**	**Reference Setting Value**
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WREN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	No effect
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory, except 2(Nor Flash)
1	NRMUX	0x0
0	NRBKEN	0x1
EXMC_SNTCFGx(Read)		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	No effect
23-20	CKDIV	No effect

19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	No effect
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx(Write)		
31-30	Reserved	0x0
29-28	WASYNCMOD	0x0
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	Reserved	0x00
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode 2/B - NOR Flash

Figure 20-10. Mode 2/B read access

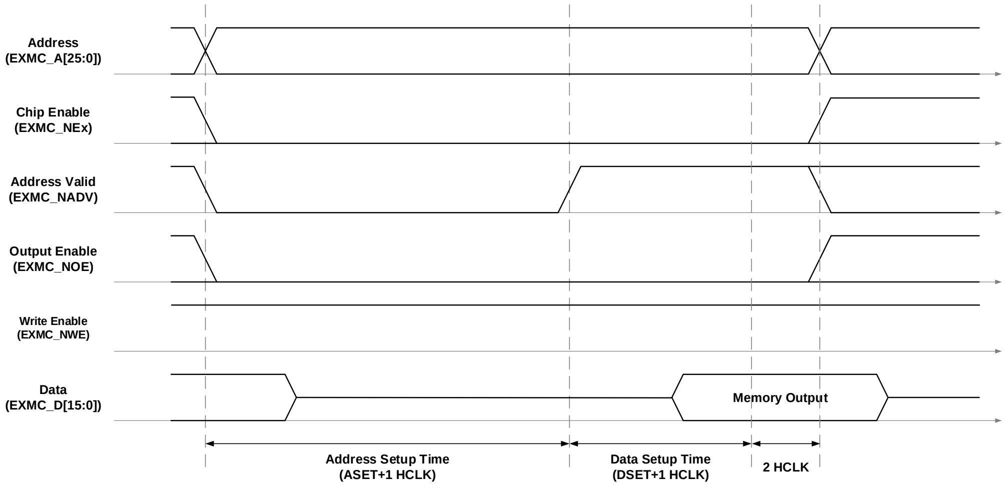


Figure 20-11. Mode 2 write access

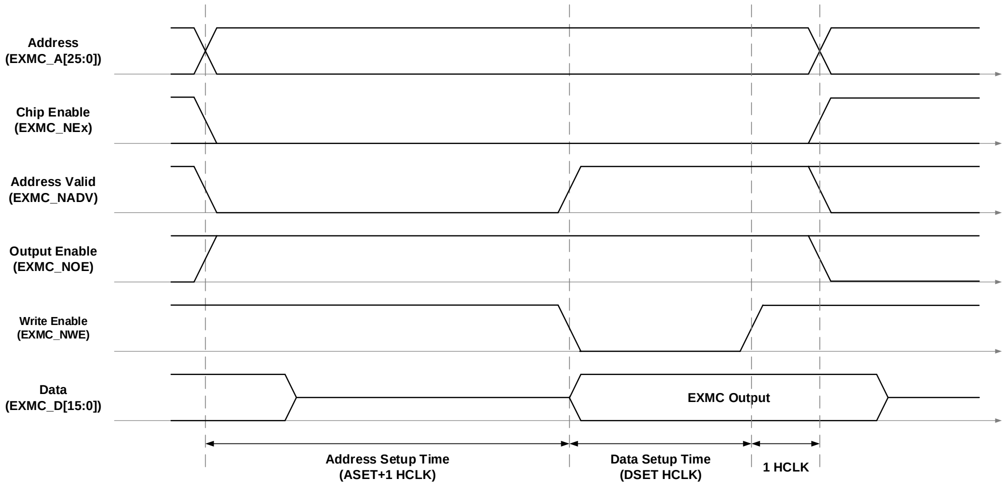


Figure 20-12. Mode B write access

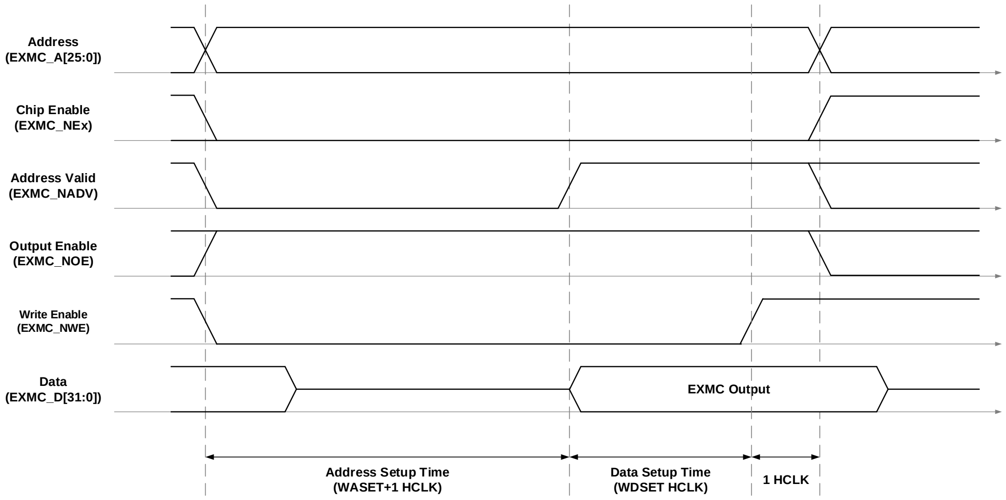


Table 20-8. Mode 2/B related registers configuration

EXMC_SNCTLx(Mode 2, Mode B)		
Bit Position	**Bit Name**	**Reference Setting Value**
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	Mode 2:0x0，Mode B:0x1
13	NRWTEN	0x0
12	WREN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1

5-4	NRW	Depends on memory
3-2	NRTP	0x2，NOR Flash
1	NRMUX	0x0
0	NRBKEN	0x1
EXMC_SNTCFGx(Read and write in mode 2,read in mode B)		
31-30	Reserved	0x0000
29-28	ASYNCMOD	Mode B:0x1
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	0x0
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx(Write in mode B)		
31-30	Reserved	0x0000
29-28	WASYNCMOD	Mode B:0x1
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	Reserved	0x000
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode C - NOR Flash OE toggling

Figure 20-13. Mode C read access

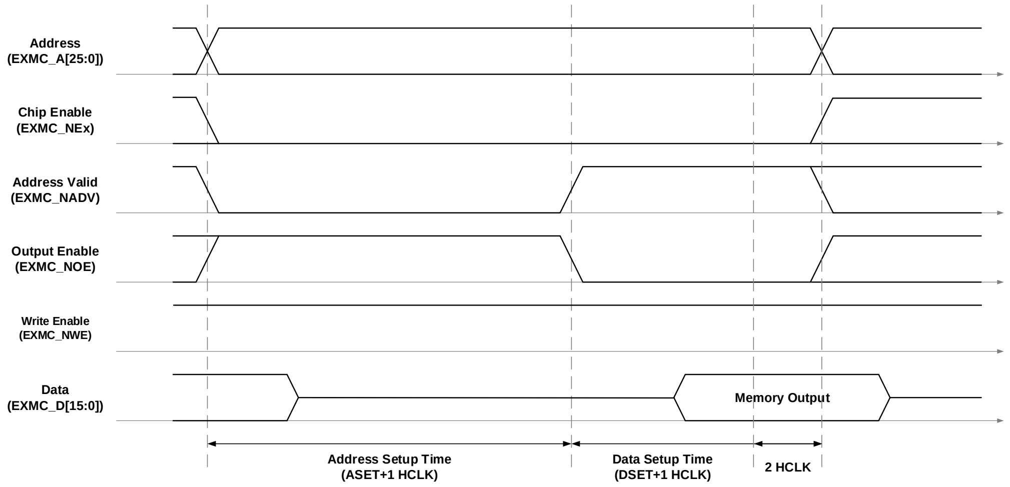
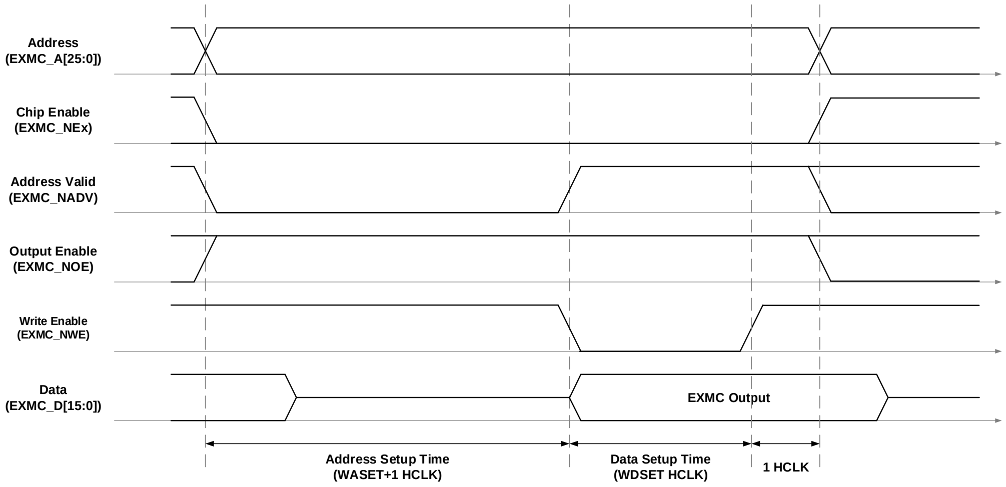


Figure 20-14. Mode C write access



The different between mode C and mode 1 write timing is that read/write timing is specified by the same set of timing configuration, while mode C write timing configuration is independent of its read configuration.

Table 20-9. Mode C related registers configuration

EXMC_SNCTLx		
Bit Position	**Bit Name**	**Reference Setting Value**
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WREN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2，NOR Flash
1	NRMUX	0x0
0	NRBKEN	0x1
EXMC_SNTCFGx		
31-30	Reserved	0x0000
29-28	ASYNCMOD	Mode C:0x2
27-24	DLAT	No effect
23-20	CKDIV	No effect

19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	0x0
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode C:0x2
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	Reserved	0x0
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode D - Asynchronous access with extended address

Figure 20-15. Mode D read access

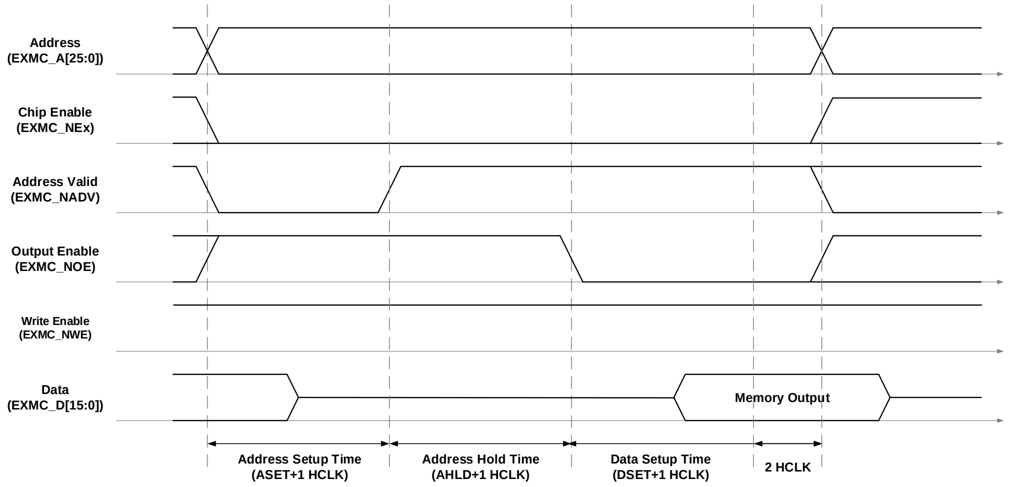


Figure 20-16. Mode D write access

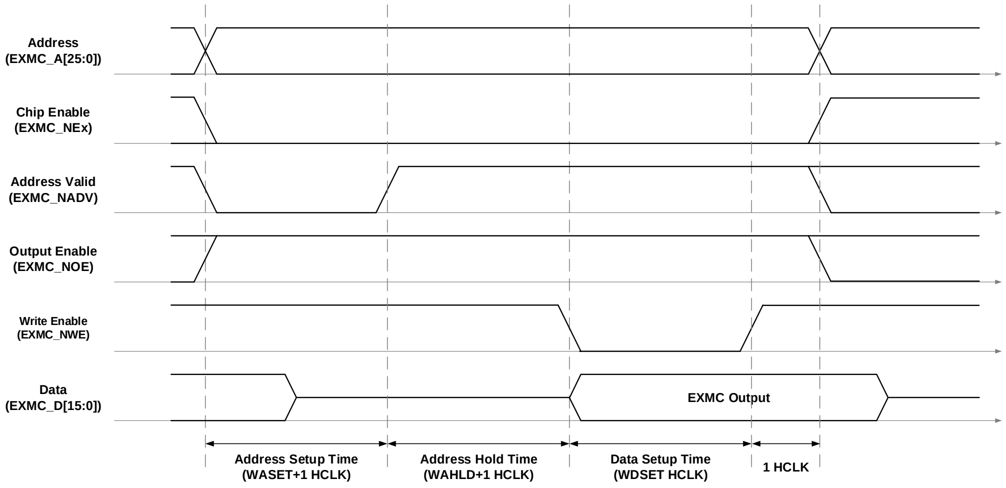


Table 20-10. Mode D related registers configuration

EXMC_SNCTLx		
Bit Position	**Bit Name**	**Reference Setting Value**
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WREN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	Depends on memory
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory
1	NRMUX	0x0
0	NRBKEN	0x1
EXMC_SNTCFGx		
31-30	Reserved	0x0
29-28	ASYNCMOD	Mode D:0x3
27-24	DLAT	Don’t care
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)

7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode D:0x3
27-24	DLAT	Don't care
23-20	CKDIV	No effect
19-16	Reserved	0x0
15-8	WDSET	Depends on memory and user
7-4	WAHLD	Depends on memory and user
3-0	WASET	Depends on memory and user

Mode AM - NOR Flash address / data bus multiplexing

Figure 20-17. Multiplex mode read access

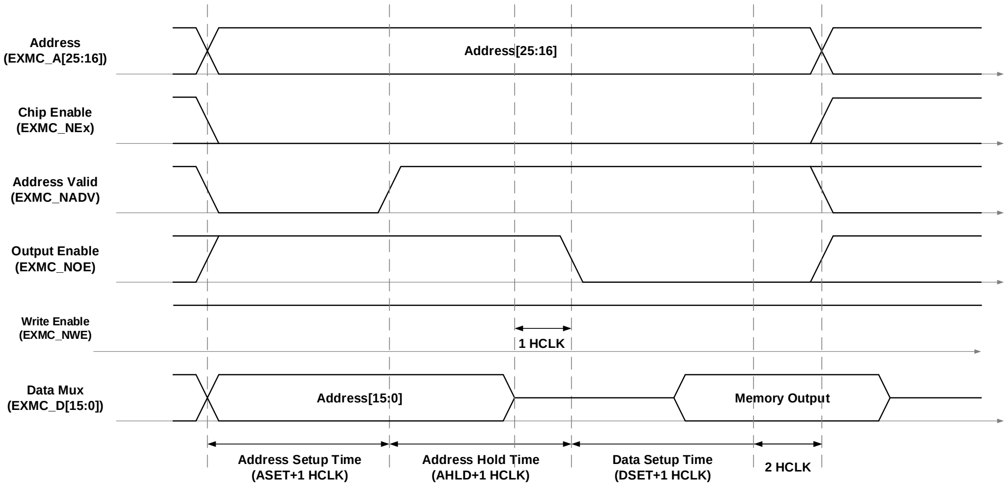


Figure 20-18. Multiplex mode write access

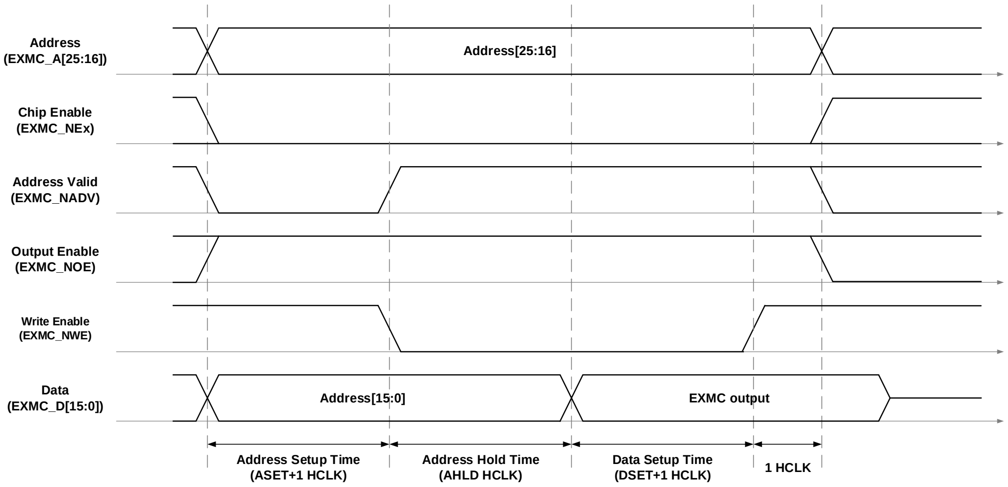


Table 20-11. Multiplex mode related registers configuration

EXMC_SNCTLx		
Bit Position	**Bit Name**	**Reference Setting Value**

31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WREN	Depends on memory
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2:NOR Flash
1	NRMUX	0x1
0	NRBKEN	0x1
EXMC_SNTCFGx		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Minimum time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user
7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user

Wait timing of asynchronous communication

Wait feature is controlled by the bit ASYNCWAIT in register EXMC_SNCTLx. During extern memory access, data setup phase will be automatically extended by the active EXMC_NWAIT signal if ASYNCWAIT bit is set. The extend time is calculated as follows:

If memory wait signal is aligned to EXMC_NOE/ EXMC_NWE:

$$T_{DATA\_SETUP} \geq maxT_{WAIT\_ASSERTION} + 4HCLK$$

If memory wait signal is aligned to EXMC_NE:

If $maxT_{WAIT\_ASSERTION} \geq T_{ADDRES\_PHASE} + T_{HOLD\_PHASE}$

$$T_{DATA\_SETUP} \geq (maxT_{WAIT\_ASSERTION} - T_{ADDRES\_PHASE} - T_{HOLD\_PHASE}) + 4HCLK$$

Otherwise

$$T_{DATA\_SETUP} \geq 4HCLK$$

Figure 20-19. Read access timing diagram under async-wait signal assertion

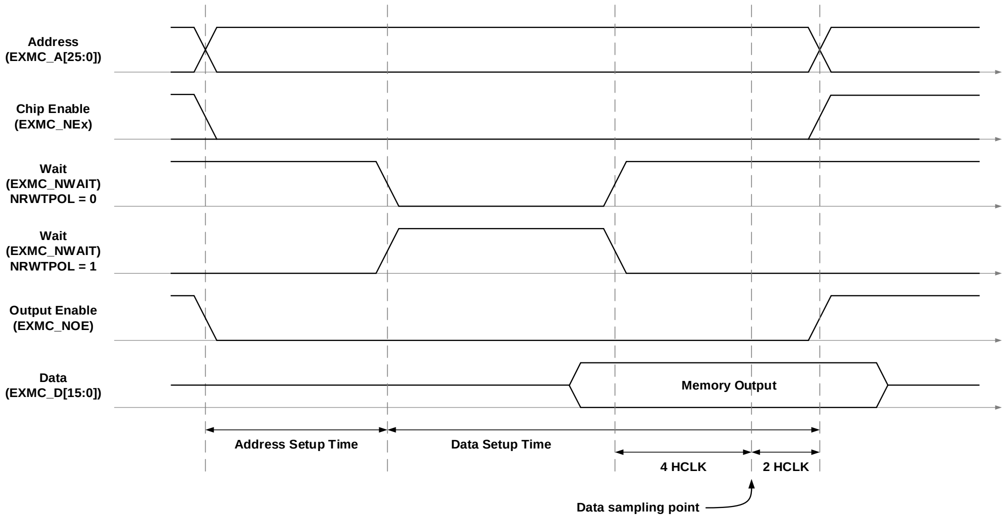
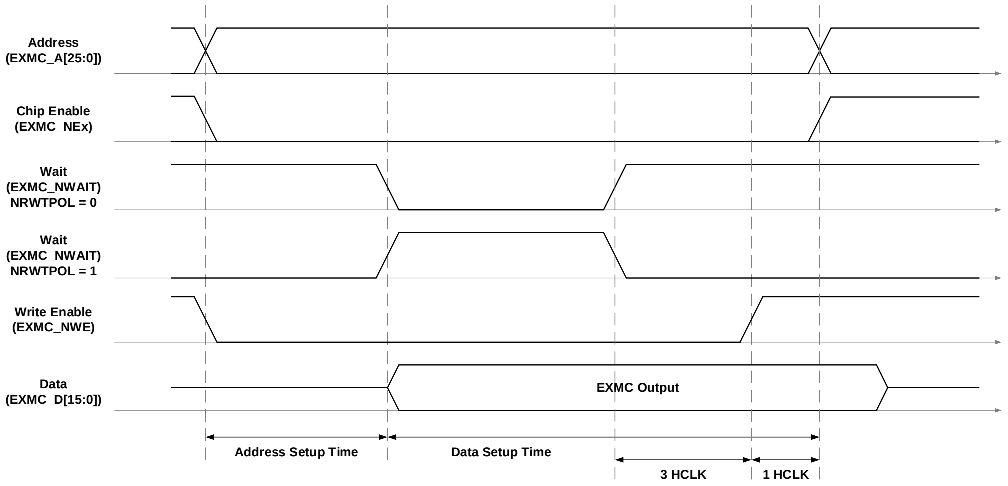


Figure 20-20. Write access timing diagram under async-wait signal assertion



Synchronous access timing diagram

The relations between memory clock (EXMC_CLK) and system clock (HCLK) clock are as follows:

$$\mathrm{EXMC\_CLK} = \frac{HCLK}{CKDIV + 1}$$

CKDIV is the synchronous clock divider ratio, it is configured through the CKDIV control field in the EXMC_SNTCFGx register.

1. Data latency and NOR Flash latency

Data latency is the number of EXMC_CLK cycles to wait before sampling the data. The relationship between data latency and NOR Flash specification's latency parameter is as follows:

For NOR Flash's specification excluding the EXMC_NADV cycle, their relationship should be:

$$\text{NOR Flash latency} = \text{DLAT} + 2$$

For NOR Flash's specification including the EXMC_NADV cycle, their relationship should be:

$$\text{NOR Flash latency} = \text{DLAT} + 3$$

2. Data wait

Users should guarantee that EXMC_NWAIT signal's behavior matches that of the external device. This signal's feature is configured through the EXMC_SNCTLx registers, it is enabled by the NRWTEN bit, and the active timing could be one data cycle before the wait state or active during the active state by the configuration NRWTCFG bit, while the wait signal's polarity is set by the NRWTPOL bit.

In NOR Flash synchronous burst access mode, when NRWTEN bit in EXMC_SNCTLx register is set, EXMC_NWAIT signal will be detected after a period of data latency. If EXMC_NWAIT signal detected as valid, wait cycles will be inserted until EXMC_NWAIT becomes invalid.

- The valid polarity of EXMC_NWAIT:

NRWTPOL= 1: valid level of EXMC_NWAIT signal is high.

NRWTPOL= 0: valid level of EXMC_NWAIT signal is low.

- In synchronous burst mode, EXMC_NWAIT signal has two kinds of configurations:

NRWTCFG = 1: When EXMC_NWAIT signal is active, current cycle data is not valid.

NRWTCFG = 0: When EXMC_NWAIT signal is active, the next cycle data is not valid. It is the default state after reset.

During wait-state inserted via the EXMC_NWAIT signal, the controller continues to send clock pulses to the memory, keep the chip select and output signals availably, and ignore the invalid data signal.

3. Mode SM - Single burst transmission

For synchronous burst transmission, if the needed data of AHB is 16-bit, EXMC will perform a burst transmission whose length is 1. If the needed data of AHB is 32-bit, EXMC will make the transmission divided into two 16-bit transmissions, that is, EXMC performs a burst transmission whose length is 2.

For other configurations please refers to ***Table 20-3. EXMC bank 0 supports all transactions***.

Synchronous mux burst read timing - NOR, PSRAM (CRAM)

Figure 20-21. Synchronous mux burst read timing

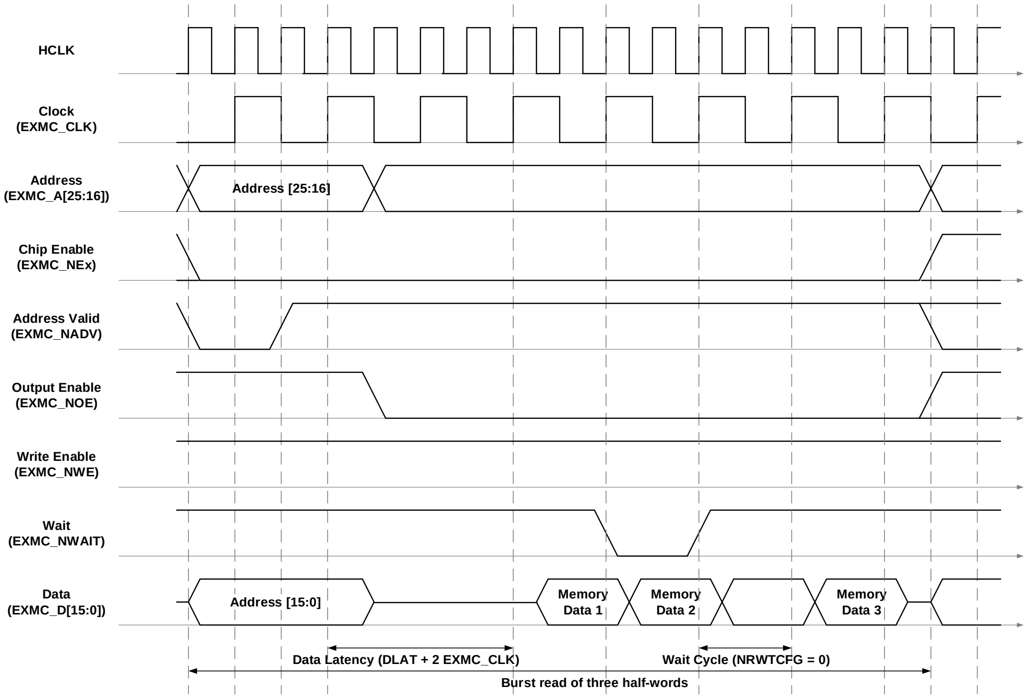


Table 20-12. Timing configurations of synchronous multiplexed read mode

EXMC_SNCTLx		
Bit Position	Bit Name	Reference Setting Value
31-20	Reserved	0x000
19	SYNCWR	No effect
18-16	Reserved	0x0
15	ASYNCWTEN	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WREN	No effect
11	NRWTCFG	Depends on memory
10	WRAPEN	0x0
9	NRWTPOL	Depends on memory
8	SBRSTEN	0x1，burst read enable
7	Reserved	0x1
6	NREN	Depends on memory
5-4	NRW	0x1
3-2	NRTP	Depends on memory，0x1/0x2
1	NRMUX	0x1, Depends on memory and users
0	NRBKEN	0x1
EXMC_SNTCFGx(Read)		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	Data latency

23-20	CKDIV	The figure above：0x1,EXMC_CLK=2HCLK
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	No effect
7-4	AHLD	No effect
3-0	ASET	No effect

Mode SM –Synchronous mux burst write timing – PSRAM (CRAM)

Figure 20-22. Synchronous mux burst write timing

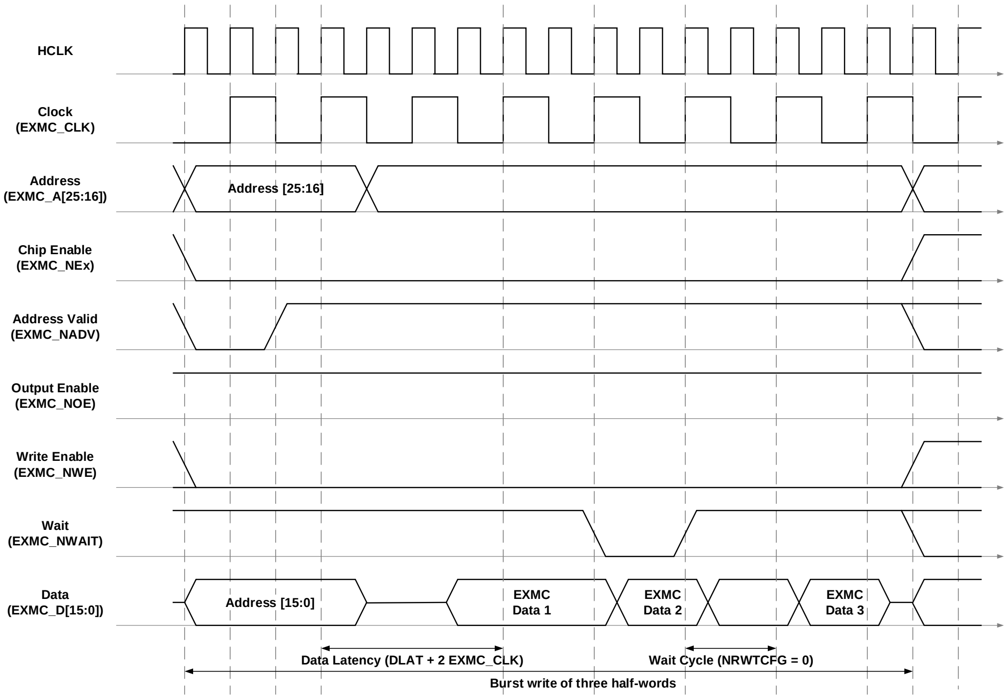


Table 20-13. Timing configurations of synchronous multiplexed write mode

EXMC_SNCTLx		
Bit Position	**Bit Name**	**Reference Setting Value**
31-20	Reserved	0x000
19	SYNCWR	0x1, synchronous write enable
18-16	Reserved	0x0
15	AYSNCWAIT	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WREN	0x1
11	NRWTCFG	0x0(Here must be zero)
10	WRAPEN	0x0
9	NTWTPOL	Depends on memory
8	SBRSTEN	No effect
7	Reserved	0x1
6	NREN	Depends on memory

5-4	NRW	0x1
3-2	NRTP	0x1
1	NRMUX	0x1, Depends on users
0	NRBKEN	0x1
EXMC_SNTCFGx(Write)		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	Data latency
23-20	CKDIV	The figure above: 0x1,EXMC_CLK=2HCLK
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	No effect
7-4	AHLD	No effect
3-0	ASET	No effect

20.3.5. NAND Flash or PC Card controller

EXMC has partitioned Bank1 and Bank2 as NAND Flash access field, bank3 as PC Card access field. Each bank has its own set of control register for access timing configuration. 8- and 16-bit NAND Flash and 16-bit PC Card are supported. An ECC hardware is provided for the NAND Flash controller to ensure the robustness of data transfer and storage.

NAND Flash or PC Card interface function

Table 20-14. 8-bit or 16-bit NAND interface signal

EXMC Pin	Direction	Functional description
EXMC_A[17]	Output	NAND Flash address latch（ALE）
EXMC_A[16]	Output	NAND Flash command latch（CLE）
EXMC_D[7:0]/ EXMC_D[15:0]	Input /Output	8-bit multiplexed, bidirectional address/data bus 16-bit multiplexed, bidirectional address/data bus
EXMC_NCE[x]	Output	Chip select, x = 1, 2
EXMC_NOE(NRE)	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT/ EXMC_INT[x]	Input	NAND Flash ready/busy input signal to the EXMC, x=1, 2

Table 20-15. 16-bit PC Card interface signal

EXMC Pin	Direction	Functional description
EXMC_A[10:0]	Output	Address bus of PC Card
EXMC_NIOS16	Input	Only for 16-bit I/O space data transmission width (Must be shorted to GND)
EXMC_NIORD	Output	I/O space read enable

EXMC Pin	Direction	Functional description
EXMC_NIOWR	Output	I/O space write enable
EXMC_NREG	Output	Register signal indicating if access is in Common space or Attribute space
EXMC_D[15:0]	Input /Output	Bidirectional data bus
EXMC_NCE3_x	Output	Chip select(x=0,1)
EXMC_NOE	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT	Input	PC Card wait input signal to the EXMC
EXMC_INTR	Input	PC Card interrupt input signal
EXMC_CD	Input	PC Card presence detection. Active high.

Supported memory access mode

Table 20-16. Bank1/2/3 of EXMC support the memory and access mode

Memory	Mode	R/W	AHB transaction size	Comments
8-bit NAND	Async	R	8	
	Async	W	8	
	Async	R	16	Automatically split into 2 EXMC accesses
	Async	W	16	
	Async	R	32	Automatically split into 4 EXMC accesses
	Async	W	32	
16-bit NAND/PC Card	Async	R	8	
	Async	W	8	Not support this operation
	Async	R	16	
	Async	W	16	
	Async	R	32	Automatically split into 2 EXMC accesses
	Async	W	32	

NAND Flash or PC Card controller timing

EXMC can generate the appropriate signal timing for NAND Flash, PC Cards and other devices. Each bank has a corresponding register to manage and control the external memory, such as EXMC_NPCTLx, EXMC_NPINTENx, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 and EXMC_NECCx. Among these registers, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 registers contain four timing parameters individually which are configured according to user specification and features of the external memory.

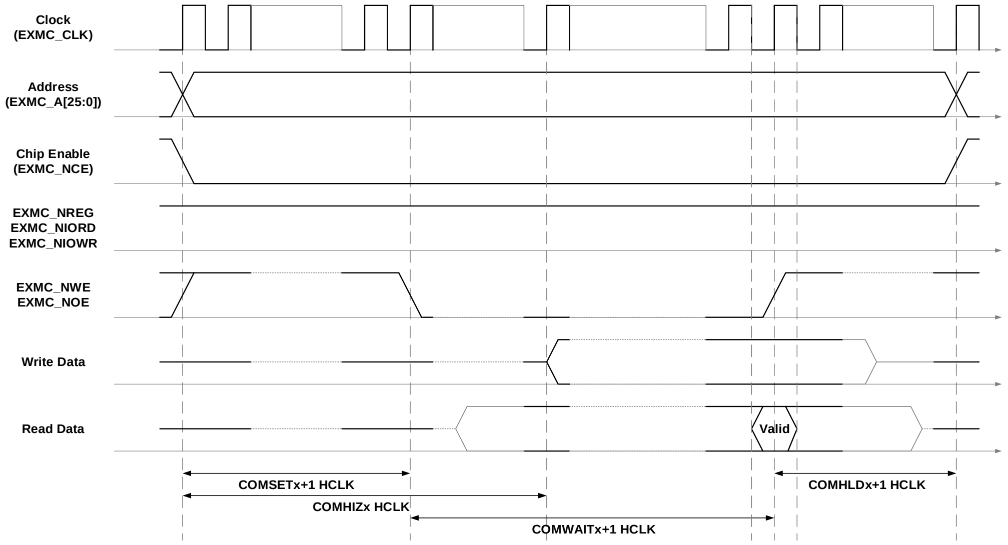
Table 20-17. NAND Flash or PC Card programmable parameters

Programmable parameter	W/R	Unit	Functional description	NAND Flash/ PC Card	
				Min	Max

High impedance time of the memory data bus （HIZ）	W/R	HCLK	Time to keep the data bus high impedance after starting write operation	0	255
Memory hold time（HLD）	W/R	HCLK	The number of HCLK clock cycles to keep address valid after sending the command. In write mode, it is also data hold time.	1	255
Memory wait time（WAIT）	W/R	HCLK	Minimum duration of sending command	1	256
Memory setup time（SET）	W/R	HCLK	The number of HCLK clock cycles to build address before sending command	1	256

The figure below shows the programmable parameters which are defined in the common memory space operations. The programmable parameters of Attribute memory space or I/O memory space (only for PC Card) are defined as well.

Figure 20-23. Access timing of common memory space of PC Card Controller



NAND Flash operation

When EXMC sends command or address to NAND Flash, it needs to use the command latch signal (A [16]) or address latch signal (EXMC_A [17]), namely, the CPU needs to perform write operation in particular address.

Example: NAND Flash read operation steps:

1. Configure EXMC_NPCTLx and EXMC_NPCTCFGx register. When pre-waiting is needed, EXMC_NPATCFGx has to be configured.
2. Send the command of NAND Flash read operation to the common space. Namely, during the valid period of EXMC_NCE and EXMC_NWE, when EXMC_CLE (EXMC_A [16])

becomes valid (high level), data on the I/O pins is regarded as a command by NAND Flash.

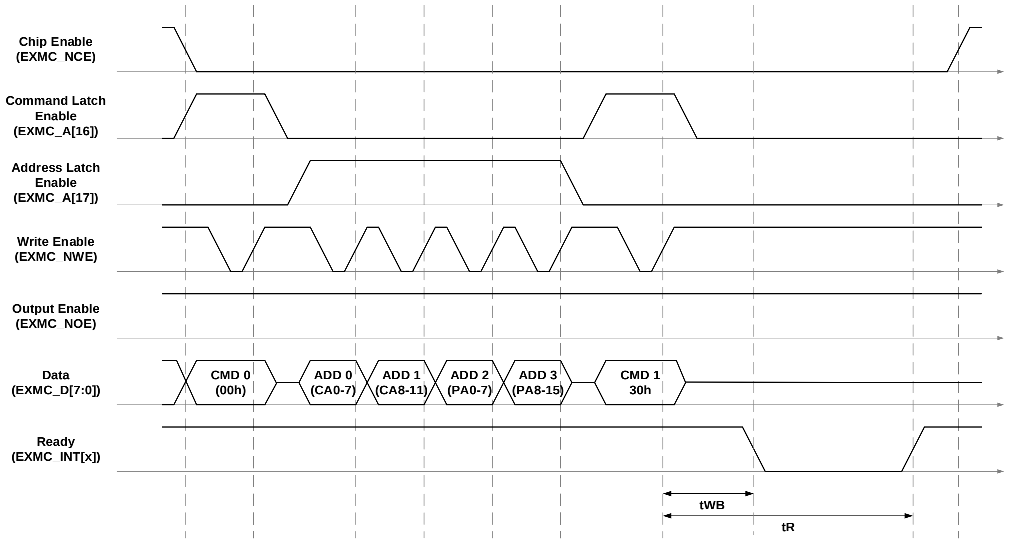
3. Send the start address of read operation to the common space. During the valid period of EXMC_NCE and EXMC_NWE, when EXMC_ALE (EXMC_A [17]) becomes valid (high level), the data on the I/O pins is regarded as an address by NAND Flash.
4. Waiting for NAND ready signal. In this period, NAND controller will maintain EXMC_NCE valid.
5. Read data byte by byte from the data area of the common space.
6. If new commands or address haven’t been written, data of the next page can be read out automatically. You can also read the data of the next page by going to step 3 and then writing a new address or writing a new command and address in step 2.

NAND Flash pre-wait functionality

Some NAND Flash requires that the controller should wait for NAND Flash to be busy after the first command byte following the address bytes are sent, and some EXMC_NCE-sensitive NAND Flash also requires that the EXMC_NCE must remain valid before it is ready.

Taking TOSHIBA128 M x 8 bit NAND Flash as an example:

Figure 20-24. Access to none "NCE don’t care" NAND Flash



1. Write CMD0 into NAND Flash bank common space command area.
2. Write ADD0 into NAND Flash bank common space address area.
3. Write ADD1 into NAND Flash bank common space address area.
4. Write ADD2 into NAND Flash bank common space address area.
5. Write ADD3 into NAND Flash bank common space address area.
6. Write CMD1 into NAND Flash bank attribute space command area.

In step 6, EXMC uses the operation timing defined in EXMC_NPATCFGx register. After a period of ATTHLD, NAND Flash waits for EXMC_INTx signal to be busy, and the time period of ATTHLD should be greater than tWB (tWB is defined as the time from EXMC_NWE high to EXMC_INTx low). For NCE-sensitive NAND Flash, after the first command byte following address bytes has been entered, EXMC_NCE must remain low until EXMC_INTx goes from low to high. The ATTHLD value of attribute space can be set in EXMC_NPATCFGx register to meet the timing requirements of tWB. CPU can use the attribute space timing when writing the first command byte following address bytes to the NAND Flash device. In other times, the CPU must use the common space timing.

NAND Flash ECC calculation module

An ECC calculation hardware is implemented in bank1 and bank2 respectively. Users can choose page size according to the ECCSZ control field in the EXMC_NPCTLx register. ECC offers one bit error correction and two bits errors detection.

When NAND memory block is enabled, ECC module will detect EXMC_D[15:0], EXMC_NCE and EXMC_NWE signals. When a data size of ECCSZ has been read or written, software must read the calculated ECC in theEXMC_NECCx register. When a recalculation of ECC is needed, software must clear the EXMC_NECCx register value by resetting ECCEN bit of EXMC_NPCTLx register to zero, and then restart ECC calculation by setting the ECCEN bit of EXMC_NPCTLx to one.

PC/CF Card access

EXMC Bank3 is used exclusively for PC/CF Card, both memory and IO mode access are supported. This bank is divided further into three sub spaces, memory, attribute and IO space.

EXMC_NCE3_0 and EXMC_NCE3_1 are the byte select signals, when only EXMC_NCE3_0 is active (Low), the lower byte or upper byte is selected depending on the EXMC_A[0], while only EXMC_NCE3_1 is active (Low), the upper byte is selected which is not supported, when both of these signals are active, 16-bit operation is performed. When NDTP is reset to select PC/CF Card as external memory device, NDW must be set to 01 in EXMC_NPCTLx register to guarantee correct EXMC operation.

EXMC PC/CF card access behavior for different spaces:

1. Common space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NWE and EXMC_NOE dictates whether the on-going operation is a write or read operation, and EXMC_NREG is high during common space access.

2. Attribute space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NWE and EXMC_NOE dictates whether the on-going operation is a write or read operation, and EXMC_NREG is low during attribute space access.

3. IO space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NIOWR and EXMC_NIORD dictate whether the on-going operation is a write or read operation, and EXMC_NREG is low during IO space access.

AHB access on 16-bit PC/CF card:

1. Common space: It is usually where data are stored, it could be accessible either in byte or in half-word mode, and odd address access is not supported in byte mode. When AHB word access is selected, EXMC automatically splits it into 2 consecutive half-word access. EXMC_NREG is high when common memory is targeted. EXMC_NOE and EXMC_NWE are the read and write enable signal for this type of access.
2. Attribute space: It is usually where configuration information are stored, for byte AHB access, only even address is possible. Half-word access converts into a single byte access automatically, and word access is converted into two consecutive byte access where only the even bytes are operational. In both half-word and word access, only EXMC_NCE3_0 will be active. EXMC_NREG is low when attribute memory is targeted. EXMC_NOE and EXMC_NWE are the read and write enable signal for this type of access.
3. IO space: Both byte and half-word AHB access are supported, in IO space memory access, EXMC_NIORD and EXMC_NIOWR act as the read and write enable signal respectively.

20.4. Register definition

20.4.1. NOR/PSRAM controller registers

SRAM/NOR Flash control registers (EXMC_SNCTLx) (x=0, 1, 2, 3)

Address offset: 0x00 + 8 * x, (x = 0, 1, 2, and 3)
Reset value: 0x0000 30DB for region0, and 0x0000 30D2 for region1, region2, and region3.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												SYNC WR	Reserved		
												rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WAIT	EXMO DEN	NRWT EN	WREN	NRWT CFG	WRAPEN	NRWT POL	SBR STEN	Reserved	NR EN	NRW[1:0]		NRTP[1:0]		NR MUX	NRBK EN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw		rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19	SYNCWR	Synchronous write 0: Asynchronous write 1: Synchronous write
18:16	Reserved	Must be kept at reset value.
15	ASYNCWAIT	Asynchronous wait 0: Disable the asynchronous wait feature 1: Enable the asynchronous wait feature
14	EXMODEN	Extended mode enable 0: Disable extended mode 1: Enable extended mode
13	NRWTEN	NWAIT signal enable For Flash memory access in burst mode, this bit enables/disables wait-state insertion via the NWAIT signal: 0: Disable NWAI signal 1: Enable NWAIT signal
12	WREN	Write enable 0: Disabled write in the bank by the EXMC, otherwise an AHB error is reported

		1: Enabled write in the bank by the EXMC (default after reset)
11	NRWTCFG	NWAIT signal configuration, only work in synchronous mode 0: NWAIT signal is active one data cycle before wait state 1: NWAIT signal is active during wait state
10	WRAPEN	Wrapped burst mode enable 0: Disable wrap burst mode support 1: Enable wrap burst mode support
9	NRWTPOL	NWAIT signal polarity 0: Low level is active of NWAIT 1: High level is active of NWAIT
8	SBRSTEN	Synchronous burst enable 0: Disable burst access mode 1: Enable burst access mode
7	Reserved	Must be kept at reset value.
6	NREN	NOR Flash access enable 0: Disable NOR Flash access 1: Enable NOR Flash access
5:4	NRW[1:0]	NOR region memory data bus width 00: 8 bits 01: 16 bits(default after reset) 10/11: Reserved
3:2	NRTP[1:0]	NOR region memory type 00: SRAM(default after reset for region1-region3) 01: PSRAM（CRAM） 10: NOR Flash(default after reset for region0) 11: Reserved
1	NRMUX	NOR region memory address/data multiplexing 0: Disable address/data multiplexing function 1: Enable address/data multiplexing function
0	NRBKEN	NOR region enable 0: Disable the corresponding memory bank 1: Enable the corresponding memory bank

SRAM/NOR Flash timing configuration registers (EXMC_SNTCFGx) (x=0, 1, 2, 3)

Address offset: 0x04 + 8 * x, (x = 0, 1, 2, and 3)
Reset value: 0x0FFF FFFF

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		ASYNCMOD[1:0]		DLAT[3:0]				CKDIV[3:0]				BUSLAT[3:0]			
		rw		rw				rw				rw			

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

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:28	ASYNCMOD[1:0]	Asynchronous access mode The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access 10: Mode C access 11: Mode D access
27:24	DLAT[3:0]	Data latency for NOR Flash. Only valid in synchronous access 0x0: Data latency of first burst access is 2 EXMC_CLK 0x1: Data latency of first burst access is 3 EXMC_CLK …… 0xF: Data latency of first burst access is 17 EXMC_CLK
23:20	CKDIV[3:0]	Synchronous clock divide ratio. This filed is only effect in synchronous mode. 0x0: Reserved 0x1: EXMC_CLK period = 2 * HCLK period …… 0xF: EXMC_CLK period = 16 * HCLK period
19:16	BUSLAT[3:0]	Bus latency The bits are defined in multiplexed read mode in order to avoid bus contention, and represent the data bus to return to a high impedance state's minimum. 0x0: Bus latency = 1 * HCLK period 0x1: Bus latency = 2 * HCLK period …… 0xF: Bus latency = 16 * HCLK period
15:8	DSET[7:0]	Data setup time This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 2 * HCLK period ……

		0xFF: Data setup time = 256 * HCLK period
7:4	AHLD[3:0]	Address hold time This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode. 0x0: Reserved 0x1: Address hold time = 2 * HCLK …… 0xF: Address hold time = 16 * HCLK
3:0	ASET[3:0]	Address setup time This field is used to set the time of address setup phase. **Note:** meaningful only in asynchronous access of SRAM,ROM,NOR Flash 0x0: Address setup time = 1 * HCLK …… 0xF: Address setup time = 16 * HCLK

SRAM/NOR Flash write timing configuration registers (EXMC_SNWTCFGx) (x=0, 1, 2, 3)

Address offset: 0x104 + 8 * x, (X = 0, 1, 2, and 3)
Reset value: 0x0FFF FFFF

This register has to be accessed by word(32-bit)

This register is meaningful only when the EXMODEN bit in EXMC_SNCTLx is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		WASYNCMOD[1:0]		DLAT[3:0]				CKDIV[3:0]				Reserved			
		rw		rw				rw							

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

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:28	WASYNCMOD[1:0]	Asynchronous access mode The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access 10: Mode C access 11: Mode D access
27:24	DLAT[3:0]	Data latency for NOR Flash. Only valid in synchronous access

		0x0: Data latency of first burst access is 2 EXMC_CLK 0x1: Data latency of first burst access is 3 EXMC_CLK …… 0xF: Data latency of first burst access is 17 EXMC_CLK
23:20	CKDIV[3:0]	Synchronous clock divide ratio. This filed is only effect in synchronous mode. 0x0: Reserved 0x1: EXMC_CLK period = 2 * HCLK period …… 0xF: EXMC_CLK period = 16 * HCLK period
19:16	Reserved	Must be kept at reset value.
15:8	WDSET[7:0]	Data setup time This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 2 * HCLK period …… 0xFF: Data setup time = 256 * HCLK period
7:4	WAHLD[3:0]	Address hold time This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode. 0x0: Reserved 0x1: Address hold time = 2 * HCLK …… 0xF: Address hold time = 16 * HCLK
3:0	WASET[3:0]	Address setup time This field is used to set the time of address setup phase. **Note:** Meaningful only in asynchronous access of SRAM,ROM,NOR Flash 0x0: Address setup time = 1 * HCLK 0x1: Address setup time = 2 * HCLK …… 0xF: Address setup time = 16 * HCLK

20.4.2. NAND Flash/PC Card controller registers

NAND Flash/PC Card control registers (EXMC_NPCTLx) (x=1, 2, 3)

Address offset: 0x40 + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0x0000 0018

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved	ECCSZ[2:0]	ATR[3]
	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATR[2:0]			CTR[3:0]				Reserved		ECCEN	NDW[1:0]		NDTP	NDBKEN	NDWTEN	Reserved
rw			rw						rw	rw		rw	rw	rw	

Bits	Fields	Description
31:20	Reserved	Must be kept at reset value.
19:17	ECCSZ[2:0]	ECC size 000: 256 bytes 001: 512 bytes 010: 1024 bytes 011: 2048 bytes 100: 4096 bytes 101: 8192 bytes
16:13	ATR[3:0]	ALE to RE delay 0x0: ALE to RE delay = 1 * HCLK …… 0xF: ALE to RE delay = 16 * HCLK
12:9	CTR[3:0]	CLE to RE delay 0x0: CLE to RE delay = 1 * HCLK 0x1: CLE to RE delay = 2 * HCLK …… 0xF: CLE to RE delay = 16 * HCLK
8:7	Reserved	Must be kept at reset value.
6	ECCEN	ECC enable 0: Disable ECC, and reset EXMC_NECCx 1: Enable ECC
5:4	NDW[1:0]	NAND bank memory data bus width 00: 8 bits 01: 16 bits Others: Reserved **Note:** for PC/CF card, 16-bit bus width must be selected.
3	NDTP	NAND bank memory type 0: PC Card, CF card, PCMCIA 1: NAND Flash
2	NDBKEN	NAND bank enable 0: Disable corresponding memory bank

		1: Enable corresponding memory bank
1	NDWTEN	Wait feature enable 0: Disable wait feature 1: Enable wait feature
0	Reserved	Must be kept at reset value.

NAND Flash/PC Card interrupt enable registers (EXMC_NPINTENx) (x=1, 2, 3)

Address offset: 0x44 + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0x0000 0040

This register has to be accessed by word(32-bit)

In addition to interrupt controlling bits, this register also contains a FIFO empty status bit, design specifically for ECC purpose. When external memory write is performed, the FIFO can hold up to 2 word from AHB access, freeing the bus temporarily for other peripherals. ECC calculation is based on the data passing through the FIFO, for correct ECC, users should read the ECC register only after the FIFO empty status flag is raised.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									FFEPT	INTFEN	INTHEN	INTREN	INTFS	INTHS	INTRS
									r	rw	rw	rw	rw	rw	rw

Bits	Fields	Description
31:7	Reserved	Must be kept at reset value.
6	FFEPT	FIFO empty flag 0: FIFO is not empty. 1: FIFO is empty.
5	INTFEN	Interrupt falling edge detection enable 0: Disable interrupt falling edge detection 1: Enable interrupt falling edge detection
4	INTHEN	Interrupt high-level detection enable 0: Disable interrupt high-level detection 1: Enable interrupt high-level detection
3	INTREN	Interrupt rising edge detection enable bit 0: Disable interrupt rising edge detection 1: Enable interrupt rising edge detection

2	INTFS	Interrupt falling edge status 0: Not detect interrupt falling edge 1: Detect interrupt falling edge
1	INTHS	Interrupt high-level status 0: Not detect interrupt high-level 1: Detect interrupt high-level
0	INTRS	Interrupt rising edge status 0: Not detect interrupt rising edge 1: Detect interrupt rising edge

NAND Flash/PC Card common space timing configuration registers (EXMC_NPCTCFGx) (x=1, 2, 3)

Address offset: 0x48 + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0xFCFC FCFC

This register has to be accessed by word(32-bit)

These operations applicable to common memory space for 16-bit PC Card, CF card and NAND Flash.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMHIZ[7:0]								COMHLD[7:0]							
rw								rw							

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

Bits	Fields	Description
31:24	COMHIZ[7:0]	Common memory data bus HiZ time The bits are defined as time of bus keep high impedance state after writing the data. 0x00: COMHIZ = 1 * HCLK …… 0xFE: COMHIZ = 255 * HCLK 0xFF: COMHIZ = 256 * HCLK
23:16	COMHLD[7:0]	Common memory hold time After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time. 0x00: Reserved 0x01: COMHLD = 1 * HCLK ……

		0xFE: COMHLD = 254 * HCLK 0xFF: COMHLD = 255 * HCLK
15:8	COMWAIT[7:0]	Common memory wait time Define the minimum time to maintain command 0x00: Reserved 0x01: COMWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFE: COMWAIT = 255 * HCLK (+NWAIT active cycles) 0xFF: COMWAIT = 256 * HCLK (+NWAIT active cycles)
7:0	COMSET[7:0]	Common memory setup time Define the time to build address before sending command 0x00: COMSET = 1 * HCLK …… 0xFE: COMSET = 255 * HCLK 0xFF: COMSET = 256 * HCLK

NAND Flash/PC Card attribute space timing configuration registers (EXMC_NPATCFGx) (x=1, 2, 3)

Address offset: 0x4C + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0xFCFC FCFC

This register has to be accessed by word(32-bit)

It is used for 8-bit accesses to the attribute memory space of the PC Card or to access the NAND Flash for the last address or command write access if another timing must be applied.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATTHIZ[7:0]								ATTHLD[7:0]							
rw								rw							

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

Bits	Fields	Description
31:24	ATTHIZ[7:0]	Attribute memory data bus HiZ time The bits are defined as time of bus keep high impedance state after writing the data. 0x00: ATTHIZ = 1 * HCLK …… 0xFE: ATTHIZ = 255 * HCLK 0xFF: ATTHIZ = 256 * HCLK
23:16	ATTHLD[7:0]	Attribute memory hold time

Bits	Fields	Description
		After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time. 0x00: Reserved 0x01: ATTHLD = 1 * HCLK …… 0xFE: ATTHLD = 254 * HCLK 0xFF: ATTHLD = 255 * HCLK
15:8	ATTWAIT[7:0]	Attribute memory wait time Define the minimum time to maintain command 0x00: Reserved 0x01: ATTWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFE: ATTWAIT = 255 * HCLK (+NWAIT active cycles) 0xFF: ATTWAIT = 256 * HCLK (+NWAIT active cycles)
7:0	ATTSET[7:0]	Attribute memory setup time Define the time to build address before sending command 0x00: ATTSET = 1 * HCLK …… 0xFE: ATTSET = 255 * HCLK 0xFF: ATTSET = 256 * HCLK

PC Card I/O space timing configuration register (EXMC_PIOTCFG3)

Address offset: 0xB0

Reset value: 0xFCFC FCFC

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IOHIZ[7:0]								IOHLD[7:0]							
rw								rw							

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

Bits	Fields	Description
31:24	IOHIZ[7:0]	IO space data bus HiZ time The bits are defined as time of bus keep high impedance state after writing the data. 0x00: IOHIZ = 0 *HCLK …… 0xFF: IOHIZ = 255 *HCLK

23:16	IOHLD[7:0]	IO space hold time After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time. 0x00: Reserved 0x01: IOHLD = 1 * HCLK …… 0xFF: IOHLD = 255 * HCLK
15:8	IOWAIT[7:0]	IO space wait time Define the minimum time to maintain command 0x00: Reserved 0x01: IOWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFF: IOWAIT = 256 * HCLK (+NWAIT active cycles)
7:0	IOSET[7:0]	IO space setup time Define the time to build address before sending command 0x00: IOSET = 1 * HCLK …… 0xFF: IOSET = 256 * HCLK

NAND Flash ECC registers (EXMC_NECCx) (x=1, 2)

Address offset: 0x54+0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC[31:16]															
r															

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

Bits	Fields	Description
31:0	ECC[31:0]	ECC result

ECCSZ[2:0]	NAND Flash page size	ECC bits
0b000	256	ECC[21:0]
0b001	512	ECC[23:0]
0b010	1024	ECC[25:0]
0b011	2048	ECC[27:0]
0b100	4096	ECC[29:0]

0b101	8192	ECC[31:0]

21. Controller area network (CAN)

21.1. Overview

CAN bus (for Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer.

The Basic Extended CAN, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. The CAN interface handles the transmission and the reception of CAN frames fully autonomously. The CAN provides 14 scalable/configurable identifier filter banks in GD32F10x XD/HD/MD and 28 scalable/configurable identifier filter banks in GD32F10x CL. The filters are used for selecting the incoming messages the software needs and discarding the others. Three transmit mailboxes are provided to the software for setting up messages. The transmission scheduler decides which mailbox has to be transmitted first. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware. Two receive FIFOs are used by hardware to store the incoming messages. The CAN controller also provides all hardware functions for supporting the time-triggered communication option for safety-critical applications.

21.2. Characteristics

- Supports CAN protocols version 2.0A, B
- Baud rates up to 1 Mbit/s
- Supports the time-triggered communication
- Interrupt enable and clear

Transmission

- Supports 3 transmit mailboxes
- Prioritization of messages
- Supports Time Stamp at SOF transmission

Reception

- Supports 2 receive FIFOs and each has 3 messages deep
- 14 scalable/configurable identifier filter banks in GDF10x XD/HD/MD
- 28 scalable/configurable identifier filter banks in GDF10x CL
- FIFO lock

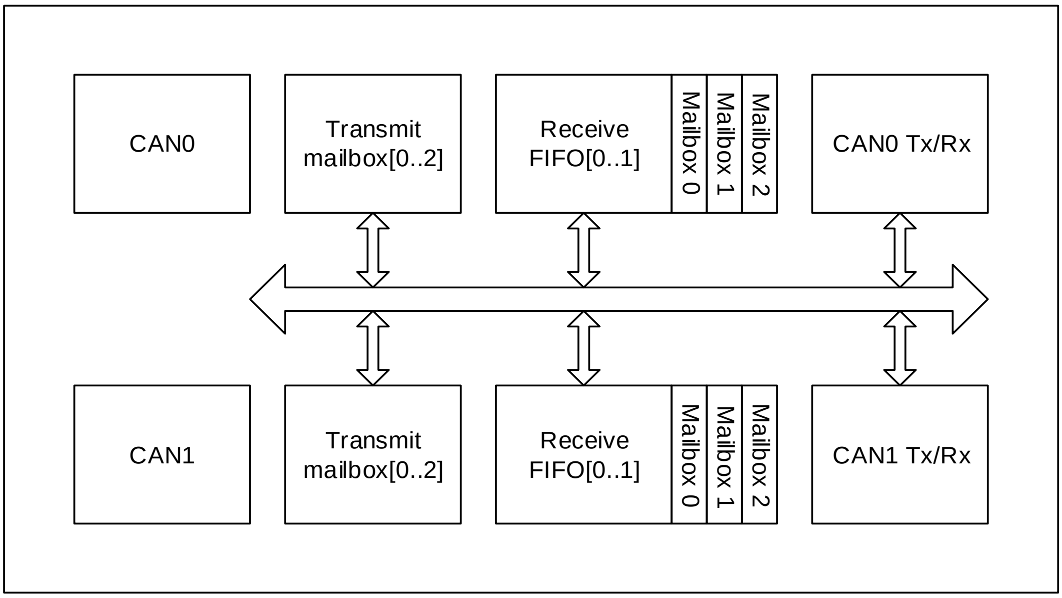
Time-triggered communication

- Disable retransmission automatically
- 16-bit free timer
- Time Stamp on SOF reception
- Time Stamp sent in last two data bytes

21.3. Function overview

Figure 21-1. CAN module block diagram shows the CAN block diagram.

Figure 21-1. CAN module block diagram



21.3.1. Working mode

The CAN interface has three working modes:

- Sleep working mode.
- Initial working mode.
- Normal working mode.

Sleep working mode

Sleep working mode is the default mode after reset. In sleep working mode, the CAN is in the low-power status while the CAN clock is stopped.

When SLPWMOD bit in CAN_CTL register is set, the CAN enters the sleep working mode. Then the SLPWS bit in CAN_STAT register is set.

To leave sleep working mode automatically: the AWU bit in CAN_CTL register is set and the CAN bus activity is detected. To leave sleep working mode by software: clear the SLPWMOD bit in CAN_CTL register.

Sleep working mode to Initial working mode: Set IWMOD bit and clear SLPWMOD bit in CAN_CTL register.

Sleep working mode to Normal working mode: Clear IWMOD and SLPWMOD bit in CAN_CTL register.

Initial working mode

When the options of CAN bus communication is needed to be changed, the CAN must enter initial working mode.

When IWMOD bit in CAN_CTL register is set, the CAN enters the initial working mode. Then the IWS bit in CAN_STAT register is set.

Initial working mode to sleep working mode: Set SLPWMOD bit and clear IWMOD bit in CAN_CTL register.

Initial working mode to Normal working mode: Clear IWMOD bit and clear SLPWMOD bit in CAN_CTL register.

Normal working mode

The CAN can enter normal working mode and to communicate with other CAN communication nodes.

To enter normal working mode: clear IWMOD and SLPWMOD bit in CAN_ CTL register.

Normal working mode to sleep working mode: Set SLPWMOD bit in CAN_CTL register and wait the current transmission or reception completed.

Normal working mode to Initial working mode: Set IWMOD bit in CAN_CTL register, and wait the current transmission or reception completed.

21.3.2. Communication modes

The CAN interface has four communication modes:

- Silent communication mode.
- Loopback communication mode.
- Loopback and silent communication mode.
- Normal communication mode.

Silent communication mode

Silent communication mode means reception available and transmission disable.

The RX pin of the CAN can get the signal from the network and the TX pin always holds logical one.

When the SCMOD bit in CAN_BT register is set, the CAN enters the silent communication mode. When it is cleared, the CAN leaves silent communication mode.

Silent communication mode is useful on monitoring the network messages.

Loopback communication mode

Loopback communication mode means the sending messages are transferred into the reception FIFOs, the RX pin is disconnected from the CAN network and the TX pin can send messages to the CAN network.

Set LCMOD bit in CAN_BT register to enter loopback communication mode or clear it to leave. Loopback communication mode is useful on self-test.

Loopback and silent communication mode

Loopback and silent communication mode means the RX and TX pins are disconnected from the CAN network while the sending messages are transferred into the reception FIFOs.

Set LCMOD and SCMOD bit in CAN_BT register to enter loopback and silent communication mode or clear them to leave.

Loopback and silent communication mode is useful on self-test. The TX pin holds logical one. The RX pin holds high impedance state.

Normal communication mode

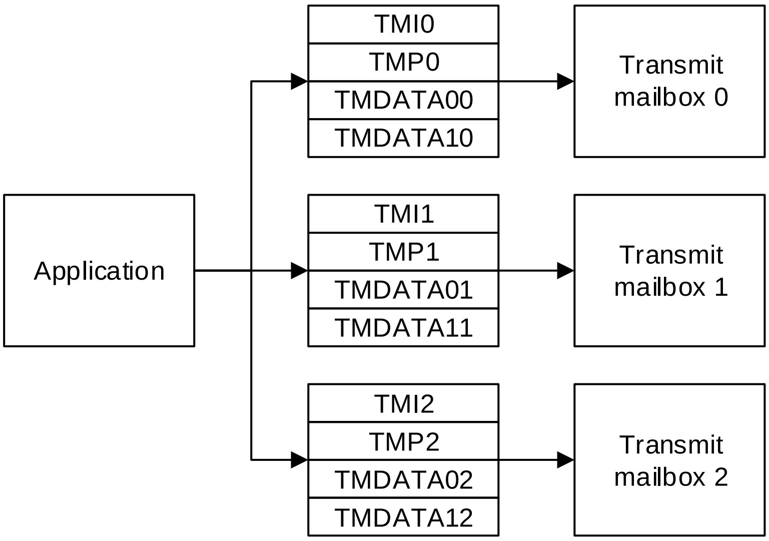
Normal communication mode is the default communication mode unless the LCMOD or SCMOD bit in CAN_BT register is set.

21.3.3. Data transmission

Transmission register

Three transmit mailboxes are transparent to the application. You can use transmit mailboxes through four transmission registers: CAN_TMIx, CAN_TMPx, CAN_TMDATA0x and CAN_TMDATA1x. As shown in ***Figure 21-2. Transmission register***.

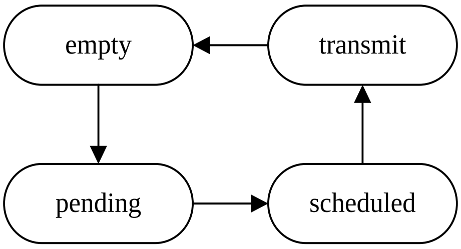
Figure 21-2. Transmission register



Transmit mailbox state

A transmit mailbox can be used when it is free: empty state. If the data is filled in the mailbox, setting TEN bit in CAN_TMIx register to prepare for starting the transmission: pending state. If more than one mailbox is in the pending state, they need schedule the transmission: scheduled state. A mailbox with priority enter transmit state and start transmitting the message. After the message has been sent, the mailbox is free: empty state. As shown in *Figure 21-3. State of transmission mailbox*.

Figure 21-3. State of transmission mailbox



Transmit status and error

The CAN_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmits finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmits finished and no error. MTFNERR is set when the frame in the transmission mailbox has been sent without any error.
- MAL: mailbox arbitration lost. MAL is set while the frame transmission is failed because of the arbitration lost.
- MTE: mailbox transmits error. MTE is set while the frame transmission is failed because

of the detection error of CAN bus.

Steps of sending a frame

To send a frame through the CAN:
Step 1: Select one free transmit mailbox.
Step 2: Fill four transmission registers with the application's acquirement.
Step 3: Set TE bit in CAN_TMIx register.
Step 4: Check the transmit status. Typically, MTF and MTFNERR are set if transmission is successful.

Transmission options

Abort

MST bit in CAN_TSTAT register can abort the transmission.

If the transmission mailbox's state is pending or scheduled, the abort of transmission can be done immediately.

In the state of transmit, the abort of transmission does not take effect immediately until the transmission is finished. In case of transmission successful, the MTFNERR and MTF in CAN_TSTAT are set and state changes to empty. In case of transmission failed, the state changes to be scheduled and then the abort of transmission can be done immediately.

Priority

When more than one transmit mailbox is pending, the transmission order is given by the TFO bit in CAN_CTL register.

In case TFO is 1, the three transmit mailboxes work as FIFO.

In case TFO is 0, the transmit mailbox with lowest identifier has the highest priority of transmission. If the identifiers are equal, the lower mailbox number will be scheduled first.

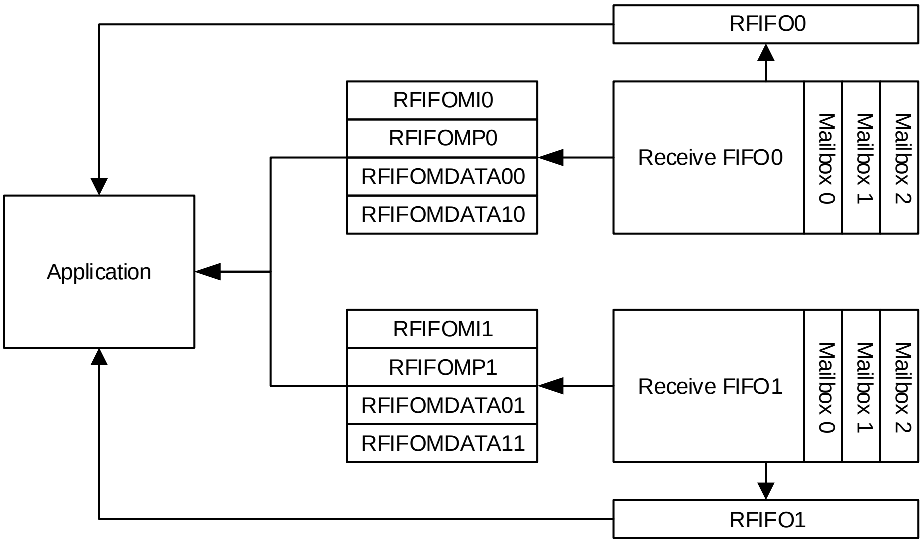
21.3.4. Data reception

Reception register

Two receive FIFOs are transparent to the application. You can use receive FIFOs through five registers: CAN_RFIFOx, CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x. FIFO's status and operation can be handled by CAN_RFIFOx register. Reception frame data can be achieved through the registers: CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x.

Each FIFO consists of three receive mailboxes. As shown in ***Figure 21-4. Reception register***.

Figure 21-4. Reception register



Receive FIFO

Receive FIFO has three mailboxes. The reception frames are stored in the mailbox ordered by the arriving sequence of the frames. First arrived frame can be accessed by application firstly.

The number of frames in the receive FIFO and the status can be accessed by the register CAN_RFIFO0 and CAN_RFIFO1.

If at least one frame has been stored in the receive FIFO0. The frame data is placed in the registers (CAN_RFIFOMI0, CAN_RFIFOMP0, CAN_RFIFOMDATA00, CAN_RFIFOMDATA10). After read the current frame, set RFD bit in CAN_RFIFO0 to release a frame in the receive FIFO and the software can read the next frame.

Receive FIFO status

RFL bit in CAN_RFIFOx register: receive FIFO length. It is 0 when no frame is stored in the reception FIFO and 3 when FIFOx is full.

RFF bit in CAN_RFIFOx register: the FIFO holds three frames. It indicates FIFOx is full.

RFO bit in CAN_RFIFOx register: one new frame arrived while the FIFO has hold three frames. It indicates FIFOx is overfull. If the RFOD bit in CAN_CTL register is set, the new frame is discarded. If the RFOD bit in CAN_CTL register is reset, the new frame is stored into the receive FIFO and the last frame in the receive FIFO is discarded.

Steps of receiving a message

Step 1: Check the number of frames in the receive FIFO.

Step 2: Reading CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_

RFIFOMDATA1x if there is data pending.

Step 3: Set the RFD bit in CAN_RFIFOx register.

21.3.5. Filtering function

The CAN would receive frames from the CAN bus. If the frame is passed through the filter, it is stored into the receive FIFOs. Otherwise, the frame will be discarded without intervention by the software.

The identifier of frame from the CAN bus takes part in the matching of the filter.

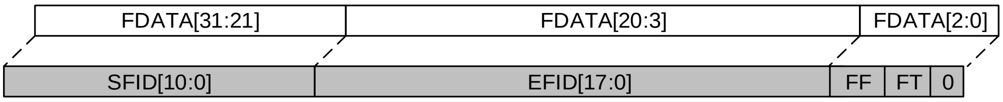
Scale

In GD32F10x XD/HD/MD , the filter consists of 14 banks: bank0 to bank13. In GD32F10x CL, the filter consists of 28 banks: bank0 to bank27. Each bank has two 32-bit registers: CAN_FxDATA0 and CAN_FxDATA1.

Each filter bank can be configured to 32-bit or 16-bit.

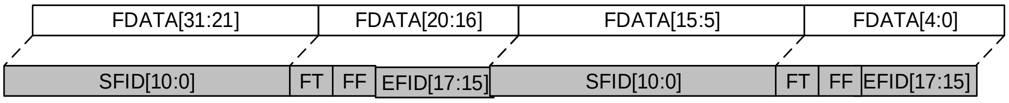
32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As shown in ***Figure 21-5. 32-bit filter***.

Figure 21-5. 32-bit filter



16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As shown in ***Figure 21-6. 16-bit filter***.

Figure 21-6. 16-bit filter



Mask mode

In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” (when the bit in mask register is ‘1’) or as “don’t care” (when the bit in mask register is ‘0’). 32-bit mask mode example is shown in ***Figure 21-7. 32-bit mask mode filter***.

Figure 21-7. 32-bit mask mode filter

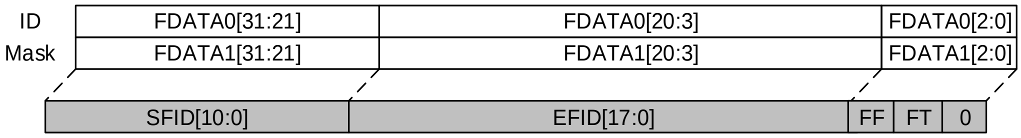
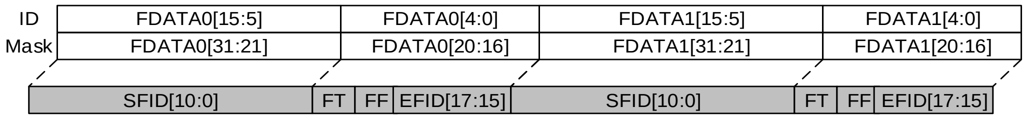


Figure 21-8. 16-bit mask mode filter



List mode

The filter consists of frame identifiers. The filter can decide whether a frame will be discarded or not. When one frame arrived, the filter will check which member can match the identifier of the frame.

32-bit list mode example is shown in ***Figure 21-9. 32-bit list mode filter***.

Figure 21-9. 32-bit list mode filter

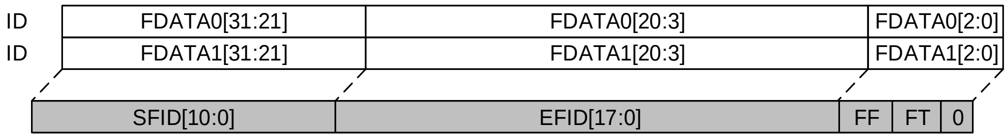
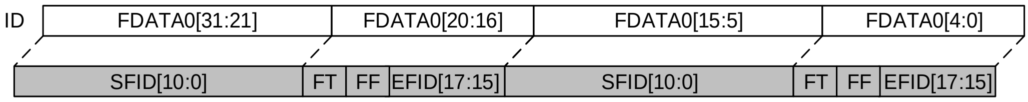


Figure 21-10. 16-bit list mode filter



Filter number

Each filter within a filter bank is numbered from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

For example, there are two filter banks. Bank 0 is configured as 32-bit mask mode. Bank 1 is configured as 32-bit list mode. The filter number is shown in ***Table 21-1. 32-bit filter number***.

Table 21-1. 32-bit filter number

Filter Bank	Filter Data Register	Filter Number
0	F0DATA0-32bit-ID	0
	F0DATA1-32bit-Mask	
1	F1DATA0-32bit-ID	1
	F1DATA1-32bit-ID	2

Associated FIFO

28 banks can be associated with FIFO0 or FIFO1. If the bank is associated with FIFO0, the frames passed through the bank will fill the FIFO0.

Active

The filter bank needs to be configured activation if the application wants the bank working and while filters not used by the application should be left deactivated.

Filtering index

Each filter number corresponds to a filtering rule. When the frame from the CAN bus passes the filters, a filter number must associate with the frame. The filter number is called filtering index. It stores in the FI bits in CAN_RFIFOMPx when the frame is read by the application.

Each FIFO numbers the filters within the banks associated with the FIFO itself whether the bank is active or not.

The example about filtering index is shown in ***Table 21-2. Filtering index***.

Table 21-2. Filtering index

Filter Bank	FIFO0	Active	Filter Nunber	Filter Bank	FIFO1	Active	Filter Nunber
0	F0DATA0-32bit-ID	Yes	0	2	F2DATA0[15:0]-16bit-ID	Yes	0
	F0DATA1-32bit-Mask				F2DATA0[31:16]-16bit-Mask		
1	F1DATA0-32bit-ID	Yes	1		F2DATA1[15:0]-16bit-ID		1
	F1DATA1-32bit-ID		2		F2DATA1[31:16]-16bit-Mask		
3	F3DATA0[15:0]-16bit-ID	No	3	4	F4DATA0-32bit-ID	No	2
	F3DATA0[31:16]-16bit-Mask				F4DATA1-32bit-Mask		
	F3DATA1[15:0]-16bit-ID		4	5	F5DATA0-32bit-ID	No	3
	F3DATA1[31:16]-16bit-Mask				F5DATA1-32bit-ID		4
7	F7DATA0[15:0]-16bit-ID	No	5	6	F6DATA0[15:0]-16bit-ID	Yes	5
	F7DATA0[31:16]-16bit-ID		6		F6DATA0[31:16]-16bit-ID		6
	F7DATA1[15:0]-16bit-ID		7		F6DATA1[15:0]-16bit-ID		7
	F7DATA1[31:16]-16bit-ID		8		F6DATA1[31:16]-16bit-ID		8
8	F8DATA0[15:0]-16bit-ID	Yes	9	10	F10DATA0[15:0]-16bit-ID	No	9
	F8DATA0[31:16]-16bit-ID		10		F10DATA0[31:16]-16bit-Mask		
	F8DATA1[15:0]-16bit-ID		11		F10DATA1[15:0]-16bit-ID		10
	F8DATA1[31:16]-16bit-ID		12		F10DATA1[31:16]-16bit-Mask		
9	F9DATA0[15:0]-16bit-ID	Yes	13	11	F11DATA0[15:0]-16bit-ID	No	11
	F9DATA0[31:16]-16bit-Mask				F11DATA0[31:16]-16bit-ID		12
	F9DATA1[15:0]-16bit-ID		14		F11DATA1[15:0]-16bit-ID		13
	F9DATA1[31:16]-16bit-Mask				F11DATA1[31:16]-16bit-ID		14
12	F12DATA0-32bit-ID	Yes	15	13	F13DATA0-32bit-ID	Yes	15

Filter Bank	FIFO0	Active	Filter Nunber	Filter Bank	FIFO1	Active	Filter Nunber
	F12DATA1-32bit-Mask				F13DATA1-32bit-ID		16

Priority

The filters have the priority:

1. 32-bit mode is higher than 16-bit mode.
2. List mode is higher than mask mode.
3. Smaller filter index value has the higher priority.

21.3.6. Time-triggered communication

The time-triggered CAN protocol is a higher layer protocol on top of the CAN data link layer. Time-triggered communication means that activities are triggered by the elapsing of time segments. In a time-triggered communication system all points of time of message transmission are defined during the development of a system. A time-triggered communication system is ideal for applications in which the data traffic is of a periodic nature.

In this mode, the 16-bit internal counter of the CAN hardware is activated and used to generate the time stamp value stored in the CAN_RFIFOMPx and CAN_TMPx registers for reception and transmission respectively. The internal counter is incremented each CAN bit time. The internal counter is captured on the sample point of the SOF (Start of Frame) bit in both reception and transmission.

The automatic retransmission is disabled in the time-triggered CAN communication.

21.3.7. Communication parameters

Nonautomatic retransmission mode

This mode has been implemented in order to fulfill the requirement of the time-triggered communication option of the CAN standard. To configure the hardware in this mode the ARD bit in the CAN_CTL register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the frame transmission.

At the end of the first transmission attempt, the hardware considers the request as finished and sets the MTF bit in the CAN_TSTAT register. The result of the transmission is indicated in the CAN_TSTAT register by the MTFNERR, MAL and MTE bits.

Bit time

On the bit-level the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also means that a sophisticated method of bit synchronization is required. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception of the start bit available with each character, a synchronous transmission protocol there is just one start bit available at the beginning of a frame. To ensure the receiver to correctly read the messages, continuous resynchronization is required. Phase buffer segments are therefore inserted before and after the nominal sample point within a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagation from sender to receiver and back to the sender must be completed within one bit-time. For synchronization purposes a further time segment, the propagation delay segment, is needed in addition to the time reserved for synchronization, the phase buffer segments. The propagation delay segment takes into account the signal propagation on the bus as well as signal delays caused by transmitting and receiving nodes.

The normal bit time simplified by the CAN from the CAN protocol has three segments as follows:

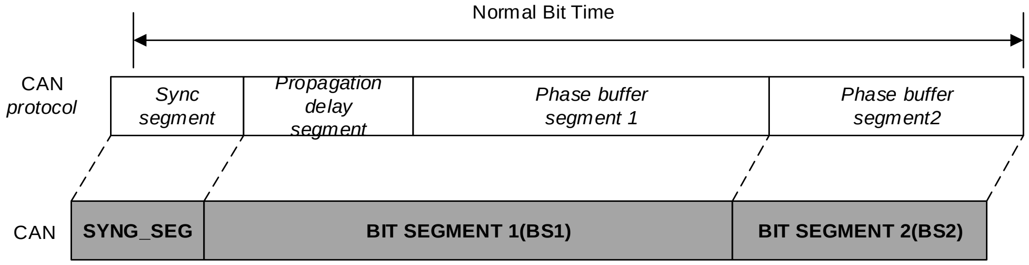
Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).

Bit segment 1 (BS1): defines the location of the sample point. It includes the *Propagation delay segment* and *Phase buffer segment 1* of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.

Bit segment 2 (BS2): defines the location of the transmit point. It represents the *Phase buffer segment 2* of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the ***Figure 21-11. The bit time***.

Figure 21-11. The bit time



The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

Baud rate

The CAN's clock derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$BaudRate = \frac{1}{Normal\ Bit\ Time} \tag{21-1}$$

$$Normal\ Bit\ Time = t_{SYNC\_SEG} + t_{BS1} + t_{BS2}$$

with:

$$t_{SYNC\_SEG} = 1 \times t_q$$

$$t_{BS1} = (1 + BS1) \times t_q$$

$$t_{BS2} = (1 + BS2) \times t_q$$

$$t_q = (1 + BAUDPSC) \times t_{PCLK1}$$

21.3.8. Error flags

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TECNT value, in CAN_ERR register) and a Receive Error Counter (RECNT value, in the CAN_ERR register), which get incremented or decremented according to the error condition. For detailed information about TECNT and RECNT management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network.

Furthermore, the CAN hardware provides detailed information on the current error status in CAN_ERR register. By means of the CAN_INTEN register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

Bus-Off recovery

The Bus-Off state is reached when TECNT is greater than 255. This state is indicated by BOERR bit in CAN_ERR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages.

Depending on the ABOR bit in the CAN_CTL register, CAN will recover from Bus-Off (becomes error active again) either automatically or on software request. But in both cases the CAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

If ABOR is set, the CAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOR is cleared, the software must initiate the recovering sequence by requesting CAN to enter and to leave initialization mode.

Note: If the Bus-off state cannot be recovery, the bus-off interrupt should be enabled and reinit in it.

21.3.9. CAN interrupts

Four interrupt vectors are dedicated to CAN. Each interrupt source can be independently enabled or disabled by setting or resetting related bits in CAN_INTEN.

The interrupt sources can be classified into:

- transmit interrupt
- FIFO0 interrupt
- FIFO1 interrupt
- error and status change interrupt

Transmit interrupt

The transmit interrupt can be generated by any of the following conditions and TMEIE bit in CAN_INTEN register will be set:

- TX mailbox 0 transmit finished: MTF0 bit in the CAN_TSTAT register is set.
- TX mailbox 1 transmit finished: MTF1 bit in the CAN_TSTAT register is set.
- TX mailbox 2 transmit finished: MTF2 bit in the CAN_TSTAT register is set.

Receive FIFO0 interrupt

The Receive FIFO0 interrupt can be generated by the following conditions:

- Reception FIFO0 not empty: RFL0 bits in the CAN_RFIFO0 register are not ‘00’ and RFNEIE0 in CAN_INTEN register is set.
- Reception FIFO0 full: RFF0 bit in the CAN_RFIFO0 register is set and RFFIE0 in CAN_INTEN register is set.
- Reception FIFO0 overrun: RFO0 bit in the CAN_RFIFO0 register is set and RFOIE0 in CAN_INTEN register is set.

Receive FIFO1 interrupt

The Receive FIFO1 interrupt can be generated by the following conditions:

- Reception FIFO1 not empty: RFL1 bits in the CAN_RFIFO1 register are not ‘00’ and RFNEIE1 in CAN_INTEN register is set.
- Reception FIFO1 full: RFF1 bit in the CAN_RFIFO1 register is set and RFFIE1 in CAN_INTEN register is set.
- Reception FIFO1 overrun: RFO1 bit in the CAN_RFIFO1 register is set and RFOIE1 in

CAN_INTEN register is set.

Error and working mode change interrupt

The error and working mode change interrupt can be generated by the following conditions:

- Error: ERRIF bit in the CAN_STAT register and ERRIE bit in the CAN_INTEN register are set. Refer to ERRIF description in the CAN_STAT register.
- Wakeup: WUIF bit in the CAN_STAT register is set and WIE bit in the CAN_INTEN register is set.
- Enter sleep working mode: SLPIF bit in the CAN_STAT register is set and SLPWIE bit in the CAN_INTEN register is set.

21.4. Register definition

21.4.1. Control register (CAN_CTL)

Address offset: 0x00

Reset value: 0x0001 0002

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															DFZ
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Reserved							TTC	ABOR	AWU	ARD	RFOD	TFO	SLPWMOD	IWMOD
rs								rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	DFZ	Debug freeze If the CANx_HOLD in DBG_CTL register is set, this bit define the CAN stop for debug or work normal. If the CANx_HOLD in DBG_CTL register is clear, this bit take not effect. 0: CAN reception and transmission working normal even during debug 1: CAN reception and transmission stop working during debug
15	SWRST	Software reset 0: No effect 1: Reset CAN with working mode of sleep. This bit is automatically reset to 0
14:8	Reserved	Must be kept at reset value
7	TTC	Time-triggered communication 0: Disable time-triggered communication 1: Enable time-triggered communication
6	ABOR	Automatic bus-off recovery 0: The bus-off state is left manually by software 1: The bus-off state is left automatically by hardware
5	AWU	Automatic wakeup If this bit is set, the sleep mode left when CAN bus activity detected, and SLPWMOD bit in CAN_CTL register will be cleared automatically. 0: The sleeping working mode is left manually by software 1: The sleeping working mode is left automatically by hardware

4	ARD	Automatic retransmission disable 0: Enable Automatic retransmission 1: Disable Automatic retransmission
3	RFOD	Receive FIFO overwrite disable 0: Enable receive FIFO overwrite when receive FIFO is full and overwrite the FIFO with the incoming frame 1: Disable receive FIFO overwrite when receive FIFO is full and discard the incoming frame
2	TFO	Transmit FIFO order 0: Order with the identifier of the frame 1: Order with first in and first out
1	SLPWMOD	Sleep working mode If this bit is set by software, the CAN enter sleep working mode after current transmission or reception complete. This bit can cleared by software or hardware. If AWU bit in CAN_CTL register is set, this bit is cleared by hardware when CAN bus activity detected. 0: Disable sleep working mode 1: Enable sleep working mode
0	IWMOD	Initial working mode 0: Disable initial working mode 1: Enable initial working mode

21.4.2. Status register (CAN_STAT)

Address offset: 0x04

Reset value: 0x0000 0C02

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RXL	LASTRX	RS	TS	Reserved			SLPIF	WUIF	ERRIF	SLPWS	IWS
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	RXL	RX level

10	LASTRX	Last sample value of RX pin
9	RS	Receiving state 0: CAN is not working in the receiving state 1: CAN is working in the receiving state
8	TS	Transmitting state 0: CAN is not working in the transmitting state 1: CAN is working in the transmitting state
7:5	Reserved	Must be kept at reset value
4	SLPIF	Status change interrupt flag of sleep working mode entering This bit is set by hardware when entering sleep working mode, and cleared by hardware when the CAN not in sleep working mode. This bit can also cleared by software when write 1 to this bit. 0: CAN is not entering the sleep working mode 1: CAN is entering the sleep working mode.
3	WUIF	Status change interrupt flag of wakeup from sleep working mode This bit is set when CAN bus activity detected on sleep working mode. This bit can cleared by software when write 1 to this bit. 0: Wakeup event is not coming 1: Wakeup event is coming
2	ERRIF	Error interrupt flag This bit is set by following event. The BOERR bit in CAN_ERR register is set and BOIE bit in CAN_INTEN register is set. Or the PERR bit in CAN_ERR register is set and PERRIE bit in CAN_INTEN register is set. Or the WERR bit in CAN_ERR register is set and WERRIE bit in CAN_INTEN register is set. Or the ERRN bits in CAN_ERR register are set to 1 to 6 (not 0 and not 7) and ERRNIE in CAN_INTEN register is set. This bit is cleared by software when write 1 to this bit. 0: No error interrupt flag 1: Any error interrupt flag has happened
1	SLPWS	Sleep working state This bit is set by hardware when the CAN enter sleep working mode after set SLPWMOD bit in CAN_CTL register. If the CAN leave from normal working mode to sleep working mode, it must wait the current frame transmission or reception completed. This bit is cleared by hardware when the CAN leave sleep working mode. Clear SLPWMOD bit in CAN_CTL register or automatically detect the CAN bus activity when AWU bit is set in CAN_CTL register. If leave sleep working mode to normal working mode, this bit will be cleared after receive 11 consecutive recessive bits from the CAN bus. 0: CAN is not the state of sleep working mode 1: CAN is the state of sleep working mode

0	IWS	Initial working state This bit is set by hardware when the CAN enter initial working mode after set IWMOD bit in CAN_CTL register. If the CAN leave from normal working mode to initial working mode, it must wait the current frame transmission or reception completed. This bit is cleared by hardware when the CAN leave initial working mode after clear IWMOD bit in CAN_CTL register. If leave initial working mode to normal working mode, this bit will be cleared after receive 11 consecutive recessive bits from the CAN bus. 0: CAN is not the state of initial working mode 1: CAN is the state of initial working mode

21.4.3. Transmit status register (CAN_TSTAT)

Address offset: 0x08

Reset value: 0x1C00 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMLS2	TMLS1	TMLS0	TME2	TME1	TME0	NUM[1:0]		MST2	Reserved			MTE2	MAL2	MTFNERR2	MTF2
r	r	r	r	r	r	r		rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MST1	Reserved			MTE1	MAL1	MTFNERR1	MTF1	MST0	Reserved			MTE0	MAL0	MTFNERR0	MTF0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31	TMLS2	Transmit mailbox 2 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 2 has the last sending order in the transmit FIFO with at least two frame are pending.
30	TMLS1	Transmit mailbox 1 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 1 has the last sending order in the transmit FIFO with at least two frame are pending.
29	TMLS0	Transmit mailbox 0 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 0 has the last sending order in the transmit FIFO with at least two frame are pending.
28	TME2	Transmit mailbox 2 empty 0: Transmit mailbox 2 not empty 1: Transmit mailbox 2 empty

27	TME1	Transmit mailbox 1 empty 0: Transmit mailbox 1 not empty 1: Transmit mailbox 1 empty
26	TME0	Transmit mailbox 0 empty 0: Transmit mailbox 0 not empty 1: Transmit mailbox 0 empty
25:24	NUM[1:0]	These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty. These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted lastly if all mailboxes are full.
23	MST2	Mailbox 2 stop transmitting This bit is set by the software to stop mailbox 2 transmitting. This bit is reset by the hardware while the mailbox 2 is empty.
22:20	Reserved	Must be kept at reset value
19	MTE2	Mailbox 2 transmit error This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
18	MAL2	Mailbox 2 arbitration lost This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
17	MTFNERR2	Mailbox 2 transmit finished and no error This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error. 0: Mailbox 2 transmit finished with error 1: Mailbox 2 transmit finished and no error
16	MTF2	Mailbox 2 transmit finished This bit set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI2 is 1. 0: Mailbox 2 transmit is progressing 1: Mailbox 2 transmit finished
15	MST1	Mailbox 1 stop transmitting This bit is set by the software to stop mailbox 1 transmitting. This bit is reset by the hardware while the mailbox 1 is empty.
14:12	Reserved	Must be kept at reset value
11	MTE1	Mailbox 1 transmit error

		This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
10	MAL1	Mailbox 1 arbitration lost This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
9	MTFNERR1	Mailbox 1 transmit finished and no error This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error. 0: Mailbox 1 transmit finished with error 1: Mailbox 1 transmit finished and no error
8	MTF1	Mailbox 1 transmit finished This bit is set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI1 is 1. 0: Mailbox 1 transmit is progressing 1: Mailbox 1 transmit finished
7	MST0	Mailbox 0 stop transmitting This bit is set by the software to stop mailbox 0 transmitting. This bit is reset by the hardware while the mailbox 0 is empty.
6:4	Reserved	Must be kept at reset value
3	MTE0	Mailbox 0 transmit error This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
2	MAL0	Mailbox 0 arbitration lost This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
1	MTFNERR0	Mailbox 0 transmit finished and no error This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error. 0: Mailbox 0 transmit finished with error 1: Mailbox 0 transmit finished and no error
0	MTF0	Mailbox 0 transmit finished This bit is set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI0 is 1.

0: Mailbox 0 transmit is progressing

1: Mailbox 0 transmit finished

21.4.4. Receive message FIFO0 register (CAN_RFIFO0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFD0	RFO0	RFF0	Reserved	RFL0[1:0]	
										rs	rc_w1	rc_w1		r	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	RFD0	Receive FIFO0 dequeue This bit is set by the software to start dequeuing a frame from receive FIFO0. This bit is reset by the hardware while the dequeuing is done.
4	RFO0	Receive FIFO0 overfull This bit is set by hardware when receive FIFO0 is overfull and reset by software when write 1 to this bit. 0: The receive FIFO0 is not overfull 1: The receive FIFO0 is overfull
3	RFF0	Receive FIFO0 full This bit is set by hardware when receive FIFO0 is full and reset by software when write 1 to this bit. 0: The receive FIFO0 is not full 1: The receive FIFO0 is full
2	Reserved	Must be kept at reset value
1:0	RFL0[1:0]	Receive FIFO0 length These bits are the length of the receive FIFO0.

21.4.5. Receive message FIFO1 register (CAN_RFIFO1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFD1	RFO1	RFF1	Reserved	RFL1[1:0]	
										rs	rc_w0	rc_w1		r	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	RFD1	Receive FIFO1 dequeue This bit is set by the software to start dequeuing a frame from receive FIFO1. This bit is reset by the hardware while the dequeuing is done.
4	RFO1	Receive FIFO1 overfull This bit is set by hardware when receive FIFO1 is overfull and reset by software when write 0 to this bit. 0: The receive FIFO1 is not overfull 1: The receive FIFO1 is overfull
3	RFF1	Receive FIFO1 full This bit is set by hardware when receive FIFO1 is full and reset by software when write 1 to this bit. 0: The receive FIFO1 is not full 1: The receive FIFO1 is full
2	Reserved	Must be kept at reset value
1:0	RFL1[1:0]	Receive FIFO1 length These bits are the length of the receive FIFO1.

21.4.6. Interrupt enable register (CAN_INTEN)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SLPWIE	WIE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			ERRNIE	BOIE	PERRIE	WERRIE	Reserved	RFOIE1	RFFIE1	RFNEIE1	RFOIE0	RFFIE0	RFNEIE0	TMEIE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	SLPWIE	Sleep working interrupt enable 0: Sleep working interrupt disable 1: Sleep working interrupt enable
16	WIE	Wakeup interrupt enable 0: Wakeup interrupt disable 1: Wakeup interrupt enable
15	ERRIE	Error interrupt enable 0: Error interrupt disable 1: Error interrupt enable
14:12	Reserved	Must be kept at reset value
11	ERRNIE	Error number interrupt enable 0: Error number interrupt disable 1: Error number interrupt enable
10	BOIE	Bus-off interrupt enable 0: Bus-off interrupt disable 1: Bus-off interrupt enable
9	PERRIE	Passive error interrupt enable 0: Passive error interrupt disable 1: Passive error interrupt enable
8	WERRIE	Warning error interrupt enable 0: Warning error interrupt disable 1: Warning error interrupt enable
7	Reserved	Must be kept at reset value
6	RFOIE1	Receive FIFO1 overfull interrupt enable 0: Receive FIFO1 overfull interrupt disable 1: Receive FIFO1 overfull interrupt enable
5	RFFIE1	Receive FIFO1 full interrupt enable 0: Receive FIFO1 full interrupt disable 1: Receive FIFO1 full interrupt enable
4	RFNEIE1	Receive FIFO1 not empty interrupt enable 0: Receive FIFO1 not empty interrupt disable 1: Receive FIFO1 not empty interrupt enable
3	RFOIE0	Receive FIFO0 overfull interrupt enable

		0: Receive FIFO0 overfull interrupt disable 1: Receive FIFO0 overfull interrupt enable
2	RFFIE0	Receive FIFO0 full interrupt enable 0: Receive FIFO0 full interrupt disable 1: Receive FIFO0 full interrupt enable
1	RFNEIE0	Receive FIFO0 not empty interrupt enable 0: Receive FIFO0 not empty interrupt disable 1: Receive FIFO0 not empty interrupt enable
0	TMEIE	Transmit mailbox empty interrupt enable 0: Transmit mailbox empty interrupt disable 1: Transmit mailbox empty interrupt enable

21.4.7. Error register (CAN_ERR)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RECNT[7:0]								TECNT[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ERRN[2:0]			Reserved	BOERR	PERR	WERR
									rw				r	r	r

Bits	Fields	Descriptions
31:24	RECNT[7:0]	Receive Error Count defined by the CAN standard
23:16	TECNT[7:0]	Transmit Error Count defined by the CAN standard
15:7	Reserved	Must be kept at reset value
6:4	ERRN[2:0]	Error number These bits indicate the error status of bit transformation. They are updated by the hardware. While the bit transformation is successful, they are equal to 0. Software can set these bits to 0b111. 000: No Error 001: Stuff Error 010: Form Error 011: Acknowledgment Error 100: Bit recessive Error 101: Bit dominant Error

		110: CRC Error 111: Set by software
3	Reserved	Must be kept at reset value
2	BOERR	Bus-off error Whenever the CAN enters bus-off state, the bit will be set by the hardware. The bus-off state is entered on TECNT overflow, greater than 255.
1	PERR	Passive error Whenever the TECNT or RECNT is greater than 127, the bit will be set by the hardware.
0	WERR	Warning error Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by the hardware.

21.4.8. Bit timing register (CAN_BT)

Address offset: 0x1C

Reset value: 0x0123 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCMOD	LCMOD	Reserved				SJW[1:0]		Reserved	BS2[2:0]			BS1[3:0]			
rw	rw					rw			rw			rw			

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

Bits	Fields	Descriptions
31	SCMOD	Silent communication mode 0: Silent communication disable 1: Silent communication enable
30	LCMOD	Loopback communication mode 0: Loopback communication disable 1: Loopback communication enable
29:26	Reserved	Must be kept at reset value
25:24	SJW[1:0]	Resynchronization jump width Resynchronization jump width time quantum= SJW[1:0]+1
23	Reserved	Must be kept at reset value

22:20	BS2[2:0]	Bit segment 2 Bit segment 2 time quantum=BS2[2:0]+1
19:16	BS1[3:0]	Bit segment 1 Bit segment 1 time quantum=BS1[3:0]+1
15:10	Reserved	Must be kept at reset value
9:0	BAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

21.4.9. Transmit mailbox identifier register (CAN_TMIx) (x=0..2)

Address offset: 0x180, 0x190, 0x1A0

Reset value: 0xXXXX XXXX (bit0=0)

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SFID[10:0]/EFID[28:18]											EFID[17:13]				
rw											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFID[12:0]													FF	FT	TEN
rw													rw	rw	rw

Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	TEN	Transmit enable This bit is set by the software when one frame will be transmitted and reset by the hardware when the transmit mailbox is empty.

0: Transmit disable

1: Transmit enable

21.4.10. Transmit mailbox property register (CAN_TMPx) (x=0..2)

Address offset: 0x184, 0x194, 0x1A4

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">TS[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">Reserved</td><td>TSEN</td><td colspan="4">Reserved</td><td colspan="4">DLENC[3:0]</td></tr>
<tr><td colspan="7"></td><td>rw</td><td colspan="4"></td><td colspan="4">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:9	Reserved	Must be kept at reset value
8	TSEN	Time stamp enable 0: Time stamp disable 1: Time stamp enable. The TS[15:0] will be transmitted in the DB6 and DB7 in DL This bit is available while the TTC bit in CAN_CTL is set.
7:4	Reserved	Must be kept at reset value
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame.

21.4.11. Transmit mailbox data0 register (CAN_TMDATA0x) (x=0..2)

Address offset: 0x188, 0x198, 0x1A8

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">DB3[7:0]</td><td colspan="8">DB2[7:0]</td></tr>
<tr><td colspan="8">rw</td><td colspan="8">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">DB1[7:0]</td><td colspan="8">DB0[7:0]</td></tr>
</table>

rw								rw							

Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

21.4.12. Transmit mailbox data1 register (CAN_TMDATA1x) (x=0..2)

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DB7[7:0]								DB6[7:0]							
rw								rw							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DB5[7:0]								DB4[7:0]							
rw								rw							

Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

21.4.13. Receive FIFO mailbox identifier register (CAN_RFIFOMIx) (x=0,1)

Address offset: 0x1B0, 0x1C0

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SFID[10:0]/EFID[28:18]											EFID[17:13]				
r											r				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EFID[12:0]													FF	FT	Reserved

r r r

Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	Reserved	Must be kept at reset value

21.4.14. Receive FIFO mailbox property register (CAN_RFIFOMPx) (x=0,1)

Address offset: 0x1B4, 0x1C4

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FI[7:0]								Reserved				DLENC[3:0]			
r												r			

Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:8	FI[7:0]	Filtering index The index of the filter by which the frame is passed.
7:4	Reserved	Must be kept at reset value
3:0	DLENC[3:0]	Data length code

DLENC[3:0] is the number of bytes in a frame.

21.4.15. Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x=0,1)

Address offset: 0x1B8, 0x1C8

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DB3[7:0]								DB2[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
r								r							

Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

21.4.16. Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x=0,1)

Address offset: 0x1BC, 0x1CC

Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DB7[7:0]								DB6[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB5[7:0]								DB4[7:0]							
r								r							

Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5

7:0	DB4[7:0]	Data byte 4

21.4.17. Filter control register (CAN_FCTL)

Address offset: 0x200

Reset value: 0x2A1C 0E01

This register has to be accessed by word(32-bit)

The filter control register with GD32F10x XD/HD/MD:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FLD
															rw

The filter control register with GD32F10x CL:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		HBC1F[5:0]						Reserved							FLD
		rw													rw

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13:8	HBC1F[5:0]	Header bank of CAN1 filter These bits are set and cleared by software to define the first bank for CAN1 filter. Bank0 ~ Bank HBC1F-1 used to CAN0. Bank HBC1F ~ Bank27 used to CAN1. When set 0, not bank used to CAN0. When set 28, not bank used to CAN1.
7:1	Reserved	Must be kept at reset value
0	FLD	Filter lock disable 0: Filter lock enable 1: Filter lock disable

21.4.18. Filter mode configuration register (CAN_FMCFG)

Address offset: 0x204

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FMOD27	FMOD26	FMOD25	FMOD24	FMOD23	FMOD22	FMOD21	FMOD20	FMOD19	FMOD18	FMOD17	FMOD16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMOD15	FMOD14	FMOD13	FMOD12	FMOD11	FMOD10	FMOD9	FMOD8	FMOD7	FMOD6	FMOD5	FMOD4	FMOD3	FMOD2	FMOD1	FMOD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FMODx	Filter mode 0: Filter x with Mask mode 1: Filter x with List mode

21.4.19. Filter scale configuration register (CAN_FSCFG)

Address offset: 0x20C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FS27	FS26	FS25	FS24	FS23	FS22	FS21	FS20	FS19	FS18	FS17	FS16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FS15	FS14	FS13	FS12	FS11	FS10	FS9	FS8	FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FSx	Filter scale 0: Filter x with 16-bit scale 1: Filter x with 32-bit scale

21.4.20. Filter associated FIFO register (CAN_FAFIFO)

Address offset: 0x214

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FAF27	FAF26	FAF25	FAF24	FAF23	FAF22	FAF21	FAF20	FAF19	FAF18	FAF17	FAF16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAF15	FAF14	FAF13	FAF12	FAF11	FAF10	FAF9	FAF8	FAF7	FAF6	FAF5	FAF4	FAF3	FAF2	FAF1	FAF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FAFx	Filter associated FIFO 0: Filter x associated with FIFO0 1: Filter x associated with FIFO1

21.4.21. Filter working register (CAN_FW)

Address offset: 0x21C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FW27	FW26	FW25	FW24	FW23	FW22	FW21	FW20	FW19	FW18	FW17	FW16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FW15	FW14	FW13	FW12	FW11	FW10	FW9	FW8	FW7	FW6	FW5	FW4	FW3	FW2	FW1	FW0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FWx	Filter working 0: Filter x working disable 1: Filter x working enable

21.4.22. Filter x data y register (CAN_FxDATAy) (x=0..27, y=0,1)

Address offset: 0x240+8*x+4*y, (x=0..27, y=0,1)
Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FD31	FD30	FD29	FD28	FD27	FD26	FD25	FD24	FD23	FD22	FD21	FD20	FD19	FD18	FD17	FD16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:0	FDx	Filter data Mask mode 0: Mask match disable 1: Mask match enable List mode 0: List identifier bit is 0 1: List identifier bit is 1

22. Ethernet (ENET)

22.1. Overview

This section applies only to GD32F107xx connectivity line devices.

This chapter describes the Ethernet peripheral module. There is a media access controller (MAC) designed in Ethernet module to support 10/100Mbps interface speed. For more efficient data transfer between Ethernet and memory, a DMA controller is designed in this module. The support interface protocol for Ethernet is media independent interface (MII) and reduced media independent interface (RMII). This module is mainly compliant with the following two standards: IEEE 802.3-2002 and IEEE 1588-2002.

22.2. Characteristics

MAC feature

- 10Mbit/s and 100Mbit/s data transfer rates support.
- MII and RMII interface support
- Loopback mode support for diagnosis
- CSMA/CD Protocol for Half-duplex back-pressure operation support.
- IEEE 802.3x flow control protocol support. Automatic delay a pause time which is decoded from a receive pause frame after current transmitting frame complete. MAC automatically transmits pause frame or back pressure feature depending on fill level of RxFIFO in Full-duplex mode or in Half-duplex mode.
- Automatic transmission of pause frame on assertion and de-assertion of flow control input frame. Zero-quanta pause time length frame for Full-duplex operation. IEEE 802.3x flow control for Full-duplex operation support. Back pressure feature to the MAC core based on RxFIFO fill level (Cut-Through mode) support. IEEE 802.3x flow control for Half-duplex operation support.
- Software configurable for automatic PAD/CRC generation in transmits operation.
- Software configurable for automatic PAD/CRC stripping in receives operation.
- Software configurable for frame length to support standard frames with sizes up to 16 KB.
- Software configurable for inter-frame gap (40-96 bit times in steps of 8).
- Support different receiving filter mode.
- IEEE 802.1Q VLAN tag detection function support for reception frames.
- Support mandatory network statistics standard (RFC2819/RFC2665).
- Two types of wakeup frame detection: LAN remote wakeup frame and AMD Magic Packet^{TM} frames.
- Support checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagram.

- Support Ethernet frame time stamping for both transmit and receive operation, which describes in IEEE 1588-2008, and 64 bit time stamps are given in each frame's status.
- Two independent FIFO of 2K Byte for transmitting and receiving.
- Support special condition frame discards handling, e.g. late collision, excessive collisions, excessive deferral or underrun.
- Calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frame transmit under Store-and-Forward mode.

DMA Feature

- Two types of descriptor addressing: Ring and Chain.
- Each descriptor can transfer up to 8 KB of data.
- Programmable normal and abnormal interrupt for many status conditions
- Round-robin or fixed-priority arbitration between reception and transmission controller.

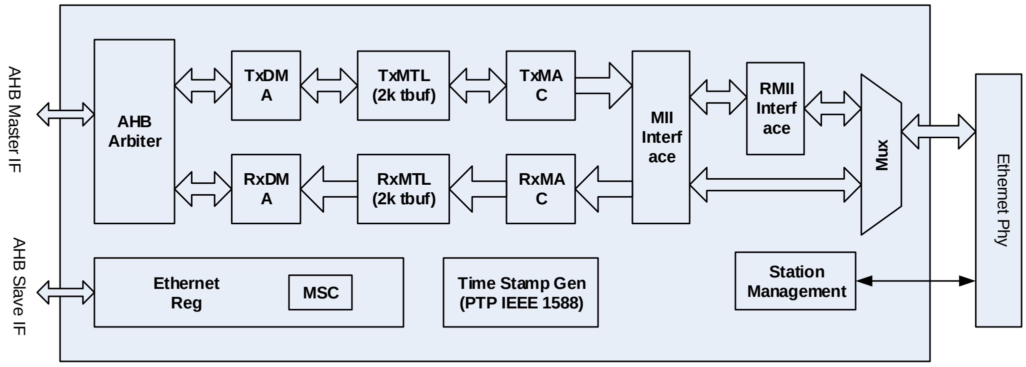
PTP Feature

- Support IEEE 1588 time synchronization function.
- Support two correction methods: Coarse or Fine.
- Pulse per second output.
- Preset target time reaching trigger and interrupt.

22.2.1. Block diagram

The Ethernet module is composed of a MAC module, MII/RMII module and a DMA module by descriptor control.

Figure 22-1. ENET module block diagram



The MAC module is connected to the external PHY by MII or RMII through one selection bit (refer to AFIO_PCF0 register). The SMI (Station Management Interface) is used to configure and manage external PHY.

Transmitting data module includes:

- TxDMA controller, used to read descriptors and data from memory and writes status to memory.

- TxMTL, used to control, management and store the transmit data. TxFIFO is implemented in this module and used to cache transmitting data from memory for MAC transmission.
- The MAC transmission relative control registers, used to control frame transmit.

Receiving data module includes:

- RxDMA controller, used to read descriptors from memory and writes received frame data and status to memory.
- RxMTL, used to control, management and store reception data. RxFIFO is implemented in this module and used to temporarily store received frame data before forwarding them into the system physical memory.
- The MAC reception relative control registers, used to control frame receive and marked the receiving state. Also a receiving filter with a variety of filtering mode is implemented in MAC, used to filter out specific Ethernet frame

Note: The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

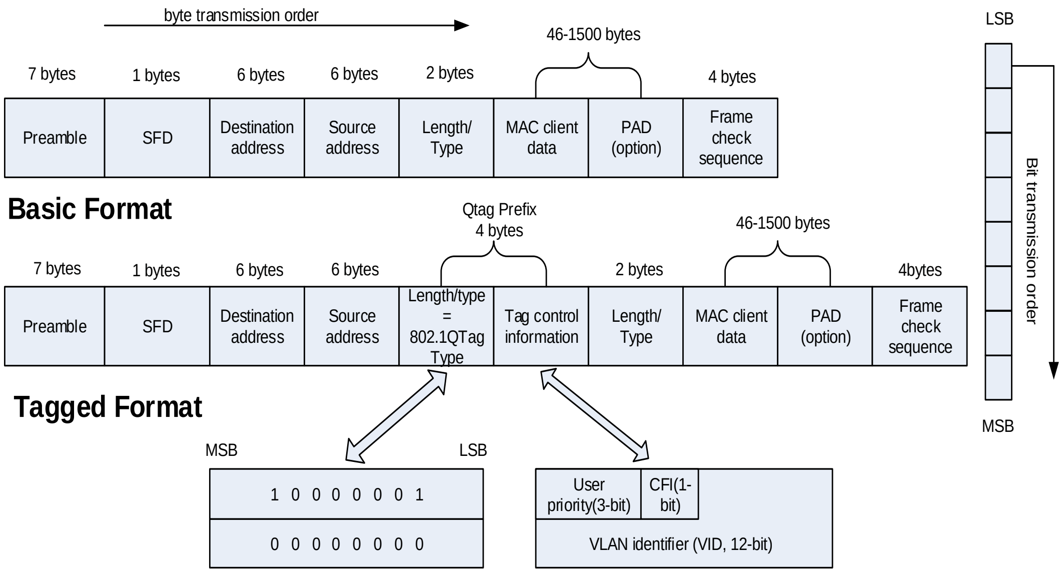
22.2.2. MAC 802.3 Ethernet packet description

Data communication of MAC can use two frame formats:

- Basic MAC frame format.
- Tagged MAC frame format (extension of the basic MAC frame format).

Figure 22-2. MAC/Tagged MAC frame format describes the structure of the frame (Basic and Tagged) that includes the following fields:

Figure 22-2. MAC/Tagged MAC frame format



Note: The Ethernet controller transmits each byte at LSB first except FCS field.

CRC calculation data comes from all bytes in the frame except the Preamble and SFD domain. The Ethernet frame’s 32-bit CRC calculation value generating polynomial is fixed

0x04C11DB7 and this polynomial is used in all 32-bit CRC calculation places in Ethernet module, as follows:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

22.2.3. Ethernet signal description

Table below shows the MAC module that pin is used default and remapping functions and specific configuration in MII/RMII mode.

Table 22-1. Ethernet pin configuration

MAC signals	Pin	Pin configuration	MII default	MII remap	RMII default	RMII remap
ETH_MDC	PC1	AF output push-pull highspeed (50 MHz)	MDC		MDC	
ETH_MII_TXD2	PC2	AF output push-pull highspeed (50 MHz)	TXD2			
ETH_MII_TX_CLK	PC3	Floating input (reset state)	TX_CLK			
ETH_MII_CRS	PA0	Floating input (reset state)	CRS			
ETH_RX_CLK ETH_RMII_REF_CLK	PA1	Floating input (reset state)	RX_CLK		REF_CLK	
ETH_MDIO	PA2	AF output push-pull highspeed (50 MHz)	MDIO		MDIO	
ETH_MII_COL	PA3	Floating input (reset state)	COL			
ETH_MII_RX_DV ETH_RMII_CRS_DV	PA7	Floating input (reset state)	RX_DV		CRS_DV	
ETH_MII_RXD0 ETH_RMII_RXD0	PC4	Floating input (reset state)	RXD0		RXD0	
ETH_MII_RXD1 ETH_RMII_RXD1	PC5	Floating input (reset state)	RXD1		RXD1	
ETH_MII_RXD2	PB0	Floating input (reset state)	RXD2			
ETH_MII_RXD3	PB1	Floating input (reset state)	RXD3			
ETH_PPS_OUT	PB5	AF output push-pull highspeed (50 MHz)	PPS_OUT		PPS_OUT	
ETH_MII_TXD3	PB8	AF output push-pull highspeed (50 MHz)	TXD3			
ETH_MII_RX_ER	PB10	Floating input (reset state)	RX_ER			
ETH_MII_TX_EN	PB11	AF output push-pull	TX_EN		TX_EN	

MAC signals	Pin	Pin configuration	MII default	MII remap	RMII default	RMII remap
ETH_RMII_TX_EN		highspeed (50 MHz)				
ETH_MII_TXD0 ETH_RMII_TXD0	PB12	AF output push-pull highspeed (50 MHz)	TXD0		TXD0	
ETH_MII_TXD1 ETH_RMII_TXD1	PB13	AF output push-pull highspeed (50 MHz)	TXD1		TXD1	
ETH_RMII_CRS_DV	PD8	Floating input (reset state)		RX_DV		CRS_DV
ETH_MII_RXD0 ETH_RMII_RXD0	PD9	Floating input (reset state)		RXD0		RXD0
ETH_MII_RXD1 ETH_RMII_RXD1	PD10	Floating input (reset state)		RXD1		RXD1
ETH_MII_RXD2	PD11	Floating input (reset state)		RXD2		
ETH_MII_RXD3	PD12	Floating input (reset state)		RXD3		

22.3. Function overview

22.3.1. Interface configuration

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the MII/RMII interface. MII or RMII mode is selected by software and carry on the PHY management through the SMI interface.

SMI: Station management interface

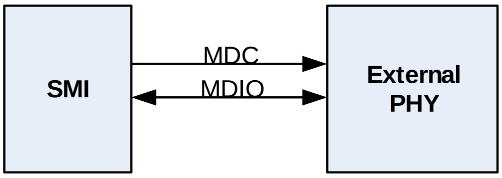
SMI is designed to access and configure PHY’s configuration.

Station management interface (SMI) is performed through two wires to communicate with the external PHY: one clock line (MDC) and one data line (MDIO), it can access to the any PHY register. The interface supports accessing up to 32 PHYs, but only one register in one PHY can be addressed at the same time.

MDC and MDIO specific functions as follows:

- **MDC**: A clock of maximum frequency is 2.5 MHz. The pin remains low level when it is in idle state. The minimum high or low level lasts time of MDC must be 160ns, and the minimum period of MDC must be 400ns when it is in data transmission state.
- **MDIO**: Used to transfer data in conjunction with the MDC clock line, receiving data from external PHY or sending data to external PHY.

Figure 22-3. Station management interface signals



SMI write operation

Applications need to write transmission data to the ENET_MAC_PHY_DATA register and operate the ENET_MAC_PHY_CTL register as follows:

1) Set the PHY device address and PHY register address, and set PW to 1, so that can select write mode.

2) Set PB bit to start transmission. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

SMI read operation

Applications need to operate the ENET_MAC_PHY_CTL register as follows:

1) Set the PHY device address and PHY register address and set PW to 0, so that can select read mode.

2) Set PB bit to start reception. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

Note: Because the PHY register address 16-31 register function is defined by each manufacturer, access different PHY device’s this part should see according to the manufacturer’s manual to adjust the parameters of applications. Details of catalog that firmware library currently supports the PHY device can refer to firmware library related instructions.

SMI clock selection

The SMI clock is generated by dividing application clock (AHB clock). In order to guarantee the MDC clock frequency is no more than 2.5MHz, application should set appropriate division factor according to the different AHB clock frequency. The following table lists the frequency factor corresponding AHB clock selection.

Table 22-2. Clock range

AHB clock	MDC clock	Selection
20~35MHz	AHB clock/16	0x3
35~60MHz	AHB clock/26	0x2
90~108 MHz	AHB clock/64	0x1
60~90MHz	AHB clock/42	0x0

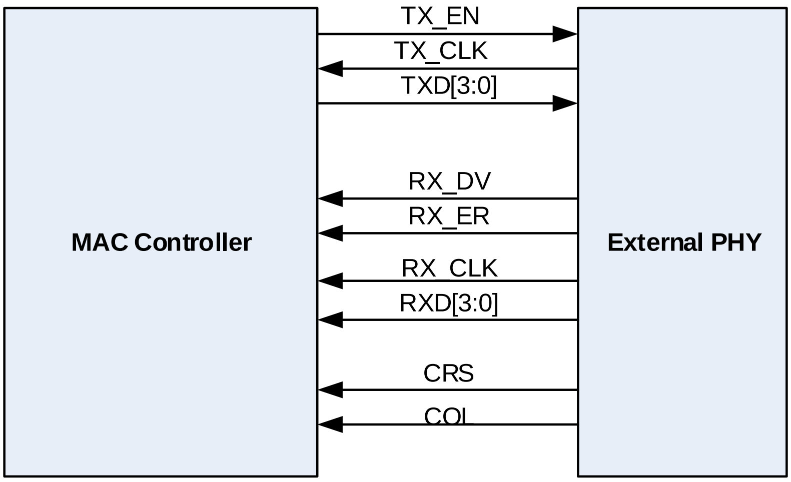
MII/RMII selection

The application can select the MII or RMII mode through the configuration bit 23 of the AFIO_PCF0 register ENET_PHY_SEL while the Ethernet controller is under reset state or before enabling the clocks. The MII mode is set by default.

MII: Media independent interface

The media-independent interface (MII) defines the interconnection between the MAC sub-layer and the PHY for data transfer at 10 Mbit/s or 100 Mbit/s.

Figure 22-4. Media independent interface signals



- **MII_TX_CLK**: clock signal for transmitting data. For the data transmission speed of 10Mbit/s, the clock is 2.5MHz, for the data transmission speed of 100Mbit/s, the clock is 25MHz.

- **MII_RX_CLK**: Clock signal for receiving data. For the data transmission speed of 10Mbit/s, the clock is 2.5MHz, for the data transmission speed of 100Mbit/s, the clock is 25MHz.

- **MII_TX_EN**: Transmission enable signal. It must be asserted synchronously with the first bit of the preamble and must remain asserted while all bits to be transmitted are presented to the MII.

- **MII_TXD[3:0]**: Transmit data line, each 4 bit data transfer, data is valid when the MII_TX_EN signal is effective. MII_TXD[0] is the least significant bit and MII_TXD[3] is the most significant bit. While MII_TX_EN is de-asserted the transmit data must have no effect upon the PHY.

- **MII_CRS**: Carrier sense signal, only working in Half-duplex mode and controlled by the PHY. It is active when either transmit or receive medium is in non idle state. The PHY must ensure that the MII_CRS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the TX and RX clock.

- **MII_COL**: Collision detection signal, only working in Half-duplex mode, controlled by the PHY. It is active when a collision on the medium is detected and must it will remain active while the collision condition continues. This signal is not required to transition synchronously with respect to the TX and RX clock.

- **MII_RXD[3:0]**: Receive data line, each 4 bit data transfer; data are valid when the MII_RX_DV signal is effective. MII_RXD[0] is the least significant bit and MII_RXD[3] is the most significant bit. While MII_RX_DV is de-asserted and MII_RX_ER is asserted, a specific MII_RXD[3:0] value is used to indicate specific information (see Table 27-3).

- **MII_RX_DV**: Receive data valid signal, controlled by the PHY. It is asserted when PHY is presenting data on the MII for reception. It must be asserted synchronously with the first 4-bit of the frame and must remain asserted while all bits to be transmitted are presented on the MII. It must be de-asserted prior to the first clock cycle that follows the final 4-bit. In order to receive the frame correctly, the effective signal starting no later than the SFD field.

- **MII_RX_ER**: Receive error signal. It must be asserted for one or more RX clock to indicate MAC detected an error in the receiving process. The specific error reason needs to cooperate with the state of the MII_RX_DV and the MII_RXD[3:0] data value (see ***Table 22-3. Rx interface signal encoding***).

Table 22-3. Rx interface signal encoding

MII_RX_ER	MII_RX_DV	MII_RXD[3:0]	Description
0	0	0000 to 1111	Normal inter-frame
0	1	0000 to 1111	Normal reception frame data
1	0	0000	Normal inter-frame
1	0	0001 to 1101	Reserved
1	0	1110	False carrier indication
1	0	1111	Reserved
1	1	0000 to 1111	Data reception with errors

MII clock sources

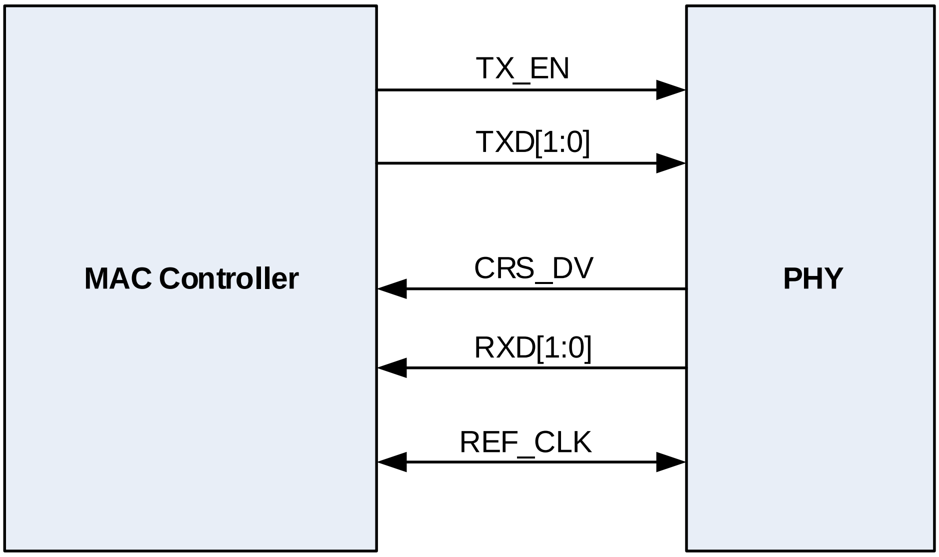
To generate both TX_CLK and RX_CLK clock signals, the external PHY needs an external 25MHz clock. This 25MHz clock does not require the same one with MAC clock. It can use the external 25MHz crystal or the output clock of microcontroller’s CK_OUT0 pin. If the clock source is selected from CK_OUT0 pin, the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUT0 pin is 25MHz.

RMII: Reduced media independent interface

The reduced media-independent interface (RMII) specification reduces the pin count during Ethernet communication. According to the IEEE 802.3 standard, an MII contains 16 pins for data and control. The RMII specification is dedicated to reduce the pin count to 7.

The RMII block has the following characteristics:
- The clock signal needs to be increased to 50MHz and only one clock signal.
- MAC and external PHY use the same clock source
- Using the 2-bit wide data transceiver

Figure 22-5. Reduced media-independent interface signals



MII/RMII bit transmission order

No matter which interface (MII or RMII) is selected, the bit order of transmit/receive is LSB first.

The deference between MII and RMII is bit number and sending times. MII is low 4bits first and then high 4bits, but RMII is the lowest 2bits, low 2bits, high 2bits and the highest 2bits.

For example: a byte value is: 10011101b (left to right order: high to low)
Transmission order for MII use 2 cycles: 1101 -> 1001 (left to right order: high to low)
Transmission order for RMII use 4 cycles: 01 -> 11 -> 01 -> 10 (left to right order: high to low)

RMII clock sources

To ensure the synchronization of the clock source, the same clock source is selected for the MAC and external PHY which is called REF_CLK. The REF_CLK input clock can be connected to the external 50MHz crystal or microcontroller CK_OUT0 pin. If the clock source is from CK_OUT0 pin, then the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUT0 pin is 50MHz.

22.3.2. MAC function overview

MAC module can achieve the following functions:

Data package (transmission and reception)

- Frame detecting/decoding and frame boundary delimitation.
- Addressing (handling source address and destination address).
- Error conditions detect.

Medium access management in Half-duplex mode

- Medium allocation (prevent conflicts).
- Conflict resolution (dealing with conflicts).

The MAC module can work in two modes

- Half-duplex mode: with the CSMA/CD algorithm to contend for using of the physical medium, at the same time only one transmission direction is active between two stations is active.
- Full-duplex mode: simultaneous transmission and reception without any conflict mode, if all of the following conditions are satisfied: 1) PHY supports the feature of transmission and reception operations at the same time. 2) Only two devices connect to the LAN and the two devices are both configured for Full-duplex mode.

Transmission process of MAC

All transactions are controlled by the dedicated DMA controller and MAC in Ethernet. After received the sending instruction, the TxDMA fetches the transmit frames from system memory and pushes them into the TxFIFO, then the data in TxFIFO are poped to MAC for sending on MII/RMII interface. The method of popping is according to the selected TxFIFO mode (Cut-Through mode or Store-and-Forward mode, the specific definition see the next paragraph).For convenient, application can configure automatically hardware calculated CRC and insert it to the FCS domain of Ethernet frame function. The entire transmission process complete when the MAC received the frame termination signal from transmit FIFO. When transmission completed, the transmission status information will be composed of MAC and write return to the DMA controller, the application can query through the DMA current transmit descriptor.

Operation for popping data from FIFO to the MAC has two modes:

- In Cut-Through mode, as soon as the number of bytes in the FIFO crosses or equals the configured threshold level or when the end-of-frame flag in descriptor is written, the data is ready to be popped out and forwarded to the MAC. The threshold level is configured using the TTHC bits of ENET_DMA_CTL.
- In Store-and-Forward mode, only after an integrated frame is stored in the FIFO, the frame is popped towards the MAC. But there is another condition for FIFO popping out data but the frame is not integrated. This is when the transmit FIFO size is smaller than

the Ethernet frame to be transmitted, the frame is popped towards the MAC when the transmit FIFO becomes almost full.

Handle special cases

In the transmission process, due to the insufficient TxDMA descriptor or misuse of FTF bit in ENET_DMA_CTL register (when this bit is set, it will clear FIFO data and reset the FIFO pointer, after clear operation is completed, it will be reset), there will be a transmit data underflow fault occurs because of insufficient data in FIFO. At the same time MAC will identify such data underflow state and write relevant status flag.

If one transmit frame uses two TxDMA descriptors for sending data, then the first segment (FSG) and the last segment (LSG) of the first descriptor should be 10b and the second ones should be 01b.If both the FSG bit of the first and the second descriptor are set and the LSG bit in the first descriptor is reset, then the FSB bit of the second descriptor will be ignored and these two descriptors are considered to sending the only one frame.

If the byte length of one transmission MAC frame’s data field is less than 46 (for Tagged MAC frame is less than 42), application can configure the MAC for automatically adding a load of content of ‘0’ bit to make the byte length of frame’s data field in accordance with the relevant domain of definition of IEEE802.3 specification. At the same time, if automatically adding zeros function is performed, the MAC will certainly calculate CRC value of the frame and append it to the frame’s FCS field domain no matter what configuration of DCRC bit in the descriptor is.

Transmission management of MAC

Jabber timer

In case of one station occupies the PHY for a long time, there is a jabber timer designed for cutting off the frame whose length is more than 2048 bytes. By default, jabber timer is enabled so when application is transmitting a frames whose byte length is more then 2048, the MAC will only transmit 2048 bytes and drop the last ones.

Collision condition solve mechanism – Re-transmission

When the MAC is running under Half-duplex mode, collision may happen when MAC is transmitting frame data on interface. When no more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, the re-transmission function is active. In this case, MAC will stop current transmitting and then read frame data from FIFO again and send them on interface again. When more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, MAC will abort transmitting current frame data and not re-transmit it. Also MAC will set late collision flag in descriptor to inform application.

Transmit status word

Transmit status word includes many transmit state flags for application and are updated after the complete the transmission of the frame. If timestamp function is enable, the timestamp value is also write back to transmit descriptor along with transmit status.

Transmit FIFO flush operation

Application can clear TxFIFO and reset the FIFO data pointer by setting FTF bit (bit 20) of ENET_DMA_CTL register. The flush operation will be executed at once no matter whether TxFIFO is popping data to MAC. This results in an underflow event in the MAC transmitter, and the makes frame transmission abort. At the same time, MAC returns state information of frame and transmit status words transferred to the application. The status of such a frame is marked with both underflow and frame flush events (TDES0 bits 1 and 13). When the transmit data in TxFIFO is flushed, the transmit status word will be written back to descriptor. After the status is written, the flush operation is complete. When a flush operation is received, all the following data which should be popped from TxFIFO into MAC will be dropped unless a new FSG bit of descriptor is received. After operation completed, the FTF bit of ENET_DMA_CTL register is then automatically cleared.

Note: If the amount of the receive data is large, the flush operation need to be opened. Reset the ENET module and initialize data structure during the receive overflow interrupt.

Transmit flow control

The MAC manages transmission frame through back pressure (in Half-duplex mode) and the pause control frame (in Full-duplex mode) for flow control.

- Half-duplex mode flow control : Back Pressure

When MAC is configured in Half-duplex mode, there are two conditions to trigger the back pressure feature. Both of the two conditions are triggered to enable back pressure function which is implemented by sending a special pattern (called jam pattern) 0x5555 5555 once to notify conflict to all other sites. The first condition is triggered by application setting the FLCB/BKPA bit in ENET_MAC_FCTL register. The second condition occurs during receiving frame. When MAC receiver is receiving frame, the byte number of RxFIFO is more and more great. When this number is greater than the high threshold (RFA bits in ENET_MAC_FCTH), MAC will set the back pressure pending flag. If this flag is set and a new frame presents on interface, MAC will send a jam pattern to delay receiving this new frame a back pressure time. After this back pressure time is end, external PHY will send this new frame again. If the number of the RxFIFO is not less than low threshold (RFD bits in ENET_MAC_FCTH) during this back pressure time, a jam pattern is send again. If the number of the RxFIFO is less than low threshold (RFD bits in ENET_MAC_FCTH) during this back pressure time, MAC resets the back pressure pending flag and is enable to receive the new frame instead of sending jam pattern.

- Full-duplex mode flow control : Pause Frame

The MAC uses a mechanism named "pause frame" for flow control in Full-duplex mode. Receiver can send a command to the sender for informing it to suspend transmission a period of time. If the application sets transmit flow control bit TFCEN in ENET_MAC_FCTL register, MAC will generate and transmit a pause frame when either of two conditions is satisfied in Full-duplex mode. There are two conditions to start transmit pause frames:

1) Application sets FLCB/BKPA bit in ENET_MAC_FCTL register to immediately send a pause frame. When doing this, MAC sends a pause frame right now with the pause time value PTM configured in ENET_MAC_FCTL register. If application considers the pause time is no need any more because the transmit frame can be transmitted without pause time, it can end the pause time by setting the pause time value PTM bits in ENET_MAC_FCTL register to 0 and set FLCB/BKPA bit to send this zero pause time frame.

2) MAC automatically sends pause time when the RxFIFO is in some condition. When MAC is receiving frame, RxFIFO will be fill in many receive data. At same time RxFIFO pops out data to RxDMA for forwarding to memory. If the popping frequency is lower than MAC pushing frequency, the number of bytes in RxFIFO is getting great. Once the data amount in RxFIFO is greater than the active threshold value (RFA bits in ENET_MAC_FCTH) of flow control, MAC will send a pause frame with PTM value in it. After sending pause frame, MAC will start a counter with configured reload value PLTS in ENET_MAC_FCTL register, when configured PLTS time has spent, the MAC will check RxFIFO again. If the byte number in RxFIFO is also greater than active threshold value, the MAC sends a pause time again. When the byte number of RxFIFO is lower than the de-active threshold value, MAC maybe send a pause frame with zero time value in frame's pause time field if DZQP bit in ENET_MAC_FCTL register is reset. This zero-pause time frame can inform send station that RxFIFO is almost empty and can receive new data again.

Transmit inter-frame gap management

MAC can manage the interval time between two frames. This interval time is called frame gap time. For Full-duplex mode, after complete sending a frame or MAC entered idle state, the gap time counter starts counting. If another transmit frame presents before this counter has not reach the configured IGBS bit time in ENET_MAC_CFG register, this transmit frame will be pended unless the counter reach the gap time. But if the second transmit frame presents after the gap time counter has reached the configured gap time, this frame will send immediately. For Half-duplex mode, the gap time counter follows the Truncated Binary Exponential Backoff algorithm. Briefly speaking, the gap time counter starts after the previous frame has completed transmitting on interface or the MAC entered idle state, and there are three conditions may occur during the gap time:

1) The carrier sense signal active in the first 2/3 gap period. In this case, the counter will reload and restart.
2) The carrier sense signal active in the last 1/3 gap period. In this case, the counter will not reload but continue counting, and when reaches gap time, the MAC sends the second frame

3) The carrier sense signal not active during the whole gap period. In this case, the counter stops after reaches the configured gap time and sends frame if the second frame has pended.

Transmit checksum offload

The MAC supports transmit checksum offload. This feature can calculate checksum and insert it in the transmit frame, and detect error in the receive frame. This section describes the operation of the transmit checksum offload.

Note: This function is enabled only when the TSFD bit in the ENET_DMA_CTL register is set (TxFIFO is configured to Store-and-Forward mode) and application must ensure the TxFIFO deep enough to store the whole transmit frame. If the depth of the TxFIFO is less than the frame length, the MAC only does calculation and insertion for IPv4 header checksum field.

See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460 and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6 and ICMPv6 packet header specifications, respectively.

- IP header checksum

If the value is 0x0800 in type field of Ethernet frame and the value is 0x4 in the IP datagram's version field, checksum offload module marks the frame as IPv4 package and calculated value replace the checksum field in frame. Because of IPv6 frame header does not contain checksum field, the module will not change any value of the IPv6's header field. After IP header checksum calculation end, the result is stored in IPHE bit (bit 16 in TDES0). The following shows the conditions under which the IPHE bit can be set:

1) For IPv4 frame type:
A). type field is 0x0800 but version filed in IP header is not 0x4.
B). IPv4 header length field value is greater than total frame byte length
C). the value of IPv4 header length field is less than 0x5 (20 bytes)

2) For IPv6 frame type:
A). type field is 0x86dd but version field in IP header is not 0x6
B). the frame ends before the IPv6 standard header or extension header (as given in the corresponding header length field in an extension header) has been completely received.

- TCP/UDP/ICMP payload checksum

The checksum offload module processes the IPv4 or IPv6 header (including extension headers) and marks the type of frame (TCP, UDP or ICMP).

But when the following frame cases are detected, the checksum offload function will be bypassed and these frames will not be processed by the checksum offload module:

1) Incomplete IPv4 or IPv6 frames

2) IP frames with security features (e.g. authentication header, security payload)

3) IP frames without TCP/UDP/ICMPv4/ICMPv6 payload

4) IPv6 frames with routing headers

The checksum offload module calculates the TCP, UDP, or ICMP payload and inserts the result into its corresponding field in the header. It has two modes when working, as follows:

1) TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame’s checksum field. The checksum field is included in the checksum calculation, and then replaced by the final calculated checksum.

2) Checksum offload module clears the contents of the checksum field in the transmission frame and make calculation which includes TCP, UDP, or ICMPv6 pseudo-header data and will instead the transmission frame’s original checksum field by the final calculation results.

After calculated by checksum offload module, the result can be found in IPPE bit (bit 12 in TDES0). The following shows the conditions under which the IPPE bit can be set:

1) In Store-and-Forward mode, frame has been forwarded to MAC transmitter but no EOF is written to TxFIFO

2) Frame is ended but the byte numbers which the payload length field of the frame indicates has not been reached

If the packet length is greater than the marked length, checksum module does not report errors, the excess data will be discarded as padding bytes. If the first condition of IPPE error is detected, the value of the checksum does not insert a TCP, UDP or ICMP header. If the second condition of IPPE error is detected, checksum calculation results will still insert the appropriate header fields.

Note: For ICMP packets over IPv4 frame, the checksum field in the ICMP packet must always be 0x0000 in both modes due to such packets are not defined pseudo-headers.

MAC receive filters

The MAC filter is divided into error filtering (such as too short frame, CRC error and other bad frame filtering) and address filtering. This section mainly describes the address filtering.

Note: If the receive filters are used, the Store-and-Forward mode is recommended for receiving.

Address filtering

Address filtering use the static physical address (MAC address) filter and hash list filter for implementing the function. If the FAR bit in the ENET_MAC_FRMF register is '0' (by default), only the frame passed the filter will be received. This function is configured according to the parameters of the application (frame filter register) to filter the destination or/and source address of unicast or multicast frame (The difference between an individual address and a group address is determined by I/G bit in the destination address field) and report the result of the corresponding address filtering. The frame not pass through the filter will be discarded.

Note: If the FAR bit in the ENET_MAC_FRMF register is set to 1, frames are all thought passed the filter. In this case, even the filter result will also be updated in receive descriptor but the result will not affect whether current frame passes the filter or not.

Unicast frame destination address filter

For a unicast frame, application has two modes for filtering: the one is using static physical address (by setting HUF bit to '0'), the other is using hash list (by setting HUF bit to '1').

- Static physical address (SPA) filtering

In the filter mode, MAC supports using four MAC addresses for unicast frame filtering. In this way, the MAC compares all 6 bytes of the received unicast address to the programmed MAC address. MAC address 0 is always used and MAC address 1 to address 3 can be configured to use or not. Each byte of MAC address 1 to MAC address 3 register can be masked for comparison with the corresponding destination address byte of received frame by setting the corresponding mask byte bits (MB) in the corresponding register.

- Hash list filtering

In this filter mode, MAC uses a HASH mechanism. MAC uses a 64-bit hash list to filter the received unicast frame. This filter mode obeys the followings two filtering steps:

1) The MAC calculates the CRC value of the received frame's destination address

2) Using the high 6 bits of the calculated CRC value as the index to retrieve the hash list. If the corresponding value of hash list is 1, the received frame passes through the filter, conversely, fail the filter.

The advantage of this type of filter is that it can cover any possible address just using a small table. But the disadvantage is that the filter is imperfect and sometimes the frames should be dropped are also be received by mistake.

Multicast frame destination address filter

Application can enable the multicast frame MAC address filtering by cleaning the MFD bit in register ENET_MAC_FRMF. By configuring the value of HMF bit in ENET_MAC_FRMF register application can choose two ways just like unicast destination address filtering for address filtering.

Hash or perfect address filter

The destination address (DA) filter can be configured to pass a frame when its DA matches either the hash list filter or the static physical address filter by setting the HPFLT bit in the ENET_MAC_FRMF register and setting the corresponding HUF or HMF bit in the ENET_MAC_FRMF register.

Broadcast frame destination address filter

At default, the MAC unconditionally receives the broadcast frames. But when setting BFRMD bit in register ENET_MAC_FRMF, MAC discards all received broadcast frames.

Unicast frame source address filter

Enable MAC address 1 to MAC address 3 register and set the corresponding SAF bit in the MAC address high register, the MAC compares and filter the source address (SA) field in the received frame with the values programmed in the SA registers. MAC also supports the group filter on the source address. If the SAFLT bit in frame filter register ENET_MAC_FRMF is set, MAC drops the frame that failed the source address filtering; meanwhile the filter result will reflect by SAFF bit in RDES0 of DMA receive descriptor. When the SAFLT bit is set, the destination address filter is also at work, so the result of the filter is simultaneous determined by DA and SA filter. This means that, as long as a frame does not pass any one of the filters (DA filter or SA filter), it will be discarded. Only a frame passing the entire filter can be forwarded to the application.

Reverse filtering operation

MAC can reverse filter-match result at the final output whether the destination address filtering or source address filtering. By setting the DAIFLT and SAIFLT bits in ENET_MAC_FRMF register, this address filter reverse function can be enabled. DAIFLT bit is used for unicast and multicast frames' DA filtering result, SAIFLT bit is used for unicast and multicast frames SA filtering result.

The following two tables summarize the destination address and source address filters working condition at different configuration.

Table 22-4. Destination address filtering table

Frame Type	PM	HPFLT	HUF	DAIFLT	HMF	MFD	BFRMD	DA filter operation
Broadcast	1	-	-	-	-	-	-	Pass
	0	-	-	-	-	-	0	Pass
	0	-	-	-	-	-	1	Fail
Unicast	1	-	-	-	-	-	-	Pass all frames
	0	-	0	0	-	-	-	Pass on perfect/group filter match
	0	-	0	1	-	-	-	Fail on perfect/group filter match
	0	0	1	0	-	-	-	Pass on hash filter match
	0	0	1	1	-	-	-	Fail on hash filter match
	0	1	1	0	-	-	-	Pass on hash or perfect/group filter match
	0	1	1	1	-	-	-	Fail on hash or perfect/group filter match
Multicast	1	-	-	-	-	-	-	Pass all frames
	-	-	-	-	-	1	-	Pass all frames
	0	-	-	0	0	0	-	Pass on perfect/group filter match and drop PAUSE control frames if PCFRM = 0x

	0	0	-	0	1	0	-	Pass on hash filter match and drop PAUSE control frames if PCFRM = 0x
	0	1	-	0	1	0	-	Pass on hash or perfect/group filter match and drop PAUSE control frames if PCFRM = 0x
	0	-	-	1	0	0	-	Fail on perfect/group filter match and drop PAUSE control frames if PCFRM = 0x
	0	0	-	1	1	0	-	Fail on hash filter match and drop PAUSE control frames if PCFRM = 0x
	0	1	-	1	1	0	-	Fail on hash or perfect/group filter match and drop PAUSE control frames if PCFRM = 0x

Table 22-5. Source address filtering table

Frame type	PM	SAIFLT	SAFLT	SA filter operation
Unicast	1	-	-	Pass all frames
	0	0	0	Pass status on perfect/group filter match but do not drop frames that fail
	0	1	0	Fail status on perfect/group filter match but do not drop frame
	0	0	1	Pass on perfect/group filter match and drop frames that fail
	0	1	1	Fail on perfect/group filter match and drop frames that fail

Promiscuous mode

If the PM bit in ENET_MAC_FRMF register is set, promiscuous mode is enable. Then the address filter function is bypassed, all frames are thought passed through the filter. At the same time the receive status information DA / SA error bit is always '0'.

Pause control frame filter

When MAC received pause frame, it will detect 6 bytes DA field in the frame. If UPFDT bit in ENET_MAC_FCTL register is 0, it is determined by whether the value of the DA field conforms to the unique value (0x0180C2000001) with IEEE-802.3 specification control frames. If UPFDT bit in ENET_MAC_FCTL register is set, MAC additionally compares DA field with the programmed MAC address for bit match. If DA field match and receive flow control is enabled (RFCEN bit in ENET_MAC_FCTL register is set), the corresponding pause control frame function will be triggered. Whether this filter passed pause frame is forwarded to memory is depending on the PCFRM[1:0] bit in ENET_MAC_FRMF register.

Reception process of MAC

Received frames will be pushed to the RxFIFO. The MAC strips the preamble and SFD of the frame, and starts pushing the frame data beginning with the first byte following the SFD to the RxFIFO. If IEEE 1588 time stamp function is enabled, the MAC will record the current system time when a frame's SFD is detected. If the frame passes the address filter, this time stamp is passed on to the application by writing it to descriptor.

The MAC can automatically strip PAD and FCS field data when the length/type field of received frame is less than 1536 if APCD bit is set. MAC pushes the data of the frame into RxFIFO up to the count specified in the length/type field, then starts dropping bytes (including the FCS field). If the value of Length/Type field is greater than or equal to 0x600, regardless of whether the option of automatic CRC and pad stripping function is enabled, the MAC pushes all received frame data to Rx FIFO.

If the watchdog timer is enabled (WDD bit in ENET_MAC_CFG is reset), a frame has more than 2048 bytes (DA + SA + LT + DATA + FCS) will be cut off receiving when has received 2048 bytes. If the watchdog timer is disabled, the MAC can extend the max receiving data bytes to 16384(16K Bytes), any data beyond this number will be cut off.

When RxFIFO works at Cut-Through mode, it starts popping out data from RxFIFO when the number of FIFO is greater than threshold value (RTHC bits in ENET_DMA_CTL register). After all data of a frame pop out, receive status word is sent to DMA for writing back to descriptor. In this mode, if a frame has started to forward to application by DMA from FIFO, the forwarding will continue until the frame is end even if frame error is detected. Although the error frame is not discarded, the error status will reflect in descriptor status field.

When RxFIFO works at Store-and-Forward mode (set by RSFD bit in ENET_DMA_CTL), DMA reads frame data from RxFIFO only after RxFIFO has completed received the whole frame. In this mode, if the MAC is configured to discard all error frames, then only valid frames without any error can be read out from RxFIFO and forward to the application. Once the MAC detects an SFD signal on the interface, a receive operation is started. The MAC strips the preamble and SFD before processing the frame. The header fields are checked by filtering and the FCS field used to verify the CRC for the frame. The frame is discarded by MAC if it fails to pass the address filter.

Reception management of MAC

Receive operation on multi-frame handling

It is different from transmit operation, after receiving the last byte of a frame, the MAC can judge the status of the receiving operation, so the second received frame’s forwarding is surely followed by the first received frame data and status.

Receive flow control

In Full-duplex mode, the MAC can detect the pause control frames, and perform it by suspending a certain time which is indicated in pause time field of detected pause control frame and then to transmit data. This function can set by RFCEN bit in ENET_MAC_FCTL register. If this function is not enabled, the MAC will ignore the received pause frames. If this function is enabled, MAC can decode this frame. Type field, opcode field and pause time field in the frame are all recognized by the MAC. During the pause time period, if MAC received a new pause frame, the new pause time filed value is loaded to the pause time counter immediately. If the new pause time filed is zero, then the pause time counter stops and transmit operation recovers. Application can configure PCFRM bit in ENET_MAC_FRMF register to decide the solving method for such control frame.

Receive checksum offload

Receive checksum offload is enabled when IPFCO bit in ENET_MAC_CFG register is set. Receive checksum offload can calculate the IPv4 header checksum and check whether it matches the contents of the IPv4 header checksum field. The MAC identifies IPv4 or IPv6 frames by checking for the value of 0x0800 or 0x86DD respectively in the received Ethernet frame type field. This method is also used to identify frames with VLAN tags. Header checksum error bits in DMA receive descriptor (the 7 bit in RDES0) reflects the header checksum result. This bit is set if received IP header has the following errors:

- Any mismatch between the IPv4 calculation result by checksum offload module and the value in received frame’s checksum field.
- Any inconsistent between the data type of Ethernet type field and IP header version field.
- Received frame length is less than the length indicated in IPv4 header length field, or IPv4 or IPv6 header is less than 20 bytes.

Receive checksum offload also identifies the data type of the IP packet is TCP, UDP or ICMP, and calculate their checksum according to TCP, UDP or ICMP specification. Calculation process can include data of TCP/UDP/ICMPv6 pseudo-header. Payload checksum error bits in DMA receive descriptor (bit 0 in RDES0) reflects the payload checksum result. This bit is set if received IP payload has the following errors:

- Any mismatch between the TCP, UDP or ICMP checksum calculation result by checksum offload and the received TCP/UDP/ICMP frame’s checksum field.
- Any inconsistent between the received TCP, UDP or ICMP data length and length of IP header.

The received checksum offload does not calculate the following conditions: Incomplete IP packets, IP packets with security features, packets of IPv6 routing header and data type is not TCP, UDP or ICMP.

Error handling

- If RxFIFO becomes full but the last received byte is not the end of frame (EOF), the RxFIFO will discard the whole frame data and return an overflow status. Also the counter of counting the overflow condition times will plus 1.

- If the RxFIFO is configured in Store-and-Forward mode, the MAC can filter and discard all error frames. But according to the configuration of FERF and FUF bit in ENET_DMA_CTL register, RxFIFO can also receive and forward such error frame and the frame that length is less than the minimum length.
- If the RxFIFO is configured in Cut-Through mode, not all the error frames can be dropped. Only when the start of frame (SOF) has not been read from RxFIFO and the receive frame has been detected error status, the RxFIFO will discard the whole error frame.

Receive status word

After receiving a complete frame, the MAC will analysis and record some state information about the frame and receiving process. These detail status information will write back to the receive descriptor and DMA status flag. Application can check these flags for upper protocol implementation.

Note: The value of frame length is 0 means that for some reason (such as FIFO overflow or dynamically modify the filter value in the receiving process, resulting did not pass the filter, etc), frame data is not written to FIFO completely.

MAC loopback mode

Often, loopback mode is used for testing and debugging hardware and software system for application. The MAC loopback mode is enabled by setting the LBM bit in ENET_MAC_CFG register. In this mode, the MAC transmitter sends the Ethernet frame to its own receiver. This mode is disabled by default.

22.3.3. MAC statistics counters: MSC

For knowing the statistics situation of transmitting and receiving frames, there is a group of counters designed for gathering statistics data. These MAC counters are called statistics counters (MSC).In Section ‘Register Description’, there is a detailed description of the function of these registers.

When the transmit frame does not appear the following situation, it can be called ‘fine frame’ and MSC transmit counters will automatically update:

- Frame underflow
- No carrier
- Lose of carrier
- Excessive deferral
- Late collision
- Excessive collision
- Jabber Timeout

When the receiving frame does not appear the following situation, it can be called ‘fine frame’ and MSC reception counters will automatically update:

- Alignment error
- CRC mismatch(calculated CRC value is different from FSC field value)
- Runt frame (frame length is shorter than 64 bytes)
- Length error (length field value is different from the actual received data bytes)
- Range error (length field value is larger than maximum size of defined in IEEE802.3,which is 1518 for untagged frame and 1522 for VLAN tagged frame)
- Error signal valid on pin MII_RX_ER

Note: Only when the discarded frame is a short frame whose length is less than 6 bytes (no complete receives the DA), MSC reception counter is updated.

22.3.4. Wake up management: WUM

Ethernet (ENET) module supports two wakeup methods from Deep-sleep mode. The one is remote wakeup frame and the other is Magic Packet wakeup frame. For reduce power consuming, the host system and Ethernet can be powered down and thus the circuit driven by HCLK or transmit clock is stop working. But the circuit driven by receive clock will continues working for listening wakeup frame. If application sets the PWD bit in ENET_MAC_WUM register, the Ethernet enters into power-down state. In power-down state, MAC ignores all the frame data on the interface until the power-down state is exited. For exiting power-down state, application can choose one of or both of the two methods mentioned above. Setting WFEN bit in ENET_MAC_WUM register can make Ethernet wakeup if a remote wakeup frame received and setting MPE bit in ENET_MAC_WUM register can make Ethernet wakeup if a Magic Packet frame is received. When any type of wakeup frame is present on interface and corresponding wakeup function is enabled, Ethernet will generate a wakeup interrupt and exit power-down state at once.

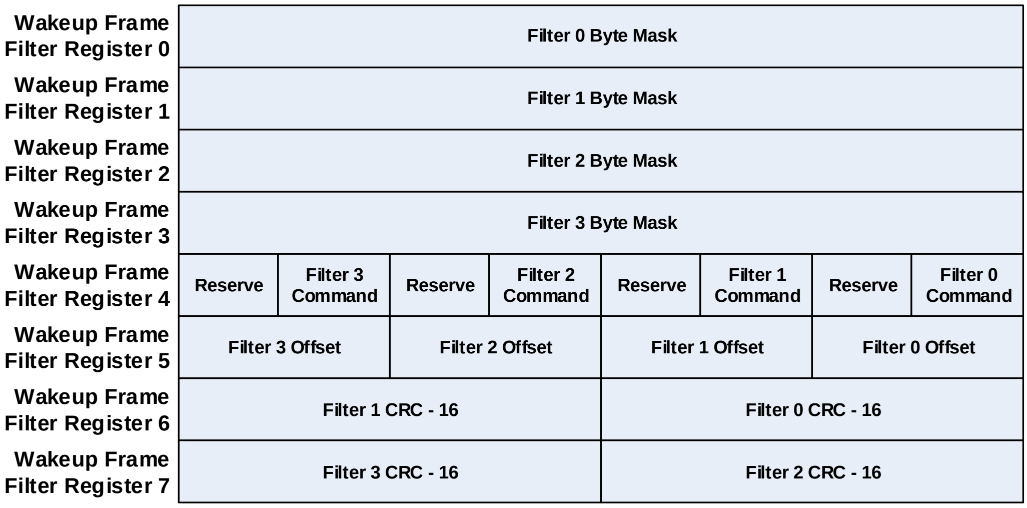
Remote wakeup frame detection

Setting WFEN bit in ENET_MAC_WUM register can enable remote wakeup detection. When the MAC is in power-down state and remote wakeup function enable bit is set, MAC wakeup frame filter is active. If the received frame passes the address filter and filter CRC-16 matches the incoming examined pattern, then MAC identified the received wakeup frame, and then MAC returns to normal working state. Even if the length of the wakeup frame exceeds 512 bytes, as long as the frame has a correct CRC value, it is still considered to be effective. After received the remote wakeup frame, the WUFR bit in ENET_MAC_WUM register will be set. If remote wakeup interrupt is not masked, then a WUM interrupt is generated.

Remote wakeup frame filter register

Wakeup frame filter register is made up of eight different registers but shared the same register offset address. So the inner pointer points the next filter register when the filter register address is accessed by writing or reading. Whatever operation, write or read, it is strongly recommended to operate eight times sequentially. This means continuously write 8 times will configure the filter registers and continuously read 8 times will get the values of filter registers.

Figure 22-6. Wakeup frame filter register



<table>
<tr><td>Wakeup Frame Filter Register 0</td><td colspan="8">Filter 0 Byte Mask</td></tr>
<tr><td>Wakeup Frame Filter Register 1</td><td colspan="8">Filter 1 Byte Mask</td></tr>
<tr><td>Wakeup Frame Filter Register 2</td><td colspan="8">Filter 2 Byte Mask</td></tr>
<tr><td>Wakeup Frame Filter Register 3</td><td colspan="8">Filter 3 Byte Mask</td></tr>
<tr><td>Wakeup Frame Filter Register 4</td><td>Reserve</td><td>Filter 3 Command</td><td>Reserve</td><td>Filter 2 Command</td><td>Reserve</td><td>Filter 1 Command</td><td>Reserve</td><td>Filter 0 Command</td></tr>
<tr><td>Wakeup Frame Filter Register 5</td><td colspan="2">Filter 3 Offset</td><td colspan="2">Filter 2 Offset</td><td colspan="2">Filter 1 Offset</td><td colspan="2">Filter 0 Offset</td></tr>
<tr><td>Wakeup Frame Filter Register 6</td><td colspan="4">Filter 1 CRC - 16</td><td colspan="4">Filter 0 CRC - 16</td></tr>
<tr><td>Wakeup Frame Filter Register 7</td><td colspan="4">Filter 3 CRC - 16</td><td colspan="4">Filter 2 CRC - 16</td></tr>
</table>

- Filter n Byte mask

This register field defines using which bytes of the frame to determine the received frame is wakeup frame or not by filter n (n=0, 1, 2, 3). Bit 31 must be set to 0. Bit 30 to bit 0 are valid byte mask. If bit m(m means byte number) is set, the filter n offset + m of the receiving frame is calculated by the CRC unit, conversely, filter n offset + m is ignored.

- Filter n command

This four bits command controls the operation of the filter n. The bit 3 of the field is address type selection bit. If this bit is 1, the detection only detects a multicast frame and if this bit is 0, the detection only detects a unicast frame. Bit 2 and bit 1 must be set to 0. Bit 0 is the filter switch bit. Setting it to 1 means enable and 0 means disable.

- Filter n offset

It is used in conjunction with filter n byte mask field. This register specifies offset (within the frame) of the first byte which the filter n uses to check. The minimum allowable value is 12, it represents the byte 13 in the frame (offset value 0 indicates the first byte of the frame).

- Filter n CRC-16

This register field contains the filter comparing CRC-16 code which is used for comparing the calculated CRC-16 from frame data.

Magic packet detection

Another wakeup method is detecting Magic Packet frame (see ‘Magic Packet Technology’, Advanced Micro Devices). A Magic Packet frame is a special frame with formed packet solely intended for wakeup purposes. This packet can be received, analyzed and recognized by the Ethernet block and used to trigger a wakeup event. Setting MPE bit in ENET_MAC_WUM register can enable this function. This type of frame's format is as follows: starts by 6 continuous bytes of the value 0xFF (0xFFFF FFFF FFFF) in anywhere of the frame behind the destination and source address field, then there are 16 duplicate MAC addresses without any interruption and pause. If there is any discontinuity between repeating it 16 times, MAC needs to re-detect 0xFFFF FFFF FFFF in the receive frame. WUM module continuously monitors each frame received. When a Magic Packet frame passing the address filter, MAC will detect its format with Magic Packet format, once the format is matched the WUM will make MAC wakeup from power down state. Then the MAC wakes up from power-down state after receiving a Magic Packet frame. Module also accepts multicast frames as Magic Packet frame.

Example: An example of a Magic Packet with station address 0xAABB CCDD EEFF is the following (MISC indicates miscellaneous additional data bytes in the packet):

<DESTINATION><SOURCE><MISC>

……………………………………………………….FF FF FF FF FF FF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

<MISC><FCS>

Upon detecting a Magic Packet, the MPKR bit in ENET_MAC_WUM register will be set. If the Magic Packet interrupt is enabled, the corresponding interrupt will generate.

Precautions during system power-down state

When the MCU is in Deep-sleep mode, if external interrupt line 19 is enabled, Ethernet WUM module can still detecting frames. Because the MAC in power-down state needs detecting Magic Packet or remote wakeup frame, the REN bit in ENET_MAC_CFG register must be maintained set. The transmit function should be turned disable during the power-down state by clearing the TEN bit in the ENET_MAC_CFG register. Moreover, the Ethernet DMA block should be disabled during the power-down state, because it is not necessary that the Magic Packet or remote wakeup frame is forwarded to the application. Application can disable the Ethernet DMA block by clearing the STE bit and the SRE bit (for the TxDMA and the RxDMA, respectively) in the ENET_DMA_CTL register.

Follow steps are recommended for application to enter and exit power-down state:

1. Wait the current sending frame completes and then reset the TxDMA block by clearing STE bit in ENET_DMA_CTL register.

2. Clear the TEN and REN bit in ENET_MAC_CFG register to disable the MAC's transmit and

receive function.

3. Check the RS bit in ENET_DMA_STAT register, waiting receive DMA read out all the frames in the receive FIFO and then close RxDMA.

4. Configure and enable the external interrupt line 19, so that it can generate an interrupt or event. If EXTI line 19 is configured to generate an interrupt, application still needs to modify ENET_WKUP_IRQ interrupt handling procedures to clear the pending bit of the EXTI line 19.

5. Set the MPEN or WFEN (or both) bit in ENET_MAC_WUM register to enable Magic Packet or Remote Wakeup frame(or both) detection.

6. Setting PWD bit in ENET_MAC_WUM register to enter power-down state.

7. Setting REN bit in ENET_MAC_CFG register to make MAC’s receive function work.

8. Make MCU enter Deep-sleep mode.

9. After received a wakeup type frame, the Ethernet module exits the power-down state.

10. Reading the ENET_MAC_WUM register to clear the power management event flags. Enable MAC’s transmit function and enable TxDMA and RxDMA.

11. Initialize the MCU system clock: enable HXTAL and configure the RCU unit.

22.3.5. Precision time protocol: PTP

The majority of protocols are implemented by the UDP layer application software. The PTP module of the MAC is mainly to recording the transmitting and receiving PTP packets’ precision time and returning it to application.

Specific details about the precise time protocol (PTP) please see the document “IEEE Standard 1588™”.

Reference clock source

System reference time in Ethernet is maintained by a 64-bit register whose high 32-bit indicates ‘second’ time and low 32-bit indicates ‘subsecond’, this is defined in IEEE 1588 specification.

The input PTP reference clock is used to drive the system reference time (also called system time for short) and capture timestamp value for PTP frame. The frequency of this reference clock must be configured no less than the resolution of timestamp counter. The synchronization accuracy between the master node and slave node is around 0.1us.

Synchronization accuracy

The accuracy of time synchronization depends on the following factors:

1) PTP reference clock input period

2) Characteristics of the oscillator (drift)
3) Frequency of the synchronization procedure.

System time correction method

The 64-bit PTP system time update by the PTP input reference clock. The PTP system time is used as the source to record transmission/reception frame's timestamp. The system time initialization and calibration support two methods: coarse method and fine method. The purpose of calibration is to correct the frequency offset.

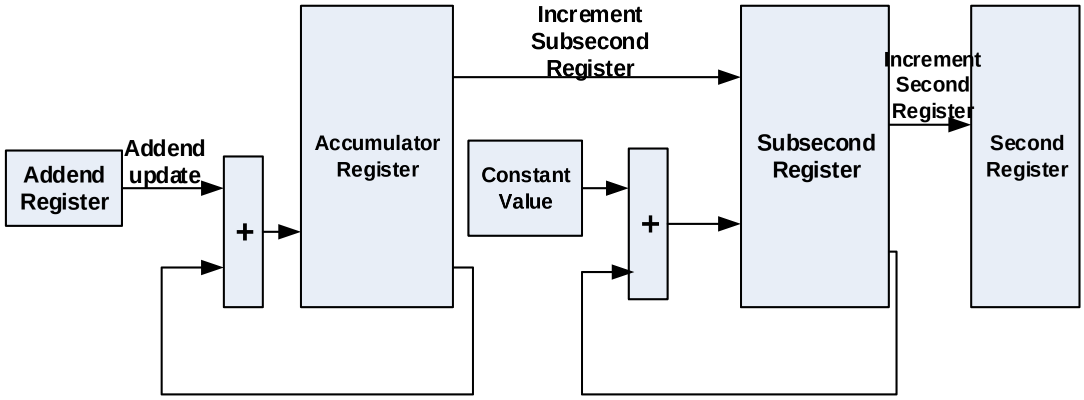
If the coarse correction method is selected, application can configure PTP timestamp update register (ENET_PTP_TSUH and ENET_PTP_TSUL) for system time initialization or correction. If TMSSTI bit is set, PTP timestamp update register is used for initialization and if TMSSTU bit is set, PTP timestamp update register is used for adjust system time by adding or subtracting.

If fine correction method is selected, operation is different. The fine correction method corrects system time not in a single clock cycle. The fine correction frequency can be configured by application to make slave clock frequency smoothly adapt master clock without unpredictability large jitter.

This method is referred to the value of ENET_PTP_TSADDEND added to the accumulator in each HCLK cycle. PTP module will produce a pulse to increase the value of ENET_PTP_TSL register when the accumulator overflowed. The increased value when this pulse occurs is in ENET_PTP_SSINC register.

The following illustration shows the fine correction algorithm process:

Figure 22-7. System time update using the fine correction method



The following concrete example is used to descript the fine correction method how to update the system time:

Assuming the accuracy of the system time update circuit required to achieve 20ns, which means the frequency of update is 50MHz. If the reference clock of HCLK is 75MHz, the frequency ratio is calculated as 75/50, result is 1.5. Hence, the addend (TMSA bit in ENET_PTP_TSADDEND register) value to be set is 2^{32}/1.5, which is equal to 0xAAAA AAAA.

If the reference clock frequency drifts lower, for example, down to 65MHz, the frequency ratio changes to 65/50=1.3, the value to be set in the addend register is 2^{32}/1.30 = 0xC4EC 4EC4. If the reference clock drift higher, for example, up to 85MHz, the value addend register must be 0xA000 0000. Initially, the slave clock frequency is set to Clock Addend Value (0) in the addend register. This value is calculated as above. In addition to configuring the addend counter, application also needs to set subsecond increment register to ensure to achieve the precision of 20ns. The value of the register is to update values of timestamp low 32-bit register after accumulator register overflow. Because the timestamp low register (bit 0 to 30) represents the subsecond value of system time, the precision is 10^9ns/2^{31}=0.46ns. So in order to make the system time accuracy to 20ns, sub second increment register value should be set to 20/0.46 = 0d43.

Note: The algorithm described below based on constant delay transferred between master and slave devices (Master-to-Slave-Delay). Synchronous frequency ratio will be confirmed by the algorithm after a few Sync cycles.

Algorithm is as follows:

- Define the master sends a SYNC message to slave time: MSYNCT (n).
  Define the slave local time SLOCALT (n).
  Define the master local time MLOCALT (n).
  Calculation: MLOCALT (n) = MSYNCT (n) + Master-to-Slave-Delay (n)
- Define the master clock count number between two SYNC message sent: MCLOCKC(n)
  Calculation: MCLOCKC (n) = MLOCALT (n) – MLOCALT (n-1)
  Define the slave clock count number between two received SYNC messages: SCLOCKC (n)
  Calculation: SCLOCKC (n) = SLOCALT (n) - SLOCALT (n-1)
- Define the difference between these two count numbers: DIFFCC (n)
  Calculation: DIFFCC (n) = MCLOCKC (n) - SCLOCKC (n)
- Define the slave clock frequency-adjusting factor: SCFAF (n)
  Calculation: SCFAF (n) = (MCLOCKC (n) + DIFFCC (n)) / SCLOCKC (n)
- Define the Clock Addend Value for addend register: Clock Addend Value (n)
  Clock Addend Value (n) = SCFAF (n) * Clock Addend Value (n-1)

Note: During the actual operation, application may need more than once SYNC message between master and slave to lock.

System time initialization procedure

Setting TMSEN bit in ENET_PTP_TSCTL register to 1, timestamp function is enabled. Each time after this bit is set from reset, application must initialize the timestamp counter at first. Initialization steps as follow:

1. Setting bit 9 in the ENET_MAC_INTMSK register to mask the timestamp trigger interrupt
2. Setting bit 0 in the ENET_PTP_TSCTL register to enable timestamp function
3. Configure the subsecond increment register according to the PTP clock frequency precision
4. If application hopes to use fine correction method, configure the timestamp addend register and set bit 5 in the ENET_PTP_TSCTL register to 1. If application hopes to use coarse correction method, please jump directly to step 7 and step 4-6 can be ignored.
5. Poll the bit 5 in the ENET_PTP_TSCTL register until it is cleared
6. Set bit 1 in the ENET_PTP_TSCTL register to 1 to choose fine correction method
7. Configure the timestamp update high and low register with the value of system time application wants to initialize
8. Send initialization command by setting bit 2 in the ENET_PTP_TSCTL register
9. The timestamp counter starts counting as soon as the initialization process complete

System time update steps under coarse correction method

1. Program the offset (may be negative) value in the timestamp update high and low registers
2. Set bit 3 (TMSSTU) in the ENET_PTP_TSCTL register to update the timestamp register
3. Poll TMSSTU bit until it is cleared.

System time update steps under fine correction method

1. Calculate the value of the desired system clock rate corresponding to the addend register (calculation formula has explained before)
2. Program the addend register, and set the bits 5 in ENET_PTP_TSCTL register
3. Program the target high and low register and reset the bit 9 of the ENET_MAC_INTMSK register to allow time stamp interrupt
4. Set bit 4 (TMSITEN) in ENET_PTP_TSCTL register
5. When an interrupt is generated by this event, read out the value of ENET_MAC_INTF register and clear the corresponding interrupt flag
6. Rewrite the old value of addend register to timestamp addend register and set bit 5 in ENET_PTP_TSCTL register

Transmission and reception of frames with the PTP feature

After enabled the IEEE 1588 (PTP) timestamp function, timestamp is recorded when the frame’s SFD field is outputting from the MAC or the MAC receives a frame’s SFD field. Each transmitted frame can be marked in TxDMA descriptor to indicate whether a timestamp should be captured or not.

Together with the state information of frame, the recorded timestamp value will also be stored in the corresponding transmission/reception descriptor. The 64-bit timestamp information of transmission frame is written back to the transmit descriptor and the 64-bit timestamp information of reception frame is written back to the receive descriptor. See the detailed description in “Transmit DMA descriptor” and “Receive DMA descriptor”.

PTP trigger internal connection with TIMER1

MAC can provide trigger interrupt when the system time is no less than the target time. Using an interrupt imports a known latency and an uncertainty in the command execution time. In order to calculate the time of this known latency part, when the system time is greater than target time, the PTP module sets an output signal. Set bit 29 of AFIO_PCF0 register to 0 can make this signal internally connected to the ITI0 input of TIMER1. For this feature designed, no uncertainty is introduced because the clock of the TIMER1 and PTP reference clock (HCLK) are synchronous.

PTP pulse-per-second (PPS) output signal

Application set bit 30 of AFIO_PCF0 register to 1 to enable the PPS output function. This function can output a signal with the pulse width of 125ms, which can be used to check the synchronization between all nodes in the network. To test the difference between the slave clock and the master clock, both of the slave and master can output PPS and connect them to one oscilloscope for clock measurement.

22.3.6. DMA controller description

Ethernet DMA controller is designed for frame transmission between FIFO and system memory which can reduce the occupation of CPU. Communication between the CPU and the DMA is achieved by the following two kinds of data structures:

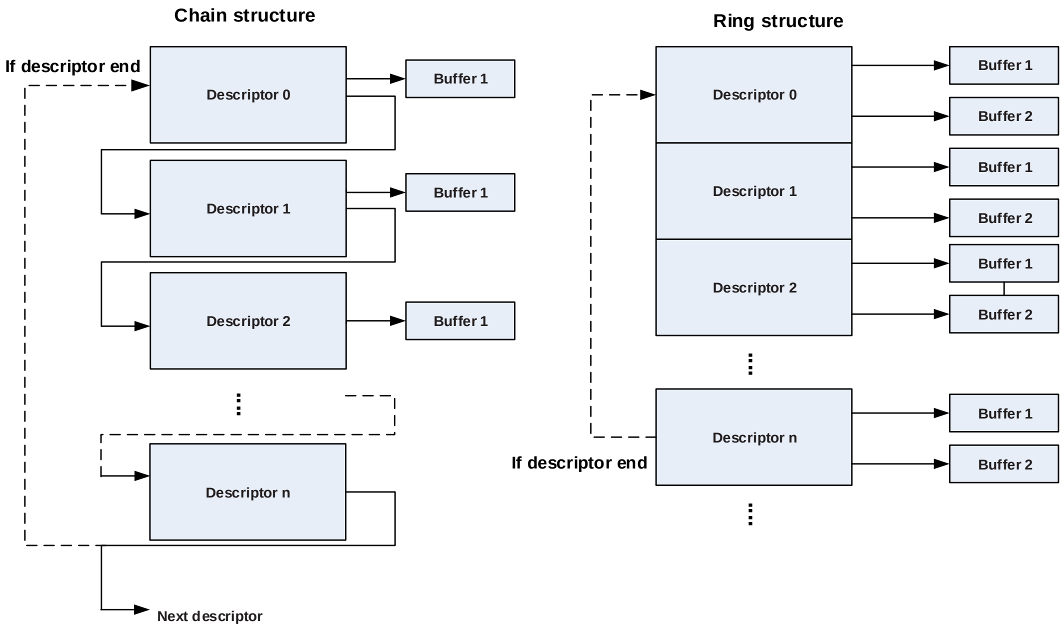
- Descriptor table (ring or chain type) and data buffer
- Control and status register

Applications need to provide the memory for storage of descriptor tables and data buffers. Descriptors that reside in the memory act as pointers to these buffers. Transmission has transmission descriptor and reception has reception descriptor. The base address of each table is stored in ENET_DMA_TDTADDR and ENET_DMA_RDTADDR register. Descriptors of transmission constituted by 4 descriptor word (TDES0-TDES3). Likewise, reception descriptors constituted by 4 descriptor word (RDES0-RDES3). Each descriptor can point to a maximum of two buffers. The value of the buffer 2 can be programmed to the second data address or the next descriptor address which is determined by the configured descriptor table type: Ring or Chain. Buffer space only contains frame data which are located in host's physical memory space. One buffer can store only one frame data but one frame data can be stored in more than one buffer which means one buffer can only store a part of a frame. When chain structure is set, descriptor table is an explicitly one and when ring structure is set, descriptor table is an implicitly one. Explicit chaining of descriptors is accomplished by configuring the second address chained in both receive and transmit descriptors (RDES1[14] and TDES0[20]), at this time RDES2 and TDES2 are stored the data buffer address, RDES3 and TDES3 should be stored the next descriptor address, this connection method of descriptor table is called chain structure. Implicitly chaining of descriptors is accomplished by clearing the RDES1[14] and TDES0 [20], at this time RDES2, TDES2 and RDES3, TDES3 should be all stored the data buffer address, this connection method of descriptor table is called ring structure. When current descriptor's buffer address is used, descriptor pointer will point to the next descriptor. If chain structure is selected, the pointer points to the value of buffer 2. If ring structure is selected, the pointer points to an address calculated as below:

Next descriptor address = Current descriptor address + 16 + DPSL*4

If current descriptor is the last one in descriptor table, application needs to set the bit 21 in TDES0 or bit 15 in RDES1 to inform DMA the current descriptor is the last one of the table in ring structure. At this time, the next descriptor pointer points back to the first descriptor address of the descriptor table. In chain structure, can also set TDES3 or RDES3 value to point back to the first descriptor address of the descriptor table. The DMA skips to the next frame buffer when the end of frame is detected.

Figure 22-8. Descriptor ring and chain structure



Alignment rule for data buffer address

The DMA controller supports all alignment types: byte alignment, half-word alignment and word alignment. This means application can configure the buffer address to any address. But during the operation of the DMA controller, access address is always word align and is different between write and read access. Follow example describes the detail:

Buffer Reading: Assuming the transmit buffer address is 0x2000 0AB2, and 15 bytes need to be transferred. After starting operating, the DMA controller will read five word addresses which are 0x2000 0AB0, 0x2000 0AB4, 0x2000 0AB8, 0x2000 0ABC and 0x2000 0AC0. But when sending data to the FIFO, the first two bytes (0x2000 0AB0 and 0x2000 0AB1) and the last 3 bytes (0x2000 0AC1, 0x2000 0AC2 and 0x2000 0AC3) will be dropped.

Buffer Writing: Assuming the receive buffer address is 0x2000 0CD2, and 16 bytes need to be stored. After starting operating, the DMA controller will write five times 32-bit data from address 0x2000 0CD0 to 0x2000 0CE0. But the first 2 bytes (0x2000 0CD0 and 0x2000 0CD1) and the last 2 bytes (0x2000 0CE2 and 0x2000 0CE3) will be substituted by the virtual bytes.

Note: DMA controller will not write any data out of the defined buffer range.

The effective length of the buffer

For the frame transmitting process, the effective length of the buffer is the same as the value configured by application in TDES1. As mentioned before, a transmitting frame can use one or more descriptors to indicate the frame information which means a frame data can be located in many buffers. When the DMA controller reads a descriptor which the FSG bit in TDES0 is set, it knows the current buffer is pointing to a new frame and the first byte of the frame is included. When the DMA controller reads a descriptor with FSG bit and LSG bit in TDES0 are both reset, it knows the current buffer is pointing to a part of current frame. When the DMA controller reads a descriptor with LSG bit in TDES0 is set, it know the current buffers is pointing to the last part of the current frame. Normally one frame is stored only in one buffer (because buffer size is large enough for a normal frame), so FSG bit and LSG bit are set in the same descriptor.

For the frame receiving process, the receive buffer size must be word align. But for word-align buffer address or not word-align buffer address, the operation is different from transmitting. When the receive buffer address is word align, it’s no difference with transmitting process, the effective length of the buffer is the same as the value configured by application in RDES1. When the receive buffer address is not word align, the effective length of the buffer is less than the value configured by application in RDES1. The effective length of the buffer should be the size value minus the low two bits value of buffer address. For example, assuming the total buffer size is 2048 bytes and buffer address is 0x2000 0001, the low two bits are 0b01, the effective length of the buffer is 2047 bytes whose address range is from 0x20000001 (for the first received frame byte) to 0x2000 07FF.

When a start of frame (SOF) is received, the FSG bit is set by DMA controller and when the end of the frame (EOF) is received, the LSG bit is set. If the receive buffer size is programmed to be large enough to store the whole frame, the FSG and the LSG bit are set in the same descriptor. The actual frame length FRML can be read from RDES0. So application can calculate the left unused buffer space. The RxDMA always uses a new descriptor to receive the start of next frame.

Arbitration for TxDMA and RxDMA controller

There are two types of arbitration method designed for improving the efficiency of DMA controller between transmission and reception: fixed-priority and round-robin. When DAB bit in ENET_DMA_BCTL register is reset, arbiter selects round-robin method. The arbiter allocates the data bus in the ratio set by the RTPR bits in ENET_DMA_BCTL, when both of TxDMA and RxDMA controller request access simultaneously. When DAB bit in ENET_DMA_BCTL register is set, arbiter selects fixed-priority, and the RxDMA controller always has higher priority over the TxDMA.

Error response to DMA controller

During the operation of the DMA controller, when a response error presents on the bus, the DMA controller considers a fatal error occurs and stops operating at once with error flags written to the DMA status register (ENET_DMA_STAT). After such fatal error (response error) occurs, application must reset the Ethernet module and reinitialize the DMA controller.

DMA controller initialization for transmission and reception

Before using the DMA controller, the initialization must be done as follow steps:

1. Set the bus access parameters by writing the ENET_DMA_BCTL register
2. Mask unnecessary interrupt source by configuring the ENET_DMA_INTEN register
3. Program the Tx and Rx descriptor table start address by writing the ENET_DMA_TDTADDR register and the ENET_DMA_RDTADDR register
4. Configure filter option by writing related registers
5. According to the auto-negotiation result with external PHY, set the SPD bit and DPM bit for selecting the communication mode (Half-duplex/Full-duplex) and the communication speed (10Mbit/s or 100Mbit/s). Set the TEN and REN bit in ENET_MAC_CFG register to enable MAC transmit and receive operations.
6. Set STE bit and SRE bit in ENET_DMA_CTL register to enable TxDMA controller and RxDMA controller

Note: If the HCLK frequency is too much low, application can enable RxDMA before set REN bit in ENET_MAC_CFG register to avoid RxFIFO overflow at start time.

TxDMA configuration

Operate on second frame in buffer

When OSF bit in ENET_DMA_CTL is reset, the order of the transmitting is follows: the first is reading transmit descriptor, followed by reading data from memory and writing to FIFO, then sending frame data on interface through MAC and last wait frame data transmitting complete and writing back transmitting status.

Above procedure is TxDMA's standard transmitting procedure but when HCLK is much faster than TX_CLK, the efficiency of transmitting two frames will be greatly reduced.

To avoid the case mentioned above, application can set OSF to 1.If so, the second frame data can be read from the memory and push into FIFO without waiting the first frame's status writing back. OSF function is only performed between two neighboring frames.

TxDMA operation mode (A) (default mode): Non-OSF

The TxDMA controller in Non-OSF mode proceeds as follows:

1. Initialize the frame data into the buffer space and configure the descriptor (TDES0-3) with DAV bit of TDES0 sets to 1
2. Enable TxDMA controller by setting STE bit in ENET_DMA_CTL register
3. The TxDMA controller starts continue polling and performing transmit descriptor. When the DAV bit in TDES0[31] that TxDMA controller read is cleared, or any error condition occurs, the controller will enter suspend state and at the same time both the transmit buffer unavailable bit in ENET_DMA_STAT and normal interrupt summary bit in ENET_DMA_STAT register are set. If entered into suspend state, operation proceeds to Step 8
4. When the DAV bit in TDES0[31] of the acquired descriptor is set, the DMA decodes the transmit frame configured and the data buffer address from the acquired descriptor
5. DMA retrieve data from the memory and push it into the TxFIFO of MAC
6. The TxDMA controller continues polling the descriptor table until the EOF data (LSG bit is set) is transferred. If the LSG bit of current descriptor is reset, it will be closed by resetting the DAV bit after all buffer data pushed into TxFIFO. Then the TxDMA controller waits to write back descriptor status and IEEE 1588 timestamp value if enabled
7. After the whole frame is transferred, the transmit status bit (TS bit in ENET_DMA_STAT register) is set only when INTC bit in TDES0[30] is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame

8. In the suspend state, application can make TxDMA returns to running state by writing any data to ENET_DMA_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process turns to Step 3.

TxDMA operation mode (B): OSF

The TxDMA controller supports transmitting two frames without waiting status write back of the first frame, this mode is called operation on second frame (OSF). When the frequency of system is much faster than the frequency of the MAC interface (10Mbit/s or 100Mbit/s), the OSF mode can improve the sending efficiency. Setting OSF bit in ENET_DMA_CTL register can enable this mode. When the TxDMA controller received EOF of the first frame, it will not enter the state of waiting status write back but to fetch the next descriptor, if the DAV bit and FSG bit of the next descriptor is set, the TxDMA controller immediately read the second frame data an push them into the MAC FIFO.

The TxDMA controller in OSF mode proceeds as follows:

1. Follow steps 1-6 operation in TxDMA default mode

2. The TxDMA controller retrieves the next descriptor without closing the previous frame’s last descriptor in which the LSG bit is set

3. If the DAV bit of the next descriptor is set, the TxDMA controller starts reading the next frame’s data from the buffer address. If the DAV bit of the next descriptor is reset, TxDMA controller enters suspend state and the next operation goes to Step 7.

4. TxDMA controller continues polling descriptor and frame data until the EOF is transferred. If a frame is described with more than one descriptor, the intermediate descriptors are all closed by TxDMA controller after fetched.

5. The TxDMA controller enters the state of waiting for the transmission status and time stamp of the previous frame (if timestamp enabled). With writing back status to descriptor, the DAV bit is also cleared by TxDMA controller

6. After the whole frame is transferred, the transmit status bit (TS bit in ENET_DMA_STAT register) is set only when INTC bit in TDES0[30] is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame if no underflow error occurred in previous frame. If underflow error of the previous frame is occurred, the TxDMA controller enters in suspend state and the next operation goes to Step 7.

7. In suspend state, when the status information and timestamp value (if the function is enable) of the transmitting frame is available, the TxDMA controller writes them back to descriptor and then close it by setting DAV=0 of descriptor.

8. In suspend state, application can make TxDMA returns to running state by writing any data to ENET_DMA_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process goes to Step 1 or Step 2.

Transmit frame format in buffer

According to IEEE 802.3 specification described before, a frame structure is made up of such fields: Preamble, SFD, DA, SA, QTAG (option), LT, DATA, PAD (option), and FCS.

The Preamble and SFD are automatically generated by the MAC, so the application only need store the DA, SA, QTAG(if needed), LT, DATA, DATA, PAD(if needed), FCS(if needed) parts. If the frame needs padding which means PAD and FCS parts are not stored in buffer, then application can configure the MAC to generate the PAD and FCS. If the frame only need FCS which means only FCS part is not stored in buffer, the application can configure the MAC to generate FCS. The DPAD bit and DCRC bit are designed to achieve the generate function of the PAD and FCS field.

Transmit frame processing

As mentioned before, a frame can span over several buffers which means several descriptors. When the FSG bit is set, the descriptor indicates the start of the frame and when the LSG bit is set, the descriptor indicates the end of the frame. All the buffers among these descriptors store the whole frame data. When the last descriptor is fetched and buffer finished reading, the transmitting status will write back to it. The other descriptors (here means the descriptor whose LSG bit is reset) of the current frame will not be changed by TxDMA controller except the DAV bit will be reset to 0. After starting transfer frame data from memory to FIFO, the transmitting has not actually start. The real start time for sending frame on interface is depended on TxDMA mode: Cut-Through mode or Store-and-Forward mode. The former mode starts sending when the byte number of FIFO is greater than configured threshold and the latter mode starts sending when the whole frame data are transferred into FIFO or when the FIFO is almost full.

Suspend during transmit polling

The DMA controller keeps querying the transmit descriptor after the transmission is started. If either of the following conditions happens, the DMA controller will enter suspend state and the transmit polling will stop. Though the DMA entered suspend state, the descriptor pointer is maintained to the descriptor following of the last closed descriptor.

- The DMA controller fetches a descriptor with DAV=0, then it enters suspend state and stops polling. In this case, the NI bit and TBU bit in ENET_DMA_STAT register are set.
- The MAC FIFO is empty during sending a frame on interface which means an error of underflow occurs. In this case, the AI bit and TU bit in ENET_DMA_STAT register are set. Also the transmit error status will write back to transmit descriptor.

Transmit DMA descriptor with IEEE 1588 timestamp format

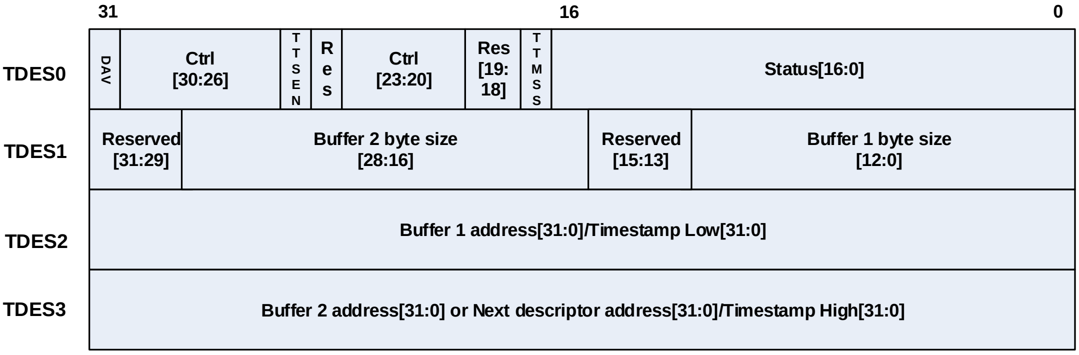
When TTSEN bit is set, the timestamp function is enabled. The TxDMA controller writes transmit timestamp status TTMSS and timestamp back to descriptor TDES2 and TDES3 after the frame transmission complete.

TxDMA descriptors

The TxDMA descriptor structure consists of four 32-bit words: TDES0 ~ TDES3. The descriptions of TDES0, TDES1, TDES2 and TDES3 are given below:

Note: When a frame is described by more than one descriptor, only the control bits of the first descriptor are accept by TxDMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

Figure 22-9. Transmit descriptor



TDES0	DAV	Ctrl [30:26]	TTSEN	Res	Ctrl [23:20]	Res [19: 18]	TTMSS	Status[16:0]
TDES1	Reserved [31:29]	Buffer 2 byte size [28:16]	Reserved [15:13]	Buffer 1 byte size [12:0]				
TDES2	Buffer 1 address[31:0]/Timestamp Low[31:0]							
TDES3	Buffer 2 address[31:0] or Next descriptor address[31:0]/Timestamp High[31:0]							

- TDES0: Transmit descriptor word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAV	INTC	LSG	FSG	DCRC	DPAD	TTSEN	Reserved	CM[1:0]		TERM	TCHM	Reserved		TTMSS	IPHE
rw	rw	rw	rw	rw	rw	rw		rw		rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FRMF	IPPE	LCA	NCA	LCO	ECO	VFRM	COCNT[3:0]				EXD	UFE	DB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits	Fields	Descriptions
31	DAV	DAV bit The DMA clears this bit either when it completes the frame transmission or the buffer allocated in the descriptor is read completely. This bit of the frame’s first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. 0: The descriptor is available for CPU not for DMA 1: The descriptor is available for DMA not for CPU
30	INTC	Interrupt on completion bit This is valid only when the last segment (TDES0[29]) is set. 0: TS bit in ENET_DMA_STAT is not set when frame transmission complete. 1: TS bit in ENET_DMA_STAT is set when frame transmission complete.
29	LSG	Last segment bit This bit indicates that the buffer contains the last segment of the frame. 0: The buffer of descriptor is not stored the last part of frame

		1: The buffer of descriptor is stored the last part of frame
28	FSG	First segment bit This bit indicates that the buffer contains the first segment of a frame. 0: The buffer of descriptor is not stored the first block of frame 1: The buffer of descriptor is stored the first block of frame
27	DCRC	Disable CRC bit This is valid only when the first segment (TDES0[28]) is set. 0: The MAC automatic append a CRC to the end of the transmitted frame 1: The MAC does not append a CRC to the end of the transmitted frame
26	DPAD	Disable adding pad bit This is valid only when the first segment (TDES0[28]) is set. 0: The DMA automatically adds padding byte and CRC to a frame shorter than 64 bytes. Only the padding actually acts, the CRC is also appended. The DCRC bit is don’t care. 1:The MAC does not automatically add padding to a frame
25	TTSEN	Transmit timestamp function enable bit. This field is only valid when the First segment control bit (TDES0[28]) is set. 0: Disable transmit timestamp function 1: When TMSEN is set (ENET_PTP_TSCTL bit 0), IEEE 1588 hardware time stamping is activated for the transmit frame
24	Reserved	Must be kept at reset value
23:22	CM[1:0]	Checksum mode bits 0x0: Disabled checksum insertion function 0x1: Only enable function for IP header checksum calculation and insertion 0x2: Enable IP header checksum and payload checksum calculation and insertion, pseudo-header checksum is not calculated in hardware 0x3: Enable IP Header checksum and payload checksum calculation and insertion, pseudo-header checksum is calculated in hardware.
21	TERM	Transmit end for ring mode bit This bit is used only in ring mode and has higher priority than TCHM 0: The current descriptor is not the last descriptor in the table 1: The descriptor table reached its final descriptor. The DMA descriptor pointer returns to the start address of the table.
20	TCHM	The second address chained mode bit This bit is used only in chain mode. When this bit, TCHM (TDES0[20]), is set, TB2S (TDES1[28:16]) is don’t care. 0: The second address in the descriptor is the second buffer address 1:The second address in the descriptor is the next descriptor address

19:18	Reserved	Must be kept at reset value
17	TTMSS	Transmit timestamp status bit This bit is only valid when the descriptor’s last segment (LSG) control bit (TDES0[29]) is set. 0: Timestamp was not captured 1:A timestamp was captured for the described transmit frame and push into TDES2 and TDES3.
16	IPHE	IP header error bit IP header error occurs when any case of below happen: IPv4 frames: 1) The header length field has a value less than 0x5. 2) The header length field value in transmitting IPv4 frame is mismatch with the number of header bytes 3) The version field value does not match the length/type field value IPv6 frames: 1) The main header length is not 40 bytes 2) The version field value does not match the length/type field value 0:The MAC transmitter did not detect error in the IP datagram header 1:The MAC transmitter detected an error in the IP datagram header
15	ES	Error summary bit Following bits are logical ORed to generate this bit: TDES0[16]: IP header error TDES0[14]: Jabber timeout TDES0[13]: Frame flush TDES0[12]: IP payload error TDES0[11]: Loss of carrier TDES0[10]: No carrier TDES0[9]: Late collision TDES0[8]: Excessive collision TDES0[2]:Excessive deferral TDES0[1]: Underflow error
14	JT	Jabber timeout bit Only set when the JBD bit is reset 0:No jabber timeout occurred 1:The MAC transmitter has experienced a jabber timeout
13	FRMF	Frame flushed bit This bit is set to flush the Tx frame by software
12	IPPE	IP payload error bit The transmitter checks the payload length received in the IPv4 or IPv6

		header against the actual number of TCP, UDP or ICMP packet bytes received from the application and issues an error status in case of a mismatch 0:No IP payload error occurred 1: MAC transmitter detected an error in the TCP, UDP, or ICMPIP datagram payload
11	LCA	Loss of carrier bit When the interface signal 'CRS' lost one or more cycles and no collision happened during transmitting, the loss of carrier condition occurs. This is valid only in Half-duplex mode. 0:No loss of carrier occurred 1:A loss of carrier occurred during frame transmission
10	NCA	No carrier bit 0: PHY carrier sense signal is active 1: The carrier sense signal from the PHY was not asserted during transmission
9	LCO	Late collision bit If a collision occurs when 64 bytes (including preamble and SFD) has already transferred, this situation called late collision. 0: No late collision occurred 1: Late collision situation occurred **Note:** This bit is not valid if the UFE bit is set
8	ECO	Excessive collision bit If the RTD=1 (retry function disable), this bit is set after the first collision. If the RTD=0 (retry function enable), this bit is set when failed 16 successive retry transmitting. When this bit is set, the transmission of current frame is aborted. 0: No excessive collision occurred 1: Excessive collision occurred
7	VFRM	VLAN frame bit 0:The transmitted frame was a normal frame 1: The transmitted frame was a VLAN-type frame
6:3	COCNT[3:0]	Collision count bits This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the ECO bit (TDES0[8]) is set
2	EXD	Excessive deferral bit This is valid when the DFC bit in the MAC configuration register is set 0: No excessive deferral occurred 1: The transmission has ended because of excessive deferral time is over 3036 bytes
1	UFE	Underflow error bit

		This bit indicates that the TxDMA comes across an empty TxFIFO while transmitting the frame before EOF which is caused by pushing data to TxFIFO late from memory. The transmission process enters the suspend state and sets both the TU (bit 5) and the TS (bit 0) in ENET_DMA_STAT 0: No underflow error occurred 1: Underflow error occurred and the MAC aborted the frame transmitting
0	DB	Deferred bit This bit indicates whether the transmitting frame is deferred because of interface signal CRS is active before MAC transmit frame. Valid only in Half-duplex mode 0:No transmission deferred 1:The MAC is deferred before transmission

- TDES1: Transmit descriptor word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			TB2S[12:0]												
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			TB1S[12:0]												
			rw												

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:16	TB2S[12:0]	Transmit buffer 2 size bits These bits indicate byte size of the second data buffer. This field is not valid if the TCHM bit (TDES0[20]) is set.
15:13	Reserved	Must be kept at reset value
12:0	TB1S[12:0]	Transmit buffer 1 size bits These bits indicate the byte size of the first data buffer. If this field is 0, the TxDMA ignores this buffer and uses buffer 2 (for TCHM=0) or the next descriptor (for TCHM=1).

- TDES2: Transmit descriptor word 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TB1AP/TTSL[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB1AP/TTSL[15:0]															
rw															

Bits	Fields	Descriptions
31:0	TB1AP/TTSL[31:0]	Transmit buffer 1 address pointer/Transmit frame timestamp low 32-bit value bits Before transmitting frame, application must configure these bits for transmit buffer 1 address (TB1AP). When the transmitting frame is complete, these bits can be changed to the timestamp low 32-bit value (TTSL) for transmitting frame. When these bits stand for buffer 1 address (TB1AP), the alignment is no limitation. When these bits stand for timestamp low 32-bit value, the TTSEN and LSG bit of current descriptor must be set.

- TDES3: Transmit descriptor word 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TB2AP/TTSH[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB2AP/TTSH[15:0]															
rw															

Bits	Fields	Descriptions
31:0	TB2AP/TTSH[31:0]	Transmit buffer 2 address pointer (or next descriptor address) / Transmit frame timestamp high 32-bit value bits Before transmitting frame, application must configure these bits for transmit buffer 2 address (TB2AP) or the next descriptor address which is decided by descriptor type is ring or chain. When the transmitting frame is complete, these bits can be changed to the timestamp high 32-bit value (TTSH) for transmitting frame TTSEN =1. When these bits stand for buffer 2 address (TCHM=0), the alignment is no limitation. When these bits stand for the next descriptor address (TCHM=1), these bits must be word-alignment. When these bits stand for timestamp high 32-bit value, the TTSEN and LSG bit of current descriptor must be set.

RxDMA configuration

The receiving process of the RxDMA controller is described detailed as below:

1. Applications initialize the receive descriptors with the DAV bit (RXDES0[31]) is set
2. Setting the SRE bit in ENET_DMA_CTL register to make RxDMA controller entering running state. In running state, the RxDMA controller continually fetching the receive descriptors from descriptor table whose starting address is configured in ENET_DMA_RDTADDR register by application. If the DAV bit of the fetched receive descriptor is set, then this descriptor is used for receiving frame. But if the DAV bit is reset which means this receive descriptor cannot be used by RxDMA, the RxDMA controller will enter suspend state and operation goes to Step 9

3. From the valid receive descriptor (DAV=1), the RxDMA controller marks the receiving control bit and data buffer address

4. Processing the received frames and transfer data to the receive buffer from the RxFIFO.

5. If all frame data has completely transferred or the buffer is full, the RxDMA controller fetches the next descriptor from receive descriptor table.

6. If the current receiving frame transfer is complete, the operation of RxDMA goes to Step7. But if not complete, two conditions may occur:

   1) The next descriptor’s DAV bit is reset. The RxDMA controller sets descriptor error bit DERR in RDES0 if flushing function is enabled. The RxDMA controller closes current descriptor by resetting DAV bit and sets the LSG bit (if flushing is enabled) or resets the LSG bit (if flushing is disabled). Then the operation goes to Step 8.

   2) The next descriptor’s DAV bit is set. The RxDMA controller closes current descriptor by resetting DAV bit and operation goes to Step 4.

7. If IEEE 1588 time stamping function is enabled, the RxDMA controller writes the time stamp value (if receiving frame meets the configured time stamping condition) to the current descriptor’s RDES2 and RDES3. At the same time (writing timestamp value) the RxDMA controller also writes the received frame’s status word to the RDES0 with the DAV bit cleared and the LSG bit set.

8. The latest descriptor is fetched by RxDMA controller. If the fetched descriptor bit 31 (DAV) is set, the RxDMA controller operation goes to Step 4. If the fetched descriptor bit 31 is reset, the RxDMA controller enters the suspend state and sets the RBU bit in register ENET_DMA_STAT. If flushing function is enabled, the RxDMA controller will flush the received frame data in the RxFIFO before entering suspend state.

9. In suspended state, there are two conditions to exit. The first is writing data in the ENET_DMA_RPEN register by application. The second is when a new received frame is available which means the byte number of receiving frame is greater than threshold in Cut-Through mode or when the whole frame is received in Store-and-Forward mode. Once exiting suspend mode, the RxDMA controller fetches the next descriptor and the following operation goes to Step 2.

Receive descriptor fetching regulation

Descriptor fetching occurs if any one or more of the following conditions are met:

- The time SRE bit is configured from 0 to 1 which makes the RxDMA controller entering running state
- The total buffer size (buffer 1 for chain mode or buffer 1 plus buffer 2 for ring mode) of the current descriptor cannot hold the current receiving frame. In other word, the last byte stored in buffer space is not the EOF byte
- After a complete frame is transferred to buffer and before current descriptor is closed

- In suspend state, the MAC received a new frame
- Writing any value to receive poll enable register ENET_DMA_RPEN

Process of receiving frame

When a frame is presented on the interface, the MAC starts to receive it. At the same time, the address filter block is running for this received frame. If the received frame fails the address filtering it will be discarded from RxFIFO in MAC and not be forwarded to buffer by RxDMA controller. If the received frame passes the address filtering, it will be forwarded to buffer when the available time comes. If the RxDMA controller is configured in Cut-Through mode, the available time means the byte number of the received frame is equal or greater than the configured threshold. If the RxDMA controller is configured in Store-and-Forward mode, the available time means the complete frame is stored in RxFIFO. During receiving frame, if any one of the below cases occurs the MAC can discard the received frame data in RxFIFO and the RxDMA controller will not forward these data: 1) The received frame bytes is less than 64. 2) Collision occurred during frame receiving. 3) The premature termination for the receiving frame.

When the available time comes, the RxDMA controller starts transfer frame data from RxFIFO to the receive buffer. If the SOF is included in current receive buffer, the FDES bit in RDES0 is set when the RxDMA controller writing receive frame status to indicate this descriptor is used for storing the first part of the frame. If the EOF is included in current receive buffer, the LDES bit in RDES0 is set when RxDMA controller writing receive frame status to indicate this descriptor is used for storing the last part of the frame. Often when the buffer size is larger than received frame, the FDES and LDES bit are set in the same descriptor. When the EOF is transferred to buffer or the receive buffer space is exhausted, the RxDMA controller fetches the next receive descriptor and closes previous descriptor by writing RDES0 with DAV=0. If the LDES bit is set, the other status are also be updated and the RS bit in ENET_DMA_STAT register will be set when DINTC=0, when DINTC=1 the RS bit will not be set. If the DAV bit of the next descriptor is set, the RxDMA controller repeats above operation when received a new frame. If the DAV bit of the next descriptor is reset, the RxDMA controller enters suspend state and sets RBU bit in ENET_DMA_STAT register. The pointer value of descriptor address table is retained and be used for the starting descriptor address after exiting suspend state.

Processing after a new frame received in suspend state

When a new frame is available (see available definition in the previous paragraph), the RxDMA controller fetches the descriptor. If the DAV bit in RDES0 is set, the RxDMA controller exits suspend state and returns to running state for frame reception. But if the DAV bit in RDES0 is reset, application can choose whether these received frame data in RxFIFO are flushed or not by configuring DAFRF bit in ENET_DMA_CTL register. If DAFRF=0, the RxDMA controller discards these received frame data and makes the missed frame counter (MSFC) increase one. If DAFRF=1, these frame data are will not be flushed and MSFC counter will not increase until the RxFIFO is full. If the DAV bit is reset in fetched descriptor, the RBU bit in ENET_DMA_STAT register will be set and the RxDMA controller will be still in suspend state.

Receive DMA descriptor with IEEE 1588 timestamp format

If the IEEE 1588 function enabled, the MAC writes the timestamp value to RDES2 and RDES3 after a frame with timestamp reception complete and before the RxDMA controller clears the DAV bit.

RxDMA descriptors

In normal descriptor mode, the descriptor structure consists of four 32-bit words: RDES0 ~ RDES3. The detailed description of RDES0, RDES1, RDES2 and RDES3 are given below.

Figure 22-10. Receive descriptor

	31		16			0
RDES0	DAV	Status[30:0]				
RDES1	CRTL	Res [30:29]	Buffer 2 byte size [28:16]	CTRL [15:14]	Res	Buffer 1 byte size [12:0]
RDES2	Buffer 1 address[31:0]/Timestamp Low[31:0]					
RDES3	Buffer 2 address[31:0] or Next descriptor address[31:0]/Timestamp High[31:0]					

- RDES0: Receive descriptor word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAV	DAFF	FRML[13:0]													
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRS	DERR	SAFF	LERR	OERR	VTAG	FDES	LDES	IPHERR	LCO	FRMT	RWDT	RERR	DBERR	CERR	PCERR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	DAV	Descriptor available bit This bit indicates the DMA controller can use this descriptor. The DMA clears this bit either when it completes the frame reception or when the buffers in this descriptor are full 0: The descriptor is owned by the CPU 1: The descriptor is owned by the DMA
30	DAFF	Destination address filter fail bit 0: A frame passed the destination address filter 1: A frame failed the destination address filter

29:16	FRML[13:0]	Frame length bits These bits indicate the byte length of the received frame that was transferred to the buffer (including CRC). This field is valid only when LDES=1 (RDES0[8]) and DERR=0 (RDES0[14]). If LDES=0 and ERRS=0, these bits indicate the accumulated number of bytes that have been transferred for the current frame.
15	ERRS	Error summary bit This field is valid only when the LDES (RDES0[8]) is set. This bit is logical ORed by the following bits: RDES0[14]: Descriptor error. RDES0[11]: Overflow error RDES0[6]: Late collision RDES0[4]: Watchdog timeout RDES0[3]: Receive error RDES0[1]: CRC error REDS0[7] = 0, REDS0[5] = 1 and REDS0[0] = 1: payload checksum error REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 0: header checksum error REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 1: both header and payload checksum errors
14	DERR	Descriptor error bit This field is valid only when the LDES (RDES0[8]) is set. When the current buffer cannot hold current received frame and the next descriptor’s DAV bit is reset, the descriptor error occurs. 0: No descriptor error occurred 1: Descriptor error occurred
13	SAFF	SA filtering fail bit 0: No source address filter fail occurred 1: A received frame failed the SA filter
12	LERR	Length error bit This bit is valid only when the FRMT (RDES0[5]) bit is reset. This bit indicates the mismatch between the length field in received and the actual frame length. 0: No length error occurred 1: Length error occurred
11	OERR	Overflow error bit When RxFIFO is overflow and the frame data has been partly forwarded to descriptor buffer, the overflow error bit sets. 0: No overflow error occurred 1: RxFIFO overflowed and frame data is not valid
10	VTAG	VLAN tag bit 0: Received frame is not a tag frame

		1: Received frame is a tag frame
9	FDES	First descriptor bit This bit indicates that current descriptor contains the SOF of the received frame. 0: The current descriptor does not store the SOF of the received frame 1: The current descriptor buffer saves the SOF of the received frame
8	LDES	Last descriptor bit This bit indicates that current descriptor contains the EOF of the received frame 0: The current descriptor buffer does not store EOF of the received frame 1: The current descriptor buffer saves the EOF of the received frame
7	IPHERR	IP frame header checksum error bit This error can be due to inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes. 0: No IPv header checksum error occurred 1: An error in the IPv4 or IPv6 header
6	LCO	Late collision bit This bit indicates a collision occurs after 64 bytes have been received This bit only valid in Half-duplex mode. 0: No late collision occurred 1: Late collision has occurred
5	FRMT	Frame type bit This bit is not valid for Runt frames less than 14 bytes. 0: The received frame is an IEEE802.3 frame 1: The receive frame is an Ethernet-type frame (the LT field is greater than or equal to 0x0600)
4	RWDT	Receive watchdog timeout bit When WDD=0, this bit indicates a frame with more than 2048 bytes was detected. When WDD=1, this bit indicates a frame with more than 16384 bytes was detected. 0: No receive watchdog timeout occurred 1: Watchdog timer overflowed during receiving and current frame is only a part of frame.
3	RERR	Receive error bit This bit indicates the interface signal RX_ER asserted when RX_DV signal is active during frame receiving process. 0: No receive error occurred 1: Receive error occurred
2	DBERR	Dribble bit error bit This bit is valid only in MII interface mode and indicates there is an incomplete byte (odd cycles during reception) received.

		0: No dribble bit error occurred 1: Dribble bit error occurred
1	CERR	CRC error bit This bit is valid only when the LDES (RDES0[8]) is set and indicates FCS field in received frame is mismatch with the calculation result of the hardware 0: No CRC error occurred 1: A CRC error occurred
0	PCERR	Payload checksum error bit 0: No payload checksum error occurred 1: The TCP, UDP or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP or ICMP segment's Checksum field or when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame.

The following table shows the combination meaning for bit 7, 5, and 0 in RDES0:

Table 22-6. Error status decoding in RDES0, only used for normal descriptor

Bit 7: IPHERR	Bit 5: FRMT	Bit 0: PCERR	Frame status
0	0	0	IEEE 802.3 normal frame (Length field value is less than 0x0600 and not tagged)
0	0	1	IPv4 or IPv6 frame, no header checksum error, payload checksum is bypassed because of unsupported payload type
0	1	0	IPv4 or IPv6 frame, checksum checking pass
0	1	1	IPv4 or IPv6 frame, payload checksum error. This error may cased by following condition: 1) Calculated checksum value mismatch the checksum field 2) byte number of received payload mismatch length field
1	0	0	Reserved
1	0	1	A type (length/type field equal or greater than 0x0600) or tagged frame but neither IPv4 nor IPv6. Offload check engine is bypassed.
1	1	0	IPv4 or IPv6 frame, but an header checksum error detected This error may cased by following condition: 1) Type value inconsistent with version value 2) Calculated header checksum mismatch the header checksum field 3) Expected IP header bytes is not received enough
1	1	1	IPv4 or IPv6 frame, both header and payload checksum detected errors

- RDES1: Receive descriptor word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

DINTC	Reserved		RB2S[12:0]												
rw			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RERM	RCHM	Reserved	RB1S[12:0]												
rw	rw		rw												

Bits	Fields	Descriptions
31	DINTC	Disable interrupt on completion bit 0: RS bit in ENET_DMA_STAT register will immediately set after receiving the completed, then if enabled the corresponding interrupt, the interrupt will trigger. 1: RS bit in ENET_DMA_STAT register will not be set after receiving the completed, so the corresponding interrupt will not be triggered.
30:29	Reserved	Must be kept at reset value
28:16	RB2S[12:0]	Receive buffer 2 size bits The second buffer size in bytes. The buffer size must be a multiple of 4. This field is ignored if RCHM (RDES1[14]) is set
15	RERM	Receive end of ring mode bit This bit indicates the final descriptor in table is arrived and the next descriptor address is automatically set to the configured start descriptor address. 0: Current descriptor is not the last descriptor in table 1: Current descriptor is the last descriptor in table
14	RCHM	Receive chained mode for second address bit 0: The second address points to the second buffer address. 1: The second address points to the next descriptor address. RB2S (RDES1[28:16]) is ignored. **Note:** If the RERM=1, the next descriptor returns to base address even this bit is set to 1.
13	Reserved	Must be kept at reset value
12:0	RB1S[12:0]	Receive buffer 1 size bits The first buffer size in bytes. The buffer size must be a multiple of 4. If this field is 0, the RxDMA controller ignores this buffer and uses buffer 2 (RCHM=0) or the next descriptor (RCHM=1)

- RDES2: Receive descriptor word 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RB1AP/RTSL[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RB1AP/RTSL[15:0]
rw

Bits	Fields	Descriptions
31:0	RB1AP/RTSL[31:0]	Receive buffer 1 address pointer / Receive frame timestamp low 32-bit These bits are designed for two different functions: buffer address pointer (RB1AP) or timestamp low 32-bit value (RTSL). **RB1AP**: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 1 address by application. This buffer 1 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0. The buffer address alignment has no limitation. **RTSL**: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition. If the received frame does not meet the snapshoot condition, these bits will keep RB1AP value.

- RDES3: Receive descriptor word 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RB2AP/RTSH[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB2AP/RTSH[15:0]															
rw															

Bits	Fields	Descriptions
31:0	RB2AP/RTSH[31:0]	Receive buffer 2 address pointer (next descriptor address) / Receive frame timestamp high 32-bit value bits These bits are designed for two different functions: buffer address pointer or next descriptor address (RB1AP) or timestamp high 32-bit value (RTSH). **RB2AP**: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 2 address (RCHM=0) or the next descriptor address (RCHM=1) by application. This buffer 2 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0 when RCHM=0. If RCHM=1 and RERM=0, this address pointer is used for fetching the next descriptor. If RCHM=1 and RERM=1, these bits are ignored. When this address is used for next descriptor address, the word alignment is needed. The other conditions have no limitation for these bits. **RTSH**: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition. If the received frame does not meet the snapshoot condition, these bits will keep RB2AP value.

22.3.7. Example for a typical configuration flow of Ethernet

After power-on reset or system reset, the following operation flow is a typical process for application to configure and run Ethernet:

- **Enable Ethernet clock.**

  Program the RCU module to enable the HCLK and Ethernet Tx/Rx clock.

- **Setup the communication interface.**

  Configure AFIO_PCF0 to define which interface mode is selected (MII or RMII). Configure GPIO module to make selected PADs to alternate function.

- **Wait the resetting complete**

  Polling the ENET_DMA_BCTL register until the SWR bit is reset. (SWR bit is set by default after power-on reset or system reset)

- **Obtain and configure the parameters in PHY register**

  According to the frequency of HCLK, configure the SMI clock frequency and access external PHY register to obtain the information of PHY (e.g. support Half/Full duplex or not, support 10M/100Mbit speed or not, and so on). Based on supported mode of external PHY, configure ENET_MAC_CFG register consistent with PHY register.

- **Initialize the DMA in Ethernet module for transaction**

  Configure the ENET_DMA_BCTL, ENET_DMA_RDTADDR, ENET_DMA_TDTADDR, ENET_DMA_CTL registers to initialize the DMA module. (Detailed information refer to ***DMA controller description***)

- **Initialize the physical memory space for descriptor table and data buffer**

  According to the address value in ENET_DMA_RDTADDR and ENET_DMA_TDTADDR register, program transmitting and receiving descriptors (with DAV=1) and data buffer.

- **Enable MAC and DMA module to start transmit and receive**

  Set TEN and REN bit in ENET_MAC_CFG register to make MAC work for transmit and receive. Set STE and SRE bit in ENET_DMA_CTL register to make DMA controller work for transmit and receive.

- **If transmitting frames is needed**

  1) Choose one or more programmed transmitting descriptor, write the transmit frame data into buffer address which is decided in TDES.

  2) Set the DAV bit in these one or more transmit frame descriptor.

  3) Write any value in ENET_DMA_TPEN register to make TxDMA exit suspend state and start transmitting

4) There are two methods for application to confirm whether current transmitting frame is complete or not. The first method is that application can poll the DAV bit of current transmit descriptor until it is reset, this means the transmitting is complete. The second method can be used only when INTC=1. Application can poll the TS bit in ENET_DMA_STAT register until it is set, this means the transmitting is complete.

- **If receiving frames is enabled**

1) Check the first receive descriptor in descriptor table (whose address is configured in ENET_DMA_RDTADDR register).

2) If DAV bit in RDES0 is reset, then the descriptor is used and receive buffer space has stored the receive frame.

3) Handling this receive frame data.

4) Set DAV bit of this descriptor to release this descriptor for new frame receiving.

5) Check next descriptor in table, then goes to Step 2.

22.3.8. Ethernet interrupts

There are two interrupt vectors in Ethernet module. The first interrupt vector is made up of normal operation interrupts and the second vector is made up of WUM events for wakeup which is mapped to the EXTI line 19.

All of the MAC and DMA controller interrupt are connected to the first interrupt vector. The description for the MAC interrupt and DMA controller interrupt are showed behind.

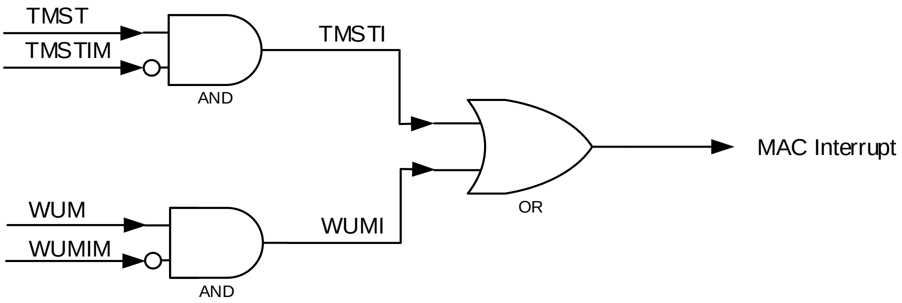
The WUM block event is connected to the second interrupt vector. The event can be remote wakeup frame received event or/and Magic Packet wakeup frame received event. This interrupt is inner mapped on the EXTI line 19. So, if the EXTI line 19 is enabled and configured to trigger by rising edge, the Ethernet WUM event can make the system exiting Deep-sleep mode after a WUM event occurred. In addition, if the WUM interrupt is not masked, both the EXTI line 19 interrupt and Ethernet normal interrupt to CPU are both generated.

Note: Because of the WUM registers are designed in RX_CLK domain, clear these registers by reading them will need a long time delay (depends on the frequency disparity between HCLK and RX_CLK). To avoid entering the same event interrupt twice, it’s strongly recommended that application polls the WUFR and MPKR bit until they reset to zero during the interrupt service routine.

MAC interrupts

All of the MAC events can be read from ENET_MAC_INTF and each of them has a mask bit for masking corresponding interrupt. The MAC interrupt is logical ORed of all interrupts.

Figure 22-11. MAC interrupt scheme



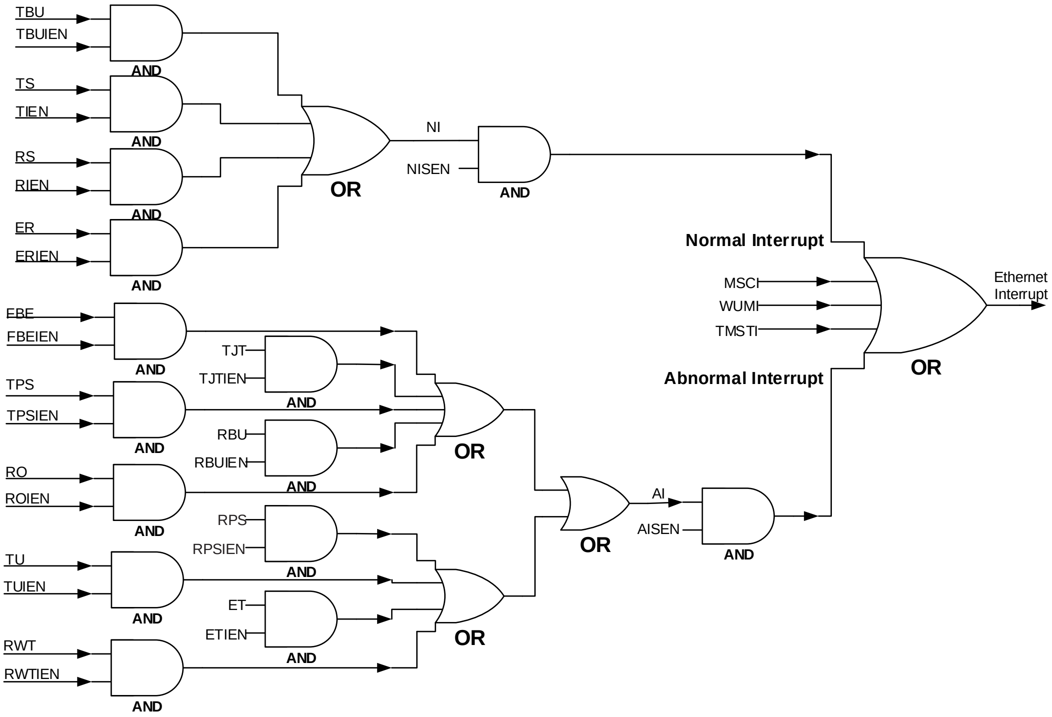
DMA controller interrupts

The DMA controller has two types of event: Normal and Abnormal.

No matter what type the event is, it has an enable bit (just like mask bit) to control the generating interrupt or not. Each event can be cleared by writing 1 to it. When all of the events are cleared or all of the event enable bits are cleared, the corresponding summary interrupt bit is cleared. If both normal and abnormal interrupts are cleared, the DMA interrupt will be cleared.

Below block diagram illustrates the Ethernet module interrupt connection:

Figure 22-12. Ethernet interrupt scheme



22.4. Register definition

Byte (8-bit) access, half word (16-bit) access and word (32-bit) access are all supported for application.

22.4.1. MAC configuration register (ENET_MAC_CFG)

Address offset: 0x0000

Reset value: 0x0000 8000

This register configures the operation mode of the MAC. It also configures the MAC receiver and MAC transmitter operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								WDD	JBD	Reserved		IGBS[2:0]			CSD
								rw	rw			rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SPD	ROD	LBM	DPM	IPFCO	RTD	Reserved	APCD	BOL[1:0]		DFC	TEN	REN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw		

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	WDD	Watchdog disable bit This bit indicates the maximum bytes for receiving, data beyond this will be cut off. 0: The MAC allows no more than 2048 bytes of the frame being received 1: The MAC disables the watchdog timer on the receiver, and can receive frames of up to 16384 bytes
22	JBD	Jabber disable bit This bit indicates the maximum bytes for transmitting data, data beyond this will be cut off. 0: The maximum transmission byte is 2048 1: The maximum transmission byte can be 16384
21:20	Reserved	Must be kept at reset value
19:17	IGBS[2:0]	Inter frame gap bit selection bits These bits can select the minimum inter frame gap bit time between two neighboring frames during transmission. 0x0: 96 bit times 0x1: 88 bit times 0x2: 80 bit times 0x3: 72 bit times 0x4: 64 bit times

		0x5: 56 bit times(For Half-duplex, must be reserved) 0x6: 48 bit times(For Half-duplex, must be reserved) 0x7: 40 bit times(For Half-duplex, must be reserved)
16	CSD	Carrier sense disable bit 0: The MAC transmitter generates carrier sense error and aborts the transmission 1: The MAC transmitter ignores the MII CRS signal during frame transmission in Half-duplex mode. Loss of carrier error and no carrier error will not be generated.
15	Reserved	Must be kept at reset value
14	SPD	Fast Ethernet speed bit Indicates the speed in Fast Ethernet mode: 0: 10 Mbit/s 1: 100 Mbit/s
13	ROD	Receive own disable bit This bit is not applicable if the MAC is operating in Full-duplex mode 0: The MAC receives all packets that are given by the PHY while transmitting 1: The MAC disables the reception of frames in Half-duplex mode
12	LBM	Loopback mode bit 0: The MAC operates in normal mode 1: The MAC operates in loopback mode at the MII.
11	DPM	Duplex mode bit 0: Half-duplex mode enable 1: Full-duplex mode enable
10	IPFCO	IP frame checksum offload bit 0: The checksum offload function in the receiver is disabled 1: IP frame checksum offload function enabled for received IP frame
9	RTD	Retry disable bit This bit is applicable only in the Half-duplex mode 0: The MAC attempts retries up to 16 times based on the settings of BOL 1: The MAC attempts only 1 transmission.
8	Reserved	Must be kept at reset value
7	APCD	Automatic pad/CRC drop bit This bit only valid for a non tagged frame and its length field value is equal or less than 1536. 0: The MAC forwards all received frames without modify it 1: The MAC strips the Pad/FCS field on received frames
6:5	BOL[1:0]	Back-off limit bits When a collision occurred, the MAC needs to retry sending current frame after delay some time. The base time unit for this delay time (dt) called slot time which means

		1 slot time is equal to 512 bit times. This delay time (dt) is a random integer number calculated by following formula : 0⩽dt <2^k 0x0: k = min (n, 10) 0x1: k = min (n, 8) 0x2: k = min (n, 4) 0x3: k = min (n, 1), n = number of times for retransmission attempt **Note:** This bit is valid only in Half-duplex mode
4	DFC	Deferral check bit 0: The deferral check function is disabled. MAC defers sending until the CRS goes inactive. 1: The deferral check function is enabled in the MAC. If deferred more than 24288 bit times, excessive deferral error occurs and MAC abort transmitting frame. If CRS signal active during deferral time running, the deferral time will reset and restart. **Note:** This bit is valid only in Half-duplex mode
3	TEN	Transmitter enable bit 0:The MAC transmit function is disabled after finish the transmission of the current frame, and no frames to be transmitted anymore 1: The transmit function of the MAC is enabled for transmission
2	REN	Receiver enable bit 0: The MAC reception function is disabled after finish the reception of the current frame, and no frames will be received anymore. 1: The MAC reception function is enabled for receiving frames
1:0	Reserved	Must be kept at reset value

22.4.2. MAC frame filter register (ENET_MAC_FRMF)

Address offset: 0x0004

Reset value: 0x0000 0000

This register configures the filtering method for receiving frames

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FAR	Reserved														
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					HPFLT	SAFLT	SAIFLT	PCFRM[1:0]		BFRMD	MFD	DAIFLT	HMF	HUF	PM
					rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	FAR	Frames all receive bit

		This bit controls the receive filter function. 0: Only the frame passed the filter can be forwarded to application. 1: All received frame are forwarded to application. But filter result will also be updated to receive descriptor status.
30:11	Reserved	Must be kept at reset value
10	HPFLT	Hash or perfect filter bit 0: If the HUF or HMF bit is set, only frames that match the hash filter are passed 1: If the HUF or HMF bit is set, the receive filter passes frames that match either the perfect filtering or the hash filtering
9	SAFLT	Source address filter bit Enable source address filtering function besides destination address filtering. The filter also compares the SA field value in received frames with the values configured in the enabled SA registers. If SA comparison matches, the SA match bit in the receive descriptor status is set high 0: Source address function in filter disable 1: Source address function in filter enable
8	SAIFLT	Source address inverse filtering bit This bit makes the result of SA matching inverse. 0: Not inverse for source address filtering 1: Inverse source address filtering result. When SA matches the enabled SA registers, filter marks it as failing the SA address filter
7:6	PCFRM[1:0]	Pass control frames bits These bits set the forwarding conditions for all control frames (including unicast and multicast pause frame). For pause control frame, the processing (not forwarding) depends only on RFCEN in ENET_MAC_FCTL[2] 0x0: MAC prevents all control frames from reaching the application 0x1: MAC only forwards all other control frames except pause control frame 0x2: MAC forwards all control frames to application even if they fail the address filter 0x3: MAC forwards control frames that only pass the address filter
5	BFRMD	Broadcast frames disable bit 0:The address filters pass all received broadcast frames 1:The address filters filter all incoming broadcast frames
4	MFD	Multicast filter disable bit 0:Filtering of multicast frame depends on the HMF bit 1:All received frames with a multicast destination address (first bit in the destination address field is '1' and not all bits in the destination are ‘1’) are passed
3	DAIFLT	Destination address inverse filtering bit

		This bit makes the result of DA filtering inverse 0: Not inverse DA filtering result 1: Inverse DA filtering result
2	HMF	Hash multicast filter bit 0: The filter uses perfect mode for filtering multicast frame. 1: The filter uses hash mode for filtering multicast frame
1	HUF	Hash unicast filter bit 0: The filter uses perfect mode for filtering unicast frame 1: The filter uses hash mode for filtering unicast frame
0	PM	Promiscuous mode bit This bit can make the filter bypassed which means all received frames are thought pass the filer and DA/SA filtering result status in descriptor is always ‘0’. 0: Promiscuous mode disabled 1: Promiscuous mode enabled

22.4.3. MAC hash list high register (ENET_MAC_HLH)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HLH[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	HLH[31:0]	Hash list high bits These bits take the high 32-bit value of hash list

22.4.4. MAC hash list low register (ENET_MAC_HLL)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HLL[31:16]															
rw															

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

rw

Bits	Fields	Descriptions
31:0	HLL[31:0]	Hash list low bits These bits take the low 32-bit value of hash list

22.4.5. MAC PHY control register (ENET_MAC_PHY_CTL)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[4:0]					PR[4:0]					Reserved	CLR[2:0]			PW	PB
rw					rw						rw			rw	rc_w1

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:11	PA[4:0]	PHY address bits These bits choose which PHY device is to be accessed
10:6	PR[4:0]	PHY register bits These bits choose the register address in selected PHY device
5	Reserved	Must be kept at reset value
4:2	CLR[2:0]	Clock range bits MDC clock divided factor select which is decided by HCLK frequency range 0x0: HCLK/42 (HCLK range: 60-90 MHz) 0x1: HCLK/64 (HCLK range: 90-108 MHz) 0x2: HCLK/16 (HCLK range: 20-35 MHz) 0x3: HCLK/26 (HCLK range: 35-60 MHz) other: Reserved
1	PW	PHY write bit This bit indicate the PHY operation mode 0: Sending read operation to PHY 1: Sending write operation to PHY
0	PB	PHY busy bit This bit indicates the running state of operation on PHY. Application sets this bit to 1 and should wait it cleared by hardware. Application must make sure this bit is zero

before writing data to ENET_MAC_PHY_CTL register and reading/writing data from/to ENET_MAC_PHY_DATA register

22.4.6. MAC MII data register (ENET_MAC_PHY_DATA)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PD[15:0]	PHY data bits For reading operation, these bits contain the data from external PHY. For writing operation, these bits contain the data will be sent to external PHY.

22.4.7. MAC flow control register (ENET_MAC_FCTL)

Address offset: 0x0018

Reset value: 0x0000 0000

This register configures the generation and reception of the control frames.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTM[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DZQP	Reserved	PLTS[1:0]		UPFDT	RFCEN	TFCEN	FLCB/BKPA
								rw		rw		rw	rw	rw	rw

Bits	Fields	Descriptions
31:16	PTM[15:0]	Pause time bits These bits configured the pause time filed value in transmit pause control frame.
15:8	Reserved	Must be kept at reset value

7	DZQP	Disable Zero-quanta pause bit 0: Enable automatic zero-quanta generation function for pause control frame. 1: Disable the automatic zero-quanta generation function for pause control frame
6	Reserved	Must be kept at reset value
5:4	PLTS[1:0]	Pause low threshold bits These bits configure the threshold of the pause timer for retransmitting frames automatically. Application must make sure the low threshold bits are greater than 0 and less than configured pause time. The low threshold calculation formula is PTM-PLTS. For example, if PTM = 0x80 (128 slot-times), and PLTS = 0x1 (28 slot-times), then the second pause frame is automatically transmitted when pause timer counted at 100 (128 - 28) slot-times after the first pause frame is transmitted 0x0: Pause time minus 4 slot times 0x1: Pause time minus 28 slot times 0x2: Pause time minus 144 slot times 0x3: Pause time minus 256 slot times **Note:** One slot time equals the time of transmitting 512 bits on the MII interface
3	UPFDT	Unicast pause frame detect bit 0: Only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected. 1: Besides the unique multicast address, MAC can also use the MAC0 address (ENET_MAC_ADDR0H and ENET_MAC_ADDR0L register) to detecting pause frame.
2	RFCEN	Receive flow control enable bit 0: Decode function for pause frame is disabled 1: Enable decoding function for the received pause frame and process it. The MAC disables its transmitter for a specified (pause time field value in received frame) time
1	TFCEN	Transmit flow control enable bit 0: Disable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode are not performed. 1: Enable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode can be performed by transmitter.
0	FLCB/BKPA	Flow control busy/back pressure activate bit This bit only valid when TFCEN is set. This bit can send a pause frame in Full-duplex mode or activate the back pressure function in Half-duplex mode by application. For Full-duplex mode, application must make sure this bit is 0 before writing ENET_MAC_FCTL register. After set by application, MAC sends a pause frame to interface and this bit will keep set until the pause frame has completed transmitting.

For Half-duplex mode, MAC can enter back-pressure state by application setting this bit. When the MAC is in back-pressure state, any frame presented on interface will make the MAC send a JAM pattern to inform outside a collision occurred.

22.4.8. MAC flow control threshold register (ENET_MAC_FCTH)

Address offset: 0x1080

Reset value: 0x0000 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6:4	RFD[2:0]	Threshold of deactive flow control This field configures the threshold of the deactive flow control. The value should always be less than the Threshold of active flow control value configured in bits[2:0]. When the value of the unprocessed data in RxFIFO is less than this value configured, the flow control function will deactive. 0x0: 256 bytes 0x1: 512 bytes 0x2: 768 bytes 0x3: 1024 bytes 0x4: 1280 bytes 0x5: 1536 bytes 0x6,0x7: 1792 bytes
3	Reserved	Must be kept at reset value
2:0	RFA[2:0]	Threshold of active flow control This field configures the threshold of the active flow control. If flow control function is enabled, when the value of the unprocessed data in RxFIFO is more than this value configured, the flow control function will active. 0x0: 256 bytes 0x1: 512 bytes 0x2: 768 bytes 0x3: 1024 bytes 0x4: 1280 bytes 0x5: 1536 bytes

0x6,0x7: 1792 bytes

22.4.9. MAC VLAN tag register (ENET_MAC_VLT)

Address offset: 0x001C

Reset value: 0x0000 0000

This register configures the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th byte (length/type field) of the receiving frame with 0x8100, and the following 2 bytes (the 15th and 16th byte) are compared with the VLAN tag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															VLTC
															rw

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

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	VLTC	12-bit VLAN tag comparison bit This bit selects 12 or 16 bit VLAN tag for comparison. 0: All 16 bits (the 15th and 16th byte) of the VLAN tag in received frame are used for comparison. 1: Only low 12 bits of the VLAN tag in received frame are used for comparison.
15:0	VLTI[15:0]	VLAN tag identifier (for receive frames) bits These bits are configured for detecting VLAN frame using 802.1Q VLAN tag format. The format shows below: VLTI[15:13]: UP(user priority) VLTI[12]: CFI(canonical format indicator) VLTI[11:0]: VID(VLAN identifier) When comparison bits (VLTI[11:0] if VLTC=1 or VLTI[15:0] if VLTC=0) are all zeros, VLAN tag comparison is bypassed and every frame with type filed value of 0x8100 is considered a VLAN frame. When comparison bits not all zeros, VLAN tag comparison use bit VLTI[11:0] (if VLTC=1) or VLTI[15:0] (if VLTC=0) for checking.

22.4.10. MAC remote wakeup frame filter register (ENET_MAC_RWFF)

Address offset: 0x0028

Reset value: 0x0000 0000

The MAC remote wakeup frame filter register is actually a pointer to eight (with same address offset) such wakeup frame filter registers. Eight sequential write operations to this address with the offset (0x0028) will write all wakeup frame filter registers. Eight sequential read operations from this address with the offset (0x0028) will read all wakeup frame filter registers.

Figure 22-13. Wakeup frame filter register

	31							0
Wakeup frame filter reg0	Byte Mask of Filter-0							
Wakeup frame filter reg1	Byte Mask of Filter-1							
Wakeup frame filter reg2	Byte Mask of Filter-2							
Wakeup frame filter reg3	Byte Mask of Filter-3							
Wakeup frame filter reg4	Reserved	Filter 3 Command	Reserved	Filter 2 Command	Reserved	Filter 1 Command	Reserved	Filter 0 Command
Wakeup frame filter reg5	Offset of Filter 3		Offset of Filter 2		Offset of Filter 1		Offset Filter 0	
Wakeup frame filter reg6	Filter 1 CRC - 16				Filter 0 CRC - 16			
Wakeup frame filter reg7	Filter 3 CRC - 16				Filter 2 CRC - 16			

22.4.11. MAC wakeup management register (ENET_MAC_WUM)

Address offset: 0x002C

Reset value: 0x0000 0000

This register configures the request of wakeup events and monitors the wakeup events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUFFRPR	Reserved														
rs															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						GU	Reserved		WUFR	MPKR	Reserved		WFEN	MPEN	PWD
						rw			rc_r	rc_r			rw	rw	rs

Bits	Fields	Descriptions
31	WUFFRPR	Wakeup frame filter register pointer reset bit This bit can reset the inner pointer of ENET_MAC_RWFF register by application set it to 1. Hardware clears it when resetting completes. 0: No effect 1: Reset the ENET_MAC_RWFF register inner pointer
30:10	Reserved	Must be kept at reset value
9	GU	Global unicast bit 0: Not all of received unicast frame is considered to be a wakeup frame

		1: Any received unicast frame passed address filtering is considered to be a wakeup frame
8:7	Reserved	Must be kept at reset value
6	WUFR	Wakeup frame received bit This bit is cleared when this register is read 0:Has not received the wake-up frame 1:The wakeup event was generated due to reception of a wakeup frame
5	MPKR	Magic packet received bit This bit is cleared when this register is read 0:Has not received the Magic Packet frame 1:The wakeup event was generated by the reception of a Magic Packet frame
4:3	Reserved	Must be kept at reset value
2	WFEN	Wakeup frame enable bit 0: Disable generating a wakeup event due to wakeup frame reception 1: Enable generating a wakeup event due to wakeup frame reception
1	MPEN	Magic Packet enable bit 0:Disable generating a wakeup event due to Magic Packet reception 1: Enable generating a wakeup event due to Magic Packet reception
0	PWD	Power down bit This bit is set by application and reset by hardware. When this bit is set, MAC drops all received frames. When power-down mode exit because of wakeup event occurred, hardware resets this bit.

22.4.12. MAC interrupt flag register (ENET_MAC_INTF)

Address offset: 0x0038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TMST	Reserved		MSCT	MSCR	MSC	WUM	Reserved		
						rc_r			r	r	r	r			

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	TMST	Time stamp trigger status bit This bit is cleared when ENET_PTP_TSF register is read

		0: The system time value is less than the value specified in the target time registers 1: The system time value equals or exceeds the value specified in the target time registers
8:7	Reserved	Must be kept at reset value
6	MSCT	MSC transmit status bit 0: All the bits in register ENET_MSC_TINTF are cleared 1: An interrupt is generated in the ENET_MSC_TINTF register
5	MSCR	MSC receive status bit 0: All the bits in register ENET_MSC_RINTF are cleared 1: An interrupt is generated in the ENET_MSC_RINTF register
4	MSC	MSC status bit This bit is logic ORed from MSCT and MSCR bit. 0: Both MSCT and MSCR bits in this register are low 1: Any of bit 6 (MSCT) or bit 5 (MSCR) is set high
3	WUM	WUM status bit This bit is logic ORed from WUFR and MPKR bit in ENET_MAC_WUM register. 0: Wakeup frame or Magic Packet frame is not received 1: A Magic packet or remote wakeup frame is received in power down Mode
2:0	Reserved	Must be kept at reset value

22.4.13. MAC interrupt mask register (ENET_MAC_INTMSK)

Address offset: 0x003C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TMSTIM	Reserved					WUMIM	Reserved		
						rw						rw			

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	TMSTIM	Timestamp trigger interrupt mask bit 0:Unmask the timestamp interrupt generation 1:Mask the timestamp interrupt generation
8:4	Reserved	Must be kept at reset value

3	WUMIM	WUM interrupt mask bit 0: Unmask the interrupt generation due to the WUM bit in ENET_MAC_INTF register 1: Mask the interrupt generation due to the WUM bit in ENET_MAC_INTF register
2:0	Reserved	Must be kept at reset value

22.4.14. MAC address 0 high register (ENET_MAC_ADDR0H)

Address offset: 0x0040

Reset value: 0x8000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MO	Reserved														
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR0H[15:0]															
rw															

Bits	Fields	Descriptions
31	MO	Always read 1 and must be kept
30:16	Reserved	Must be kept at reset value
15:0	ADDR0H[15:0]	MAC address0 high16-bit These bits contain the high 16-bit (bit 47 to 32) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control.

22.4.15. MAC address 0 low register (ENET_MAC_ADDR0L)

Address offset: 0x0044

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR0L[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR0L[15:0]															
rw															

Bits	Fields	Descriptions
31:0	ADDR0L[31:0]	MAC addresss0 low 32-bit

These bits contain the low 32-bit (bit 31 to 0) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control.

22.4.16. MAC address 1 high register (ENET_MAC_ADDR1H)

Address offset: 0x0048

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFE	SAF	MB[5:0]						Reserved							
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR1H[15:0]															
rw															

Bits	Fields	Descriptions
31	AFE	Address filter enable bit 0: The address filter ignores the MAC address1 for filtering 1: The address filter uses the MAC address1 for perfect filtering
30	SAF	Source address filter bit 0: The MAC address1[47:0] is used to comparing with the DA field of the received frame 1: The MAC address1[47:0] is used to comparing with the SA field of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address1 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR1H [15:8] MB[4]: ENET_MAC_ADDR1H [7:0] MB[3]: ENET_MAC_ADDR1L [31:24] MB[2]: ENET_MAC_ADDR1L[23:16] MB[1]: ENET_MAC_ADDR1L[15:8] MB[0]: ENET_MAC_ADDR1L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR1H[15:0]	MAC address1 high [47:32] bits This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address1

22.4.17. MAC address 1 low register (ENET_MAC_ADDR1L)

Address offset: 0x004C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR1L[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR1L[15:0]															
rw															

Bits	Fields	Descriptions
31:0	ADDR1L[31:0]	MAC address1 low 32-bit This field contains the low 32-bit of the 6-byte MAC address1

22.4.18. MAC address 2 high register (ENET_MAC_ADDR2H)

Address offset: 0x0050

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFE	SAF	MB[5:0]						Reserved							
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR2H[15:0]															
rw															

Bits	Fields	Descriptions
31	AFE	Address filter enable bit 0:The address filter ignores the MAC address2 for filtering 1:The address filter uses the MAC address2 for perfect filtering
30	SAF	Source address filter bit 0:The MAC address2[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address2[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address2 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR2H [15:8]

Bits	Fields	Descriptions
		MB[4]: ENET_MAC_ADDR2H [7:0] MB[3]: ENET_MAC_ADDR2L [31:24] MB[2]: ENET_MAC_ADDR2L[23:16] MB[1]: ENET_MAC_ADDR2L[15:8] MB[0]: ENET_MAC_ADDR2L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR2H[15:0]	MAC address2 high 16-bit This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address2

22.4.19. MAC address 2 low register (ENET_MAC_ADDR2L)

Address offset: 0x0054

Reset value: 0xFFFF FFFF

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">ADDR2L[31:16]</td></tr>
<tr><td colspan="16">rw</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ADDR2L[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31:0	ADDR2L[31:0]	MAC address2 low 32-bit This field contains the low 32-bit of the 6-byte MAC address2

22.4.20. MAC address 3 high register (ENET_MAC_ADDR3H)

Address offset: 0x0058

Reset value: 0x0000 FFFF

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>AFE</td><td>SAF</td><td colspan="6">MB[5:0]</td><td colspan="8">Reserved</td></tr>
<tr><td>rw</td><td>rw</td><td colspan="6">rw</td><td colspan="8"></td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ADDR3H[15:0]</td></tr>
<tr><td colspan="16">rw</td></tr>
</table>

Bits	Fields	Descriptions
31	AFE	Address filter enable bit 0:The address filter ignores the MAC address3 for filtering 1:The address filter use the MAC address3 for perfect filtering

30	SAF	Source address filter bit 0:The MAC address3[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address3[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address3 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR3H [15:8] MB[4]: ENET_MAC_ADDR3H [7:0] MB[3]: ENET_MAC_ADDR3L [31:24] MB[2]: ENET_MAC_ADDR3L[23:16] MB[1]: ENET_MAC_ADDR3L[15:8] MB[0]: ENET_MAC_ADDR3L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR3H[15:0]	MAC address3 high 16-bit This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address3

22.4.21. MAC address 3 low register (ENET_MAC_ADDR3L)

Address offset: 0x005C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR3L[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR3L[15:0]															
rw															

Bits	Fields	Descriptions
31:0	ADDR3L[31:0]	MAC address3 low 32-bit This field contains the low 32-bit of the 6-byte MAC address3

22.4.22. MSC control register (ENET_MSC_CTL)

Address offset: 0x0100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												MCFZ	RTOR	CTSR	CTR
												rw	rw	rw	rw

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3	MCFZ	MSC counter freeze bit 0: MSC counters are not frozen 1: Freezes all the MSC counters to their current value. RTOR bit can work on this frozen state.
2	RTOR	Reset on read bit 0: The MSC counters are not reset after reading MSC counter 1: The MSC counters are reset to zero after read them
1	CTSR	Counter stop rollover bit 0: The counters roll over to zero after they reached the maximum value 1: The counters do not roll over to zero after they reached the maximum value
0	CTR	Counter reset bit Cleared by hardware 1 clock after set. This bit is cleared automatically after 1 clock cycle 0: No effect 1: Reset all counters

22.4.23. MSC receive interrupt flag register (ENET_MSC_RINTF)

Address offset: 0x0104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														RGUF	Reserved
														rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RFAE	RFCE	Reserved				
									rc_r	rc_r					

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	RGUF	Received good unicast frames bit 0: Good unicast frame received counter is less than half of the maximum value

		1: Good unicast frame received counter reaches half of the maximum value
16:7	Reserved	Must be kept at reset value
6	RFAE	Received frames alignment error bit 0: Alignment error frame received counter is less than half of the maximum value 1: Alignment error frame received counter reaches half of the maximum value
5	RFCE	Received frames CRC error bit 0: CRC error frame received counter is less than half of the maximum value 1: CRC error frame received counter reaches half of the maximum value
4:0	Reserved	Must be kept at reset value

22.4.24. MSC transmit interrupt flag register (ENET_MSC_TINTF)

Address offset: 0x0108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TGF	Reserved				
										rc_r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFMSC	TGFSC	Reserved													
rc_r	rc_r														

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TGF	Transmitted good frames bit 0: Good frame transmitted counter is less than half of the maximum value 1: Good frame transmitted counter reaches half of the maximum value
20:16	Reserved	Must be kept at reset value
15	TGFMSC	Transmitted good frames more single collision bit 0: Good frame after more than a single collision transmitted counter is less than half of the maximum value 1:Good frame after more than a single collision transmitted counter reaches half of the maximum value
14	TGFSC	Transmitted good frames single collision bit 0: Good frame after a single collision transmitted counter is less than half of the maximum value 1: Good frame after a single collision transmitted counter reaches half of the maximum value

13:0	Reserved	Must be kept at reset value

22.4.25. MSC receive interrupt mask register (ENET_MSC_RINTMSK)

Address offset: 0x010C

Reset value: 0x0000 0000

The Ethernet MSC receive interrupt mask register maintains the masks for interrupts generated when receive statistic counters reach half their maximum value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														RGUFIM	Reserved
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RFAEIM	RFCEIM	Reserved				
									rw	rw					

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	RGUFIM	Received good unicast frames interrupt mask bit 0: Unmask the interrupt when the RGUF bit is set 1: Mask the interrupt when RGUF bit is set
16:7	Reserved	Must be kept at reset value
6	RFAEIM	Received frames alignment error interrupt mask bit 0: Unmask the interrupt when the RFAE bit is set 1: Mask the interrupt when the RFAE bit is set
5	RFCEIM	Received frame CRC error interrupt mask bit 0: Unmask the interrupt when RFCE bit is set 1: Mask the interrupt when the RFCE bit is set
4:0	Reserved	Must be kept at reset value

22.4.26. MSC transmit interrupt mask register (ENET_MSC_TINTMSK)

Address offset: 0x0110

Reset value: 0x0000 0000

The MSC transmit interrupt mask register configures the mask bits for interrupts generation

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TGFIM	Reserved				
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFMSCIM	TGFSCIM	Reserved													
rw	rw														

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TGFIM	Transmitted good frames interrupt mask bit 0: Unmask the interrupt when the TGF bit is set 1:Mask the interrupt when the TGF bit is set
20:16	Reserved	Must be kept at reset value
15	TGFMSCIM	Transmitted good frames more single collision interrupt mask bit 0: Unmask the interrupt when the TGFMSC bit is set 1: Mask the interrupt when the TGFMSC bit is set
14	TGFSCIM	Transmitted good frames single collision interrupt mask bit 0: Unmask the interrupt when the TFGSC bit is set 1: Mask the interrupt when the TFGSC bit is set
13:0	Reserved	Must be kept at reset value

22.4.27. MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT)

Address offset: 0x014C

Reset value: 0x0000 0000

This register counts the number of successfully transmitted frames after a single collision in Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCC[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	SCC[31:0]	Transmitted good frames single collision counter bits These bits count the number of a transmitted good frames after only a single collision

22.4.28. MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT)

Address offset: 0x0150

Reset value: 0x0000 0000

This register counts the number of successfully transmitted frames after more than one single collision in Half-duplex mode.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">MSCC[31:16]</td></tr>
<tr><td colspan="16">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">MSCC[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:0	MSCC[31:0]	Transmitted good frames more one single collision counter bits These bits count the number of a transmitted good frames after more than one single collision

22.4.29. MSC transmitted good frames counter register (ENET_MSC_TGFCNT)

Address offset: 0x0168

Reset value: 0x0000 0000

This register counts the number of good frames transmitted.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">TGF[31:16]</td></tr>
<tr><td colspan="16">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">TGF[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:0	TGF[31:0]	Transmitted good frames counter bits These bits count the number of transmitted good frames

22.4.30. MSC received frames with CRC error counter register (ENET_MSC_RFCECNT)

Address offset: 0x0194

Reset value: 0x0000 0000

This register counts the number of frames received with CRC error.

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

Bits	Fields	Descriptions
31:0	RFCER[31:0]	Received frames with CRC error counter bits These bits count the number of receive frames with CRC error

22.4.31. MSC received frames with alignment error counter register (ENET_MSC_RFAECNT)

Address offset: 0x0198

Reset value: 0x0000 0000

This register counts the number of received frames with alignment error.

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

Bits	Fields	Descriptions
31:0	RFAER[31:0]	Received frames alignment error counter bits These bits count the number of receive frames with alignment error

22.4.32. MSC received good unicast frames counter register (ENET_MSC_RGUFCNT)

Address offset: 0x01C4

Reset value: 0x0000 0000

This register counts the number of good unicast frames received.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RGUF[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	RGUF[31:0]	Received good unicast frames counter bits These bits count the number of good unicast frames received.

22.4.33. PTP time stamp control register (ENET_PTP_TSCTL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the generation and updating for timestamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TMSARU	TMSITEN	TMSSTU	TMSSTI	TMSFCU	TMSEN
										rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	TMSARU	Time stamp addend register update bit This bit is cleared when the update is completed. This register bit must be read as zero before application set it. 0: The timestamp addend register’s contents are not updated to the PTP block for fine correction

		1: The timestamp addend register's contents are updated to the PTP block for fine correction
4	TMSITEN	Timestamp interrupt trigger enable bit 0: Disable timestamp interrupt 1: A timestamp interrupt is generated when the system time becomes greater than the value written in target time register. **Note:** When the timestamp trigger interrupt generated, this bit is cleared
3	TMSSTU	Timestamp system time update bit Both the TMSSTU and TMSSTI bits must be read as zero before application set this bit 0: The system time is maintained without any change 1: The system time is updated (added to or subtracted from) with the value specified in the timestamp update (high and low) registers. It is cleared by hardware when the update finished.
2	TMSSTI	Timestamp system time initialize bit This bit must be read as zero before application set it. 0: The system time is maintained without any change 1: Initializing the system time with the value in timestamp update (high and low) registers. It is cleared by hardware when the initialization finished.
1	TMSFCU	Timestamp fine or coarse update bit 0:The system timestamp uses the coarse method for updating 1:The system timestamp uses the fine method for updating
0	TMSEN	Timestamp enable bit 0: Disable timestamp function 1: Enable timestamp function for transmit and receive frames **Note:** After setting this to 1, application must initialize the system time.

22.4.34. PTP subsecond increment register (ENET_PTP_SSINC)

Address offset: 0x0704

Reset value: 0x0000 0000

This register configures the 8-bit value for the incrementing subsecond register. In coarse mode, this value is added to the system time every HCLK clock cycle. In fine mode, this value is added to the system time when the accumulator reaches overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								STMSSI[7:0]							

rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	STMSSI[7:0]	System time subsecond increment bits In every update operation, these bits are added to the subsecond value of system time.

22.4.35. PTP time stamp high register (ENET_PTP_TSH)

Address offset: 0x0708

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STMS[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	STMS[31:0]	System time second bits These bits show the current second of the system time.

22.4.36. PTP time stamp low register (ENET_PTP_TSL)

Address offset: 0x070C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STS	STMSS[30:16]														
r	r														

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

Bits	Fields	Descriptions
31	STS	System time sign bit 0: Time value is positive 1: Time value is negative
30:0	STMSS[30:0]	System time subseconds bits

		These bits show the current subsecond of the system time with 0.46 ns accuracy if required accuracy is 20 ns.

22.4.37. PTP time stamp update high register (ENET_PTP_TSUH)

Address offset: 0x0710

Reset value: 0x0000 0000

This register configures the high 32-bit of the time to be written to, added to, or subtracted from the system time value. The timestamp update registers (high and low) initialize or update the system time maintained by the MAC core. Application must write both of these registers before setting the TMSSTI or TMSSTU bits in the timestamp control register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSUS[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	TMSUS[31:0]	Time stamp update second bits These bits are used for initializing or adding/subtracting to second of the system time

22.4.38. PTP time stamp update low register (ENET_PTP_TSUL)

Address offset: 0x0714

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSUPNS	TMSUSS[30:16]														
rw	rw														

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

Bits	Fields	Descriptions
31	TMSUPNS	Timestamp update positive or negative sign bit When TMSSTI is set, this bit must be 0. 0: Timestamp update value is added to system time

		1: Timestamp update value is subtracted from system time
30:0	TMSUSS[30:0]	Timestamp update subsecond bits These bits are used for initializing or adding/subtracting to subsecond of the system time

22.4.39. PTP time stamp addend register (ENET_PTP_TSADDEND)

Address offset: 0x0718

Reset value: 0x0000 0000

This register value is used only in fine update mode for adjusting the clock frequency. This register value is added to a 32-bit accumulator in every clock cycle and the system time updates when the accumulator reaches overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSA[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	TMSA[31:0]	Time stamp addend bits These registers contain a 32-bit time value which is added to the accumulator register to achieve time synchronization

22.4.40. PTP expected time high register (ENET_PTP_ETH)

Address offset: 0x071C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETSH[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	ETSH[31:0]	Expected time high bits These bits store the expected target second time.

22.4.41. PTP expected time low register (ENET_PTP_ETL)

Address offset: 0x0720

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETSL[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	ETSL[31:0]	Expected time low bits These bits store the expected target nanosecond time (signed).

22.4.42. DMA bus control register (ENET_DMA_BCTL)

Address offset: 0x1000

Reset value: 0x0000 2101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						AA	FPBL	UIP	RXDP[5:0]						FB
						rw	rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTPR[1:0]		PGBL[5:0]						Reserved	DPSL[4:0]					DAB	SWR
rw		rw							rw					rw	rs

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25	AA	Address-aligned bit 0: Disable address-aligned 1: Enabled address-aligned. If the FB=1, all AHB interface address is aligned to the start address LS bits (bit 1 to 0). If the FB=0, the AHB interface first access address (accessing the data buffer’s start address) is not aligned, but subsequent burst access addresses are aligned to the address
24	FPBL	Four times PGBL mode bit 0: The PGBL value programmed (bits [22:17] and bits [13:8]) for the DMA data number of beats to be transferred 1: Multiple the PGBL value programmed (bits [22:17] and bits [13:8]) four times for the DMA data number of beats to be transferred

23	UIP	Use independent PGBL bit 0: The PGBL value in bits [13:8] is applicable for both TxDMA and RxDMA engines 1: The RxDMA uses the RXDP[5:0] bits as burst length while the PGBL[5:0] is used by TxDMA
22:17	RXDP[5:0]	RxDMA PGBL bits If UIP=0, these bits are not valid. Only when UIP=1, these bits is configured for the maximum number of beats to be transferred in one RxDMA transaction. 0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8 0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved
16	FB	Fixed burst bit 0: The AHB can use SINGLE and INCR burst transfer operations 1: The AHB can only use SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers **Note:** MB and FB should be and must be only one of bit is set.
15:14	RTPR[1:0]	RxDMA and TxDMA transfer priority ratio bits These bits indicate the access ratio between RxDMA and TxDMA. 0x0: RxDMA : TxDMA = 1:1 0x1: RxDMA : TxDMA = 2:1 0x2: RxDMA : TxDMA = 3:1 0x3: RxDMA : TxDMA = 4:1 **Note:** This bit is valid only when the arbitration mode is Round-robin (DAB=0)
13:8	PGBL[5:0]	Programmable burst length bits These bits indicate the maximum number of beats to be transferred in one DMA transaction. When UIP=1, the PGBL value is only used for TxDMA. When UIP=0, the PGBL value is used for both TxDMA and RxDMA. 0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8 0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved
7	Reserved	Must be kept at reset value
6:2	DPSL[4:0]	Descriptor skip length bit These bits are valid only between two ring mode descriptors. They define the number of words (32-bit) to skip between two unchained descriptors. The address

		skipping starts from the end of current descriptor to the start of next descriptor. When DPSL value equals zero, the descriptor table is taken as contiguous by the DMA, in ring mode
1	DAB	DMA arbitration bit This bit indicates the arbitration mode between RxDMA and TxDMA. 0: Round-robin mode and DMA access priority is given in RTPR 1: Fixed mode. RxDMA has higher priority than TxDMA
0	SWR	Software reset bit This bit can reset all core internal registers located in CLK_TX and CLK_RX. It is cleared by hardware when the reset operation is complete in all clock domains. **Note:** Application must make sure this bit is 0 before writing any MAC core registers. 0: Core and inner register are not in reset state 1: Reset all core internal registers

22.4.43. DMA transmit poll enable register (ENET_DMA_TPEN)

Address offset: 0x1004

Reset value: 0x0000 0000

This register is used by the application to make the TxDMA controller poll the transmit descriptor table. The TxDMA controller can go into suspend state because of an underflow error in a transmitted frame or the descriptor unavailable (DAV=0). Application can write any value into this register for attempting to re-fetch the current descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TPE[31:16]															
rw_wt															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPE[15:0]															
rw_wt															

Bits	Fields	Descriptions
31:0	TPE[31:0]	Transmit poll enable bits Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CTDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the TBU bit in ENET_DMA_STAT register will be set.

22.4.44. DMA receive poll enable register (ENET_DMA_RPEN)

Address offset: 0x1008

Reset value: 0x0000 0000

This register is used by the application to make the RxDMA controller poll the receive descriptor table. Writing to this register makes the RxDMA controller exit suspend state.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPE[31:16]															
rw_wt															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPE[15:0]															
rw_wt															

Bits	Fields	Descriptions
31:0	RPE[31:0]	Receive poll enable bits Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CRDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the RBU bit in ENET_DMA_STAT register will be set.

22.4.45. DMA receive descriptor table address register (ENET_DMA_RDTADDR)

Address offset: 0x100C

Reset value: 0x0000 0000

This register points to the start of the receive descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when RxDMA controller is in stop state. Before starting RxDMA reception process, this register must be configured correctly.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRT[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	SRT[31:0]	Start address of receive table bits These bits indicate the start address of the receive descriptor table. SRT[1:0] are internally taken as zero so SRT[1:0] are read only.

22.4.46. DMA transmit descriptor table address register (ENET_DMA_TDTADDR)

Address offset: 0x1010

Reset value: 0x0000 0000

This register points to the start of the transmit descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when TxDMA controller is in stop state. Before starting TxDMA transmission process, this register must be configured correctly.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STT[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	STT[31:0]	Start address of transmit table bits These bits indicate the start address of the transmit descriptor table. STT[1:0] are internally taken as zero so STT[1:0] are read only.

22.4.47. DMA status register (ENET_DMA_STAT)

Address offset: 0x1014

Reset value: 0x0000 0000

This register contains all the status bits that the DMA controller recorded. Writing 1 to meaningful bits in this register clears them but writing 0 has no effect. Each bit (bits [16:0]) can be masked by masking the corresponding bit in the ENET_DMA_INTEN register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		TST	WUM	MSC	Reserved	EB[2:0]			TP[2:0]			RP[2:0]			NI
		r	r	r		r			r			r			rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AI	ER	FBE	Reserved		ET	RWT	RPS	RBU	RS	TU	RO	TJT	TBU	TPS	TS
rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	TST	Timestamp trigger status bit

		This bit indicates a timestamp event occurred. It is cleared by application through clearing TMST bit. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: Timestamp event has not occurred 1: Timestamp event has occurred
28	WUM	WUM status bit This bit indicates a WUM event occurred. It is cleared when both two source event status bits are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: WUM event has not occurred 1: WUM event has occurred
27	MSC	MSC status bit This bit indicates a MSC event occurred. It is cleared when all of event sources are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: MSC event has not occurred 1: MSC event has occurred
26	Reserved	Must be kept at reset value
25:23	EB[2:0]	Error bits status bit When FBE=1, these bits decode the type of error that caused a bus response error on AHB bus. EB[0] 1: Error during data transfer by TxDMA 0: Error during data transfer by RxDMA EB[1] 1: Error during read transfer 0: Error during write transfer EB[2] 1: Error during descriptor access 0: Error during data buffer access
22:20	TP[2:0]	Transmit process state bit These bits decode the TxDMA state. 0x0: Stopped; Reset or Stop Transmit Command issued 0x1: Running; Fetching transmit transfer descriptor 0x2: Running; Waiting for status 0x3: Running; Reading Data from host memory buffer and queuing it to transmit buffer (TxFIFO) 0x4, 0x5: Reserved 0x6: Suspended; Transmit descriptor unavailable or transmit buffer underflow 0x7: Running; Closing transmit descriptor
19:17	RP[2:0]	Receive process state bit These bits decode the RxDMA state. 0x0: Stopped: Reset or Stop Receive Command issued 0x1: Running: Fetching receive transfer descriptor

		0x2: Reserved 0x3: Running: Waiting for receive packet 0x4: Suspended: Receive descriptor unavailable 0x5: Running: Closing receive descriptor 0x6: Reserved 0x7: Running: Transferring the receive packet data from receive buffer to host memory
16	NI	Normal interrupt summary The NI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register: TS (ENET_DMA_STAT [0]): Transmit interrupt TBU (ENET_DMA_STAT [2]): Transmit buffer unavailable RS (ENET_DMA_STAT [6]): Receive interrupt ER (ENET_DMA_STAT [14]): Early receive interrupt **Note:** Each time when this bit is set, application must cleared its source bit by writing 1 to that bit.
15	AI	Abnormal interrupt summary bit The AI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register: TPS (ENET_DMA_STAT [1]):Transmit process stopped TJT (ENET_DMA_STAT [3]):Transmit jabber timeout RO (ENET_DMA_STAT [4]): Receive FIFO overflow TU (ENET_DMA_STAT [5]): Transmit underflow RBU (ENET_DMA_STAT [7]): Receive buffer unavailable RPS (ENET_DMA_STAT [8]): Receive process stopped RWT (ENET_DMA_STAT [9]): Receive watchdog timeout ET (ENET_DMA_STAT [10]): Early transmit interrupt FBE (ENET_DMA_STAT [13]): Fatal bus error **Note:** Each time when this bit is set, application must cleared its source bit by writing 1 to that bit.
14	ER	Early receive status bit This bit is automatically cleared when the ENET_DMA_STAT [6] is set. 0: The first buffer has not been filled 1: The first buffer has filled with received frame
13	FBE	Fatal bus error status bit This bit indicates a response error on AHB interface is occurred and the error type can be decoded by EB[2:0] bits. 0: Bus error has not occurred 1: A bus error occurred and the corresponding DMA stops all operations
12:11	Reserved	Must be kept at reset value

10	ET	Early transmit status bit 0: The frame to be transmitted has not fully transferred into the TxFIFO 1: The frame to be transmitted has fully transferred into the TxFIFO
9	RWT	Receive watchdog timeout status bit 0: No received a frame with a length greater than 2048 bytes 1: A frame with a length greater than 2048 bytes is received
8	RPS	Receive process stopped status bit 0: The receive process is not in stop state 1: The receive process is in stop state
7	RBU	Receive buffer unavailable status bit 0: The DAV bit in fetched next receive descriptor is set 1: The DAV bit in fetched next receive descriptor is reset and RxDMA enters suspend state.
6	RS	Receive status bit 0: Frame reception has not completed 1: Frame reception has completed
5	TU	Transmit underflow status bit 0: Underflow error has not occurred during frame transmission 1: The TxFIFO encountered an underflow error during frame transmission and entered suspend state
4	RO	Receive overflow status bit 0: Receive overflow error has not occurred during frame reception 1: The RxFIFO encountered an overflow error during frame reception. If a part of frame data has transferred to the memory, the overflow status in RDES0[11] is also set
3	TJT	Transmit jabber timeout status bit 0: Transmit jabber timeout has not occurred during frame transmission 1: The transmit jabber timer expired. The TxDMA controller cancels the current transmission and enters stop state. This also causes JT bit in TDES0 set.
2	TBU	Transmit buffer unavailable status bit 0: The DAV bit in fetched next transmit descriptor is set 1: The DAV bit in fetched next transmit descriptor is reset and TxDMA enters suspend state.
1	TPS	Transmit process stopped status bit 0: The transmission is not in stop state 1: The transmission is in stop state
0	TS	Transmit status bit This bit can only be set when both LSG and INTC are set in TDES0.

0: Current frame transmission is not finished

1: Current frame transmission is finished.

22.4.48. DMA control register (ENET_DMA_CTL)

Address offset: 0x1018

Reset value: 0x0000 0000

This register configures both the transmitting and receiving operation modes and commands. This register should be written at last during the process of DMA initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DTCERFD	RSFD	DAFRF	Reserved		TSFD	FTF	Reserved			TTHC[2]
					rw	rw	rw			rw	rs				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTHC[1:0]		STE	Reserved					FERF	FUF	Reserved	RTHC[1:0]		OSF	SRE	Reserved
rw		rw						rw	rw		rw		rw	rw	

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26	DTCERFD	Dropping of TCP/IP checksum error frames disable bit 0: All error frames will be dropped when FERF=0 1: The received frame with only payload error but no other errors will not be dropped.
25	RSFD	Receive Store-and-Forward bit 0: The RxFIFO operates in Cut-Through mode. The forwarding threshold depends on the RTHC bits 1: The RxFIFO operates in Store-and-Forward mode. The RTHC bits are don't care and the frame forwarding starts after the whole frame has pushed into RxFIFO.
24	DAFRF	Disable flushing of received frames bit 0: The RxDMA flushes all frames because of unavailable receive descriptor 1: The RxDMA does not flush any frames even though receive descriptor is unavailable
23:22	Reserved	Must be kept at reset value
21	TSFD	Transmit Store-and-Forward bit 0: The TxFIFO operates in Cut-Through mode. The TTHC bits in ENET_DMA_CTL register defines the start popping time from TxFIFO 1: The TxFIFO operates in Store-and-Forward mode. Transmission on interface starts after the full frame has been pushed into the TxFIFO. The TTHC bits are don't care in this mode. **Note:** This bit can be changed when transmission is in stop state

20	FTF	Flush transmit FIFO bit This bit can be set by application to reset TxFIFO inner control register and logic. If set, all data in TxFIFO are flushed. It is cleared by hardware after the flushing operation is finish. **Note:** Before this bit is reset, this register (ENET_DMA_CTL) must not be written.
19:17	Reserved	Must be kept at reset value
16:14	TTHC[2:0]	Transmit threshold control bit These bits control the start transmitting byte threshold of the TxFIFO. When TSFD=1, these bits are ignored. 0x0: 64 0x1: 128 0x2: 192 0x3: 256 0x4: 40 0x5: 32 0x6: 24 0x7: 16
13	STE	Start/stop transmission enable bit 0: The TxDMA controller will enter stop state after transmitting complete if the current frame is being transmitted. After complete transmitting, the next descriptor address will become current descriptor address for the address pointer. If the TxDMA controller is in suspend state, reset this bit make the controller entering stop state. 1: The TxDMA controller will enter running state. TxDMA controller fetches current descriptor address for frame transmitting. Transmit descriptor's fetching can either from base address in ENET_DMA_TDTADDR register or from the pointer position when transmission was stopped previously. If the DAV bit of current descriptor is reset, TxDMA controller enters suspend state and the TBU bit will be set. This bit should be set after all other DMA registers have been configured otherwise the action of TxDMA is unpredictable.
12:8	Reserved	Must be kept at reset value
7	FERF	Forward error frames bit 0: When RxFIFO is in Cut-Through mode (RSFD=0), if frame error (CRC error, collision error, checksum error, watchdog timeout, overflow error) is detected before popping RxFIFO data to memory, RxFIFO drops this error frame. But if frame error is detected after popping RxFIFO data to memory, RxFIFO will not drop this frame data. When RxFIFO is in Store-and-Forward mode, once frame error is detected during reception the RxFIFO drops this frame. 1: All frame received with error except runt error are forwarded to memory
6	FUF	Forward undersized good frames bit

		0: The RxFIFO drops all frames whose length is less than 64 bytes. However, if this frame has already started forwarding (may due to lower value of receive threshold in Cut-Through mode), the whole frame will be forwarded. 1: The RxFIFO forwards received frame whose frame length is less than 64 bytes but without any other error.
5	Reserved	Must be kept at reset value
4:3	RTHC[1:0]	Receive threshold control bit These bits control the threshold bytes of the RxFIFO. **Note:** These bits are valid only when the RSFD=0 and are ignored when the RSFD=1. 0x0: 64 0x1: 32 0x2: 96 0x3: 128
2	OSF	Operate on second frame bit 0: The TxDMA controller process the second transmit frame after the status of the first frame is written back to descriptor 1: The TxDMA controller process the second transmit frame after pushed all first frame data into TxFIFO but before the status of the first frame is written back to descriptor
1	SRE	Start/stop receive enable bit 0: The RxDMA controller will enter stop state after transfer complete if current received frame is transmitting to memory by RxDMA. After transfer complete, the next descriptor address in the receive table will become the current descriptor address when restart the RxDMA controller. Only RxDMA controller is in running state or suspend state, this bit can be reset by application. 1: The RxDMA controller will enter running state. RxDMA controller fetches receive descriptor from receive descriptor table for receiving frames. The descriptor address can either from current address in the ENET_DMA_RDTADDR register or the address after previous frame stopped by application. If the DAV bit in fetched descriptor is reset, RxDMA controller will enter suspend state and RBU bit will be set. Setting this bit can only when RxDMA controller is in stop state or suspend state. This bit should be set after all other DMA registers have been configured otherwise the action of RxDMA is unpredictable.
0	Reserved	Must be kept at reset value

22.4.49. DMA interrupt enable register (ENET_DMA_INTEN)

Address offset: 0x101C

Reset value: 0x0000 0000

This register configures the interrupts which are reflected in ENET_DMA_STAT register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															NIE
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIE	ERIE	FBEIE	Reserved		ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE
rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	NIE	Normal interrupt summary enable bit 0: A normal interrupt is disabled. 1: A normal interrupt is enabled This bit enables the following bits: TS (ENET_DMA_STAT [0]): Transmit interrupt TBU (ENET_DMA_STAT [2]): Transmit buffer unavailable RS (ENET_DMA_STAT [6]): Receive interrupt ER (ENET_DMA_STAT [14]): Early receive interrupt
15	AIE	Abnormal interrupt summary enable bit 0: An abnormal interrupt is disabled. 1: An abnormal interrupt is enabled This bit enables the following bits: TPS (ENET_DMA_STAT [1]):Transmit process stopped TJT (ENET_DMA_STAT [3]):Transmit jabber timeout RO (ENET_DMA_STAT [4]): Receive FIFO overflow TU (ENET_DMA_STAT [5]): Transmit underflow RBU (ENET_DMA_STAT [7]): Receive buffer unavailable RPS (ENET_DMA_STAT [8]): Receive process stopped RWT (ENET_DMA_STAT [9]): Receive watchdog timeout ET (ENET_DMA_STAT [10]): Early transmit interrupt FBE (ENET_DMA_STAT [13]): Fatal bus error
14	ERIE	Early receive interrupt enable bit 0: The early receive interrupt is disabled 1: The early receive interrupt is enabled
13	FBEIE	Fatal bus error interrupt enable bit 0: The fatal bus error enable interrupt is disabled 1: The fatal bus error enable interrupt is enabled
12:11	Reserved	Must be kept at reset value
10	ETIE	Early transmit interrupt enable bit

		0: The early transmit interrupt is disabled 1: The early transmit interrupt is enabled
9	RWTIE	Receive watchdog timeout interrupt enable bit 0: The receive watchdog timeout interrupt is disabled 1: The receive watchdog timeout interrupt is enabled
8	RPSIE	Receive process stopped interrupt enable bit 0: The receive stopped interrupt is disabled 1: The receive stopped interrupt is enabled
7	RBUIE	Receive buffer unavailable interrupt enable bit 0: The receive buffer unavailable interrupt is disabled 1: The receive buffer unavailable interrupt is enabled
6	RIE	Receive interrupt enable bit 0: The receive interrupt is disabled 1: The receive interrupt is disabled
5	TUIE	Transmit underflow interrupt enable bit 0: The underflow interrupt is disabled 1: The underflow interrupt is enabled
4	ROIE	Receive overflow interrupt enable bit 0: The overflow interrupt is disabled 1: The overflow interrupt is enabled
3	TJTIE	Transmit jabber timeout interrupt enable bit 0: The transmit jabber timeout interrupt is disabled 1: The transmit jabber timeout interrupt is enabled
2	TBUIE	Transmit buffer unavailable interrupt enable bit 0: The transmit buffer unavailable interrupt is disabled 1: The transmit buffer unavailable interrupt is enabled
1	TPSIE	Transmit process stopped interrupt enable bit 0: The transmission stopped interrupt is disabled 1: The transmission stopped interrupt is enabled
0	TIE	Transmit interrupt enable bit 0: The transmit interrupt is disabled 1: The transmit interrupt is enabled

22.4.50. DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT)

Address offset: 0x1020

Reset value: 0x0000 0000

There are two counters designed in DMA controller for tracking the number of missed frames during receiving. The counter value can be read from this register for debug purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			OBFOC	MSFA[10:0]											OBMFC
			rc_r	rc_r											rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSFC[15:0]															
rc_r															

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28	OBFOC	Overflow bit for FIFO overflow counter bit Overflow bit for FIFO overflow counter
27:17	MSFA[10:0]	Missed frames by the application bits These bits indicate the number of frames dropped by RxFIFO
16	OBMFC	Overflow bit for missed frame counter
15:0	MSFC[15:0]	Missed frames by the controller bits These bits indicate the number of frames missed by the RxDMA controller because of the unavailable receive buffer. Each time the RxDMA controller flushes one frame, this counter will plus 1.

22.4.51. DMA current transmit descriptor address register (ENET_DMA_CTDADDR)

Address offset: 0x1048

Reset value: 0x0000 0000

This register points to the start descriptor address of the current transmit descriptor read by the TxDMA controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDAP[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	TDAP[31:0]	Transmit descriptor address pointer bits These bits are automatically updated by TxDMA controller during operation.

22.4.52. DMA current receive descriptor address register (ENET_DMA_CRDADDR)

Address offset: 0x104C

Reset value: 0x0000 0000

This register points to the start descriptor address of the current receive descriptor read by the RxDMA controller.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RDAP[31:16]</td></tr>
<tr><td colspan="16">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RDAP[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:0	RDAP[31:0]	Receive descriptor address pointer bits These bits are automatically updated by RxDMA controller during operation.

22.4.53. DMA current transmit buffer address register (ENET_DMA_CTBADDR)

Address offset: 0x1050

Reset value: 0x0000 0000

This register points to the current transmit buffer address being read by the TxDMA controller.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">TBAP[31:16]</td></tr>
<tr><td colspan="16">r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">TBAP[15:0]</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

Bits	Fields	Descriptions
31:0	TBAP[31:0]	Transmit buffer address pointer bits These bits are automatically updated by TxDMA controller during operation.

22.4.54. DMA current receive buffer address register (ENET_DMA_CRBADDR)

Address offset: 0x1054

Reset value: 0x0000 0000

This register points to the current receive buffer address being read by the RxDMA controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBAP[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	RBAP[31:0]	Receive buffer address pointer bits These bits are automatically updated by RxDMA controller during operation.

23. Universal Serial Bus full-speed device interface (USBD)

The USBD is only available on GD32F103 series.

23.1. Overview

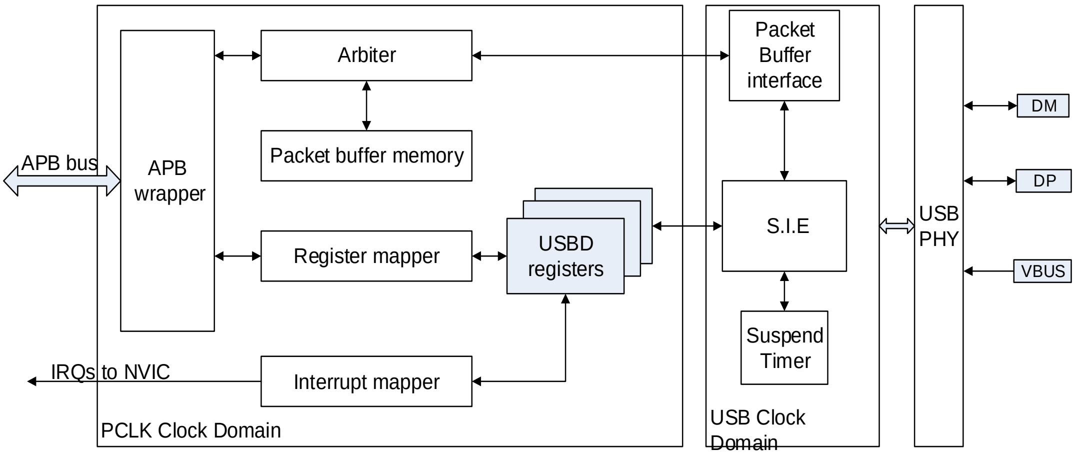
The Universal Serial Bus full-speed device interface (USBD) module provides a device solution for implementing a USB 2.0 full-speed compliant peripheral. It contains a full-speed internal USB PHY and no more external PHY chip is needed. USBD supports all the four types of transfer (control, bulk, interrupt and isochronous) defined in USB 2.0 protocol.

23.2. Characteristics

- USB 2.0 full-speed device controller.
- Support up to 8 configurable bidirectional endpoints.
- Support double-buffered bulk/isochronous endpoints.
- Each endpoint supports control, bulk, isochronous or interrupt transfer types (exclude endpoint 0, endpoint 0 only support control transfer).
- Support USB suspend/resume operations.
- Shared dedicated 512-byte SRAM used for data packet buffer with CAN.
- Integrated USB PHY.

23.3. Block diagram

Figure 23-1. USBD block diagram



23.4. Signal description

Table 23-1. USBD signal description

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D-
DP	Input/Output	Differential D+

Note: As soon as the USBD is enabled, these pins are connected to the USBD internal transceiver automatically.

23.5. Clock configuration

According to the USB standard definition, the USB full-speed module adopt fixed 48MHz clock. It is necessary to configure two clock for using USBD, one is the USB controller clock, its frequency must be configured to 48MHz, and the other one is the APB1 to USB interface clock which is also APB1 bus clock, its frequency can be above or below 48MHz.

Note: In order to meet the system requirements of packet buffer interface and USB data transfer rate, the frequency of the APB1 bus clock must be greater than 24MHz, so as to avoid data buffer overflow and underflow.

48MHz clock of USB controller can be generated by dividing MCU internal or external crystal oscillator by a programmable prescaler, then multiplicating the frequency through PLL.

- Regard two frequency division of 8MHz internal oscillator as the input of the PLL, then 12 frequencies doubling the clock.
- Regard 8MHz external oscillator as the input of the PLL, firstly frequency doubling, then adopt USB frequency divider to divide frequency.

When the USB clock is generated by external crystal, only 4 USB frequency prescaler can be used as 1, 1.5, 2 and 2.5 (2.5 cannot be used, because the frequency of GD32F10x MCU cannot reach above 108MHz). Thus, for obtaining 48MHz clock, PLL frequencies doubling can only be 48MHz, 72MHz and 96MHz.

Note: Regardless of using internal or external crystal oscillator to generate USB clock, the clock accuracy must reach ±500ppm. If the accuracy of the USB clock cannot meet the condition, data transfer may not conform to the requirements of the USB specification, and even it may cause USB not working directly.

23.6. Function overview

23.6.1. USB endpoints

USBD supports 8 USB endpoints that can be individually configured.

Each endpoint supports:

- Single/Double buffer (endpoint 0 can’t use double buffer).
- One endpoint buffer descriptor.
- Programmable buffer starting address and buffer length.
- Configurable response to a packet.
- Control transfer (endpoint 0 only).

Endpoint buffer

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. To efficiently manage USB endpoint communications, USBD implements a dedicated data packet buffer of 512-bytes SRAM memory accessed directly by the USB peripheral. It is mapped to the APB1 peripheral memory, from 0x4000 6000 to 0x4000 6400. The total capacity is 1KB, but USBD uses actually only 512 bytes for the bus width reason.

Each endpoint can be associated with one or two data packet buffers used to store the current data payload. The bidirectional endpoint has usually two buffers, one is used for transmission and the other one is for reception. The mono-directional endpoint only has one buffer for data operation.

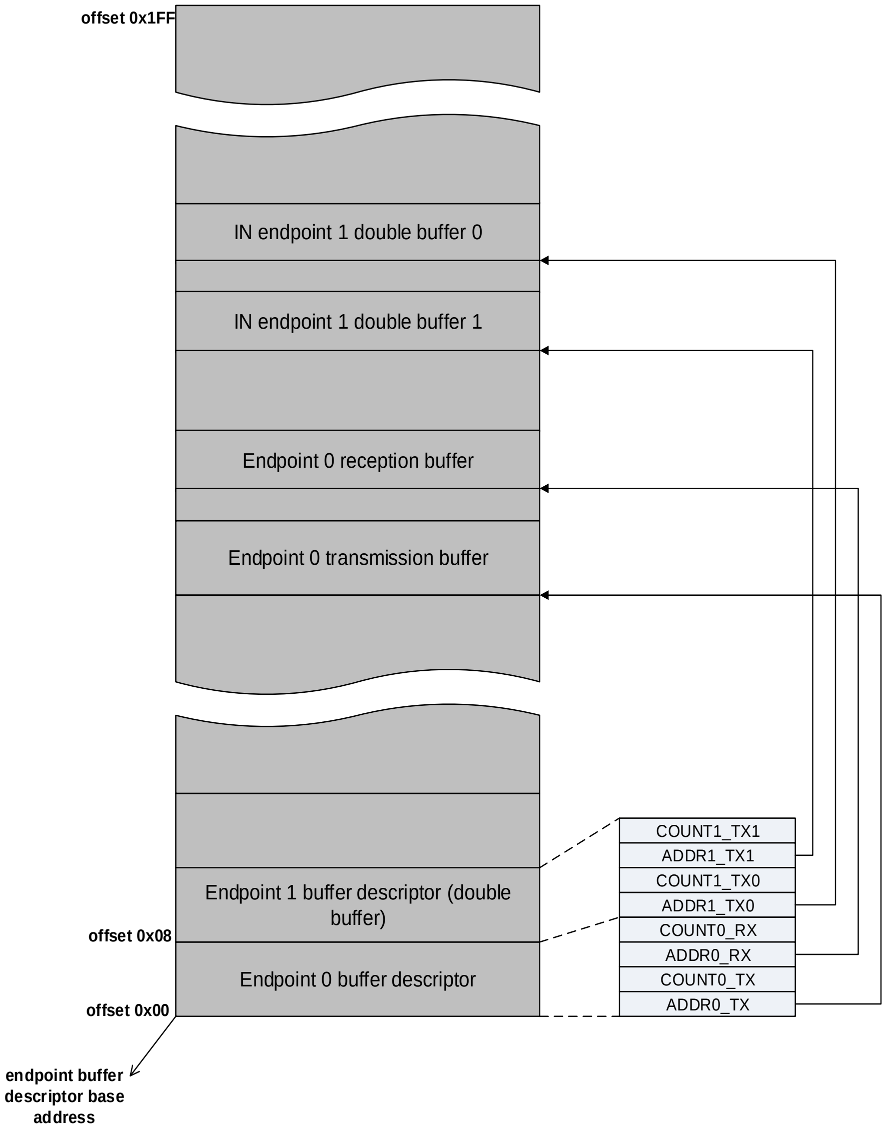
Note: The USBD and CAN share the dedicated 512-byte SRAM memory.

Endpoint buffer descriptor table

To indicate where the endpoint-related buffer is located, how large it is or how many bytes must be transmitted, USBD implements an endpoint buffer descriptor table which defines the buffer address and length and which is also located in the endpoint data packet buffer. The endpoint buffer descriptor is used as a communication port between the application firmware and the SIE in system memory. Every endpoint direction requires two 16-bit words buffer descriptor. Therefore, each table entry includes 4 16-bit words (Tx and Rx two direction) and is aligned to 8-byte boundary. When an endpoint is double-buffered, the SIE will use the two buffers in ping-pong fashion. The endpoint buffer descriptor table is pointed to by the USBD endpoint buffer address register.

The relationship between endpoint buffer descriptor table entries and packet buffer areas is depicted in ***Figure 23-2. An example with buffer descriptor table usage (USBD_BADDR = 0).***

Figure 23-2. An example with buffer descriptor table usage (USBD_BADDR = 0)


offset 0x1FF
IN endpoint 1 double buffer 0
IN endpoint 1 double buffer 1
Endpoint 0 reception buffer
Endpoint 0 transmission buffer
Endpoint 1 buffer descriptor (double buffer)
Endpoint 0 buffer descriptor
offset 0x08
offset 0x00
endpoint buffer descriptor base address
COUNT1_TX1
ADDR1_TX1
COUNT1_TX0
ADDR1_TX0
COUNT0_RX
ADDR0_RX
COUNT0_TX
ADDR0_TX

Note: This figure is not drawn on the actual scale, and it is addressed through the USB bus 16-bit mode.

Double-buffered endpoints

The double-buffered feature is used to improve bulk transfer performance. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USBD_EPxCS register, there are two data toggle bits (TX_DTG and RX_DTG) but only one is used by USBD for hardware data handling (due to the unidirectional constraint required by double-buffering feature), the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called software buffer bit (SW_BUF). In ***Table 23-2. Double-buffering buffer flag definition***, the correspondence between USBD_EPxCS register bits and DTG/SW_BUF definition is explained.

Table 23-2. Double-buffering buffer flag definition

Buffer flag	Tx endpoint	Rx endpoint
DTG	TX_DTG (USBD_EPxCS bit 6)	RX_DTG (USBD_EPxCS bit 14)
SW_BUF	USBD_EPxCS bit 14	USBD_EPxCS bit 6

The DTG bit and the SW_BUF bit are responsible for the flow control. When a transfer completes, the USB peripheral toggle the DTG bit; when the data have been copied, the application software need to toggle the SW_BUF bit. Except for the first time, if the value of DTG bit is equal to the SW_BUF's, the transfer will pause, and the host is NAK. When the two bits are not equal, the transfer resume.

Table 23-3. Double buffer usage

Endpoint Type	DTOG	SW_BUF	Packet buffer used by the USB peripheral	Packet buffer used by the application software
OUT	0	1	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.
	1	0	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
IN	0	1	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
	1	0	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.

Endpoint memory requests arbitration

As the USBD is connected to the APB1 bus through an APB1 interface, so USB APB1 interface will accept memory requests coming from the APB1 bus and from the USB interface. The arbiter will resolve the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, when an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.

23.6.2. Operation procedure

USB transaction process

After the endpoint is configured and a transaction is required, the hardware will detect the token packet. When a token is recognized by the USBD, the data transfer is performed. When all the data has been transferred, the proper handshake packet over the USBD is generated or expected according to the direction of the transfer.

After the transaction process is completed, an endpoint-specific interrupt is generated. In the interrupt routine, the application can process it accordingly.

Transaction formatting is performed by the hardware, including CRC generation and checking.

Once the endpoint is enabled, endpoint control and status register, buffer address and COUNT filed should not be modified by the application software. When the data transfer operation is completed, notified by a STIF interrupt event, they can be accessed again to re-enable a new operation.

IN transaction

When a configured and valid endpoint receives an IN token packet, it will send the data packet to the host. If the endpoint is not valid, a NAK or STALL handshake is sent according to the endpoint status.

In the data packet transfer process, a configured data PID will be sent firstly, then the actual data in endpoint buffer memory is loaded into the output shift register to be transmitted. After the data are sent, the computed CRC will be sent by hardware.

When receiving the ACK sent from the host, then the USB peripheral will toggle the data PID and set the endpoint status to be NAK. At the same time, the successful transfer interrupt will be triggered. In the interrupt service routine, application fill the data packet memory with data, start next transfer by re-enable the endpoint by setting the endpoint status VALID.

OUT and SETUP transaction

USBD handles these two tokens more or less in the same way, the differences in the handling of SETUP packets will be detailed in the following section about control transfer.

After the received endpoint is configured and enabled, host will send OUT/SETUP token to the device. When receiving the token, USBD will access the endpoint buffer descriptor to initialize the endpoint buffer address and length. Then the received data bytes subsequently are packed in words (LSB mode) and transferred to the endpoint buffer. When detecting the end of data packet, the computed CRC and received CRC are compared. If no errors occur, an ACK handshake packet is sent to the host.

When the transaction is completed correctly, USBD will toggle the data PID and set the endpoint status to be NAK. Then the endpoint successful transfer interrupt will be triggered by hardware. In the interrupt service routine, the application can get the transaction type and read the received data from the endpoint buffer. After the received data is processed, the application should initiate further transactions by setting the endpoint status valid.

If any error happens during reception, the USBD set the error interrupt bit and still copy data into the packet memory buffer, but will not send the ACK packet. The USBD itself can recover from reception errors and continue to handle next transfer. The USBD never override outside the data buffer, which is controlled by the internal register configured. The received 2-byte CRC is also copied to the packet memory buffer, immediately following data bytes. If the length of data is greater than actually allocated length, the excess data are not copied. This is a buffer overrun situation. A STALL handshake is sent, and this transaction fails.

If an addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to the endpoint status and no data is written to the endpoint data buffers.

If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated endpoint data buffer, the USBD detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

Control transfers

Control transfers require that a SETUP transaction be started from the host to a device to describe the type of control access that the device should perform. The SETUP transaction is followed by zero or more control DATA transactions that carry the specific information for the requested access. Finally, a STATUS transaction completes the control transfer and allows the endpoint to return the status of the control transfer to the client software. After the STATUS transaction for a control transfer is completed, the host can advance to the next control transfer for the endpoint.

USBD always use endpoint 0 in two directions as default control endpoint to handle control transfers. It is aware of the number and direction of data stages by interpreting the contents of SETUP transaction, and is required to set the unused direction endpoint 0 status to STALL except the last data stage.

At the last data stage, the application software set the opposite direction endpoint 0 status to NAK. This will keep the host waiting for the completion of the control operation. If the operation completes successfully, the software will change NAK to VALID, otherwise to STALL. If the status stage is an OUT, the STATUS_OUT bit should be set, so that a status transaction with non-zero data will be answered STALL to indicate an error happen.

As USB specification states, a device must answer a SETUP packet with an ACK handshake. The USB doesn’t allow a control endpoint to answer with a NAK or STALL handshake packet to a SETUP token when device aborts a previously issued command to start the new one eventually.

When the configured control endpoint 0 receives a SETUP token, the USBD accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued successful receive interrupt request not yet acknowledged by the application (i.e. RX_ST bit is still set from a previously completed reception), the USBD discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the successful receive interrupt.

Isochronous transfers

Isochronous transfers can guarantee constant data rate and bounded latency, but do not support data retransmission in response to errors on the bus. A receiver can determine that a transmission error occurred. The low-level USB protocol does not allow handshakes to be returned to the transmitter of an isochronous pipe. Normally, handshakes would be returned to tell the transmitter whether a packet was successfully received or not. Consequently, the isochronous transaction does not have a handshake phase, and have no ACK packet after the data packet. Data toggling is not supported, and DATA0 PID is only used to start a data packet.

The isochronous endpoint status only can be set DISABLED and VALID, any other value is illegal. The application software can implement double-buffering to improve performance. By swapping transmission and reception data packet buffer on each transaction, the application software can copy the data into or out of a buffer, at the same time the USB peripheral handle the data transmission or reception of data in another buffer. The DTOG bit indicates which buffer that the USB peripheral is currently using.

The application software initializes the DTOG according to the first buffer to be used. At the end of each transaction, the RX_ST or TX_ST bit is set, depending on the enabled direction regardless of CRC errors or buffer-overrun conditions (if errors occur, the ERRIF bit will be set). At the same time, The USB peripheral will toggle the DTOG bit, but will not affect the STAT bit.

23.6.3. USB events and interrupts

Each USB action is always initiated by the application software, driven by one USB interrupt or event. After system reset, the application needs to wait for a succession of USB interrupts and events.

Reset events

System and power-on reset

Upon system and power-on reset, the application software should first provide all required clock to the USB module and interface, then de-assert its reset signal so to be able to access its registers, last switch on the analog part of the device related to the USB transceiver.

The USB firmware should do as follows:

- Reset CLOSE bit in USBD_CTL register.
- Wait for the internal reference voltage to be stable.
- Clear SETRST bit in USBD_CTL register.
- Clear the IFR register to remove the spurious pending interrupt and then enable other unit.

USB reset (RESET interrupt)

When this event occurs, the USB peripheral status is the same as the moment system reset.

The USB firmware should do as follows:

- Set USBEN bit in AR register to enable USB module in 10ms.
- Initialize the USBD_EP0CS register and its related packet buffers.

Suspend and resume events

The USB module can be forced to place in low-power mode (SUSPEND mode) by writing in the USB control register (USBD_CTL) whenever required. At this time, all static power dissipation is avoided and the USB clock can be slowed down or stopped. It will be resumed when detect activity at the USB bus while in low-power mode.

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500mA.
- A suspended device should draw a maximum of 500uA.

A device will go into the suspend state if there is no activity on the USB bus for more than 3ms. A suspended device wakes up, if RESUME signaling is detected.

USBD also supports software initiated remote wakeup. To initiate remote wakeup, the application software must enable all clocks and clear the suspend bit after MCU is waked up. This will cause the hardware to generate a remote wakeup signal upstream.

Setting the SETSPS bit to 1 enables the suspend mode, and it will disable the check of SOF reception. Setting the LOWM bit to 1 will shut down the static power consumption in the analog USB transceivers, but the RESUME signal is still able to be detected.

USB Interrupts

USBD has three interrupts: low-priority interrupt, high-priority interrupt and wakeup interrupt. Software can configure these interrupts to route the interrupt condition to these entries in the NVIC table. An interrupt will be generated when both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit).

- Low-priority interrupt (Channel 20): triggered by all USB events.
- High-priority interrupt (Channel 19): triggered only by a correct transfer event for isochronous and double-buffer bulk transfer.
- Wakeup interrupt (Channel 42): triggered by the wakeup events.

23.6.4. Operation guide

This section describes the operation guide for USBD.

USBD register initialization sequence

1. Clear the CLOSE bit in USBD_CTL register, then clear the SETRST bit.
2. Clear USBD_INTF register to remove any spurious pending interrupt.
3. Program USBD_BADDR register to set endpoint buffer base address.
4. Set USBD_CTL register to enable interrupts.
5. Wait for the reset interrupt (RSTIF).
6. In the reset interrupt, initialize default control endpoint 0 to start enumeration process and program USBD_ADDR to set the device address to 0 and enable USB module function.
7. Configure endpoint 0 and prepare to receive SETUP packet.

Endpoint initialization sequence

1. Program USBD_EPxTBADDR or USBD_EPxRBADDR registers with transmission or reception data buffer address.
2. Program the EP_CTL and EP_KCTL bits in USBD_EPxCS register to set endpoint type and buffer kind according to the endpoint usage.
3. If the endpoint is a single buffer endpoint:

   1) Initialize the endpoint data toggle bit by programming the TX_DTG or RX_DTG bit in USBD_EPxCS register, but endpoint 0 needs to set them to 1 and 0 respectively for control transfer.

   2) Configure endpoint status by programming the TX_STA bit or RX_STA bit in USBD_EPxCS register, but both of them are set to ‘10 (NAK) if use endpoint 0 to initialize the control transfer.

   If the endpoint is a double buffer endpoint:

   1) Both transmission and reception toggle fields need to be programmed. If the endpoint is a Tx endpoint, clear the TX_DTG and RX_DTG bit in USBD_EPxCS register, or if endpoint is a Rx endpoint, it needs to toggle TX_DTG bit.

   2) Program USBD_EPxTBCNT and USBD_EPxRBCNT register to set transfer data bit count.

   3) Endpoint transmission and reception status both need to be configured. If the endpoint is a Tx endpoint, set the TX_STA bit to be NAK and RX_STA bit to be DISABLED, or the endpoint is a Rx endpoint, set the RX_STA bit to be VALID and

TX_STA bit to be DISABLED.

SETUP and OUT data transfers

1. Program USBD_EPxRBCNT register to set BLKNUM and RXCTNT filed, these filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to receive data by programming USBD_EPxCS register.
3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application can get the transaction type by reading the STEUP bit in USBD_EPxCS register. Then application will read the data payload from the endpoint data buffer with the start address defined in USBD_EPxRBAR register. Last application will interpret the data and process the corresponding transaction.

IN data transfers

1. Program USBD_EPxTBCNT register to set TCNT filed, this filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to transmit data by programming USBD_EPxCS register.
3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application needs to update user buffer length and location pointer. Then application fill the endpoint buffer with user buffer data. Last application will configure the endpoint status to be VALID to start next transfer.

23.7. Register definition

23.7.1. USBD control register (USBD_CTL)

Address offset: 0x40

Reset value: 0x0003

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIE	PMOUIE	ERRIE	WKUPIE	SPSIE	RSTIE	SOFIE	ESOFIE	Reserved			RSREQ	SETSPS	LOWM	CLOSE	SETRST
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15	STIE	Successful transfer interrupt enable. 0: Successful transfer interrupt disabled. 1: Interrupt generated when STIF bit in USBD_INTF register is set.
14	PMOUIE	Packet memory overrun/underrun interrupt enable. 0: No interrupt generated when packet memory overrun / underrun. 1: Interrupt generated when PMOUIF bit in USBD_INTF register is set.
13	ERRIE	Error interrupt enable. 0: Error interrupt disabled 1: Interrupt generated when ERRIF bit in USBD_INTF register is set.
12	WKUPIE	Wakeup interrupt enable 0: Wakeup interrupt disabled 1: Interrupt generated when WKUPIF bit in USBD_INTF register is set.
11	SPSIE	Suspend state interrupt enable 0: Suspend state interrupt disabled 1: Interrupt generated when SPSIF bit in USB_IFR register is set.
10	RSTIE	USB reset interrupt enable. 0: USB reset interrupt disabled 1: Interrupt generated when RSTIF bit in USBD_INTF register is set.
9	SOFIE	Start of frame interrupt enable 0: Start of frame interrupt disabled 1: Interrupt generated when SOFIF bit in USBD_INTF register is set.

8	ESOFIE	Expected start of frame interrupt enable 0: Expected start of frame interrupt disabled 1: Interrupt generated when ESOFIF bit in USBD_INTF register is set.
7:5	Reserved	Must be kept at reset value
4	RSREQ	Resume request The software set a resume request to the USB host, and the USB host should drive the resume sequence according the USB specifications 0: No resume request 1: Send resume request.
3	SETSPS	Set suspend The software should set suspend state when SPSIF bit in USBD_INTF register is set. 0: Not set suspend state. 1: Set suspend state.
2	LOWM	Low-power mode When set this bit, the USB goes to low-power mode at suspend state. If resume from suspend state, the hardware reset this bit. 0: No effect 1: Go to low-power mode at suspend state.
1	CLOSE	Close state When this bit is set, the USBD goes to close state, and completely close the USBD and disconnected from the host. 0: Not in close state 1: In close state.
0	SETRST	Set reset When this bit is set, the USBD peripheral should be reset. 0: No reset 1: A reset generated.

23.7.2. USBD interrupt flag register (USBD_INTF)

Address offset: 0x44
Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIF	PMOUIF	ERRIF	WKUPIF	SPSIF	RSTIF	SOFIF	ESOFIF	Reserved			DIR	EPNUM[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r			

Bits	Fields	Descriptions
15	STIF	Successful transfer interrupt flag This bit set by hardware when a successful transaction completes
14	PMOUIF	Packet memory overrun/underrun interrupt flag This bit set by hardware to indicate that the packet memory is inadequate to hold transfer data. The software writes 0 to clear this bit.
13	ERRIF	Error interrupt flag This bit set by hardware when an error happens during transaction. The software writes 0 to clear this bit.
12	WKUPIF	Wakeup interrupt flag This bit set by hardware in the SUSPEND state to indicate that activity is detected. The software writes 0 to clear this bit.
11	SPSIF	Suspend state interrupt flag When no traffic happen in 3ms, hardware set this bit to indicate a SUSPEND request. The software writes 0 to clear this bit.
10	RSTIF	USB reset interrupt flag Set by hardware when the USB RESET signal is detected. The software writes 0 to clear this bit.
9	SOFIF	Start of frame interrupt flag Set by hardware when a new SOF packet arrives, The software writes 0 to clear this bit.
8	ESOFIF	Expected start of frame interrupt flag Set by the hardware to indicate that a SOF packet is expected but not received. The software writes 0 to clear this bit.
7:5	Reserved	Must be kept at reset value
4	DIR	Direction of transaction Set by the hardware to indicate the direction of the transaction 0: IN type 1: OUT type
3:0	EPNUM[3:0]	Endpoint Number Set by the hardware to identify the endpoint which the transaction is directed to

23.7.3. USBD status register (USBD_STAT)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>RX_DP</td><td>RX_DM</td><td>LOCK</td><td colspan="2">SOFLN[1:0]</td><td colspan="11">FCNT[10:0]</td></tr>
<tr><td>r</td><td>r</td><td>r</td><td colspan="2">r</td><td colspan="11">r</td></tr>
</table>

Bits	Fields	Descriptions
15	RX_DP	Receive data + line status Represent the status on the DP line
14	RX_DM	Receive data - line status Represent the status on the DM line
13	LOCK	Locked the USB Set by the hardware indicate that at the least two consecutive SOF have been received
12:11	SOFLN[1:0]	SOF lost number Increment every ESOFIF happens by hardware Cleared once the reception of SOF
10:0	FCNT[10:0]	Frame number counter The Frame number counter incremented every SOF received.

23.7.4. USBD device address register (USBD_DADDR)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td>USBEN</td><td colspan="7">USBDAR[6:0]</td></tr>
<tr><td colspan="8"></td><td>rw</td><td colspan="7">rw</td></tr>
</table>

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	USBEN	USB device enable Set by software to enable the USB device 0: The USB device disabled. No transactions handled.

		1: The USB device enabled.
6:0	USBDAR[6:0]	USBD device address After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR

23.7.5. USBD buffer address register (USBD_BADDR)

Address offset: 0x50

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:3	BAR[12:0]	Buffer address Start address of the allocation buffer(512byte on-chip SRAM), used for buffer descriptor table, packet memory
2:0	Reserved	Must be kept at reset value

23.7.6. USBD endpoint x control and status register (USBD_EPxCS), x=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_ST	RX_ DTG	RX_STA[1:0]		SETUP	EP_CTL[1:0]		EP_ KCTL	TX_ST	TX_ DTG	TX_STA[1:0]		EP_ADDR[3:0]			
rc_w0	t	t		r	rw		rw	rc_w0	t	t		rw			

Bits	Fields	Descriptions
15	RX_ST	Reception successful transferred Set by hardware when a successful OUT/SETUP transaction complete Cleared by software by writing 0
14	RX_DTG	Reception data PID toggle This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint Used to implement the flow control for double-buffered endpoint

		Used to swap buffer for isochronous endpoint
13:12	RX_STA[1:0]	Reception status bits Toggle by writing 1 by software Remain unchanged by writing 0 Refer to the table below
11	SETUP	Setup transaction completed Set by hardware when a SETUP transaction completed.
10:9	EP_CTL[1:0]	Endpoint type control Refer to the table below
8	EP_KCTL	Endpoint kind control The exact meaning depends on the endpoint type Refer to the table below
7	TX_ST	Transmission successful transfer Set by hardware when a successful IN transaction complete Clear by software
6	TX_DTG	Transmission data PID toggle This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint Used to implement the flow control for double-buffered endpoint Used to swap buffer for isochronous endpoint
5:4	TX_STA[1:0]	Status bits, for transmission transfers Refer to the table below
3:0	EP_ADDR	Endpoint address Used to direct the transaction to the target endpoint

Table 23-4. Reception status encoding

RX_STA[1:0]	Meaning
00	**DISABLED:** ignore all reception requests of this endpoint
01	**STALL:** STALL handshake status
10	**NAK**: NAK handshake status
11	**VALID**: enable endpoint for reception

Table 23-5. Endpoint type encoding

EP_CTL[1:0]	Meaning
00	**BULK:** bulk endpoint
01	**CONTROL:** control endpoint
10	**ISO**: isochronous endpoint
11	**INTERRUPT**: interrupt endpoint

Table 23-6. Endpoint kind meaning

EP_CTL[1:0]		EP_KCTL Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT

Table 23-7. Transmission status encoding

TX_STA[1:0]	Meaning
00	**DISABLED:** ignore all transmission requests of this endpoint
01	**STALL:** STALL handshake status
10	**NAK**: NAK handshake status
11	**VALID**: enable endpoint for transmission

23.7.7. USBD endpoint x transmission buffer address register (USBD_EPxTBADDR), x=[0..7]

Address offset: [USBD_BADDR] + x * 16
USB local address: [USBD_BADDR] + x * 8

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:1	EPTXBAR[15:1]	Endpoint transmission buffer address Start address of the packet buffer containing data to be sent when receive next IN token
0	EPTXBAR[0]	Must be set to 0

23.7.8. USBD endpoint x transmission buffer byte count register (USBD_EPxTBCNT), x=[0..7]

Address offset: [USBD_BADDR] + x * 16 + 4
USB local Address: [USBD_BADDR] + x * 8 + 2

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions

15:10	Reserved	Must be kept at reset value
9:0	EPTXCNT[9:0]	Endpoint transmission byte count The number of bytes to be transmitted at next IN token

23.7.9. USBD endpoint x reception buffer address register (USBD_EPxRBADDR), x=[0..7]

Address offset: [USBD_BADDR] + x * 16 + 8
USB local Address: [USB_BADDR] + x * 8 + 4

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:1	EPRBAR[15:1]	Endpoint reception buffer address Start address of packet buffer containing the data received by the endpoint at the next OUT/SETUP token
0	EPRBAR[0]	Must be set to 0

23.7.10. USBD endpoint x reception buffer byte count register (USBD_EPxRBCNT), x=[0..7]

Address offset: [USBD_BADDR] + x * 16 + 12
USB local Address: [USBD_BADDR] + x * 8 + 6

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLKSIZ	BLKNUM[4:0]					EPRCNT[9:0]									
rw	rw					r									

Bits	Fields	Descriptions
15	BLKSIZ	Block size 0: block size is 2 bytes 1: block size is 32 bytes
14:10	BLKNUM[4:0]	Block number

		The number of blocks allocated to the packet buffer
9:0	EPRCNT[9:0]	Endpoint reception byte count The number of bytes to be received at next OUT/SETUP token

24. Universal serial bus full-speed interface (USBFS)

The USBFS is available on GD32F105 and GD32F107 series.

24.1. Overview

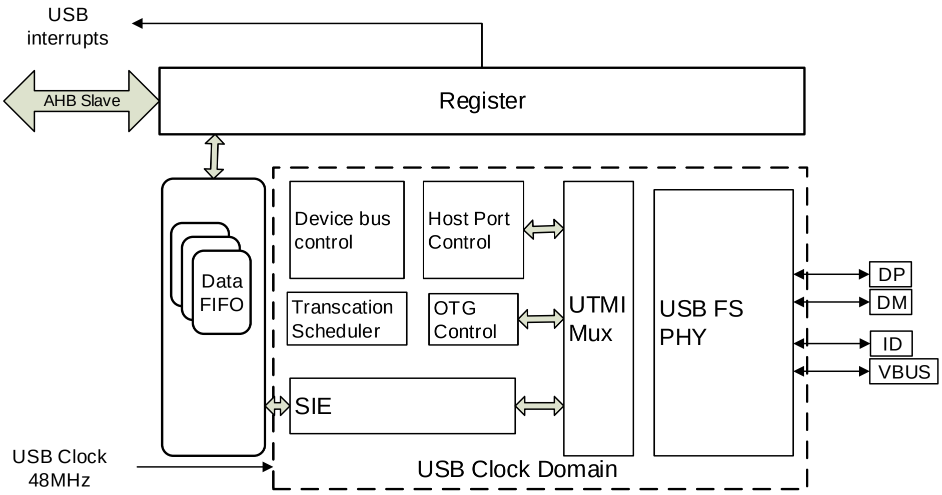
USB Full-Speed (USBFS) controller provides a USB-connection solution for portable devices. USBFS supports host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBFS contains a full-speed internal USB PHY and external PHY chip is not contained. USBFS supports all the four types of transfer (control, bulk, Interrupt and isochronous) which defined in USB 2.0 protocol.

24.2. Characteristics

- Supports USB 2.0 host mode at Full-Speed(12Mb/s) or Low-Speed(1.5Mb/s)
- Supports USB 2.0 device mode at Full-Speed(12Mb/s)
- Supports OTG protocol with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol)
- Supports all the 4 types of transfer: control, bulk, interrupt and isochronous
- Includes a USB transaction scheduler in host mode to handle USB transaction request efficiently.
- Includes a 1.25KB FIFO RAM.
- Supports 8 channels in host mode.
- Includes 2 transmit FIFOs (periodic and non-periodic) and a receive FIFO (shared by all channels) in host mode.
- Includes 4 transmit FIFOs (one for each IN endpoint) and a receive FIFO (shared by all OUT endpoints) in device mode.
- Supports 4 OUT and 4 IN endpoints in device mode.
- Supports remote wakeup in device mode.
- Includes a Full-Speed USB PHY with OTG protocol supported.
- Time intervals of SOFs is dynamic adjustable in host mode
- SOF pulse supports output to PAD.
- Supports detecting ID pin level and VBUS voltage.
- Needs external component to supply power for connected USB device in host mode or OTG A-device mode.

24.3. Block diagram

Figure 24-1. USBFS block diagram



24.4. Signal description

Table 24-1. USBFS signal description

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D-
DP	Input/Output	Differential D+
ID	Input	USB identification: Mini connector identification port

24.5. Function overview

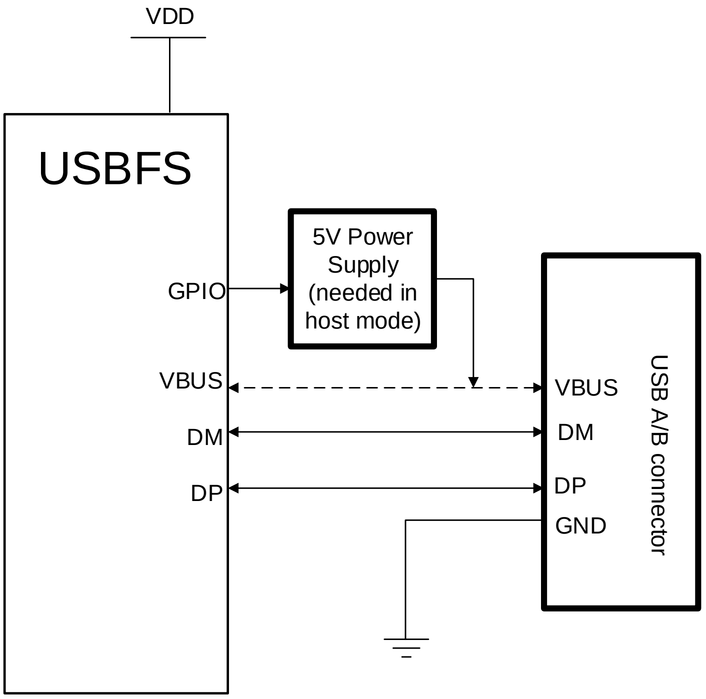
24.5.1. USBFS clocks and working modes

USBFS can operate as a host, a device or a DRD (Dual-role-Device), it contains an internal full-speed PHY.The maximum speed supported by USBFS is full-speed.

The internal PHY supports Full-Speed and Low-Speed in host mode, supports Full-speed in device mode, and supports OTG mode with HNP and SRP. The USB clock used by the USBFS should be 48MHz. The 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU.

The pull-up and pull-down resistors have already been integrated into the internal PHY and they could be controlled by USBFS automatically according to the current mode (host, device or OTG mode) and connection status. A typical connection is shown in ***Figure 24-2. Connection with host or device mode.***

Figure 24-2. Connection with host or device mode

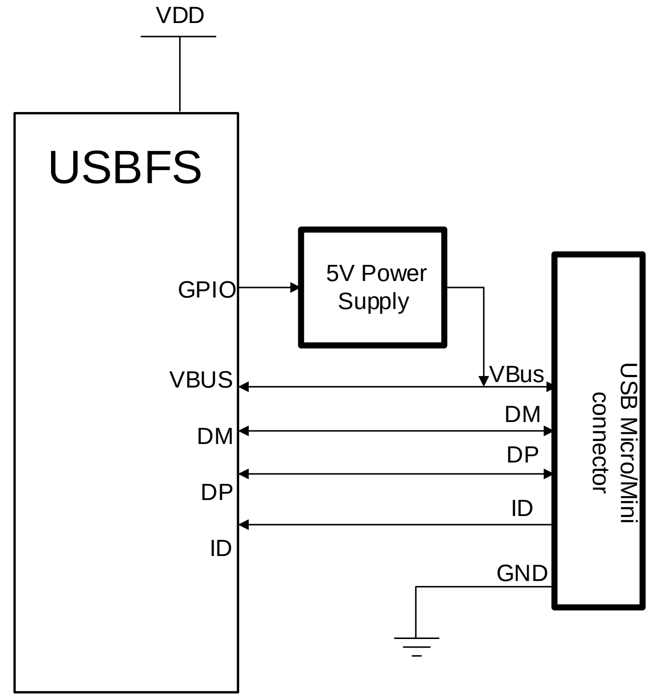


When USBFS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power detecting pin used for voltage detection defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuits on VBUS line. So if application needs VBUS power, an external power supply IC is needed. The VBUS connection between USBFS and the USB connector can be omitted in host mode, so USBFS doesn’t detect the voltage level on VBUS pin and always assumes that the 5V power is present.

When USBFS works in device mode (FHM bit is cleared and FDM bit is set), the VBUS detection circuit is connected to a GPIO pin. USBFS continuously monitor the VBUS voltage by the GPIO pin and will immediately switch on the pull-up resistor on DP line once that the VBUS voltage rise above the needed valid value. This will cause a connection. If the VBUS voltage falls below the needed valid value, the pull-up resistor on DP line will be switched off and a disconnection will happen.

The OTG mode connection is described in the ***Figure 24-3. Connection with OTG mode***. When USBFS works in OTG mode, the FHM, FDM bits in USBFS_GUSBCS and VBUSIG bit in USBFS_GCCFG should be cleared. In this mode, the USBFS needs all the four pins: DM, DP, VBUS and ID, and needs to use several voltage comparers to monitor the voltage on these pins. USBFS also contains VBUS charge and discharge circuits to perform SRP request described in OTG protocol. The OTG A-device or B-device is decided by the level of ID pins. USBFS controls the pull-up or pull-down resistor during performing the HNP protocol.

Figure 24-3. Connection with OTG mode

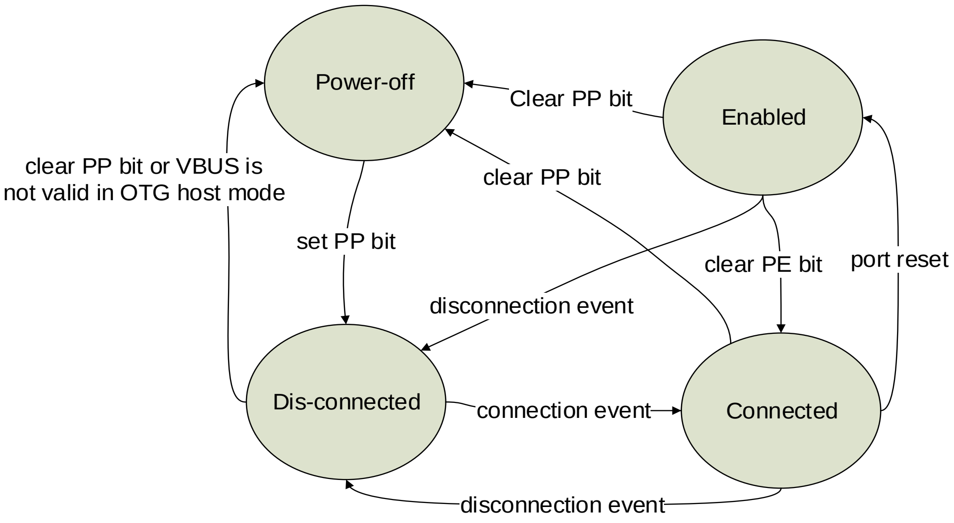


24.5.2. USB host function

USB Host Port State

Host application may control state of the USB port via USBFS_HPCS register. After system initialization, the USB port stays at power-off state. After PP bit is set by software, the internal USB PHY is powered on, and the USB port changes into disconnected state. After a connection is detected, USB port changes into connected state. The USB port changes into enabled state after a port reset is performed on USB bus.

Figure 24-4. State transition diagram of host port



Connection, Reset and Speed identification

As a USB host, USBFS will trigger a connection flag for application after a connection is detected and will trigger a disconnection flag after a disconnection event.

PRST bit is used for USB reset sequence. Application may set this bit to start a USB reset and clear this bit to finish the USB reset. This bit only takes effect when port is at connected or enabled state.

The USBFS performs speed identification during connection, and the speed information will be reported in PS filed in USBFS_HPCS register. USBFS identifies the device speed by the voltage level of DM or DP. As described in USB protocol, full-speed device pulls up DP line while low-speed device pulls up DM line.

Suspend and resume

USBFS supports suspend state and resume operation. When USBFS port is at enabled state, writing 1 to PSP bit in USBFS_HPCS register will cause USBFS to enter suspend state. In suspend state, USBFS stops sending SOFs on USB bus and this will cause the connected USB device to enter suspend state after 3ms. Application can set the PREM bit in USBFS_HPCS register to start a resume sequence to wake up the suspended device and clear this bit to stop the resume sequence. The WKUPIF bit in USBFS_GINTF will be set and the USBFS wake up interrupt will be triggered if a host in suspend state detects a remote wakeup signal.

SOF generate

USBFS sends SOF tokens on USB bus in host mode. As described in USB 2.0 protocol, SOF packets are generated (by the host controller or hub transaction translator) every 1ms in full-speed links.

Each time after USBFS enters into enabled state, it will send the SOF packet periodically which the time is defined in USB 2.0 protocol. In addition, application may adjust the length of a frame by writing FRI filed in USBFS_HFT registers. The FRI bits define the number of USB clock cycles in a frame, so its value should be calculated based on the frequency of USB clock which is used by USBFS. The FRT filed bits show that the remaining clock cycles of the current frame and stop changing during suspend state.

USBFS is able to generate a pulse signal each SOF packet and output it to a pin. The pulse length is 16 HCLK cycle. If application desires to use this function, it needs to set SOFOEN bit in USBFS_GCCFG register and configure the related pin registers in GPIO.

USB Channels and Transactions

USBFS includes 8 independent channels in host mode. Each channel is able to communicate with an endpoint in USB device. The transfer type, direction, packet length and other information are all configured in channel related registers such as USBFS_HCHxCTL and USBFS_HCHxLEN.

USBFS supports all the four kinds of transfer types: control, bulk, interrupts and isochronous. USB 2.0 protocol divides these transfers into 2 kinds: non-periodic transfer (control and bulk) and periodic transfer (interrupt and isochronous). Based on this, USBFS includes two request queues: periodic request queue and non-periodic request queue, to perform efficient transaction schedule. A request entry in a request queue described above may represent a USB transaction request or a channel operation request.

Application needs to write packet into data FIFO via AHB register interface if it wants to start an OUT transaction on USB bus. USBFS hardware will automatically generate a transaction request entry in request queue after the application writes a whole packet.

The request entries in request queue are processed in order by transaction control module. USBFS always tries to process periodic request queue firstly and secondly process non-periodic request queue.

After a start of frame, USBFS begins to process periodic queue until the queue is empty or bus time required by the current periodic request is not enough, and then process the non-periodic queue. This strategy ensures the bandwidth of periodic transactions in a frame. Each time the USBFS reads and pops a request entry from request queue. If this is a channel disable request, it immediately disables the channel and prepares to process the next entry.

If the current request is a transaction request and the USB bus time is enough for this transaction, USBFS will employ SIE to generate this transaction on USB bus.

When the required bus time for the current request is not enough in the current frame, and if this is a periodic request, USBFS stops processing the periodic queue and starts to process non-periodic request. If this is a non-periodic queue the USBFS will stop processing any queue and wait until the end of current frame.

24.5.3. USB device function

USB Device Connection

In device mode, USBFS stays at power-off state after initialization. After connecting to a USB host with 5V power supply through VBUS pin, USBFS enters into powered state. USBFS begins to switch on the pull-up resistor on DP line and thus, host side will detect a connection event.

Note: the VBUS pin must be connected to the PA9 for detecting the level.

Reset and Speed-Identification

The USB host always starts a USB reset when it detects a device connection, and USBFS in device mode will trigger a reset interrupt by hardware when it detects the reset event on USB bus.

After reset sequence, USBFS will trigger an ENUMF interrupt in USBFS_GINTF register and reports current enumerated device speed in ES bits in USBFS_DSTAT register, this bit field is always 11(full-speed).

As required by USB 2.0 protocol, USBFS doesn’t support low-speed in device mode.

Suspend and Wake-up

A USB device will enter into suspend state when the USB bus stays at IDLE state and there is no change on data lines for 3ms. When USB device is in suspend state, most of its clock are closed to save power. The USB host is able to wake up the suspended device by generating a resume signal on USB bus. When USBFS detects the resume signal, the WKUPIF flag in USBFS_GINTF register will be set and the USBFS wake up interrupt will be triggered.

In suspend mode, USBFS is also able to remotely wake up the USB bus. Software may set RWKUP bit in USBFS_DCTL register to send a remote wake-up signal, and if remote wake-up is supported in USB host, the host will begin to send resume signal on USB bus.

Soft Disconnection

USBFS supports soft disconnection. After the device is powered on, USBFS will switch on the pull-up resistor on DP line so that the host can detect the connection. It is able to force a disconnection by setting the SD bit in USBFS_DCTL register. After the SD bit is set, USBFS will directly switch off the pull-up resistor, so that USB host will detect a disconnection on USB bus.

SOF tracking

When USBFS receives a SOF packet on USB bus, it will trigger a SOF interrupt and begin to count the bus time using local USB clock. The frame number of the current frame is reported in FNRSOF filed in USBFS_DSTAT register. When the USB bus time reaches EOF1 or EOF2 point (End of Frame, described in USB 2.0 protocol), USBFS will trigger an EOPFIF interrupt in USBFS_GINTF register. These flags and registers can be used to get current bus time and position information.

24.5.4. OTG function overview

USBFS supports OTG function described in OTG protocol 1.3, OTG function includes SRP and HNP protocols.

A-Device and B-Device

A-Device is an OTG capable USB device with a Standard-A or Micro-A plug inserted into its receptacle. The A-Device supplies power to VBUS and it is host at the start of a session. B-Device is an OTG capable USB device with a Standard-B, Micro-B or Mini-B plug inserted into its receptacle, or a captive cable ending being a Standard-A plug. The B-Device is a peripheral at the start of a session. USBFS uses the voltage level of ID pin to identify A-Device or B-Device. The ID status is reported in IDPS bit in USBFS_GOTGCS register. For the details of transfer states between A-Device and B-Device, please refer to OTG 1.3 protocol.

HNP

The Host Negotiation Protocol (HNP) allows the host function to be switched between two directly connected On-The-Go devices and eliminates the necessity of switching the cable connections for the change of control of communications between the devices. HNP will be initialized typically by the user or an application on the On-The-Go B-Device. HNP may only be implemented through the Micro-AB receptacle on a device.

Since On-The-Go devices have a Micro-AB receptacle, an On-The-Go device can default to being either a host or a device, depending that which type of plug (Micro-A plug for host, Micro-B plug for device) is inserted. By utilizing the Host Negotiation Protocol (HNP), an On-The-Go B-Device, which is the default device, may make a request to be a host. The process for the exchange of the role to a host is described in this section. This protocol eliminates the necessity of switching the cable connection for the change of the roles of the connected devices.

When USBFS is in OTG A-Device host mode and it wants to give up its host role, it may firstly set PSP bit in USBFS_HPCS register to make the USB bus enter in suspend status. Then, the B-Device will enter in suspend state 3ms later. If the B-Device wants to change to be a host, HNPREQ bit in USBFS_GOTGCS register should be set and the USBFS will begin to perform HNP protocol on bus, and at last, the result of HNP is reported in HNPS bit in USBFS_GOTGCS register. Besides, it is always available to get the current role (host or device) from COPM bit in USBFS_GINTF register.

SRP

The Session Request Protocol (SRP) allows a B-Device to request the A-Device to turn on VBUS and start a session. This protocol allows the A-Device, which may be battery powered, to conserve power by turning VBUS off when there is no bus activity while still providing a means for the B-Device to initiate bus activity. As described in OTG protocol, an OTG device must compare VBUS voltage with several threshold values and the compare result should be reported in ASV and BSV bits in USBFS_GOTGCS register.

Set SRPREQ bit in USBFS_GOTGCS register to start a SRP request when USBFS is in B-Device OTG mode and USBFS will generate a success flag SRPS in USBFS_GOTGCS register if the SRP request successfully.

When USBFS is in OTG A-Device mode and it has detected a SRP request from a B-Device, it sets a SESIF flag in USBFS_GINTF register. The 5V power supply for VBUS pin should be prepared to switch on after getting this flag.

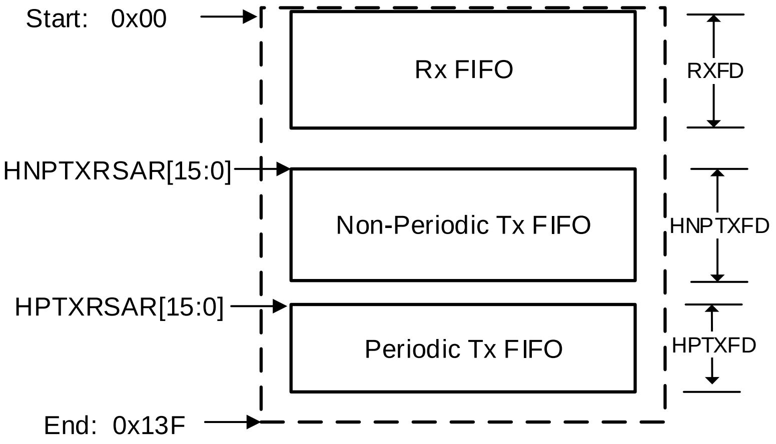
24.5.5. Data FIFO

The USBFS contains a 1.25K bytes data FIFO for packet data storage. The data FIFO is implemented by using an internal SRAM in USBFS.

Host Mode

In host mode, the data FIFO space is divided into 3 parts: Rx FIFO for received packet, Non-Periodic Tx FIFO for non-period transmission packet and Periodic Tx FIFO for periodic transmission packet. All IN channels shares the Rx FIFO for packets reception. All the periodic OUT channels share the periodic Tx FIFO to packets tramsmission. All the non-periodic OUT channels share the non-Periodic Tx FIFO for transmit packets. The size and start offset of these data FIFOs should be configured using these registers: USBFS_GRFLEN, USBFS_HNPTFLEN and USBFS_HPTFLEN. ***Figure 24-5. HOST mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

Figure 24-5. HOST mode FIFO space in SRAM



USBFS provides a special register area for the internal data FIFO reading and writing. ***Figure 24-6. Host mode FIFO access register map*** describes the register memory area that the data FIFO can write. This area can be read by any channel data FIFO.The addresses in the figure are addressed in bytes. Each channel has its own FIFO access register space, although all Non-periodic channels share the same FIFO and all the Periodic channels also share the same FIFO. It is important for USBFS to know which channel the current pushed packet belongs to. Rx FIFO is also able to be accessed using USBFS_GRSTATR/ USBFS_GRSTATP register.

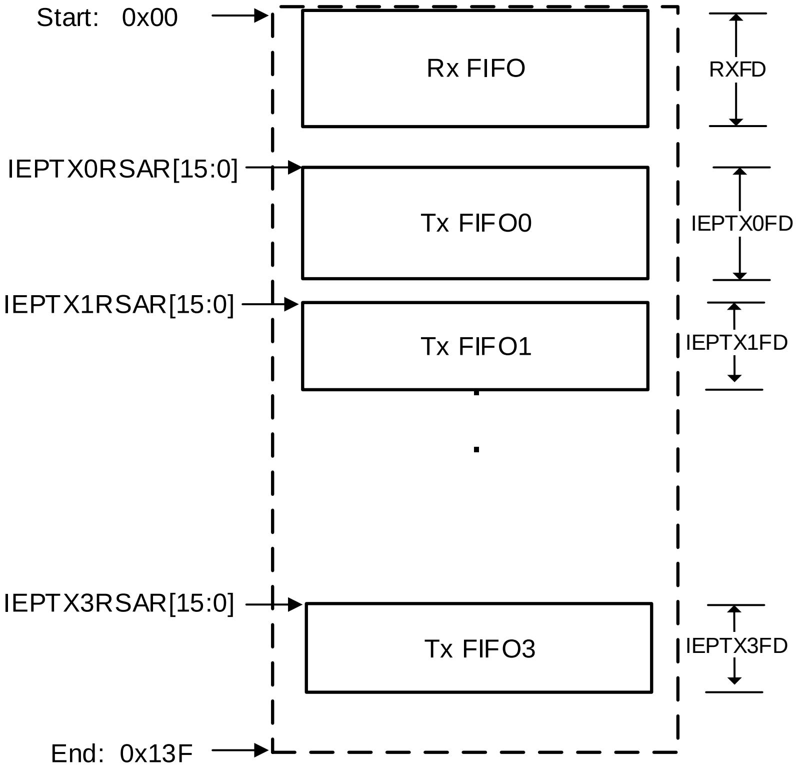
Figure 24-6. Host mode FIFO access register map

1000h-1FFFh	CH0 FIFO Write/Read
2000h-2FFFh	CH1 FIFO Write/Read
	⋮
8000h-8FFFh	CH7 FIFO Write/Read

Device mode

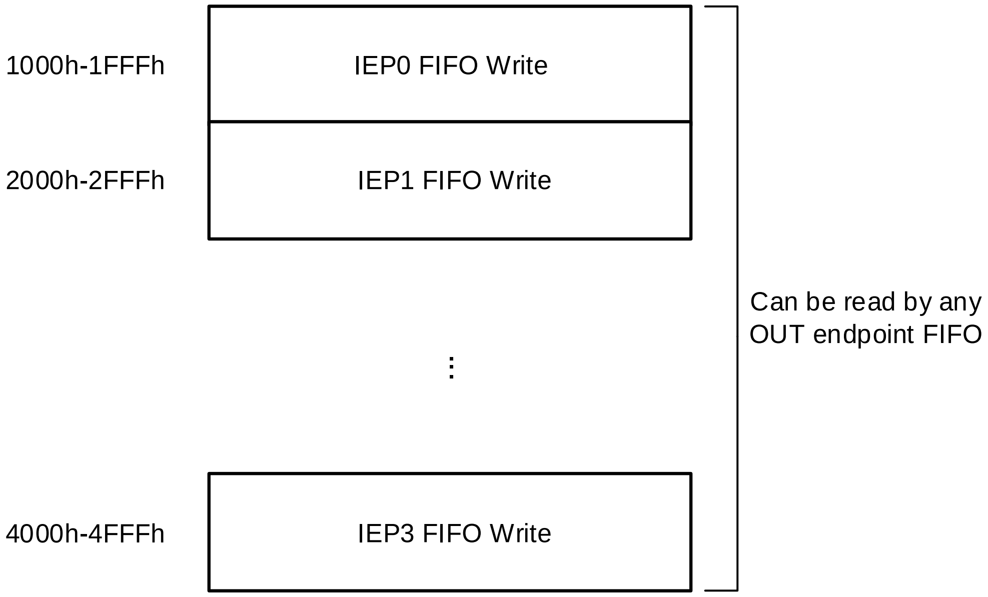
In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 4 Tx FIFOs (one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets. The size and start offset of these data FIFOs should be configured using USBFS_GRFLEN and USBFS_DIEPxTFLEN (x=0…3) registers. ***Figure 24-7. Device mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

Figure 24-7. Device mode FIFO space in SRAM



USBFS provides a special register area for the internal data FIFO reading and writing. ***Figure 24-8. Device mode FIFO access register map*** describes the register memory area where the data FIFO can write. This area can be read by any endpoint FIFO. The addresses in the figure are addressed in bytes. Each endpoint has its own FIFO access register space. Rx FIFO is also able to be accessed using USBFS_GRSTATR/USBFS_GRSTATP register.

Figure 24-8. Device mode FIFO access register map



24.5.6. Operation guide

This section describes the advised operation guide for USBFS.

Host mode

Global register initialization sequence

1. Program USBFS_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBFS_GUSBCS register according to application's demand, such as the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.
3. Program USBFS_GCCFG register according to application's demand.
4. Program USBFS_GRFLEN, USBFS_HNPTFLEN_DIEP0TFLEN and USBFS_HPTFLEN register to configure the data FIFOs according to application's demand.
5. Program USBFS_GINTEN register to enable Mode Fault and Host Port interrupt and set GINTEN bit in USBFS_GAHBCS register to enable global interrupt.
6. Program USBFS_HPCS register and set PP bit.
7. Wait for a device's connection, and once a device is connected, the connection interrupt PCD in USBFS_HPCS register will be triggered. Then set PRST bit to perform a port reset. Wait for at least 10ms and then clear PRST bit.
8. Wait PEDC interrupt in USBFS_HPCS register and then read PE bit to ensure that the port is successfully enabled. Read PS [1:0] bits to get the connected device's speed and then program USBFS_HFT register to change the SOF interval if needed.

Channel initialization and enable sequence

1. Program USBFS_HCHxCTL registers with desired transfer type, direction, packet size, etc. Ensure that CEN and CDIS bits keep cleared during configuration.

2. Program USBFS_HCHxINTEN register. Set the desired interrupt enable bits.

3. Program USBFS_HCHxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

   For OUT channel: If PCNT=1, the single packet's size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBFS_HCHxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If software want s to send out a zero-length packet, it should program TLEN=0, PCNT=1.

   For IN channel: Because the application doesn't know the actual received data size before the IN transaction finishes, TLEN can be set to a maximum possible value supported by Rx FIFO.

4. Set CEN bit in USBFS_HCHxCTL register to enable the channel.

Channel disable sequence

Software can disable the channel by setting both CEN and CDIS bits at the same time. USBFS will generate a channel disable request entry in request queue after the register setting operation. When the request entry reaches the top of request queue, it is processed by USBFS immediately:

For OUT channels, the specified channel will be disabled immediately. Then, a CH flag will be generated and the CEN and CDIS bits will be cleared by USBFS.

For IN channels, USBFS pushes a channel disable status entry into Rx FIFO. Software should then handle the Rx FIFO not empty event: read and pop this status entry, then, a CH flag will be generated and the CEN and CDIS bits will be cleared.

IN transfers operation sequence

1. Initialize USBFS global registers.

2. Initialize the channel.

3. Enable the channel.

4. After the IN channel is enabled by software, USBFS generates an Rx request entry in the corresponding request queue.

5. When the Rx request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the IN transaction indicated by the request entry is enough, USBFS starts the IN transaction on USB bus.

6. If the IN transaction finishes successfully (ACK handshake received), USBFS pushes the

received data packet into the Rx FIFO and triggers ACK flag. Otherwise, the status flag (NAK) reports the transaction result.

7. If the IN transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to receive the remaining packets. If the IN transaction described in step 5 is not successful, return to step 3 to re-receive the packet again.

8. After all the transactions in a transfer are successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and poping all the received data packet, the TF status entry is need, USBFS generates TF flag to indicate that the transfer successfully finishes.

9. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

OUT transfers operation sequence

1. Initialize USBFS global registers.

2. Initialize and enable the channel.

3. Write a packet into the channel's Tx FIFO (Periodic Tx FIFO or non-periodic Tx FIFO). After the whole packet data is written into the FIFO, USBFS generates a Tx request entry in the corresponding request queue and decreases the TLEN field in USBFS_HCHxLEN register by the written packet's size.

4. When the request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the transaction indicated by the request entry is enough, USBFS starts the OUT transaction on USB bus.

5. When the OUT transaction indicated by the request entry finishes on USB bus, PCNT in USBFS_HCHxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flag (NAK) reports the transaction result.

6. If the OUT transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to send the remaining packets. If the OUT transaction described in step 5 is not successful, return to step 3 to resend the packet again.

7. After all the transactions in a transfer are successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully finishes.

8. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

Device mode

Global register initialization sequence

1. Program USBFS_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.

2. Program USBFS_GUSBCS register according to application’s demand, such as: the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.

3. Program USBFS_GCCFG register according to application’s demand.

4. Program USBFS_GRFLEN, USBFS_HNPTFLEN_DIEP0TFLEN, USBFS_DIEPxTFLEN register to configure the data FIFOs according to application’s demand.

5. Program USBFS_GINTEN register to enable Mode Fault, Suspend, SOF, Enumeration Done and USB Reset interrupt and then, set GINTEN bit in USBFS_GAHBCS register to enable global interrupt.

6. Program USBFS_DCFG register according to application’s demand, such as the device address, etc.

7. After the device is connected to a host, the host will perform port reset on USB bus and this will trigger the RST interrupt in USBFS_GINTF register.

8. Wait for ENUMF interrupt in USBFS_GINTF register.

Endpoint initialization and enable sequence

1. Program USBFS_DIEPxCTL or USBFS_DOEPxCTL register with desired transfer type, packet size, etc.

2. Program USBFS_DIEPINTEN or USBFS_DOEPINTEN register. Set the desired interrupt enable bits.

3. Program USBFS_DIEPxLEN or USBFS_DOEPxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

   For IN endpoint：If PCNT=1, the single packet’s size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBFS_DIEPxCTL register, and the last packet’s size is calculated based on PCNT, TLEN and MPL. If a zero-length packet is required to be sent, it should program TLEN=0, PCNT=1.

   For OUT endpoint：Because the application doesn’t know the actual received data size before the OUT transaction finishes, TLEN can be set to a maximum possible value supported by Rx FIFO.

4. Set EPEN bit in USBFS_DIEPxCTL or USBFS_DOEPxCTL register to enable the endpoint.

Endpoint disable sequence

The endpoint can be disabled anytime when the EPEN bit in USBFS_DIEPxCTL or USBFS_DOEPxCTL registers is cleared.

IN transfers operation sequence

1. Initialize USBFS global registers.

2. Initialize and enable the IN endpoint.

3. Write packets into the endpoint's Tx FIFO. Each time a data packet is written into the FIFO, USBFS decreases the TLEN field in USBFS_DIEPxLEN register by the written packet's size.

4. When an IN token received, USBFS transmits the data packet, and after the transaction finishes on USB bus, PCNT in USBFS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags reports the transaction result.

5. After all the data packets in a transfer are successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the IN endpoint.

OUT transfers operation sequence

1. Initialize USBFS global registers.

2. Initialize the endpoint and enable the endpoint.

3. When an OUT token received, USBFS receives the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBFS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBFS_DOEPxLEN register is decreased by 1 and the ACK flag is triggered, otherwise, the status flags report the transaction result.

4. After all the data packets in a transfer are successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and poping all the received data packet, the TF status entry is read, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the OUT endpoint.

24.6. Interrupts

USBFS has two interrupts: global interrupt and wake-up interrupt.

The source flags of the global interrupt are readable in USBFS_GINTF register and are listed in ***Table 24-2. USBFS global interrupt***.

Table 24-2. USBFS global interrupt

Interrupt Flag	Description	Operation Mode
SEIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	Host or device mode

Interrupt Flag	Description	Operation Mode
PTXFEIF	Periodic Tx FIFO empty interrupt flag	Host Mode
HCIF	Host channels interrupt flag	Host Mode
HPIF	Host port interrupt flag	Host Mode
ISOONCIF/PXNCIF	Periodic transfer Not Complete Interrupt flag /Isochronous OUT transfer Not Complete Interrupt Flag	Host or device mode
ISOINCIF	Isochronous IN transfer Not Complete Interrupt Flag	Device mode
OEPIF	OUT endpoint interrupt flag	Device mode
IEPIF	IN endpoint interrupt flag	Device mode
EOPFIF	End of periodic frame interrupt flag	Device mode
ISOOPDIF	Isochronous OUT packet dropped interrupt flag	Device mode
ENUMF	Enumeration finished	Device mode
RST	USB reset	Device mode
SP	USB suspend	Device mode
ESP	Early suspend	Device mode
GONAK	Global OUT NAK effective	Device mode
GNPINAK	Global IN Non-Periodic NAK effective	Device mode
NPTXFEIF	Non-Periodic Tx FIFO empty interrupt flag	Host Mode
RXFNEIF	Rx FIFO non-empty interrupt flag	Host or device mode
SOF	Start of frame	Host or device mode
OTGIF	OTG interrupt flag	Host or device mode
MFIF	Mode fault interrupt flag	Host or device mode

Wake-up interrupt can be triggered when USBFS is in suspend state, even when the USBFS's clocks are stopped. The source of the wake-up interrupt is WKUPIF bit in USBHS_GINTF register.

24.7. Register definition

24.7.1. Global control and status registers

Global OTG control and status register (USBFS_GOTGCS)

Address offset: 0x0000

Reset value: 0x0000 0800

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												BSV	ASV	DI	IDPS
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DHNPEN	HHNPEN	HNPREQ	HNPS	Reserved						SRPREQ	SRPS
				rw	rw	rw	r							rw	r

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19	BSV	B-Session Valid (described in OTG protocol). 0: Vbus voltage level of a OTG B-Device is below $V_{BSESSVLD}$ 1: Vbus voltage level of a OTG B-Device is not below $V_{BSESSVLD}$ **Note:** Only accessible in OTG B-Device mode.
18	ASV	A- Session valid A-host mode transceiver status. 0: Vbus voltage level of a OTG A-Device is below $V_{ASESSVLD}$ 1: Vbus voltage level of a OTG A-Device is below $V_{ASESSVLD}$ The A-Device is the default host at the start of a session. **Note:** Only accessible in OTG A-Device mode.
17	DI	Debounce interval Debounce interval of a detected connection. 0: Indicates the long debounce interval , when a plug-on and connection occurs on USB bus 1: Indicates the short debounce interval, when a soft connection is used in HNP protocol.

		Note: Only accessible in host mode.
16	IDPS	ID pin status Voltage level of connector ID pin 0: USBFS is in A-Device mode 1: USBFS is in B-Device mode **Note:** Accessible in both device and host modes.
15:12	Reserved	Must be kept at reset value
11	DHNPEN	Device HNP enable Enable the HNP function of a B-Device. If this bit is cleared, USBFS doesn’t start HNP protocol when application set HNPREQ bit in USBFS_GOTGCS register. 0: HNP function is not enabled. 1: HNP function is enabled **Note:** Only accessible in device mode.
10	HHNPEN	Host HNP enable Enable the HNP function of an A-Device. If this bit is cleared, USBFS doesn’t response to the HNP request from B-Device. 0: HNP function is not enabled. 1: HNP function is enabled **Note:** Only accessible in host mode.
9	HNPREQ	HNP request This bit is set by software to start a HNP on the USB. This bit can be cleared when HNPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the HNPEND bit in USBFS_GOTGINTF register. 0: Don’t send HNP request 1: Send HNP request **Note:** Only accessible in device mode.
8	HNPS	HNP successes This bit is set by the core when HNP succeeds, and this bit is cleared when HNPREQ bit is set. 0: HNP fails 1: HNP succeeds **Note:** Only accessible in device mode.
7:2	Reserved	Must be kept at reset value
1	SRPREQ	SRP request This bit is set by software to start a SRP on the USB. This bit can be cleared when SRPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the SRPEND bit in USBFS_GOTGINTF register. 0: No session request 1: Session request

		Note: Only accessible in device mode.
0	SRPS	SRP success This bit is set by the core when SRP succeeds, and this bit is cleared when SRPREQ bit is set. 0: SRP fails 1: SRP succeeds **Note:** Only accessible in device mode.

Global OTG interrupt flag register (USBFS_GOTGINTF)

Address offset: 0x0004

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												DF	ADTO	HNPDET	Reserved
												rc_w1	rc_w1	rc_w1	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						HNPEND	SRPEND	Reserved					SESEND	Reserved	
						rc_w1	rc_w1						rc_w1		

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19	DF	Debounce finish Set by USBFS when the debounce during device connection is done. **Note:** Only accessible in host mode.
18	ADTO	A-Device timeout Set by USBFS when the A-Device’s waiting for a B-Device’ connection has timed out. **Note:** Accessible in both device and host modes.
17	HNPDET	Host negotiation request detected Set by USBFS when A-Device detects a HNP request. **Note:** Accessible in both device and host modes.
16:10	Reserved	Must be kept at reset value
9	HNPEND	HNP end

		Set by the core when a HNP ends. Read the HNPS in USBFS_GOTGCS register to get the result of HNP. **Note:** Accessible in both device and host modes.
8	SRPEND	SRPEND Set by the core when a SRP ends. Read the SRPS in USBFS_GOTGCS register to get the result of SRP. **Note:** Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value
2	SESEND	Session end Set by the core when VBUS voltage is below $V_{b\_ses\_vld}$.
1:0	Reserved	Must be kept at reset value

Global AHB control and status register (USBFS_GAHBCS)

Address offset: 0x0008

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							PTXFTH	TXFTH	Reserved						GINTEN
							rw	rw							rw

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8	PTXFTH	Periodic Tx FIFO threshold 0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty 1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty **Note:** Only accessible in host mode.
7	TXFTH	Tx FIFO threshold **Device mode**: 0: TXFEIF will be triggered when the IN endpoint transmit FIFO is half empty 1: TXFEIF will be triggered when the IN endpoint transmit FIFO is completely

		empty **Host mode**: 0: NPTXFEIF will be triggered when the non-periodic transmit FIFO is half empty 1: NPTXFEIF will be triggered when the non-periodic transmit FIFO is completely empty
6：1	Reserved	Must be kept at reset value
0	GINTEN	Global interrupt enable 0: Global interrupt is not enabled. 1: Global interrupt is enabled. **Note:** Accessible in both device and host modes.

Global USB control and status register (USBFS_GUSBCS)

Address offset: 0x000C

Reset value: 0x0000 0A80

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	FDM	FHM	Reserved												
	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UTT[3:0]				HNPCEN	SRPCEN	Reserved					TOC[2:0]		
		rw				r/rw	r/rw						rw		

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30	FDM	Force device mode Setting this bit will force the core to device mode irrespective of the USBFS ID input pin. 0: Normal mode 1: Device mode The application must wait at least 25 ms for the change taking effect after setting the force bit. **Note:** Accessible in both device and host modes.
29	FHM	Force host mode

		Setting this bit will force the core to host mode irrespective of the USBFS ID input pin. 0: Normal mode 1: Host mode The application must wait at least 25 ms for the change taking effect after setting the force bit. **Note:** Accessible in both device and host modes.
28:14	Reserved	Must be kept at reset value
13:10	UTT[3:0]	USB turnaround time Turnaround time in PHY clocks. **Note:** Only accessible in device mode.
9	HNPCEN	HNP capability enable Controls whether the HNP capability is enabled 0: HNP capability is disabled 1: HNP capability is enabled **Note:** Accessible in both device and host modes.
8	SRPCEN	SRP capability enable Controls whether the SRP capability is enabled 0: SRP capability is disabled 1: SRP capability is enabled **Note:** Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value
2:0	TOC[2:0]	Timeout calibration USBFS always uses time-out value required in USB 2.0 when waiting for a packet. Application may use TOC [2:0] to add the value is in terms of PHY clock. (The frequency of PHY clock is 48MHZ.).

Global reset control register (USBFS_GRSTCTL)

Address offset: 0x0010

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TXFNUM[4:0]					TXFF	RXFF	Reserved	HFCRST	HCSRST	CSRST
					rw					rs	rs		rs	rs	rs

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10:6	TXFNUM[4:0]	Tx FIFO number Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set. Host Mode: 00000: Only non-periodic Tx FIFO is flushed 00001: Only periodic Tx FIFO is flushed 1XXXX: Both periodic and non-periodic Tx FIFOs are flushed Other: Non data FIFO is flushed Device Mode: 00000: Only Tx FIFO0 is flushed 00001: Only Tx FIFO1 is flushed … 00011: Only Tx FIFO3 is flushed 1XXXX: All Tx FIFOs are flushed Other: Non data FIFO is flushed
5	TXFF	Tx FIFO flush Application set this bit to flush data Tx FIFOs and TXFNUM[4:0] bits decide the FIFO number to be flushed. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS. **Note:** Accessible in both device and host modes.
4	RXFF	Rx FIFO flush Application set this bit to flush data Rx FIFO. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS. **Note:** Accessible in both device and host modes.
3	Reserved	Must be kept at reset value
2	HFCRST	Host frame counter reset Set by the application to reset the frame number counter in USBFS. After this bit is set, the frame number of the following SOF returns to 0. Hardware automatically clears this bit after the reset process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS.

		Note: Only accessible in host mode.
1	HCSRST	HCLK soft reset Set by the application to reset AHB clock domain circuit. Hardware automatically clears this bit after the reset process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS. **Note:** Accessible in both device and host modes.
0	CSRST	Core soft reset Resets the AHB and USB clock domains circuits, as well as most of the registers.

Global interrupt flag register (USBFS_GINTF)

Address offset: 0x0014

Reset value: 0x0400 0021

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPIF	SESIF	DISCIF	IDPSC	Reserved.	PTXFEIF	HCIF	HPIF	Reserved		PXNCIF/ ISOONCIF	ISOINCIF	OEPIF	IEPIF	Reserved	
rc_w1	rc_w1	rc_w1	rc_w1		r	r	r			rc_w1	rc_w1	r	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFIF	ISOOPDIF	ENUMF	RST	SP	ESP	Reserved		GONAK	GNPINAK	NPTXFEIF	RXFNEIF	SOF	OTGIF	MFIF	COPM
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bits	Fields	Descriptions
31	WKUPIF	Wakeup interrupt flag This interrupt is triggered when a resume signal (in device mode) or a remote wakeup signal (in host mode) is detected on the USB. **Note:** Accessible in both device and host modes.
30	SESIF	Session interrupt flag This interrupt is triggered when a SRP is detected (in A-Device mode) or V_{BUS} becomes valid for a B- Device (in B-Device mode). **Note:** Accessible in both device and host modes.
29	DISCIF	Disconnect interrupt flag This interrupt is triggered after a device disconnection. **Note:** Only accessible in host mode.

28	IDPSC	ID pin status change Set by the core when ID status changes. **Note:** Accessible in both device and host modes.
27	Reserved	Must be kept at reset value
26	PTXFEIF	Periodic Tx FIFO empty interrupt flag This interrupt is triggered when the periodic transmit FIFO is either half or completely empty. The threshold is determined by the periodic Tx FIFO empty level bit (PTXFTH) in the USBFS_GAHBCS register. **Note:** Only accessible in host mode.
25	HCIF	Host channels interrupt flag Set by USBFS when one of the channels in host mode has raised an interrupt. First read USBFS_ HACHINT register to get the channel number, and then read the corresponding USBFS_HCHxINTF register to get the flags of the channel that cause the interrupt. This bit will be automatically cleared after the respective channel's flags which cause channel interrupt are cleared. **Note:** Only accessible in host mode.
24	HPIF	Host port interrupt flag Set by the core when USBFS detects that port status changes in host mode. Software should read USBFS_HPCS register to get the source of this interrupt. This bit will be automatically cleared after the flags that causing a port interrupt are cleared. **Note:** Only accessible in host mode.
23:22	Reserved	Must be kept at reset value
21	PXNCIF	Periodic transfer Not Complete Interrupt flag USBFS sets this bit when there are periodic transactions for current frame not completed at the end of frame. (Host mode)
	ISOONCIF	Isochronous OUT transfer Not Complete Interrupt Flag At the end of a periodic frame (defined by EOPFT bit in USBFS_DCFG), USBFS will set this bit if there are still isochronous OUT endpoints for that not completed transactions. (Device Mode)
20	ISOINCIF	Isochronous IN transfer Not Complete Interrupt Flag At the end of a periodic frame (defined by EOPFT [1:0] bits in USBFS_DCFG), USBFS will set this bit if there are still isochronous IN endpoints for that not completed transactions. (Device Mode) **Note:** Only accessible in device mode.
19	OEPIF	OUT endpoint interrupt flag Set by USBFS when one of the OUT endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device

		number, and then read the corresponding USBFS_DOEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint’s flags which cause this interrupt are cleared. **Note:** Only accessible in device mode.
18	IEPIF	IN endpoint interrupt flag Set by USBFS when one of the IN endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device number, and then read the corresponding USBFS_DIEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint’s flags which cause this interrupt are cleared. **Note:** Only accessible in device mode.
17:16	Reserved	Must be kept at reset value
15	EOPFIF	End of periodic frame interrupt flag When USB bus time in a frame reaches the value defined by EOPFT [1:0] bits in USBFS_DCFG register, USBFS sets this flag. **Note:** Only accessible in device mode.
14	ISOOPDIF	Isochronous OUT packet dropped interrupt flag USBFS set this bit if it receives an isochronous OUT packet but cannot save it into Rx FIFO because the FIFO doesn’t have enough space. **Note:** Only accessible in device mode.
13	ENUMF	Enumeration finished USBFS sets this bit after the speed enumeration finishes. Read USBFS_DSTAT register to get the current device speed. **Note:** Only accessible in device mode.
12	RST	USB reset USBFS sets this bit when it detects a USB reset signal on bus. **Note:** Only accessible in device mode.
11	SP	USB suspend USBFS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state. **Note:** Only accessible in device mode.
10	ESP	Early suspend USBFS sets this bit when it detects that the USB bus is idle for 3 ms. **Note:** Only accessible in device mode.
9:8	Reserved	Must be kept at reset value
7	GONAK	Global OUT NAK effective Write 1 to SGONAK bit in the USBFS_DCTL register and USBFS will set GONAK flag after the writing to SGONAK takes effect.

		Note: Only accessible in device mode.
6	GNPINAK	Global Non-Periodic IN NAK effective Write 1 to SGINAK bit in the USBFS_DCTL register and USBFS will set GNPINAK flag after the writing to SGINAK takes effect. **Note:** Only accessible in device mode.
5	NPTXFEIF	Non-Periodic Tx FIFO empty interrupt flag This interrupt is triggered when the non-periodic transmit FIFO is either half or completely empty. The threshold is determined by the non-periodic Tx FIFO empty level bit (TXFTH) in the USBFS_GAHBCS register. **Note:** Only accessible in host mode.
4	RXFNEIF	Rx FIFO non-empty interrupt flag USBFS sets this bit when there is at least one packet or status entry in the Rx FIFO. **Note:** Accessible in both host and device modes.
3	SOF	Start of frame Host Mode: USBFS sets this bit when it prepares to transmit a SOF or Keep-Alive on USB bus. Software can clear this bit by writing 1. Device Mode: USBFS sets this bit after it receives a SOF token. The application can read the Device Status register to get the current frame number. Software can clear this bit by writing 1. **Note:** Accessible in both host and device modes.
2	OTGIF	OTG interrupt flag USBFS sets this bit when the flags in USBFS_GOTGINTF register generate an interrupt. Software should read USBFS_GOTGINTF register to get the source of this interrupt. This bit is cleared after the flags in USBFS_GOTGINTF causing this interrupt are cleared. **Note:** Accessible in both host and device modes.
1	MFIF	Mode fault interrupt flag USBFS sets this bit when software operates host-only register in device mode, or operates device-mode in host mode. These fault operations won’t take effect. **Note:** Accessible in both host and device modes.
0	COPM	Current operation mode 0: Device mode 1: Host mode **Note:** Accessible in both host and device modes.

Global interrupt enable register (USBFS_GINTEN)

Address offset: 0x0018

Reset value: 0x0000 0000

This register works with the global interrupt flag register (USBFS_GINTF) to interrupt the application. When an interrupt enable bit is disabled, the interrupt associated with that bit is not generated. However, the global Interrupt flag register bit corresponding to that interrupt is still set.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPIE	SESIE	DISCIE	IDPSCIE	Reserved.	PTXFEIE	HCIE	HPIE	Reserved		PXNCIE/ ISOONCIE	ISOINCIE	OEPIE	IEPIE	Reserved	
rw	rw	rw	rw		rw	rw	r			rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFIE	ISOOPDIE	ENUMFIE	RSTIE	SPIE	ESPIE	Reserved		GONAKIE	GNPINAKIE	NPTXFEIE	RXFNEIE	SOFIE	OTGIE	MFIE	Reserved
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31	WKUPIE	Wakeup interrupt enable 0: Disable wakeup interrupt 1: Enable wakeup interrupt **Note:** Accessible in both host and device modes.
30	SESIE	Session interrupt enable 0: Disable session interrupt 1: Enable session interrupt **Note:** Accessible in both host and device modes.
29	DISCIE	Disconnect interrupt enable 0: Disable disconnect interrupt 1: Enable disconnect interrupt **Note:** Only accessible in device mode.
28	IDPSCIE	ID pin status change interrupt enable 0: Disable connector ID pin status interrupt 1: Enable connector ID pin status interrupt **Note:** Accessible in both host and device modes.
27	Reserved	Must be kept at reset value
26	PTXFEIE	Periodic Tx FIFO empty interrupt enable 0: Disable periodic Tx FIFO empty interrupt 1: Enable periodic Tx FIFO empty interrupt **Note:** Only accessible in host mode.

25	HCIE	Host channels interrupt enable 0: Disable host channels interrupt 1: Enable host channels interrupt **Note:** Only accessible in host mode.
24	HPIE	Host port interrupt enable 0: Disable host port interrupt 1: Enable host port interrupt **Note:** Only accessible in host mode.
23:22	Reserved	Must be kept at reset value
21	PXNCIE	Periodic transfer not complete Interrupt enable 0: Disable periodic transfer not complete interrupt 1: Enable periodic transfer not complete interrupt **Note:** Only accessible in host mode.
	ISOONCIE	Isochronous OUT transfer not complete interrupt enable 0: Disable isochronous OUT transfer not complete interrupt 1: Enable isochronous OUT transfer not complete interrupt **Note:** Only accessible in device mode.
20	ISOINCIE	Isochronous IN transfer not complete interrupt enable 0: Disable isochronous IN transfer not complete interrupt 1: Enable isochronous IN transfer not complete interrupt **Note:** Only accessible in device mode.
19	OEPIE	OUT endpoints interrupt enable 0: Disable OUT endpoints interrupt 1: Enable OUT endpoints interrupt **Note:** Only accessible in device mode.
18	IEPIE	IN endpoints interrupt enable 0: Disable IN endpoints interrupt 1: Enable IN endpoints interrupt **Note:** Only accessible in device mode.
17:16	Reserved	Must be kept at reset value
15	EOPFIE	End of periodic frame interrupt enable 0: Disable end of periodic frame interrupt 1: Enable end of periodic frame interrupt **Note:** Only accessible in device mode.
14	ISOOPDIE	Isochronous OUT packet dropped interrupt enable 0: Disable isochronous OUT packet dropped interrupt 1: Enable isochronous OUT packet dropped interrupt

		Note: Only accessible in device mode.
13	ENUMFIE	Enumeration finish enable 0: Disable enumeration finish interrupt 1: Enable enumeration finish interrupt **Note:** Only accessible in device mode.
12	RSTIE	USB reset interrupt enable 0: Disable USB reset interrupt 1: Enable USB reset interrupt **Note:** Only accessible in device mode.
11	SPIE	USB suspend interrupt enable 0: Disable USB suspend interrupt 1: Enable USB suspend interrupt **Note:** Only accessible in device mode.
10	ESPIE	Early suspend interrupt enable 0: Disable early suspend interrupt 1: Enable early suspend interrupt **Note:** Only accessible in device mode.
9:8	Reserved	Must be kept at reset value
7	GONAKIE	Global OUT NAK effective interrupt enable 0: Disable global OUT NAK interrupt 1: Enable global OUT NAK interrupt **Note:** Only accessible in device mode.
6	GNPINAKIE	Global non-periodic IN NAK effective interrupt enable 0: Disable global non-periodic IN NAK effective interrupt 1: Enable global non-periodic IN NAK effective interrupt **Note:** Only accessible in device mode.
5	NPTXFEIE	Non-periodic Tx FIFO empty interrupt enable 0: Disable non-periodic Tx FIFO empty interrupt 1: Enable non-periodic Tx FIFO empty interrupt **Note:** Only accessible in Host mode.
4	RXFNEIE	Receive FIFO non-empty interrupt enable 0: Disable receive FIFO non-empty interrupt 1: Enable receive FIFO non-empty interrupt **Note:** Accessible in both device and host modes.
3	SOFIE	Start of frame interrupt enable 0: Disable start of frame interrupt 1: Enable start of frame interrupt

		Note: Accessible in both device and host modes.
2	OTGIE	OTG interrupt enable 0: Disable OTG interrupt 1: Enable OTG interrupt **Note:** Accessible in both device and host modes.
1	MFIE	Mode fault interrupt enable 0: Disable mode fault interrupt 1: Enable mode fault interrupt **Note:** Accessible in both device and host modes.
0	Reserved	Must be kept at reset value

Global receive status read/receive status read and pop registers (USBFS_GRSTATR/USBFS_GRSTATP)

Address offset for Read: 0x001C
Address offset for Pop: 0x0020
Reset value: 0x0000 0000

A read to the receive status read register returns the entry of the top of the Rx FIFO. A read to the Receive status read and pop register additionally pops the top entry out of the Rx FIFO.

The entries in RxFIFO have different meanings in host and device modes. Software should only read this register after when Receive FIFO non-empty interrupt flag bit of the global interrupt flag register (RXFNEIF bit in USBFS_GINTF) is triggered.

This register has to be accessed by word (32-bit)

Host mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											RPCKST[3:0]				DPID
											r				r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCOUNT[10:0]											CNUM[3:0]			
r	r											r			

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value

20:17	RPCKST[3:0]	Received packet status 0010: IN data packet received 0011: IN transfer completed (generates an interrupt if poped) 0101: Data toggle error (generates an interrupt if poped) 0111: Channel halted (generates an interrupt if poped) Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA
14:4	BCOUNT[10:0]	Byte count The byte count of the received IN data packet.
3:0	CNUM[3:0]	Channel number The channel number to which the current received packet belongs.

Device mode:

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="11">Reserved</td><td colspan="4">RPCKST[3:0]</td><td>DPID</td></tr>
<tr><td colspan="11"></td><td colspan="4">r</td><td>r</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>DPID</td><td colspan="11">BCOUNT[10:0]</td><td colspan="4">EPNUM[3:0]</td></tr>
<tr><td>r</td><td colspan="11">r</td><td colspan="4">r</td></tr>
</table>

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20:17	RPCKST[3:0]	Received packet status 0001: Global OUT NAK (generates an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (generates an interrupt) 0100: SETUP transaction completed (generates an interrupt) 0110: SETUP data packet received Others: Reserved
16:15	DPID[1:0]	Data PID

		The Data PID of the received OUT data packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA
14:4	BCOUNT[10:0]	Byte count The byte count of the received data packet.
3:0	EPNUM[3:0]	Endpoint number The endpoint number to which the current received packet belongs.

Global receive FIFO length register (USBFS_GRFLEN)

Address offset: 0x024

Reset value: 0x0000 0200

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RXFD[15:0]	Rx FIFO depth In terms of 32-bit words. 1≤RXFD≤1024

Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBFS_HNPTFLEN _DIEP0TFLEN)

Address offset: 0x028

Reset value: 0x0200 0200

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HNPTXFD/ IEP0TXFD[15:0]															

r/rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HNPTXRSAR/ IEP0TXRSAR [15:0]															

r/rw

Host Mode:

Bits	Fields	Descriptions
31:16	HNPTXFD[15:0]	Host Non-periodic Tx FIFO depth In terms of 32-bit words. 1⩽HNPTXFD⩽1024
15:0	HNPTXRSAR[15:0]	Host Non-periodic Tx RAM start address The start address for non-periodic transmit FIFO RAM is in term of 32-bit words.

Device Mode:

Bits	Fields	Descriptions
31:16	IEP0TXFD[15:0]	IN Endpoint 0 Tx FIFO depth In terms of 32-bit words. 16⩽IEP0TXFD⩽140
15:0	IEP0TXRSAR[15:0]	IN Endpoint 0 TX RAM start address The start address for endpoint0 transmit FIFO RAM is in term of 32-bit words.

Host non-periodic transmit FIFO/queue status register (USBFS_HNPTFQSTAT)

Address offset: 0x002C
Reset value: 0x0008 0200

This register reports the current status of the non-periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.

Note: In Device mode, this register is not valid.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Reserved	NPTXRQTOP[6:0]	NPTXRQS[7:0]
	r	r

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:24	NPTXRQTOP[6:0]	Top entry of the non-periodic Tx request queue Entry in the non-periodic transmit request queue. Bits 30:27: Channel number Bits 26:25: – 00: IN/OUT token – 01: Zero-length OUT packet – 11: Channel halt request Bit 24: Terminate Flag, indicating last entry for selected channel.
23:16	NPTXRQS[7:0]	Non-periodic Tx request queue space The remaining space of the non-periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries … n: n entries (0≤n≤8) Others: Reserved
15:0	NPTXFS[15:0]	Non-periodic Tx FIFO space The remaining space of the non-periodic transmit FIFO. In terms of 32-bit words. 0: Non-periodic Tx FIFO is full 1: 1 word 2: 2 words n: n words (0≤n≤NPTXFD) Others: Reserved

Global core configuration register (USBFS_GCCFG)

Address offset: 0x0038

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											SOFOEN	VBUSBCEN	VBUSACEN	Reserved	PWRON
											rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20	SOFOEN	SOF output enable 0: SOF pulse output disabled. 1: SOF pulse output enabled.
19	VBUSBCEN	The V_{BUS} B-device Comparer enable 0: V_{BUS} B-device comparer disabled 1: V_{BUS} B-device comparer enabled
18	VBUSACEN	The VBUS A-device Comparer enable 0: V_{BUS} A-device comparer disabled 1: V_{BUS} A-device comparer enabled
17	Reserved	Must be kept at reset value
16	PWRON	Power on This bit is the power switch for the internal embedded Full-Speed PHY. 0: Embedded Full-Speed PHY power off. 1: Embedded Full-Speed PHY power on.
15:0	Reserved	Must be kept at reset value.

Core ID register (USBFS_CID)

Address offset: 0x003C

Reset value: 0x0000 1000

This register contains the Product ID.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CID[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	CID[31:0]	Core ID Software can write or read this field and uses this field as a unique ID for its application

Host periodic transmit FIFO length register (USBFS_HPTFLEN)

Address offset: 0x0100

Reset value: 0x0200 0600

This register has to be accessed by word 32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HPTXFD [15:0]															
r/rw															

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

Bits	Fields	Descriptions
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth In terms of 32-bit words. 1≤HPTXFD≤1024
15:0	HPTXFSAR[15:0]	Host periodic Tx FIFO RAM start address The start address for host periodic transmit FIFO RAM is in term of 32-bit words.

Device IN endpoint transmit FIFO length register (USBFS_DIEPxTFLEN) (x = 1..3, where x is the FIFO_number)

Address offset: 0x0104 + (FIFO_number – 1) × 0x04

Reset value: 0x0200 0400

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEPTXFD[15:0]															
r/rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPTXRSAR 15:0]															
r/rw															

Bits	Fields	Descriptions
31:16	IEPTXFD[15:0]	IN endpoint Tx FIFO depth In terms of 32-bit words. 1≤HPTXFD≤1024
15:0	IEPTXRSAR[15:0]	IN endpoint FIFO Tx RAM start address The start address for IN endpoint transmit FIFOx is in term of 32-bit words.

24.7.2. Host control and status registers

Host control register (USBFS_HCTL)

Address offset: 0x0400

Reset value: 0x0000 0000

This register configures the core after power on in host mode. Do not modify it after host initialization.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1:0	CLKSEL[1:0]	Clock select for usbclock. 01: 48MHz clock others: reserved

Host frame interval register (USBFS_HFT)

Address offset: 0x0404
Reset value: 0x0000 BB80

This register sets the frame interval for the current enumerating speed when USBFS controller is enumerating.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	FRI[15:0]	Frame interval This value describes the frame time in terms of PHY clocks. Each time when port is enabled after a port reset operation, USBFS use a proper value according to the current speed, and software can write to this field to change the value. This value should be calculated using the frequency described below: Full-Speed: 48MHz Low-Speed: 6MHz

Host frame information remaining register (USBFS_HFINFR)

Address offset: 0x408

Reset value: 0xBB80 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRT[15:0]															
r															

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

Bits	Fields	Descriptions
31:16	FRT[15:0]	Frame remaining time This field reports the remaining time of current frame in terms of PHY clocks.
15:0	FRNUM[15:0]	Frame number This field reports the frame number of current frame and returns to 0 after it reaches 0x3FFF.

Host periodic transmit FIFO/queue status register (USBFS_HPTFQSTAT)

Address offset: 0x0410

Reset value: 0x0008 0200

This register reports the current status of the host periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXREQT[7:0]								PTXREQS[7:0]							
r								r							

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

Bits	Fields	Descriptions
31:24	PTXREQT[7:0]	Top entry of the periodic Tx request queue Entry in the periodic transmit request queue. Bits 30:27: Channel Number Bits 26:25: 00: IN/OUT token 01: Zero-length OUT packet 11: Channel halt request Bit 24: Terminate Flag, indicating last entry for selected channel.
23:16	PTXREQS[7:0]	Periodic Tx request queue space The remaining space of the periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries … n: n entries (0≤n≤8) Others: Reserved
15:0	PTXFS[15:0]	Periodic Tx FIFO space The remaining space of the periodic transmit FIFO. In terms of 32-bit words. 0: periodic Tx FIFO is full 1: 1 word 2: 2 words n: n words (0≤n≤PTXFD) Others: Reserved

Host all channels interrupt register (USBFS_HACHINT)

Address offset: 0x0414

Reset value: 0x0000 0000

When a channel interrupt is triggered, USBFS set corresponding bit in this register and software should read this register to know which channel is asserting interrupts.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								HACHINT[7:0]							
								r							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	HACHINT[7:0]	Host all channel interrupts Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

Host all channels interrupt enable register (USBFS_HACHINTEN)

Address offset: 0x0418

Reset value: 0x0000 0000

This register can be used by software to enable or disable a channel’s interrupt. Only the channel whose corresponding bit in this register is set is able to cause the channel interrupt flag HCIF in USBFS_GINTF register.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Reserved	CINTEN[7:0]
	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	CINTEN[7:0]	Channel interrupt enable 0: Disable channel-n interrupt 1: Enable channel-n interrupt Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

Host port control and status register (USBFS_HPCS)

Address offset: 0x0440

Reset value: 0x0000 0000

This register controls the port’s behavior and also has some flags which report the status of the port. The HPIF flag in USBFS_GINTF register will be triggered if one of these flags in this register is set by USBFS: PRST, PEDC and PCD.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													PS[1:0]		Reserved
													r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PP	PLST[1:0]		Reserved	PRST	PSP	PREM	Reserved		PEDC	PE	PCD	PCST
			rw	r			rw	rs	rw			rc_w1	rc_w1	rc_w1	r

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18:17	PS[1:0]	Port speed Report the enumerated speed of the device attached to this port. 01: Full speed 10: Low speed

		Others: Reserved
16:13	Reserved	Must be kept at reset value
12	PP	Port power This bit should be set before a port is used. Because USBFS doesn’t have power supply ability, it only uses this bit to know whether the port is in powered state. Software should ensure the true power supply on V_{BUS} before setting this bit. 0: Port is powered off 1: Port is powered on
11:10	PLST[1:0]	Port line status Report the current state of USB data lines Bit 10: State of DP line Bit 11: State of DM line
9	Reserved	Must be kept at reset value
8	PRST	Port reset Application sets this bit to start a reset signal on USB port. Application should clear this bit when it wants to stop the reset signal. 0: Port is not in reset state 1: Port is in reset state
7	PSP	Port suspend Application sets this bit to put port into suspend state. When this bit is set the port stops sending SOF tokens. This bit can only be cleared by the following operations: – PRST bit in this register is set by application – PREM bit in this register is set – A remote wakeup signal is detected – A device disconnect is detected 0: Port is not in suspend state 1: Port is in suspend state
6	PREM	Port resume Application sets this bit to start a resume signal on USB port. Application should clear this bit when it wants to stop the resume signal. 0: No resume driven 1: Resume driven
5:4	Reserved	Must be kept at reset value
3	PEDC	Port enable/disable change Set by the core when the status of the Port enable bit 2 in this register changes.
2	PE	Port Enable This bit is automatically set by USBFS after a USB reset signal finishes and

		cannot be set by software. This bit is cleared by the following events: – A disconnect condition – Software clearing this bit 0: Port disabled 1: Port enabled
1	PCD	Port connect detected Set by USBFS when a device connection is detected. This bit can be cleared by writing 1 to this bit.
0	PCST	Port connect status 0: Device is not connected to the port 1: Device is connected to the port

Host channel-x control register (USBFS_HCHxCTL) (x = 0..7 where x = channel_number)

Address offset: 0x0500 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CEN	CDIS	ODDFRM	DAR[6:0]							Reserved		EPTYPE[1:0]		LSD	Reserved
rs	rs	rw	rw									rw		rw	

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

Bits	Fields	Descriptions
31	CEN	Channel enable Set by the application and cleared by USBFS. 0: Channel disabled 1: Channel enabled Software should following the operation guide to disable or enable a channel.
30	CDIS	Channel disable Software can set this bit to disable the channel from processing transactions.

		Software should follow the operation guide to disable or enable a channel.
29	ODDFRM	Odd frame For periodic transfers (interrupt or isochronous transfer), this bit controls that whether in an odd frame or even frame this channel's transaction is desired to be processed. 0: Even frame 1: Odd frame
28:22	DAR[6:0]	Device address The address of the USB device that this channel wants to communicate with.
21:20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	Endpoint type The transfer type of the endpoint that this channel wants to communicate with. 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	LSD	Low-Speed device The device that this channel wants to communicate with is a Low-Speed Device.
16	Reserved	Must be kept at reset value
15	EPDIR	Endpoint direction The transfer direction of the endpoint that this channel wants to communicate with. 0: OUT 1: IN
14:11	EPNUM[3:0]	Endpoint number The number of the endpoint that this channel wants to communicate with.
10:0	MPL[10:0]	Maximum packet length The target endpoint's maximum packet length.

Host channel-x interrupt flag register (USBFS_HCHxINTF) (x = 0..7 where x = channel number)

Address offset: 0x0508 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of a channel, when software get a channel interrupt, it should read this register for the respective channel to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTER	REQOVR	BBER	USBER	Reserved.	ACK	NAK	STALL	Reserved.	CH	TF
					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	DTER	Data toggle error The IN transaction gets a data packet but the PID of this packet doesn’t match DPID [1:0] bits in USBFS_HCHxLEN register.
9	REQOVR	Request queue overrun The periodic request queue is full when software starts new transfers.
8	BBER	Babble error A babble condition occurs on USB bus. A typical reason for babble condition is that a device sends a data packet and the packet length exceeds the endpoint’s maximum packet length.
7	USBER	USB Bus Error The USB error flag is set when the following conditions occurs during receiving a packet: – A received packet has a wrong CRC field – A stuff error detected on USB bus – Timeout when waiting for a response packet
6	Reserved	Must be kept at reset value
5	ACK	ACK An ACK response is received or transmitted
4	NAK	NAK A NAK response is received.
3	STALL	STALL A STALL response is received.
2	Reserved	Must be kept at reset value

1	CH	Channel halted This channel is disabled by a request, and it will not response to other requests during the request processing.
0	TF	Transfer finished All the transactions of this channel finish successfully, and no error occurs. For IN channel, this flag will be triggered after PCNT bits in USBFS_HCHxLEN register reach zero. For OUT channel, this flag will be triggered when software reads and pops a TF status entry from the RxFIFO.

Host channel-x interrupt enable register (USBFS_HCHxINTEN) (x = 0..7, where x = channel number)

Address offset: 0x050C + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_HCHxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_HCHxINTF register is able to trigger a channel interrupt. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTERIE	REQOVRIE	BBERIE	USBERIE	Reserved.	ACKIE	NAKIE	STALLIE	Reserved	CHIE	TFIE
					rw	rw	rw	rw		rw	rw	rw		rw	rw

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	DTERIE	Data toggle error interrupt enable 0: Disable data toggle error interrupt 1: Enable data toggle error interrupt
9	REQOVRIE	Request queue overrun interrupt enable 0: Disable request queue overrun interrupt 1: Enable request queue overrun interrupt

8	BBERIE	Babble error interrupt enable 0: Disable babble error interrupt 1: Enable babble error interrupt
7	USBERIE	USB bus error interrupt enable 0: Disable USB bus error interrupt 1: Enable USB bus error interrupt
6	Reserved	Must be kept at reset value
5	ACKIE	ACK interrupt enable 0: Disable ACK interrupt 1: Enable ACK interrupt
4	NAKIE	NAK interrupt enable 0: Disable NAK interrupt 1: Enable NAK interrupt
3	STALLIE	STALL interrupt enable 0: Disable STALL interrupt 1: Enable STALL interrupt
2	Reserved	Must be kept at reset value
1	CHIE	Channel halted interrupt enable 0: Disable channel halted interrupt 1: Enable channel halted interrupt
0	TFIE	Transfer finished interrupt enable 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Host channel-x transfer length register (USBFS_HCHxLEN) (x = 0..7, where x = channel number)

Address offset: 0x0510 + (channel_number × 0x20)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DPID[1:0]		PCNT[9:0]										TLEN[18:16]		
	rw		rw										rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TLEN[15:0]
rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	DPID[1:0]	Data PID Software should write this field before the transfer starts. For OUT transfers, this field controls the Data PID of the first transmitted packet. For IN transfers, this field controls the expected Data PID of the first received packet, and DTERR will be triggered if the Data PID doesn't match. After the transfer starts, USBFS changes and toggles this field automatically following the USB protocol. 00: DATA0 10: DATA1 11: SETUP (For control transfer only) 01: Reserved
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted (OUT) or received (IN) in a transfer. Software should program this field before the channel is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.
18:0	TLEN[18:0]	Transfer length The total data bytes number of a transfer. For OUT transfers, this field is the total data bytes of all the data packets desired to be transmitted in an OUT transfer. Software should program this field before the channel is enabled. When software successfully writes a packet into the channel's data TxFIFO, this field is decreased by the byte size of the packet. For IN transfer each time software or DMA reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

24.7.3. Device control and status registers

Device configuration register (USBFS_DCFG)

Address offset: 0x0800

Reset value: 0x0000 0000

This register configures the core in device mode after power on or after certain control commands or enumeration. Do not change this register after device initialization.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EOPFT[1:0]		DAR[6:0]							Reserved	NZLSOH	DS[1:0]	
			rw		rw								rw	rw	

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12:11	EOPFT[1:0]	End of periodic frame time This field defines the percentage time point in a frame that the end of periodic frame (EOPF) flag should be triggered. 00: 80% of the frame time 01: 85% of the frame time 10: 90% of the frame time l 11: 95% of the frame time
10:4	DAR[6:0]	Device address This field defines the USB device's address. USBFS uses this field to match with the incoming token's device address field. Software should program this field after receiving a Set Address command from USB host.
3	Reserved	Must be kept at reset value
2	NZLSOH	Non-zero-length status OUT handshake When a USB device receives a non-zero-length data packet during status OUT stage, this field controls that either USBFS should receive this packet or reject this packet with a STALL handshake. 0: Treat this packet as a normal packet and response according to the status of NAKS and STALL bits in USBFS_DOEPxCTL register. 1: Send a STALL handshake and don't save the received OUT packet.
1:0	DS[1:0]	Device speed This field controls the device speed when the device connected to a host. 11: Full speed Others: Reserved

Device control register (USBFS_DCTL)

Address offset: 0x0804

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				POIF	CGONAK	SGONAK	CGINAK	SGINAK	Reserved			GONS	GINS	SD	RWKUP
				rw	w	w	w	w				r	r	rw	rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	POIF	Power-on initialization finished Software should set this bit to notify USBFS that the registers are initialized after waking up from power down state.
10	CGONAK	Clear global OUT NAK Software sets this bit to clear GONS bit in this register.
9	SGONAK	Set global OUT NAK Software sets this bit to set GONS bit in this register. When GONS bit is zero, setting this bit will also cause GONAK flag in USBFS_GINTF register triggered after a while. Software should clear the GONAK flag before writing this bit again.
8	CGINAK	Clear global IN NAK Software sets this bit to clear GINS bit in this register.
7	SGINAK	Set global IN NAK Software sets this bit to set GINS bit in this register. When GINS bit is zero, setting this bit will also cause GINAK flag in USBFS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again.
6:4	Reserved	Must be kept at reset value
3	GONS	Global OUT NAK status

		0: The handshake that USBFS response to OUT transaction packet and whether to save the OUT data packet are decided by Rx FIFO status, endpoint’s NAK and STALL bits. 1: USHBS always responses to OUT transaction with NAK handshake and doesn’t save the incoming OUT data packet.
2	GINS	Global IN NAK status 0: The response to IN transaction is decided by Tx FIFO status, endpoint’s NAK and STALL bits. 1: USBFS always responses to IN transaction with a NAK handshake.
1	SD	Soft disconnect Software can use this bit to generate a soft disconnect condition on USB bus. After this bit is set, USBFS switches off the pull up resistor on DP line. This will cause the host to detect a device disconnect. 0: No soft disconnect generated. 1: Generate a soft disconnection.
0	RWKUP	Remote wakeup In suspend state, software can use this bit to generate a Remote wake up signal to inform host that it should resume the USB bus. 0: No remote wakeup signal generated. 1: Generate remote wakeup signal.

Device status register (USBFS_DSTAT)

Address offset: 0x0808
Reset value: 0x0000 0000

This register contains status and information of the USBFS in device mode.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										FNRSOF[13:8]					
										r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNRSOF[7:0]								Reserved					ES[1:0]		SPST
r													r		r

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21:8	FNRSOF[13:0]	The frame number of the received SOF. USBFS always update this field after receiving a SOF token
7:3	Reserved	Must be kept at reset value
2:1	ES[1:0]	Enumerated speed This field reports the enumerated device speed. Read this field after the ENUMF flag in USBFS_GINTF register is triggered. 11: Full speed Others: reserved
0	SPST	Suspend status This bit reports whether device is in suspend state. 0: Device is in suspend state. 1: Device is not in suspend state.

Device IN endpoint common interrupt enable register (USBFS_DIEPINTEN)

Address offset: 0x810

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_DIEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_DIEPxINTF register is able to trigger an endpoint interrupt in USBFS_DAEPINT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									IEPNEEN	Reserved	EPTXFUDEN	CITOEN	Reserved	EPDISEN	TFEN
									rw		rw	rw		rw	rw

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value

6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable IN endpoint NAK effective interrupt 1: Enable IN endpoint NAK effective interrupt
5	Reserved	Must be kept at reset value
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable endpoint Tx FIFO underrun interrupt 1: Enable endpoint Tx FIFO underrun interrupt
3	CITOEN	Control In timeout interrupt enable bit 0: Disable control In timeout interrupt 1: Enable control In timeout interrupt
2	Reserved	Must be kept at reset value
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Device OUT endpoint common interrupt enable register (USBFS_DOEPINTEN)

Address offset: 0x0814

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_DOEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_DOEPxINTF register is able to trigger an endpoint interrupt in USBFS_DAEPINT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									BTBSTPEN	Reserved	EPRXFOVREN	STPFEN	Reserved	EPDISEN	TFEN

rw rw rw rw rw

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	BTBSTPEN	Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit 0: Disable back-to-back SETUP packets interrupt 1: Enable back-to-back SETUP packets interrupt
5	Reserved	Must be kept at reset value
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable endpoint Rx FIFO overrun interrupt 1: Enable endpoint Rx FIFO overrun interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable SETUP phase finished interrupt 1: Enable SETUP phase finished interrupt
2	Reserved	Must be kept at reset value
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Device all endpoints interrupt register (USBFS_DAEPINT)

Address offset: 0x0818

Reset value: 0x0000 0000

When an endpoint interrupt is triggered, USBFS sets corresponding bit in this register and software should read this register to know which endpoint is asserting an interrupt.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												OEPITB[3:0]			
												r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	IEPITB[3:0]
	r

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19:16	OEPITB[3:0]	Device all OUT endpoint interrupt bits Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value
3:0	IEPITB[3:0]	Device all IN endpoint interrupt bits Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

Device all endpoints interrupt enable register (USBFS_DAEPINTEN)

Address offset: 0x081C

Reset value: 0x0000 0000

This register can be used by software to enable or disable an endpoint’s interrupt. Only the endpoint whose corresponding bit in this register is set is able to cause the endpoint interrupt flag OEPIF or IEPIF in USBFS_GINTF register.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												OEPIE[3:0]			
												rw			

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

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value

19:16	OEPIE[3:0]	Out endpoint interrupt enable 0: Disable OUT endpoint-n interrupt 1: Enable OUT endpoint-n interrupt Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value
3:0	IEPIE[3:0]	IN endpoint interrupt enable bits 0: Disable IN endpoint-n interrupt 1: Enable IN endpoint-n interrupt Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

Device VBUS discharge time register (USBFS_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	DVBUSDT[15:0]	Device V_{BUS} discharge time There is a discharge process after V_{BUS} pulsing in SRP protocol. This field defines the discharge time of V_{BUS}. The true discharge time is 1024 * DVBUSDT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device VBUS pulsing time register (USBFS_DVBUSPT)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DVBUSPT[11:0]	Device V_{BUS} pulsing time This field defines the pulsing time for V_{BUS}. The true pulsing time is 1024*DVBUSPT[11:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device IN endpoint FIFO empty interrupt enable register (USBFS_DIEPFEINTEN)

Address offset: 0x0834

Reset value: 0x0000 0000

This register contains the enable bits for the Tx FIFO empty interrupts of IN endpoints.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	IEPTXFEIE[3:0]	IN endpoint Tx FIFO empty interrupt enable bits This field controls whether the TXFE bits in USBFS_DIEPxINTF registers are able to generate an endpoint interrupt bit in USBFS_DAEPINT register. Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3 0: Disable FIFO empty interrupt 1: Enable FIFO empty interrupt

Device IN endpoint 0 control register (USBFS_DIEP0CTL)

Address offset: 0x0900

Reset value: 0x0000 8000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPEN	EPD	Reserved		SNAK	CNAK	TXFNUM[3:0]				STALL	Reserved	EPTYPE[1:0]		NAKS	Reserved
rs	rs			w	w	rw				rs		r		r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved													MPL[1:0]	
r														rw	

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint.
29:28	Reserved	Must be kept at reset value
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK

		Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of IN endpoint 0.
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake when receiving IN token. USBFS will clear this bit after a SETUP token is received on the corresponding OUT endpoint 0. This bit has a higher priority than NAKS bit in this register and GINS bit in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect.
20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	Endpoint type This field is fixed to ‘00’ for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint’s Tx FIFO. 1: USBFS always sends NAK handshake to the IN token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	Reserved	Must be kept at reset value
15	EPACT	Endpoint active This field is fixed to ‘1’ for endpoint 0.
14:2	Reserved	Must be kept at reset value
1:0	MPL[1:0]	Maximum packet length This field defines the maximum packet length for a control data packet. As described in USB 2.0 protocol, there are 4 kinds of length for control transfers: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

Device IN endpoint-x control register (USBFS_DIEPxCTL) (x = 1..3, where x = endpoint_number)

Address offset: 0x0900 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPEN	EPD	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Reserved	EPTYPE[1:0]		NAKS	EOFRM/DPID
rs	rs	w	w	w	w	rw				rw/rs		rw		r	r

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

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint.
29	SODDFRM	Set odd frame (For isochronous IN endpoints) This bit has effect only if this is an isochronous IN endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk IN endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous IN endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk IN endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of this IN endpoint.
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake when receiving

		IN token. This bit has a higher priority than NAKS bit in this register and GINS bit in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. For control IN endpoint: Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk IN endpoint: Only software can clear this bit
20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint’s Tx FIFO. 1: USBFS always sends NAK handshake to the IN token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous IN endpoints) For isochronous transfers, software can use this bit to control that USBFS only sends data packets for IN tokens in even or odd frames. If the parity of the current frame number doesn’t match with this bit, USBFS only responses with a zero-length packet. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk IN endpoints) There is a data PID toggle scheme in interrupt or bulk transfer. Set SD0PID to set this bit before a transfer starts and USBFS maintains this bit during transfers according to the data toggle scheme described in USB protocol. 0: Data packet’s PID is DATA0 1: Data packet’s PID is DATA1
15	EPACT	Endpoint active This bit controls whether this endpoint is active. If an endpoint is not active, it ignores all tokens and doesn’t make any response.

14:11	Reserved	Must be kept at reset value
10:0	MPL[10:0]	This field defines the maximum packet length in bytes.

Device OUT endpoint 0 control register (USBFS_DOEP0CTL)

Address offset: 0x0B00

Reset value: 0x0000 8000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPEN	EPD	Reserved.		SNAK	CNAK	Reserved				STALL	SNOOP	EPTYPE[1:0]		NAKS	Reserved
rs	r			w	w					rs	rw	r		r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved													MPL[1:0]	
r														r	

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable This bit is fixed to 0 for OUT endpoint 0.
29:28	Reserved	Must be kept at reset value
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register
25:22	Reserved	Must be kept at reset value
21	STALL	STALL handshake Set this bit to make USBFS send STALL handshake during an OUT transaction. USBFS will clear this bit after a SETUP token is received on OUT endpoint 0. This bit has a higher priority than NAKS bit in this register, i.e. if both STALL and NAKS

		bits are set, the STALL bit takes effect.
20	SNOOP	Snoop mode This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn’t check the received data packet’s CRC value. 0:Snoop mode disabled 1:Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field is fixed to ‘00’ for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint’s Rx FIFO. 1: USBFS always sends NAK handshake for the OUT token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	Reserved	Must be kept at reset value
15	EPACT	Endpoint active This field is fixed to ‘1’ for endpoint 0.
14:2	Reserved	Must be kept at reset value
1:0	MPL[1:0]	Maximum packet length This is a read-only field, and its value comes from the MPL field of USBFS_DIEP0CTL register: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

Device OUT endpoint-x control register (USBFS_DOEPxCTL) (x = 1..3, where x = endpoint_number)

Address offset: 0x0B00 + (endpoint_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the operations of each logical OUT endpoint other than OUT endpoint 0.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

EPEN	EPD	SODDFRM/SD1PID	SEVNFRM/SD0PID	SNAK	CNAK	Reserved				STALL	SNOOP	EPTYPE[1:0]		NAKS	EOFRM/DPID
rs	rs	w	w	w	w					rw/rs	rw	rw		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved				MPL[10:0]										
rw										rw					

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint.
29	SODDFRM	Set odd frame (For isochronous OUT endpoints) This bit has effect only if this is an isochronous OUT endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk OUT endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous OUT endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk OUT endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	Reserved	Must be kept at reset value
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake during an OUT transaction. This bit has a higher priority than NAKS bit in this register and GINAK in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit

		takes effect. For control OUT endpoint: Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk OUT endpoint: Only software can clear this bit.
20	SNOOP	Snoop mode This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value. 0:Snoop mode disabled 1:Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared: 0: USBFS sends handshake packets according to the status of the endpoint's Rx FIFO. 1: USBFS always sends NAK handshake to the OUT token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous OUT endpoints) For isochronous transfers, software can use this bit to control that USBFS only receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBFS just drops the data packet. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk OUT endpoints) These is a data PID toggle scheme in interrupt or bulk transfer. Software should set SD0PID to set this bit before a transfer starts and USBFS maintains this bit during transfers following the data toggle scheme described in USB protocol. 0: Data packet's PID is DATA0 1: Data packet's PID is DATA1
15	EPACT	Endpoint active This bit controls whether this endpoint is active. If an endpoint is not active, it

		ignores all tokens and doesn't make any response.
14:11	Reserved	Must be kept at reset value
10:0	MPL[10:0]	This field defines the maximum packet length in bytes.

Device IN endpoint-x interrupt flag register (USBFS_DIEPxINTF) (x = 0..3, where x = endpoint_number)

Address offset: 0x0908 + (endpoint_number × 0x20)

Reset value: 0x0000 0080

This register contains the status and events of an IN endpoint, when an IN endpoint interrupt occurs, read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1 except the read-only TXFE bit.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXFE	IEPNE	Reserved	EPTXFUD	CITO	Reserved	EPDIS	TF
								r	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	TXFE	Transmit FIFO empty The Tx FIFO of this IN endpoint has reached the empty threshold value defined by TXFTH field in USBFS_GAHBCS register.
6	IEPNE	IN endpoint NAK effective The setting of SNAK bit in USBFS_DIEPxCTL register takes effect. This bit can be cleared either by writing 1 to it or by setting CNAK bit in USBFS_DIEPxCTL register.
5	Reserved	Must be kept at reset value
4	EPTXFUD	Endpoint Tx FIFO underrun

		This flag is triggered if the Tx FIFO has no packet data when an IN token is incoming
3	CITO	Control In Timeout interrupt This flag is triggered if the device waiting for a handshake is timeout in a control IN transaction.
2	Reserved	Must be kept at reset value
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's request.
0	TF	Transfer finished This flag is triggered when all the IN transactions assigned to this endpoint have been finished.

Device OUT endpoint-x interrupt flag register (USBFS_DOEPxINTF) (x = 0..3, where x = endpoint_number)

Address offset: 0x0B08 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of an OUT endpoint, when an OUT endpoint interrupt occurs, read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									BTBSTP	Reserved	EPRXFOVR	STPF	Reserved	EPDIS	TF
									rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	BTBSTP	Back-to-back SETUP packets (Only for control OUT endpoint) This flag is triggered when a control out endpoint has received more than 3 back-to-back setup packets.

5	Reserved	Must be kept at reset value
4	EPRXFOVR	Endpoint Rx FIFO overrun This flag is triggered if the OUT endpoint’s Rx FIFO has no enough space for a packet data when an OUT token is incoming. USBFS will drop the incoming OUT data packet and sends a NAK handshake in this case.
3	STPF	SETUP phase finished (Only for control OUT endpoint) This flag is triggered when a setup phase finished, i.e. USBFS receives an IN or OUT token after a setup token.
2	Reserved	Must be kept at reset value
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software’s request.
0	TF	Transfer finished This flag is triggered when all the OUT transactions assigned to this endpoint have been finished.

Device IN endpoint 0 transfer length register (USBFS_DIEP0LEN)

Address offset: 0x0910

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											PCNT[1:0]		Reserved		
											rw				

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

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20:19	PCNT[1:0]	Packet count The number of data packets desired to be transmitted in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet

		transmission.
18:7	Reserved	Must be kept at reset value
6:0	TLEN[6:0]	Transfer length The total data bytes number of a transfer. This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint’s Tx FIFO, this field is decreased by the byte size of the packet.

Device OUT endpoint 0 transfer length register (USBFS_DOEP0LEN)

Address offset: 0x0B10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	STPCNT[1:0]		Reserved									PCNT	Reserved		
	rw											rw			

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	STPCNT[1:0]	SETUP packet count This field defines the maximum number of back-to-back SETUP packets this endpoint can accept. Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS_DOEP0INTF register will be triggered. 00: 0 packet 01:1 packet 10: 2 packets 11: 3 packets
28:20	Reserved	Must be kept at reset value

19	PCNT	Packet count The number of data packets desired to receive in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.
18:7	Reserved	Must be kept at reset value
6:0	TLEN[6:0]	Transfer length The total data bytes number of a transfer. This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Program this field before the endpoint is enabled. Each time software reads out a packet from the Rx FIFO, this field is decreased by the byte size of the packet.

Device IN endpoint-x transfer length register (USBFS_DIEPxLEN) (x = 1..3, where x = endpoint_number)

Address offset: 0x910 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	MCPF[1:0]]		PCNT[9:0]										TLEN[18:16]		
	rw		rw										rw		

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	MCPF[1:0]	Multi packet count per frame This field indicates the packet count that must be transmitted per frame for periodic IN endpoints on the USB. It is used to calculate the data PID for isochronous IN endpoints by the core. 01: 1 packet 10: 2 packets

		11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.
18:0	TLEN[18:0]	Transfer length The total data bytes number of a transfer. This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.

Device OUT endpoint-x transfer length register (USBFS_DOEPxLEN) (x = 1..3, where x = endpoint_number)

Address offset: 0x0B10 + (endpoint_number × 0x20)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	RXDPID/STPCNT[1:0]		PCNT[9:0]										TLEN[18:16]		
	r/rw		rw										rw		

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	RXDPID[1:0]	Received data PID (For isochronous OUT endpoints) This field saves the PID of the latest received data packet on this endpoint. 00: DATA0 10: DATA1 Others: Reserved
	STPCNT[1:0]	SETUP packet count (For control OUT Endpoints.)

		This field defines the maximum number of back-to-back SETUP packets this endpoint can accept. Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS_DOEPxINTF register will be triggered. 00: 0 packet 01:1 packet 10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to receive in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.
18:0	TLEN[18:0]	Transfer length The total data bytes number of a transfer. This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Program this field before the endpoint is enabled. Each time after software reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

Device IN endpoint-x transmit FIFO status register (USBFS_DIEPxTFSTAT) (x = 0..3, where x = endpoint_number)

Address offset: 0x0918 + (endpoint_number × 0x20)
Reset value: 0x0000 0200

This register contains the information of each endpoint's Tx FIFO.

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	IEPTFS[15:0]	IN endpoint’s Tx FIFO space remaining I N endpoint’s Tx FIFO space remaining in 32-bit words: 0: FIFO is full 1: 1 word available … n: n words available

24.7.4. Power and clock control register (USBFS_PWRCLKCTL)

Address offset: 0x0E00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SHCLK	SUCLK
														rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	SHCLK	Stop HCLK Stop the HCLK to save power. 0:HCLK is not stopped 1:HCLK is stopped
0	SUCLK	Stop the USB clock Stop the USB clock to save power. 0:USB clock is not stopped 1:USB clock is stopped

25. Revision history

Table 25-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Dec.26, 2014
2.0	Adapt To New Name Convention	Jun.20, 2017
2.1	Revision content：remove ADC2 content in connectivity type of chip; modify AFIO_PCF0 register in GPIO chapter; modify contents of overview in ADC chapter; modify ITI1 of TIMER1 in TIMER chapter; add notices and modify contents in I2C chapter.	Aug.7, 2018
2.2	Revision content：add SPI2_REMAP bit to chapter 7.5.9 Middle-density, High-density and Extra-density devices register; modify the USART2_REMAP[1:0] in register chapter 7.5.9; modify CAN RFO1 register access attributes in chapter 21.4.5; modify I2C register information in chapter 17.4.1, 17.4.6 and 17.4.7; add description of DAC output buffer; modify the DBG_CTL register; modify the sleep mode wakeup description; modify the description of ADC, DAC, V_{REF} in PMU module chapter 3.3.2, and modify the WKUP1-4 names; ENET module receive descriptor bit 15 modification.	Oct.9, 2019
2.3	Revision content：USBD_INTF register modification; BKP_OCTL register description modification in chapter 4.3.1 and 4.4.2.	Mar.12, 2020

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.